МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

СИСТЕМИ ЗБОРУ ДАНИХ ТА ЇХ КОМПАКТНОГО ПРЕДСТАВЛЕННЯ

КОНСПЕКТ ЛЕКЦІЙ
для здобувачів освітнього ступеня бакалавра
з спеціальності 151 «Автоматизація та комп`ютерно-інтегровані технології»
денної форми навчання

Черкаси 2018

УДК 004.62

С54

Затверджено Вченою радою ФІТІС, протокол № 12 від 26.06.2018 р.
згідно з рішенням кафедри робототехніки та спеціалізованих комп'ютерних систем, протокол № 12 від 04.06.2018 р.

Упорядники: Нечипоренко О. В., к.т.н., доцент,
Корпань Я. В., к.т.н., доцент.

Рецензент Уткіна Т. Ю., к.т.н., доцент.

С54 **Системи** збору даних та їх компактного представлення: конспект лекцій для здобувачів освітнього ступеня бакалавра з спеціальності 151 «Автоматизація та комп`ютерно-інтегровані технології» денної форми навчання [Електронний ресурс] / [Упорядники О. В. Нечипоренко, Я. В. Корпань]; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси: ЧДТУ, 2018. – 240 с.

Матеріал конспекту лекцій систематизований відповідно до навчальної програми. Висвітлено 15 тем (20 лекцій), які є базовими для опанування дисципліни. До кожної лекції наводяться питання для контролю і перевірки знань.

Для студентів освітнього ступеня бакалавра спеціальності 151 «Автоматизація та комп`ютерно-інтегровані технології» денної форми навчання.

УДК 004.62

Навчальне електронне видання
комбінованого використовування

КОНСПЕКТ ЛЕКЦІЙ

для здобувачів освітнього ступеня бакалавра
зі спеціальності 151 «Автоматизація та комп`ютерно-інтегровані технології»

Упорядники: **Нечипоренко** Ольга Володимирівна,
Корпань Ярослав Васильович

В авторській редакції

ЗМІСТ

ВСТУП

Передача, зберігання і обробка інформації вимагають чималих витрат. Найбільш просте у використанні подання даних вимагає вдвічі - втричі, а іноді і в сотні разів більше місця для їх збереження і смугу частот для їх передачі, ніж насправді потрібно.

Система збору даних (англ. *Data acquisition*, DAQ) - в обчисленні і аналізі сигналів перший етап обробки даних, що полягає у накопиченні та підготовці даних для подальшої обробки та інтерпретації. Зазвичай DAQ забезпечує перетворення в цифровий код сигналів, що надходять від багатьох датчиків та передачу їх в комп'ютерних пристрій через систему зв'язку.

Стиснення даних – це процедура представлення даних в більш компактній формі за рахунок усунення надлишковості з метою економії ресурсів системи передачі і зберігання інформації.

Стиснення в даний час застосовуються практично у будь-якій більш-менш складній системі, що оперує із значними обсягами даних – достатньо навести приклади архівування файлів, форматів аудіо- та відеоданих, які використовують стиснення (MP3, JPEG, GIF, PNG, AVI, MPEG і т. д.), а також цифрового зв’язку, зокрема мобільного.

Стиснення даних можливе лише в тому випадку, коли вони містять надлишковість. Практично будь-які осмислені дані, що не були попередньо стиснуті, містять надлишковість, тобто можуть бути стиснені. Не можуть бути стиснутими лише дані, що носять абсолютно випадковий характер, з однаковою ймовірністю появи всіх можливих символів.

Найбільший ступінь надлишковості – відповідно найбільший ступінь стиснення та найбільший ефект від використання цієї процедури – характерний для зображень в растровому форматі, та, дещо меншою мірою, для оцифрованого звуку.

Дані вимірювань в системах автоматизованого контролю та управління (оцифровані сигнали аналогових датчиків) також містять надлишковість, зумовлену подібністю (близькістю) сусідніх відліків сигналу. Смуга частот, ширина якої загалом і визначає необхідний обсяг даних за одиницю часу, залежить від характеру фізичного процесу, а необхідна точніть представлення визначається в першу чергу похибкою датчика, а не суб’єктивними вимогами до якості сигналу.

Значною надлишковістю володіють повідомлення на природній мові. Навіть якщо не брати до уваги, смислову надлишковість (технічна реалізація усунення якої можлива хіба що за умови застосування апарату штучного інтелекту), статистична надлишковість, обумовлена різною ймовірністю появи символів у тексті в залежності від контексту, складає близько 60%, тобто можна сказати, що на кожних 10 символів припадає 6 надлишкових.

Системи збору даних та їх компактного представлення - це нормативна дисципліна циклу професійно-орієнтованих дисциплін підготовки бакалаврів, що викладається у сьомому та восьмому семестрах.

Предметом вивчення навчальної дисципліни є системи збору даних і кодування інформації, перетворення та компактного представлення кодової інформації.

Метою викладання навчальної дисципліни «Системи збору даних та їх компактного представлення» є вивчення методів представлення інформації (кодування), ефективного кодування, компактного представлення даних та методами компресії даних.

Основним ***завданням*** вивчення дисципліни «Системи збору даних та їх компактного представлення» є розуміння та засвоєння студентами теоретичних знань і практичних навиків і вмінь щодо проектування і використання систем компактного представлення даних.

Компетентності:

Згідно з вимогами освітньої-професійної програми студент повинен сформувати наступну компетентність - здатність проектувати багаторівневі систем керування, збору даних та їх архівування для формування бази даних параметрів процесу та їх візуалізації.

Програмні результати навчання (відповідно ОПП):

1. Знати основні поняття і принципи організації систем компактного представлення даних.
2. Знати основні стандарти та інтерфейси в області компресії різноманітних типів цифрових даних.
3. Застосовувати різні утиліти для тестування та налагодження програмних компресорів.
4. Використовувати набуті навички з використання засобів програмування в цілях створення комплексів компактного представлення даних.

На вивчення навчальної дисципліни відводиться 240 годин та 8 кредитів ECTS.

Інформаційний обсяг навчальної дисципліни

Тема 1. Багатоканальні системи збору даних – 2 год

Структура систем збору і обробки інформації. Складові системи збору даних. Загальна характеристика вимірювальних перетворювачів (датчиків). Характеристики аналого-цифрових перетворювачів. Бездротові технології передачі даних в СЗОІ

Тема 2. Системи збору даних з віддалених ресурсів – 2 год

Збір даних з Web-ресурсів. Інтелектуальна система пошуку та збирання інформації. Основні характеристики пошукових систем. Загальна модель функціонування тематичної пошукової системи

Тема 3. Теорія стиснення інформації: визначення, основні принципи та області застосування – 2 год

Вступ до дисципліни. Надлишковість та фактори, що її зумовлюють. Стиснення даних в різних інформаційних системах. Класифікація методів стиснення. Продуктивність стиснення, основні характеристики методів. Перспективи розвитку методів стиснення.

Тема 4. Кодування джерел без пам`яті. Методи стиснення без втрат – 4 год

Нерівномірні префіксні коди. Загальні відомості про методи стиснення без втрат. Оптимальні нерівномірні префіксні коди. Арифметичне стиснення. Адаптивні методи статистичного кодування.

Тема 5. Словникові методи стиснення даних – 2 год

Ідея словникових методів. Класичні алгоритми Зіва-Лемпеля. Алгоритми LZ77 та LZ78 і їх модифікації.

Тема 6. Стиснення даних із послідовностями однакових символів і сортуючі перетворення – 2 год

Методи стиснення даних із послідовностями однакових символів. Методи сортуючих перетворень.

Тема 7. Методи контекстного моделювання – 4 год

Концепція універсального моделювання та кодування. Класифікація стратегій моделювання. Основні терміни методів контекстного моделювання. Види контекстного моделювання. Алгоритми PPM. Оцінка ймовірності відходу.

Тема 8. Аналіз і оцінка методів стиснення даних для сучасних баз даних – 2 год

Актуальність задач стиснення баз даних. Аналіз основних методів стиснення даних в СУБД. Оцінка методів і алгоритмів стиснення табличних даних БД.

Тема 9. Загальний огляд алгоритмів стиснення зображень – 4 год

Особливості зображень та актуальність їх стиснення. Класи зображень. Класи додатків. Вимоги додатків до алгоритмів компресії. Критерії порівняння алгоритмів стиснення. Короткий огляд алгоритмів стиснення зображень

Тема 10. Алгоритми архівації цифрових зображень без втрат – 2 год

Алгоритми групового кодування (RLE). Словникові алгоритми. Алгоритм LZW. Алгоритм Хаффмана з фіксованою таблицею CCITT GROUP 3. Алгоритм JBIG. Алгоритм Lossless JPEG.

Тема 11. Руйнівні алгоритми стиснення цифрових зображень: алгоритм JPEG – 2 год

Алгоритми стиснення зображень з втратами. Алгоритм JPEG. Алгоритм JPEG 2000.

Тема 12. Руйнівні алгоритми стиснення цифрових зображень: рекурсивний і фрактальний алгоритми – 2 год

Рекрсивний (хвильовий) алгоритм. Фрактальний алгоритм. Відмінності між форматом і алгоритмом.

Тема 13. Загальний огляд алгоритмів стиснення відеоданих – 2 год

Основні поняття та принципи стиснення відеоданих. Вимоги додатків до алгоритмів стиснення відео. Визначення класів програмного і апаратного забезпечення для алгоритмів стиснення відео. Огляд стандартів стиснення відеоданих. Базові технології стиснення відео.

Тема 14. Технології і алгоритми стиснення відеоданих – 4 год

Технології стиснення цифрового відео. Огляд технологій та алгоритмів стиснення відео. Алгоритми стиснення відео без втрат. Алгоритми стиснення відео з втратами. Порівняння стандартів стиснення відео. Шляхи підвищення ступеня стиснення відеоданих.

Тема 15. Загальний огляд алгоритмів стиснення аудіосигналів – 4 год

Параметри звукового сигналу. Основні види алгоритмів стиснення аудіосигналів. Характеристика алгоритмів стиснення аудіосигналів. Аналіз форматів стиснення звукових даних. Аналіз можливостей алгоритмів стиснення аудіосигналів.

ЛЕКЦІЯ 1

на тему: Багатоканальні системи збору даних

*У лекції приводяться основні відомості про технічні засоби збору інформації від зовнішніх пристроїв. Розглядаються с*истеми збору даних, які здійснюють функцію перетворення первинних вхідних сигналів від одного або декількох вимірювальних перетворювачів в еквівалентні цифрові сигнали, що придатні для подальшого оброблення, відображення інформації або використання в системах управління.

План:

1. Структура систем збору і обробки інформації.
2. Складові системи збору даних.
3. Загальна характеристика вимірювальних перетворювачів (датчиків).
4. Характеристики аналого-цифрових перетворювачів.
5. Бездротові технології передачі даних в СЗОІ

При побудові пристроїв, що використовують обчислювальну техніку для управління і контролю складними процесами, виникає питання обробки в реальному масштабі часу швидкозмінні сигнали, які будуть надходити одночасно від багатьох джерел і датчиків, та відновлювати аналогову інформацію і розподіляти її між різними виконавчими пристроями. Для цього слід використовувати аналого-цифрові перетворювачі багатоканальної інформації, що відрізняються наступними характеристиками: швидкодією, собівартістю, точністю перетворення і т.д. Пристрої, які здійснюють нормалізацію і аналого-цифрове перетворення сигналів з подальшим введенням інформації в комп'ютерну систему, називають системами збору даних (СЗД).

Система збору даних (англ. *Data acquisition*, **DAQ**) - в обчисленні і аналізі сигналів перший етап обробки даних, що полягає у накопиченні та підготовці даних для подальшої обробки та інтерпретації. Зазвичай **DAQ** забезпечує перетворення в цифровий код сигналів, що надходять від багатьох датчиків та передачу їх в комп'ютерних пристрій через систему зв'язку.

Системи збору даних здійснюють функцію перетворення первинних вхідних сигналів від одного або декількох вимірювальних перетворювачів в еквівалентні цифрові сигнали, які придатні для подальшого оброблення, відображення інформації або використання в системах управління. Діапазон застосувань систем збору даних широкий - починаючи з простого поточного контролю значень однієї аналогової змінної і закінчуючи контролем та управлінням сотнею параметрів в складних автоматизованих технічних системах. Відповідно існують як досить дешеві системи збору даних прямого перетворення, так і складні багатоканальні системи, що забезпечують дуже високу точність та надійність.

Послідовні кроки збору даних:

- використання явища або фізичної властивості;
- отримання сигналу від датчика (датчиків);
- передача сигналу (телематика);
- дискретизація;
- квантування;
- обробка сигналу для подальших потреб.

Інформативними можуть бути зміна температури, тиску, об'єму рідини і т.д. Сенсор реєструє потрібну величину, а потім, після її перетворення в електричний сигнал перетворювачем (датчиком), дані посилаються на комп'ютер або записуючий пристрій. Характеристики системи збору даних, тип і діапазон значень багато в чому залежать від сенсора і АЦП.

1. Структура систем збору і обробки інформації

У теперішній час в усіх сферах діяльності людини актуальною є задача створення недорогих розподілених систем збору і обробки інформації (СЗОІ) або інформаційно-керуючих систем. Такі системи призначені для збору й обробки інформації, яка поступає від різних об'єктів, і, як правило, складаються із контрольованих об'єктів, обладнання, яке формує, оброблює, архівує інформацію, та обладнання, яке об'єднує всі згадані об'єкти в єдину систему.

Розвиток електроніки та телекомунікаційних технологій за останні 10 років привів до того, що в багатьох сферах діяльності людини почали розроблятися та впроваджуватися електронні системи збору інформації з дистанційним зніманням виміряних даних. Вони включали в себе інтелектуальні сенсори, лінії зв'язку, передаючу та приймаючу апаратуру, центральний диспетчерський пункт, в якому здійснюється збір і накопичення інформації. Приклад типової СЗОІ показано на рис. 1.1.

Багато в чому ефективність роботи СЗОІ визначається рівнем використаних технологій збору даних і середовищем передачі інформації. Чим більші масштаби системи, тим більший внесок цих складових у загальний показник ефективності роботи СЗОІ. Відповідно, перш за все, потрібно вибрати технологію передачі даних. На даний час існуючі технології передачі даних можна поділити на провідні та безпровідні.

Не сьогоднішній день перспективним є СЗОІ в яких передбачено використання інтелектуальних портативних приладів як мобільних засобів збору інформації у будь-якій галузі. В таких системах виміряні дані з портативних приладів поступають у диспетчерський центр, де ці дані обробляються, аналізуються і узагальнюються. Отримані результати представляються у графічній, табличній або іншій формі і надалі вони використовуються для побудови узагальненої карти стану системи.

Вимірювання і попередня обробка інформації інтелектуальними портативними приладами можуть здійснюватися на значній відстані від телефонних вузлів і, відповідно, швидка передача даних вимірювання

неможлива. Прокладка кабельних ліній зв'язку у даному випадку недоцільна і часто неможлива.

Рисунок 1.1 - Структура типової СЗОІ

До недоліків першої групи технологій, а саме провідних, необхідно віднести великі затрати на розгортання та експлуатацію подібних систем і відсутність у них такої важливої властивості як мобільність точок збору даних.

2. Складові системи збору даних

Для здійснення збору даних в сучасних автоматизованих системах управління технологічним процесом використовують персональні комп'ютери, програмовані логічні контролери, аналого-цифрові перетворювачі (АЦП) та інше спеціалізоване обладнання. Вибір комплексу технічного забезпечення для системи збору даних проводиться на основі відомостей про технологічний процес і обладнання, значень параметрів технологічного процесу та їх відхилення, вимог до системи автоматизації та типу датчиків і виконавчих механізмів, з врахування швидкості передачі інформації, та вимог, що ставляться до точності вимірювання параметрів. Коректність отриманих результатів визначається компонентами системи збору даних, до яких входять:

- датчики;
- виконавчі механізми;
- пристрої узгодження сигналів;
- шини передачі даних;

- обладнання для збору даних: програмовані логічні контролери, аналого-цифрові перетворювачі (АЦП) та інше спеціалізоване обладнання);
- персональний комп'ютер;
- спеціалізоване програмне забезпечення.

При виборі складу технічного забезпечення автоматизованої системи управління технологічним процесом враховують їх вартість, тому структура простою системи збору даних може складатись з датчиків, виконавчих елементів, системи узгодження сигналів, що підключені безпосередньо до одного з портів комп'ютерної системи.

3. Загальна характеристика вимірювальних перетворювачів (датчиків)

Вимірювання – єдиний засіб одержання кількісної інформації про величини, які характеризують ти чи інші фізичні явища та процеси. Вимірюванням розуміють знаходження дослідним шляхом за допомогою технічних засобів значень фізичної величини, які вибираються з прийнятої шкали цих значень.

Вимірювальне перетворення являє собою відображення розміру однієї фізичної величини розміром іншої фізичної величини, що функціонально з нею пов'язана.

Використання вимірювальних перетворень є єдиним методом практичної побудови будь-яких вимірювальних пристроїв, тому що кожний вимірювальний засіб використовує ти чи інші функціональні зв'язки (найпростіші або більш складні) між вхідною та вихідною величинами. Так, якщо розуміти під функціональним перетворенням і масштабне перетворення у вигляді множення на сталий коефіцієнт (в тому числі такий, що дорівнює одиниці), то таке перетворення має місце і в найпростіших приладах, наприклад, в вільному гіроскопі, в якому вимірювана величина (кут повороту основи) відраховується по куту повороту рамок підвісу з коефіцієнтом перетворення, що дорівнює одиниці. В інтегруючому гіроскопі таке перетворення проводиться з коефіцієнтом, який може відрізнятися від одиниці в той чи інший бік.

Вимірювальний перетворювач – це технічний пристрій, який побудований на певному фізичному принципі дії і виконує одне часткове вимірювальне перетворення. Поняття "вимірювальний перетворювач" значно більш вузьке, більш конкретне, ніж поняття "вимірювальне перетворення", тому що одне й теж вимірювальне перетворення може виконуватися цілою низкою різних за принципом дії вимірювальних перетворювачів. Так, вимірювання кутової швидкості повороту основи може бути здійснене механічними гіроскопами, хвильовими та лазерними пристроями, тощо.

Як правило, вимірювальний перетворювач називають датчиком.

Системи класифікації датчиків можуть бути різними, від дуже простих до складних. Критерій класифікації завжди вибирається залежно від мети проведення класифікації.

Класифікація за *видом* сенсора:

- активні (генераторні);
- пасивні (параметричні).

Класифікація за вибором *точки відліку*:

- абсолютні;
- відносні.

Класифікація за *зовнішньою дією* (вимірювальною величиною):

- датчики тиску:

- абсолютного тиску;
- відносного тиску;
- розрідження;
- тиску-розрідження;
- різниці тисків;
- гідростатичного тиску;

- датчик витрати:

- механічні лічильники витрати;
- перепадоміри;
- ультразвукові витратоміри;
- електромагнітні витратоміри;
- коріолісові витратоміри;
- вихрові витратоміри;

- датчик рівня:

- поплавкові;
- ємнісні;
- радарні;
- ультразвукові;

- датчик температури:

- відносні (сенсор – термопара);
- абсолютні (сенсор – терморезистор);
- пірометр;

- датчики концентрації (кондуктометри);
- датчики радіоактивності (детектори радіоактивності або випромінювання):

- іонізаційна камера;
- датчик прямого заряду;

- датчик переміщення:

- абсолютний шифратор;
- відносний шифратор;
- LVDT (лінійний диференційний перетворювач, що регулюється);

- позиційні вимикачі:

- контактні;
- безконтактні;

- датчики кутового положення:

- сельсин;
- перетворювач кут – код (енкодер);
- RVDT (обертний диференційний перетворювач, що регулюється);

- давачі вібрацій:

- п’єзоелектричний;
- вихрострумовий;

- давач механічних величин:

- відносного розширення;
- абсолютного розширення.

Класифікація за *характеристиками*:

- чутливість;
- стабільність;
- точність;
- гістерезис;
- швидкодія;
- діапазон вхідних значень;
- формат вихідного сигналу;
- інші.

Класифікація за *кількістю вхідних величин*:

- одномірні;
- багатомірні.

Класифікація за *принципом дії*:

- фотоелектричні (оптичні);
- магнітоелектричні (на підставі ефекту Холла);
- п’єзоелектричні;
- тензодавачі;
- ємнісні;
- потенціометричні;
- індуктивні;
- індукційні;
- ультразвукові.

Класифікація за *принципом реалізації вихідних перетворювачів*:

- з дискретним виходом:

- статичні;
- електромеханічні;

- з аналоговим виходом;
- з цифровим виходом.

Класифікація за *конструкцією*:

- суцільний, інакше компактний, або моноблочний;
- складений (складається з окремих конструктивних частин), тобто виступає як функціональна група, інакше, наприклад, двоблочний.

4. Характеристики аналого-цифрових перетворювачів

Основу вище вказаної системи складає АЦП. Існують три основні типи АЦП: інтегруючий, послідовного наближення і паралельний.

Інтегруючий АЦП усереднює вхідний сигнал за часом, є найточнішим, але і повільним. Час перетворення інтегруючого АЦП лежить в діапазоні від 0,001 до 50с і більше, похибка становить 0,1-0,0003%.

Похибка АЦП послідовного наближення дещо більша (0,4-0,002%), але час перетворення - від ~ 10мкс до ~ 1 мс.

Паралельні АЦП - швидкодіючі, але і найменш точні: їх час перетворення порядку 0,25 нс, похибка - від 0,4 до 2%.

В системах збору даних не ставлять такі високі вимоги до частоти дискретизації, як в цифрових осцилографах, і до роздільної здатності, як в цифрових мультиметрах. Тому в даних системах переважно використовуються АЦП послідовного наближення та сигма-дельта АЦП, що належать до класу інтегруючих перетворювачів.

АЦП паралельного типу (прямого перетворення) володіють найвищою швидкодією, яка визначається швидкодією компараторів і затримками в логічному дешифраторі. Недоліком їх є необхідність великої кількості компараторів. Кількість можливих кодових комбінацій (а отже, і загальна кількість компараторів) дорівнює 2^m-1, де m - кількість розрядів АЦП.

АЦП послідовного наближення має дещо меншу швидкодію, але істотно більшу розрядність (роздільну здатність). В ньому використовується тільки один компаратор, максимальна кількість спрацьовувань якого за один цикл вимірювання не перевищує кількості розрядів перетворювача. АЦП послідовного наближення розширили діапазон частот дискретизації до мегагерц.

Першими для високоточної реалізації алгоритмів аналого-цифрового перетворення вимірювальних сигналів використовувалися інтегруючі АЦП (ІАЦП). Поєднання високої точності і завадостійкості ІАЦП дозволили вирішити ряд проблем боротьби з перешкодами у вимірювальних каналах. Серед завадостійких ІАЦП, що поєднують аналогові і цифрові методи, лідерами є сигма-дельта АЦП, які реалізують алгоритми аналого-цифрової фільтрації для зменшення шуму квантування.

Для підвищення функціональних можливостей АЦП можна, наприклад, використовувати аналого-цифрові перетворювачі Монте-Карло. Метод Монте-Карло дозволяє вирішувати завдання моделювання випадкових процесів шляхом статистичних досліджень на основі аналітичної формалізації об'єкту та здійснювати перетворення інтегралу вхідної функції в цифровий код в реальному часі. Для цього за допомогою генератора випадкових чисел формуються випадкові відліки, значення яких порівнюються із значенням вхідного параметру, в результаті чого обчислюється сукупність позитивних

результатів статистичного дослідження вхідного параметру. Така реалізація полягає у проведенні статистично незалежних циклічних досліджень із накопиченням вказаних результатів порівняння. Характеристичною для методу Монте-Карло є оцінка математичного очікування дискретизованих значень випадкової величини вхідного параметру, яка визначає інтеграл функції перетворення за обчисленою сумою відліків позитивних результатів порівняння, внаслідок чого формується результат аналого-цифрового перетворення в області визначення, заданій відомим рівномірним імовірнісним розподілом псевдовипадкового опорного сигналу.

Від характеристик АЦП залежить кінцевий вибір користувача і їх область застосування. Для цифрової частини АЦП важливим є швидкість передачі даних, яка може перебувати в діапазоні від декількох десятків Гц до декількох ГГц. Розрядність даних на виході АЦП – у сучасних пристроях від 8 до 32 біт даних. З боку аналогових характеристик важливе відношення сигнал-шум (SNR) і динамічний діапазон, який вільний від паразитних складових, що виражаються, як правило, в децибелах (типові рівні SNR лежать в межах 70-80 дБ). Важливими параметрами при виборі АЦП є кількість каналів в одному корпусі, інтерфейс обміну з цифровим вузлом і, зрозуміло, вартість компонента.

Точність перетворення залежить від:

1. Похибки квантування і похибок, що створюються електронними вузлами, такими, як ЦАП, компаратор, джерело опорної напруги.

Точність вимірювання, як і точність встановлення будь-якого процесу в електронних вузлах, зменшується по мірі збільшення швидкості перетворення. Похибка при цьому збільшується за логарифмічною залежністю, оскільки вона в основному залежить від сталих часу встановлення сигналу.

Похибка перетворення і швидкодія аналого-цифрового перетворювача послідовного наближення визначається в основному параметрами ЦАП (роздільною здатністю, лінійністю, швидкодією) і компаратора (порогом чутливості, швидкодією). Точність АЦП з паралельним перетворенням обмежується точністю і стабільністю кожного компаратора. Похибка інтегруючих перетворювачів визначається нестабільністю порогу спрацювання компаратора. Похибки перетворення однотактного інтегруючого АЦП з пилоподібною напругою поділяються на два основні класи: похибки, що пов’язані з параметрами пилоподібної напруги, та похибки, що викликані неточністю тактування. Точність перетворення двотактного інтегруючого АЦП не залежить від ємності конденсатора, ні від частоти тактового генератора, оскільки їх вплив однаково позначається на час повернення до нуля. Істотною перевагою методу двотактного інтегрування полягає в тому, що він придатний для створення перетворювачів, що забезпечують зменшення вхідних завад при деяких частотах. Похибка АЦП Монте-Карло визначається кількістю згенерованих на області інтегрування множини випадкових чисел та якістю рівномірності їх розподілу на даній області.

2. Динамічної похибки.

Виникнення динамічних похибок викликано дискретизацією сигналів, що змінюються в часі. Можна виділити наступні параметри АЦП, що визначають її динамічну точність: максимальна частота дискретизації, час перетворення.

АЦП двотактного інтегрування мають високу точність і високу роздільну здатність, а також мають порівняно просту структуру. Це дає можливість виконувати їх у вигляді інтегральних мікросхем. Основний недолік таких АЦП - великий час перетворення.

В АЦП послідовного наближення час перетворення визначається кількістю тактів, які в свою чергу залежать від розрядності АЦП. Але слід відмітити, що робота АЦП послідовного наближення має особливість, яка пов'язану з перехідними процесами у внутрішньому ЦАП. Теоретично, напруга на виході ЦАП для кожного з n тактів перетворення має встановлюватися за однаковий проміжок часу. Але насправді цей проміжок в перших тактах значно більше, ніж в останніх. Тому час перетворення 12-розрядного АЦП послідовного наближення більш, ніж в два рази перевищує час перетворення 6-розрядного АЦП даного типу.

Для паралельного АЦП цифровий код, що відповідає вхідній аналоговій напрузі, появляється на виході майже миттєво (якщо не враховувати невеличкої затримки, що пов’язана з роботою компараторів).

Для АЦП Монте-Карло частота перетворення визначається динамічними характеристиками ЦАП і компаратора. Час перетворення являється сталою величиною та дорівнює

$$t_{пр} = 2^n T,$$

де n – розрядність АЦП;

T – період слідування тактовихімпульсів, що відповідає часу вибірки одного кванта.

5. Бездротові технології передачі даних в СЗОІ

Основними критеріями вибору технології безпровідної передачі даних є: зона покриття зв'язком; мобільність терміналів зв'язку; можливість визначення географічних координат місця здійснення сеансу зв'язку; передача невеликих об'ємів даних; двосторонній зв'язок; одночасне підключення до обладнання зв'язку диспетчерського центру кількох терміналів зв'язку; можливість швидкої, дешевої і простої модернізації і розвитку системи; можливість швидкого добавлення нових терміналів зв'язку у систему.

Кожна з безпровідних технологій має свої переваги та недоліки.

Так технологію Bluetooth зручно використовувати для поєднання недалеко встановлених пристроїв, наприклад в офісі чи в автомобілі. Вона забезпечує швидкість передачі 721 кбіт/с та дозволяє під'єднувати до одного основного пристрою (Master) до семи підлеглих (Slave) пристроїв, утворюючи пікомережу (piconet).

Технологія Wi-Fi надає високошвидкісний (понад 100 Мбіт/с) та надійний зв’язок (із 64/128-бітним шифруванням), дозволяє легко інтегруватися в існуючі проводові мережі, проте має високу вартість обладнання та велике енергоспоживання. Типовий Wi-Fi маршрутизатор стандарту 802.11b або 802.11g має радіус дії 45 м у приміщенні і 90 м на відкритому просторі.

Технологія ZigBee являє собою самоорганізуючу і самовідновлювальну мережу, яку зручно використовувати на промислових об’єктах, де необхідне встановлення багатьох датчиків з можливістю їх переміщення по території об’єкта. Ця технологія призначена для передачі невеликої кількості даних і забезпечує максимальну швидкість передачі даних разом із службовою інформацією до 250 кбіт/с.

Проте усі три дані мережі є територіально обмежені радіусом дії їх передавачів і приймачів. Для одержання даних практично з будь-якої місцевості зручно використовувати стандарт GSM/GPRS. GSM мережі забезпечують високий рівень безпеки та велику швидкість передачі даних (до 171 кбіт/с або до 473 кбіт/с за новими можливостями стандарту EDGE). Крім того, пристрої, під'єднанні до GSM мережі, мають вихід у всесвітню мережу Інтернет. Щоправда, недоліками такої системи є необхідність оплати за передачу даних, велике енергоспоживання в активному режимі, необхідність наявності покриття GSM-мережею.

Практика показала, що природним і найбільш оптимальним вирішенням поставленої прикладної проблеми є створення безпровідної мережі на базі стільникових терміналів через існуючі системи стільникового зв'язку, які працюють згідно зі стандартами GSM/GPRS. В такому випадку не потрібно оформляти дозволи на використання радіочастот, купувати дороге приймально-передавальне обладнання. До того ж мережа GSM має велике покриття. Отже, частина проблем з організації безпровідного каналу перекладається на оператора стільникового зв'язку.

У загальному випадку система обміну даними через мережу стільникового зв'язку складається з двох компонентів – певної кількості портативних приладів і центрального сервера. На портативні прилади встановлюється обладнання отримання даних і передачі необхідної інформації через мережу стільникового зв'язку. Сервер забезпечує отримання, зберігання, відображення і аналіз отриманих даних.

Існує два способи обміну даними через мережу стільникового зв'язку: пасивний і активний.

При пасивному способі інформація про вимірювання записується в енергонезалежну пам'ять приладу, а потім у певні моменти часу вся зібрана інформація відправляється на сервер.

При активному способі дані про вимірювання передаються на центральний сервер безпосередньо після виконання вимірювання. При активному режимі можливі наступні режими передачі даних: безперервний, періодичний, по події або по запиту з центрального сервера.

Активний спосіб обміну інформацією по каналах GSM/GPRS є найбільш ефективним з точки зору відношення ціна/якість в регіонах з розвинутою структурою мереж GSM. Типова система передачі даних до сервера через стільникову мережу з використанням GPRS режиму представлена на рис. 1.2. За таким способом передачі даних існує ймовірність втрати даних у випадку нестабільного GPRS-каналу. Задачу зберігання даних у таких випадках можна вирішити шляхом запису цих даних у тимчасову пам'ять.

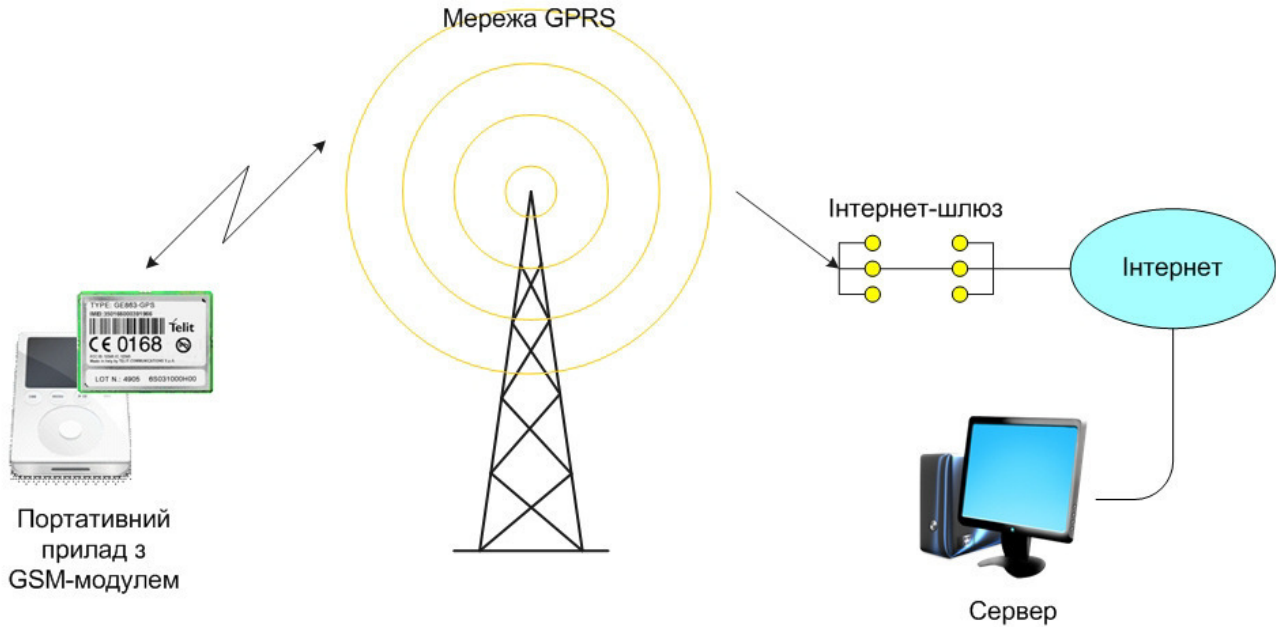

Рисунок 1.2 - Типова система передачі даних через стільникову мережу

Використання стільникового зв'язку значно спрощує і здешевлює створення СЗОІ. Все, що потрібно зробити для забезпечення радіозв'язку – це підключити GSM-модем або, як у даному випадку, вбудувати у прилад GSM-модуль і запрограмувати його відповідно до прикладної задачі. При подальшій експлуатації це не потребує спеціального обслуговування.

Можна виділити наступні переваги СЗОІ з використанням стільникового зв'язку:

– в диспетчерському центрі не обов'язково встановлювати GSM/GPRS-модем за рахунок безпосереднього підключення центрального сервера до мережі Інтернет, як це вказано на рис. 2.2. Це зумовлює зменшення вартості трафіка через стільниковий канал в 2 рази;

– можливість передачі даних обмежена зоною покриття стільниковим зв'язком;

– SIM-картки на терміналах збору і передачі даних можуть належати різним операторам в залежності від зони покриття і якості GPRS;

– у випадку тимчасової відсутності послуги GPRS можна переключитися у режим прямого GSM-з'єднання;

– подібні системи можуть функціонувати без виділення статичної IP-адреси оператором стільникового зв'язку або провайдером.

Для вирішення задач створення СЗОІ, в яких дані передаються через систему стільникового зв'язку, ряд компаній випускає GSM-модеми або GSM-модулі, які відрізняються від мобільних телефонів конструктивним виконанням, відсутністю клавіатури, дисплею, антени, а також наявністю специфічних додаткових можливостей. Подібні системи мають розширений набір АТ команд у відповідності до ряду стандартів GSM (наприклад, GSM 07.07 і GSM 07.05), що дозволяє програмувати їх як звичайні мобільні телефони з використанням програмного забезпечення, яке входить до складу стандартного пакета Microsoft Windows.

GSM-модуль є базовим безкорпусним елементом, для запуску якого в роботу потрібні додаткові комплектуючі й обладнання, зокрема, схеми живлення цифрового і радіочастотного блоків, аудіосистема, інтерфейси і роз'єми для зв'язку із зовнішніми пристроями, тримач та інтерфейс SIM-карти, гарнітура, периферія, корпус. Слід зауважити, що комплектність вказаного додаткового обладнання визначається прикладною задачею.

Як правило, GSM-модулі використовуються для інтеграції в обладнання користувача, де, відповідно, виконують роль передаючого блоку. GSM-модуль призначений для дистанційного зв'язку між сервером і обладнанням (мобільним або стаціонарним), яке знаходиться в області покриття стільниковим зв'язком різних стандартів, наприклад GSM 900 МГц, GSM 1800 МГц та ін. Для вводу команд керування й отримання інформації від даного обладнання також можна використовувати операторну станцію моніторингу на базі персонального комп'ютера з модемом.

Кількість контрольованих входів і виходів, встановлення периферійних додаткових пристроїв, інтерфейсів обміну, вбудованих інтерфейсів, протоколів зв'язку, а також періодичність сеансів зв'язку і конфіденційність даних, що передаються, визначаються характеристиками конкретного GSM-модуля та вимогами прикладної задачі.

Досвід іноземних і вітчизняних фахівців у сфері створення та експлуатації об'єктів автоматизації зі стільниковими каналами показує, що для систем з великим рівнем відповідальності доцільне використання тільки режиму "data". Хоча цей режим і не є повністю гарантованим, але за статистикою 90% загального часу канали є доступні і функціонують.

Режим GPRS можна і доцільно використовувати у системах телеметрії, де збір даних проводиться рідко і протягом короткого проміжку часу. До так званого недоліку режиму GPRS можна віднести неможливість запиту об'єкта з верхнього рівня. Ініціатива запиту зв'язку відбувається завжди зі сторони абонента. Це пов'язано з правилами IP-адресації в операторів стільникового зв'язку. При реєстрації у стільниковій мережі абонента йому призначається IP адрес внутрішньої мережі оператора стільникового зв'язку, і весь обмін інформацією відбувається через механізми NAT (Network address translation).

Слід зазначити також, що GPRS є однією із недорогих послуг, які надаються операторами стільникового зв'язку для передачі даних. Важлива особливість технології GPRS полягає в тому, що тарифікація здійснюється за об'ємом інформації, що передається, а не за часове з'єднання, як це робиться при використанні технології CSD / HSCSD.

Контрольні питання

1. Що розуміють під вимірювальною інформацією?
2. Що розуміють під вимірювальним перетворенням?
3. Що називається вимірювальним перетворювачем?
4. Що називається датчиком?
5. Структура систем збору і обробки інформації.
6. Призначення основних складових системи збору даних.
7. Класифікація датчиків.
8. Що таке аналого-цифрові перетворювачі?
9. Характеристики аналого-цифрових перетворювачів.
10. Бездротові технології передачі даних в СЗОІ.

ЛЕКЦІЯ 2

на тему: Системи збору даних з віддалених ресурсів

У лекції розглядаються деякі особливості та принципи збору та аналізу інформації, яка знаходиться на віддалених носіях, через мережу Internet, їх переваги і недоліки.

План:
1. Збір даних з Web-ресурсів
2. Інтелектуальна система пошуку та збирання інформації
3. Основні характеристики пошукових систем
4. Загальна модель функціонування тематичної пошукової системи

1. Збір даних з Web-ресурсів

Кожна компанія зацікавлена в інформації про ринок, на якому вона працює. Для розвитку потрібні дані, які допомагають генерувати ці знання. Вони надають незаперечну перевагу над конкурентами, допомагаючи знаходити потенційних клієнтів, аналізувати реакцію споживачів на нові продукти, розміщувати більш ефективні стратегії маркетингу, а також розробляти більш якісні продукти.

Процес вилучення структурованої корисної інформації з сайту називається парсингом (parsing), а інструменти для реалізації даного процесу - парсерами (parsers).

Як правило, сайти розробляються з урахуванням того, що зчитувати інформацію з їх сторінок буде людина. Але формат представлених даних, зрозумілий людині, найчастіше не настільки зрозумілий програмним засобам. Крім того, структура представлених даних змінюється від сайту до сайту, тому не існує універсального засобу для вилучення інформації з них. Однак, існують готові рішення, що дозволяють отримувати інформацію з сайту після попередньої конфігурації. Ці рішення часто коштують дорого і не мають тієї гнучкості, яку можуть дати рішення, розроблені під конкретний сайт.

Для того, щоб один раз отримати значення декількох полів з 10-15 сторінок, писати окремий парсер недоцільно, однак в разі великого числа сторінок, полів або високої періодичності збору інформації, цей процес потребує автоматизації. Тому завдання написання системи програмного забезпечення для збору даних, яка накопичує дані дуже важлива.

Web Mining - це процес отримання даних з Web-ресурсів. Так як Web-джерела, як правило, не є текстовими даними, то і підходи до процесу вилучення даних відрізняються в цьому випадку. В першу чергу необхідно пам'ятати, що інформація в інтернеті зберігається у вигляді спеціальної мови розмітки HTML (RSS, Atom та інші), Web-сторінки можуть мати додаткову метаінформацію, а також інформацію про структуру (семантику) документа.

Підходи до вилучення даних:

а) аналіз DOM дерева;

б) використання XPath;

в) використання регулярних виразів;

г) візуальний підхід.

У Web Mining можна виділити наступні етапи:

а) вхідний етап (input stage) - отримання "сирих" даних з джерел;

б) етап обробки (preprocessing stage) - дані подаються у формі, необхідній для успішної побудови тієї чи іншої моделі;

в) етап моделювання (pattern discovery stage);

г) етап аналізу моделі (pattern analysis stage) - інтерпретація отриманих результатів.

Існує багато систем для вилучення даних з інтернет-джерел. Найпопулярніші - Datacol, Content Downloader, X-Parser Light, FDE Grabber, WP Uniparser.

В процесі аналізу були виявлені наступні недоліки:

а) багато разів програма звертається в базу даних або в файл;

б) багато часу займають налаштування;

в) в більшості випадків збирається менше 95% даних з порталів;

ґ) іноді на виході отримуємо помилковий контент;

д) виникають помилки при роботі з великими обсягами даних.

Враховуючи вищесказане систему збору даних з Web-ресурсів можна представити у вигляді рис.2.1. Для реалізації такої системи, яка має назву Extractdata, було використано фреймворк Scrapy, база даних – PostgreSQL та платформа RabbitMQ.

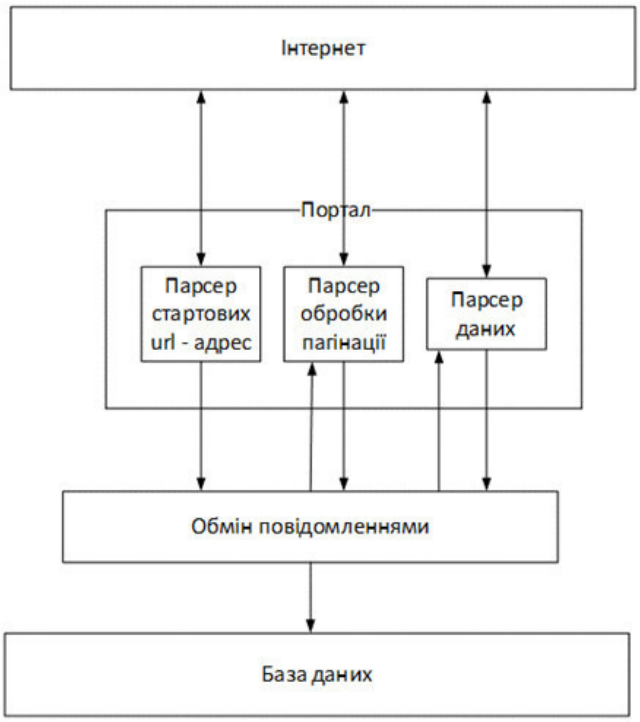

Рисунок 2.1 - Архітектура системи збору даних з Web-ресурсів

Відомо, що обсяг даних в мережі Інтернет зростає за експоненціальною залежністю, тому необхідно розробляти та оптимізувати системи збору даних з Web-ресурсів.

2. Інтелектуальна система пошуку та збирання інформації

У сучасному світі Інтернет вже давно використовують як широкий довідковий інструмент. За останні роки він став середовищем опрацювання та зберігання наукової, бізнесової та інших типів інформації. Але основними особливостями Інтернету є динаміка інформації, її постійне оновлення та поширення по всьому Інтернету.

Саме тому пошукові системи вже давно стали невід'ємною частиною Інтернету. Завдяки ним користувачі Всесвітньої павутини намагаються знайти потрібну інформацію. Переважно, пошук інформації зводиться до пошуку сторінки, на якій розміщена ця інформація, і таких сторінок в Інтернеті може бути декілька. Для отримання результатів, які дадуть змогу порівняти інформацію на однотипних веб-ресурсах, потрібно створити тематичні пошукові системи.

Створення тематичної пошукової системи містить в своїй основі такі задачі:

- підбір тематичних сайтів,
- пошук інформації на різних ресурсах з різними запитами,
- видобування інформації з сайту,
- створення єдиної системи представлення інформації користувачу.

Враховуючи поставлені задачі, алгоритм роботи тематичного пошуковика схожий з алгоритмом звичайної пошукової системи.

Пошукова система складається з таких основних компонентів:

Spider (павук) – браузероподібна програма, яка викачує Web-сторінки.

Crawler (краулер, «мандрівний» павук) – програма, яка автоматично проходить по всіх посиланнях, знайдених на сторінці.

Indexer (індексатор) – програма, яка аналізує веб-сторінки, викачані павуками.

Database (база даних) – сховище викачаних та опрацьованих сторінок.

Search engine results engine (система видачі результатів) – витягує результати пошуку з бази даних.

Web server (веб-сервер) – Web-сервер, який здійснює взаємодію між користувачем та іншими компонентами пошукової системи.

Web server та Search engine results engine часто називають просто пошуковим сервером.

У деталях реалізації пошукових механізмів можуть відрізнятися одна від однієї (наприклад, зв’язка Spider+Crawler+Indexer може бути виконана у вигляді єдиної програми, яка викачує відомі Web-сторінки, аналізує їх і шукає за посиланнями нові ресурси), проте всім пошуковим системам властиві описані загальні риси.

Spider. Забезпечує скачування сторінки і витягує всі внутрішні посилання з цієї сторінки. Викачується html-код кожної сторінки. Для скачування сторінок роботи використовують протоколи HTTP. Працює «павук» так. Робот передає на сервер запит "get/path/document" і деякі інші команди HTTP-запиту. У відповідь робот отримує текстовий потік, що містить службову інформацію і безпосередньо сам документ.

Посилання витягуються з тегів a, area, base, frame, frameset та ін. Крім посилань, роботи обробляють редиректи (перенапрямлення). Кожна викачана сторінка зберігається у такому форматі:

– URL сторінки
– дата, коли сторінка була викачана
– http-заголовок відповіді сервера
– тіло сторінки (html-код)

Crawler. Виділяє всі посилання, присутні на сторінці. Його завдання – визначити, куди далі повинен йти павук, ґрунтуючись на посиланнях або за заздалегідь заданим списком адрес. Краулер, проходячи по знайдених посиланнях, шукає нові документи, ще не відомі пошукові системи.

Indexer. Індексатор розбирає сторінку на складові частини і аналізує їх, застосовуючи власні лексичні і морфологічні алгоритми. Аналізу піддаються різні елементи сторінки, такі як текст, заголовки, посилання структурні і стильові особливості, спеціальні службові html-теги і так далі.

Отже, модуль індексування дає змогу обходити за посиланнями задану кількість ресурсів, викачувати сторінки, що зустрічаються, витягувати посилання на нові сторінки з отриманих документів і здійснювати повний аналіз цих документів.

Database. База даних – це сховище всіх даних, які пошукова система викачує і аналізує. Інколи базу даних називають індексом пошукової системи.

Пошуковий сервер є найважливішим елементом всієї системи, оскільки від алгоритмів, покладених в основу її функціонування, безпосередньо залежить якість і швидкість пошуку.

Пошуковий сервер працює так:

- Отриманий від користувача запит піддається морфологічному аналізу. Генерується інформаційне оточення кожного документа, що міститься в базі (яке і буде згодом відображено у вигляді відповідної текстової інформації на сторінці видачі результатів пошуку).
- Отримані дані передаються як вхідні параметри спеціальному модулю ранжирування. Обробляються дані за всіма документами, внаслідок чого для кожного документа розраховується власний рейтинг, що характеризує релевантність запиту, введеного користувачем, і різних складових цього документа, що зберігаються в індексі пошукової системи.
- Залежно від вибору користувача цей рейтинг може бути скоригований додатковими умовами (наприклад, так званий «розширений пошук»).

- Далі генерується сніппет, тобто для кожного знайденого документа з таблиці документів витягуються заголовок, коротка анотація, яка найбільше відповіднає запиту і посилання на сам документ, причому знайдені слова підсвічують.
- Отримані результати пошуку передаються користувачеві у вигляді сторінки видачі пошукових результатів.

3. Основні характеристики пошукових систем

Точність – ще одна основна характеристика пошукової системи, яка визначається рівнем відповідності знайдених документів запиту користувача. Наприклад, якщо за запитом «як вибрати автомобіль» знаходять 100 документів, в 50 з яких є словосполучення «як вибрати автомобіль», а в інших лише наявність цих слів («як правильно вибрати автомагнітолу та встановити її в автомобіль»), то точність пошуку буде 50/100 (=0,5). Чим точніший пошук, тим швидше користувач знайде потрібні йому документи, тим менше різного роду «сміття» буде в них зустрічатись, тимрідше знайдені документи не відповідатимуть запиту.

Актуальність – не менш важлива складова пошуку, яка характеризується часом, що пройшов від моменту публікації документа в мережі Інтернет, до занесення його до індексної бази пошукової системи. Наприклад, на наступний день після появи цікавої новини, велика кількість користувачів звернулись до пошукових систем з відповідним запитом. Об’єктивно з моменту публікації новини на цю тему пройшло менше доби, однак основні документи вже були проіндексовані та доступні для пошуку завдяки існуванню у великих пошукових системах так званої «швидкої бази», яка поновлюється декілька разів на день.

4. Загальна модель функціонування тематичної пошукової системи

Принцип роботи розробленої тематичної пошукової системи наведено на рис. 2.2.

Користувач, використовуючи можливості клієнтської частини, вводить ключове слово для пошуку інформації та вибирає із списку тематичних сайтів ті, з яких потрібно отримати інформацію за ключовим словом.

Агент формування запиту, отримавши ключове слово і список вибраних сайтів, звертається до бази правил для отримання правил формування запитів до вибраних сайтів. Отримавши ці правила, формує запит до кожного з вибраних сайтів і цю інформацію передає агенту пошуку і зберігання інформації. Своєю чергою, агент пошуку та зберігання інформації відсилає отримані запити в Інтернет і отримує html-коди сторінок, які записує в свою оперативну базу даних. Потім цей агент запитує в бази правил правила видобування інформації з html-кодів. Опрацьовує інформацію, записану в своїй оперативній базі, згідно з правилами і записує вибрану за правилами

інформацію в оперативній базі даних системи. Далі з цією базою даних працює клієнтська частина.

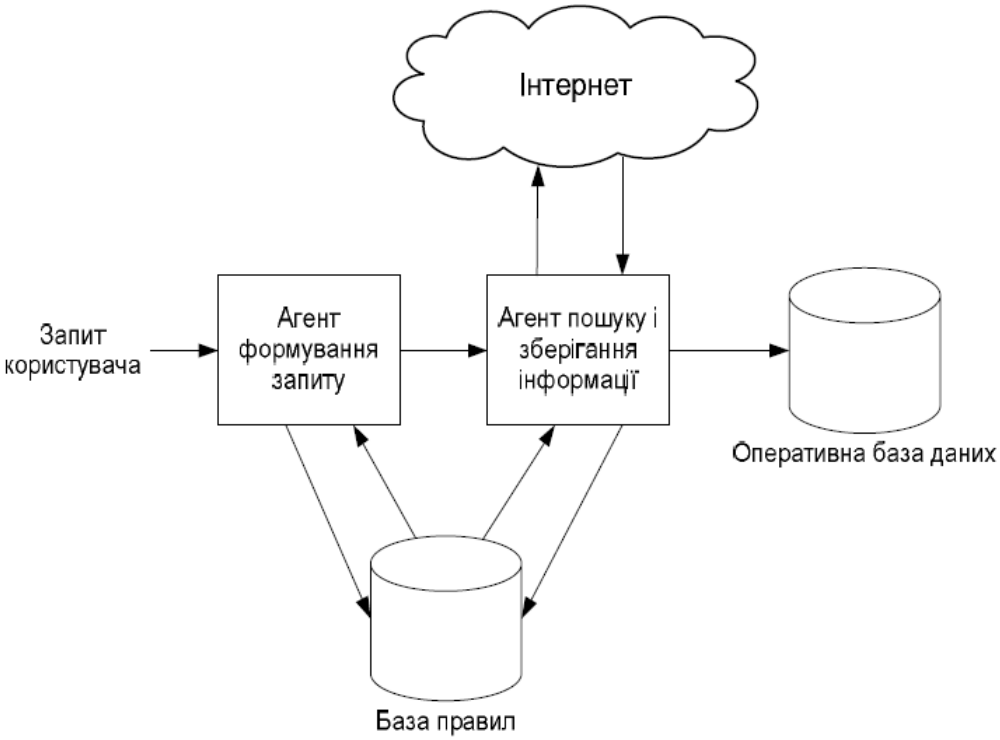

Рисунок 2.2 - Схема роботи тематичної пошукової системи

База правил формується розробником системи. Саме розробник опікується її правильним наповненням. Сама база правил складається з таких полів:

- назва сайту (єдине, що відображається користувачеві в клієнтській частині);
- приклад формування запиту – тут наведено приклад запиту до відповідного сайту;
- інформація про правило поєднання ключового слова із запитом попереднього поля;
- правила видобування інформації з html-коду отриманої в результаті пошуку сторінки відповідного сайту.

Враховуючи динаміку руху інформації в Інтернеті, база правил повинна постійно перебувати під контролем розробника для зазначення відповідних змін властивостей сайту, для внесення нових тематичних сайтів, що з’явилися в мережі. Для наповнення цієї бази можна також використовувати інтелектуальні системи, які б в кооперації зі стандартними системами пошуку відбирали сайти з конкретною тематикою. Але часто для "розкрутки" своїх сайтів веб-майстри використовують не дуже чесні прийоми, які можуть давати неправильні результати аналізу веб-ресурсу. Саме для цього у процесі наповнення бази

правил використовують людський фактор, за допомогою якого коректно відсіюють «непридатні» сайти.

Оперативна база даних складається з інформації, необхідної користувачу, яка є невпорядкованою. Агент пошуку і зберігання інформації записує в оперативну базу даних отриману з html-коду інформацію, яка зберігається під такими полями:

- назва сайту;
- зображення, що відповідає конкретному результату;
- текст результату пошуку;
- посилання на детальнішу інформацію;

Ця база даних формується для оперативного збереження даних, які користувач може переглядати, сортувати, відбирати за певними критеріями, використовуючи клієнтську частину системи.

Агент формування запиту складається з наступних кроків (рис. 2.3):

1. Отримання ключового слова від клієнтської частини.
2. Зчитування списку зазначених користувачем сайтів пошуку.
3. Перевірка кожного елементу зі списку сайтів (позначений – крок 4, позначений – крок 9).
4. Формування запиту до бази правил щодо вибраного сайту.
5. Отримання правила формування запиту до конкретного сайту.
6. Внесення ключового слова пошуку у правило формування запиту.
7. Створення запиту до вибраного сайту.
8. Формування списку запитів для агента пошуку та зберігання інформації (перехід на крок 10).
9. Додавання до списку запитів нульового запиту.
10. Повернення на крок 3, поки не закінчиться список сайтів.
11. Передача списку запитів агенту пошуку та зберігання інформації.

Агент пошуку і зберігання інформації складається з наступних кроків (рис. 2.4):

1. На початку своєї роботи отримує від внутрішнього агента список сформованих запитів до тематичних сайтів.
2. Перевірка, чи запит не нульовий (якщо так – крок 8).
3. Відсилання запиту тематичному сайту.
4. Одержання коду html-коду сторінки, яка сформувалась як відповідь на запит.
5. Звернення до бази правил та отримання правила видобування інформації з коду сторінки.
6. Отримання потрібної інформації з html-коду сторінки (малюнки, текст, посилання).
7. Запис отриманої інформації в оперативну базу даних системи.
8. Перехід на крок 2 до закінчення списку запитів.

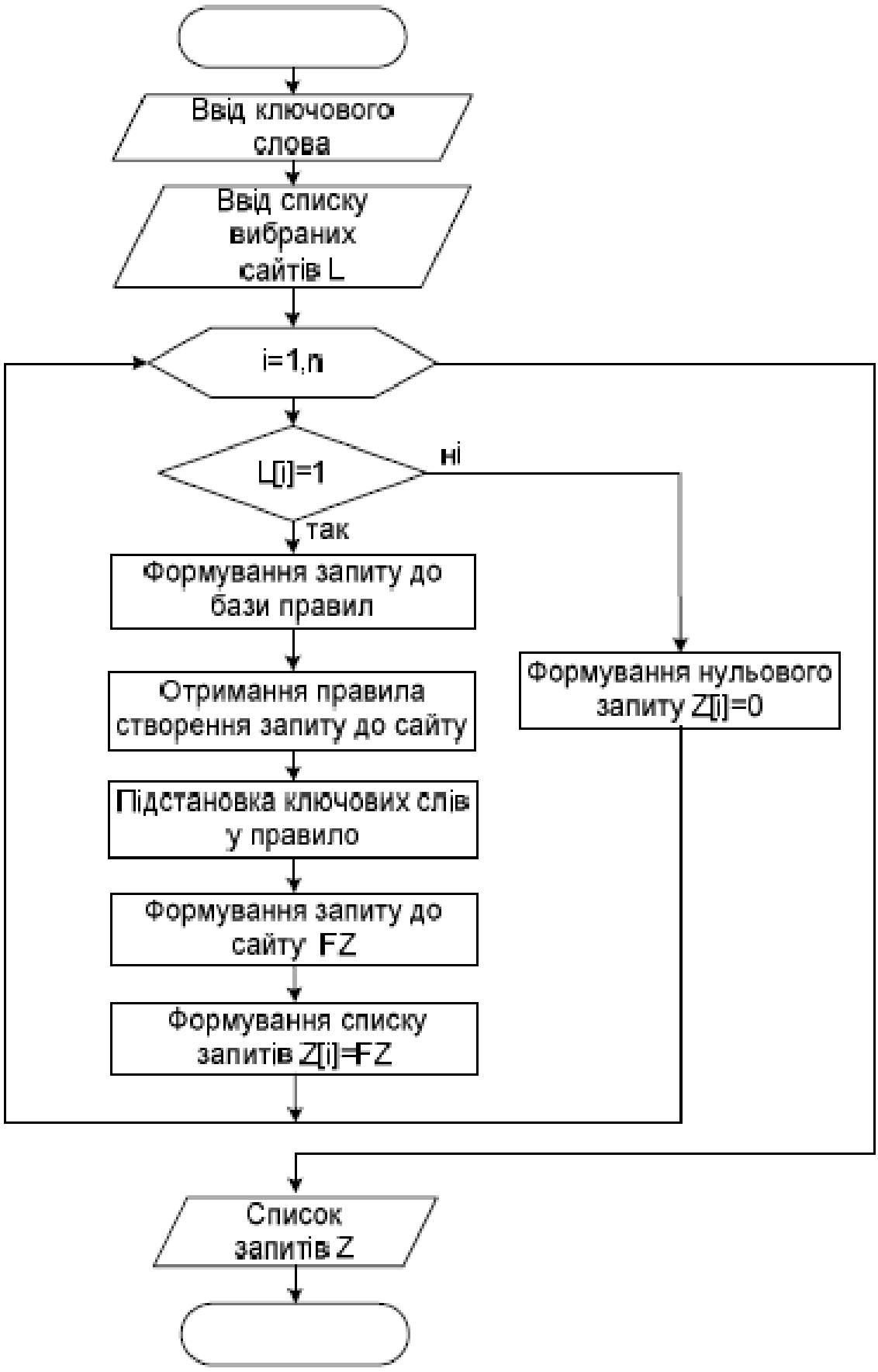

Рисунок 2.3 - Блок-схема алгоритму роботи агента формування запиту

Рисунок 2.4 - Блок-схема алгоритму роботи агента пошуку і зберігання інформації

Як видно, всі ці компоненти тісно пов'язані один з одним і працюють у взаємодії, утворюючи чіткий, достатньо складний механізм роботи тематичної пошукової системи, що вимагає величезних витрат ресурсів.

Суть описаної системи пошуку полягає в тому, що вона не відображає користувачу посилання на сторінки, які його цікавлять, а відображає саму інформацію з цих сторінок, оскільки тематика пошуку вже відома, а це дає змогу звузити коло до лише достовірних результатів.

Така система тематичного пошуку може використовуватись для вирішення багатьох задач:

- виявлення оптимальної пропозиції купівлі/продажу з кількох Інтернет магазинів або аукціонів;
- наповнення методичного матеріалу різноманітних курсів дистанційного навчання з Інтернет енциклопедій та довідників;
- пошук бізнес-партнерів у різних галузях;
- порівняння пропозицій роботодавців з різних мережевих дошок оголошень для молодих спеціалістів та багатьох інших задач, які у своїй основі вимагають малої затрати часу, оптимальних рішень та наочності результатів.

Контрольні питання

1. Як називається процес вилучення структурованої корисної інформації з сайту?
2. Як називаються інструменти для реалізації вилучення структурованої корисної інформації з сайту?
3. Назвіть Підходи до вилучення даних.
4. Основні задачі пошукових систем.
5. Основні компоненти пошукових систем.
6. Призначення основних компонентів пошукових систем.
7. Праця пошукового сервера.
8. Основні характеристики пошукових систем.
9. Поля бази правил.

ЛЕКЦІЯ 3

на тему: Теорія стиснення інформації: визначення, основні принципи та області застосування

У лекції розглянуто поняття надлишковості та фактори, що її зумовлюють, наведені приклади стиснення даних в різних інформаційних системах, класифікація і характеристики методів стиснення, проаналізовано перспективи їх розвитку.

План:

1. Вступ до дисципліни
2. Надлишковість та фактори, що її зумовлюють
3. Стиснення даних в різних інформаційних системах
4. Класифікація методів стиснення
5. Продуктивність стиснення, основні характеристики методів
6. Перспективи розвитку методів стиснення

1. Вступ до дисципліни

Передача, зберігання і обробка інформації вимагають чималих витрат. І чим з більшою кількістю інформації нам доводиться мати справу, тим дорожче це коштує. На жаль, велика частина даних, які потрібно передавати по каналах зв'язку і зберігати, має не саме компактне представлення. Скоріше, ці дані зберігаються в формі, що забезпечує їх найбільш просте використання, наприклад: звичайні книжкові тексти, ASCII коди текстових редакторів, двійкові коди даних ЕОМ, окремі відліки сигналів в системах збору даних і т.д. Однак таке найбільш просте у використанні подання даних вимагає вдвічі - втричі, а іноді і в сотні разів більше місця для їх збереження і смугу частот для їх передачі, ніж насправді потрібно.

Наприклад, супутник, який робить знімки поверхні Землі в різних частотних діапазонах, передає інформацію на Землю, яка потім поміщається в сховище зображень, де використовується геоінформаційними системами для самих різних цілей. При цьому, обсяг цієї графічної інформації настільки великий, що при її стисненні борються буквально за кожний байт, тому що його зберігання коштує грошей. За деякими оцінками додаткове стиснення хоча б на 5% дає виграш в мільйони доларів. Приблизно та ж ситуація зберігається і при передачі зображень по каналах зв’язку. Існуючої пропускної здатності не вистачає, щоб в повній мірі задовольнити потреби користувачів. Тому стиснення даних - це один з найбільш актуальних напрямків сучасної комп'ютерної інженерії.

Стиснення даних – це процедура представлення даних в більш компактній формі за рахунок усунення надлишковості з метою економії ресурсів системи передачі і зберігання інформації.

Стиснення в даний час застосовуються практично у будь-якій більш-менш складній системі, що оперує із значними обсягами даних – достатньо навести приклади архівування файлів, форматів аудіо- та відеоданих, які використовують стиснення (MP3, JPEG, GIF, PNG, AVI, MPEG і т. д.), а також цифрового зв'язку, зокрема мобільного.

Стиснення даних можливе лише в тому випадку, коли вони містять надлишковість. Практично будь-які осмислені дані, що не були попередньо стиснуті, містять надлишковість, тобто можуть бути стиснені. Не можуть бути стиснутими лише дані, що носять абсолютно випадковий характер, з однаковою ймовірністю появи всіх можливих символів.

Найбільший ступінь надлишковості – відповідно найбільший ступінь стиснення та найбільший ефект від використання цієї процедури – характерний для зображень в растровому форматі, та, дещо меншою мірою, для оцифрованого звуку. Використання растрового формату (тобто коли зберігається колір кожної точки зображення) для відеофайлів призвело б до того, що середньостатистичний фільм заледве вмістився б на середньостатистичний вінчестер. Використання кодування форми сигналу без стиснення (тобто коли зберігається оцифроване значення амплітуди звуку в кожен момент часу) призводить до того, що на компакт-диск можливо записати максимум 80 хвилин музики, в той час як при використанні формату MP3 – 5-10 годин чи більше (в залежності від ступеня стиснення). В мобільному зв'язку стандарту GSM швидкість передачі даних 13 кбіт/сек (із досить складним алгоритмом стиснення, який до того ж працює в реальному часі) забезпечує таку ж суб'єктивну якість звуку, як при передачі на швидкості 64 кбіт/сек без стиснення.

Дані вимірювань в системах автоматизованого контролю та управління (оцифровані сигнали аналогових датчиків) також містять надлишковість, зумовлену подібністю (близькістю) сусідніх відліків сигналу. Смуга частот, ширина якої загалом і визначає необхідний обсяг даних за одиницю часу, залежить від характеру фізичного процесу, а необхідна точність представлення визначається в першу чергу похибкою датчика, а не суб'єктивними вимогами до якості сигналу.

Значною надлишковістю володіють повідомлення на природній мові. Навіть якщо не брати до уваги, смислову надлишковість (технічна реалізація усунення якої можлива хіба що за умови застосування апарату штучного інтелекту), статистична надлишковість, обумовлена різною ймовірністю появи символів у тексті в залежності від контексту, складає близько 60%, тобто можна сказати, що на кожних 10 символів припадає 6 надлишкових. Так, якщо, наприклад, викинути з тексту всі голосні літери, то в абсолютній більшості випадків цей текст можна безпомилково відновити. Такий «метод стиснення даних» успішно застосовувався у ряді стародавніх складових та словесно-складових писемностей (фінікійська, палестинська, арамейська, шумерська, давньоєгипетська), де кожен символ писемності позначав склад «приголосна + довільна голосна».

В давньоєврейській, арабській та санскритській писемностях для позначення голосних використовувались діакритичні значки (невеликі штрихи, гачки і крапки над основними літерами, що позначали приголосні). Це можна вважати першим випадком застосування принципу кодування високоймовірних символів більш короткими кодами порівняно із низько ймовірними. В повній мірі цей принцип вперше було розвинуто англійським вченим-винахідником Семюелом Морзе, який не тільки створив перший діючий телеграфний апарат, а і успішно вирішив задачу, що з'явилася одразу із появою цієї системи зв'язку обмеженої пропускної здатності – задачу ефективного кодування (інший термін для стиснення даних). В абетці Морзе літери кодуються нерівномірним кодом, тобто довжина кодового слова тим більша, чим менша ймовірність появи символа.

Коди, побудовані за таким принципом, не усувають інший вид надлишковості, пов'язаний із статистичними зв'язками між близько розташованими символами. Методи стиснення, що враховують контекст, вперше були створені у 7-х рр. XX ст. Лемпелем і Зівом (так звані словникові методи) і по сьогоднішній день застосовуються у найбільш популярних програмах-архіваторах (WinRar, WinZip). Пізніше були створені методи контекстного моделювання та сортуючих перетворень, які вимагають значно більше обчислювальних ресурсів, ніж словникові, однак забезпечують для текстів на природній мові ступінь стиснення, найбільш близький до максимально можливого.

2. Надлишковість та фактори, що її зумовлюють

Всім, хто використовує комп'ютерні програми стиснення інформації, добре знайомі такі слова, як «zip», «implode», «stuffit», «diet» і «squeeze». Все це імена програм або назви методів для компресії комп'ютерної інформації. Переклад цих слів в тій чи іншій мірі означає застібання, ущільнення, набивання або стиснення. Однак звичайний, мовний сенс цих слів або їх переклад не в повній мірі відображають справжню природу того, що відбувається з інформацією в результаті компресії. Насправді, при компресії комп'ютерної інформації нічого не набивається і не скорочується, а лише видаляється деякий надлишок інформації, присутній у вхідних даних. Надмірність - ось центральне поняття в теорії стиснення інформації. Будь-які дані з надлишковою інформацією можна стиснути. Дані, в яких немає надмірності, стиснути не можна.

Ми всі добре знаємо, що таке інформація. Інтуїтивно нам це цілком зрозуміло, однак це скоріше якісне сприйняття предмета. Інформація представляється нам однією з сутностей, які неможливо точно визначити і тим більше виміряти кількісно. Однак, існує область математики, яка називається теорією інформації, де інформацію вивчають саме кількісно.

Під **ентропією** в теорії інформації розуміють міру невизначеності (наприклад міру невизначеності стану деякого об'єкта). Для того, щоб зняти цю невизначеність, необхідно повідомити деяку кількість інформації.

При цьому **ентропія** дорівнює мінімальній кількості інформації, яку необхідно повідомити для повного зняття невизначеності. Ентропія також може бути використана для оцінки найкращого можливого ступеня стиснення для деякого потоку подій.

Способи оцінки кількості інформації:

- комбінаторний
- ймовірнісний
- алгоритмічний.

Найбільш простим способом оцінки кількості інформації є ***комбінаторний*** підхід. Згідно з ним, якщо змінна *x* належить до множини з *N* елементів, то ентропія

$$H(x)=\log_2 N$$

Таким чином для передачі стану об`єкта достатньо $I=\log_2 N$ біт інформації. Основним недоліком комбінаторного підходу є його орієнтованість на системи з рівноймовірнісними станами. В реальному світі події, як правило, нерівноймовірнісні.

Ймовірнісний підхід до оцінки кількості інформації, що враховує цей фактор, є найбільш широко застосовним на сьогодні. Нехай змінна *x* може приймати *N* значень x_i з ймовірнісю $p(x_i)$. Тоді ентропія

$$H_w(x) = -\sum_{i=1}^{N} p(x_i)\log_2(p(x_i))$$

Наведені формули показують, що незалежно від того, як були отримані ймовірності настання наступних подій, для кодування подій з ймовірністю *p* достатньо - $\log_2 p$ біт (у повній відповідності з теоремою Шеннона про оптимальне кодування).

Науковою передумовою можливості стиснення даних виступає відома з теорії інформації теорема кодування для каналу без перешкод, опублікована наприкінці 40-х років у статті Клода Шеннона "Математична теорія зв'язку". Теорема підтверджує, що в каналі зв'язку без перешкод можна так перетворити послідовність символів джерела у послідовність символів коду, що середня довжина символів коду може бути як завгодно близька до ентропії джерела повідомлень *H(X)*, обумовленої як:

$$H(X) = -\sum_{i=1}^{N} p(x_i) \times \log_2 p(x_i)$$

де $p(x_i)$ – можливість появи конкретного повідомлення x_i із *N* можливих символів алфавіту джерела.

Число *N* називають об'ємом алфавіту джерела.

Алгоритмічний підхід застосовується в тих випадках, коли дані володіють певними закономірностями. Згідно цього підходу, якщо дані можна описати деякими формулами або породжуючими алгоритмами, ентропія буде чисельно дорівнювати мінімальній кількості інформації, необхідної для передачі цих формул або алгоритмів від джерела інформації до приймача.

Алгоритмічний підхід використовується самостійно або сумісно з ймовірносним, наприклад в деяких алгоритмах стиснення графічної інформації.

Іншим важливим досягненням теорії інформації є цілком суворе визначення надлишковості. Зараз ми спробуємо витлумачити це поняття інтуїтивно, вказавши, що є надмірність для двох простих типів комп'ютерних даних, і що представляють собою дані, з яких видалена надлишковість.

Перший тип інформації - це текст. Текст являє собою найважливіший вид комп'ютерних даних. Величезна кількість комп'ютерних програм і додатків є за своєю природою нечисловими; вони працюють з даними, у яких основними елементарними компонентами служать символи тексту. Комп'ютер здатний зберігати і обробляти лише двійкову інформацію, що складається з нулів і одиниць. Тому кожному символу тексту необхідно зіставити двійковий код. Сучасні комп'ютери використовують так звані коди ASCII (вимовляється «аски», а саме слово ASCII є скороченням від «American Standard Code for Information Interchange»), хоча все більше комп'ютерів і додатків використовують нові коди Unicode. ASCII представляє код фіксованої довжини, де кожному символу присвоюється 8-бітова послідовність (сам код займає сім бітів, а восьмий – для перевірки, який спочатку був задуманий для підвищення надійності коду). Код фіксованої довжини представляється найкращим вибором, оскільки дозволяє комп'ютерним програмам легко оперувати з символами різних текстів. З іншого боку, код фіксованої довжини є по суті надлишковим.

У випадковому текстовому файлі ми очікуємо, що кожен символ зустрічається приблизно рівне число раз. Однак файли, які використовуються на практиці, навряд чи є випадковими. Вони містять осмислені тексти, і з досвіду відомо, що, наприклад, в типовому українському тексті деякі букви, такі, як «О», «К» і «А», зустрічаються набагато частіше, ніж «Ч» і «Ш». Це пояснює, чому код ASCII є надлишковим, а також вказує на шляхи усунення надлишковості. ASCII надлишковий перш за все тому, що присвоює кожному символу, незалежно від того часто чи рідко він використовується, одне і те ж число біт (вісім). Щоб видалити таку надлишковість, можна скористатися кодами змінної довжини, в якому короткі коди присвоюються буквам, які трапляється частіше, а рідкісним буквах дістаються більш довгі коди. Точно так працює кодування Хаффмана.

Випадкові дані неможливо стиснути, так як в них немає надлишковості.

Другий тип комп'ютерних даних - це оцифровані зображення: фотографії, малюнки, картинки, графіки і т.п. Цифрове зображення - це прямокутна матриця забарвлених точок, званих пікселями. Кожен піксель представляється в комп'ютері за допомогою колірного коду. Для спрощення цифрової обробки

зображень передбачається, що всі пікселі мають один і той же розмір. Розмір пікселя залежить від числа кольорів в зображенні, яке, зазвичай, є ступенем 2. Якщо в ньому міститься 2^k різних кольорів, то кожен піксель - це k-бітове число.

Є два види надлишковості в цифрових зображеннях. Перший вид схожий на надлишковість в текстовому файлі. У кожному невипадковому зображенні деякі кольори можуть переважати, а інші зустрічатися рідко. Така надлишковість може бути вилучена за допомогою кодів змінної довжини, що присвоюються різним пікселам, точно так як і при стисненні текстових файлів. Інший вид надлишковості набагато важливіший, він є результатом кореляції пікселів. Коли наш погляд переміщається по картинці, він виявляє в більшості випадків, що сусідні пікселі пофарбовані в близькі кольор. Уявімо собі фотографію, на якій зображено блакитне небо, білі хмари, коричневі гори і зелені дерева. Поки ми дивимося на гори, близькі пікселі мають схожий колір; всі або майже всі з них мають різні відтінки коричневого кольору. Про близькі пікселі неба можна сказати, що вони носять різні відтінки блакитного кольору. І тільки на обрії, там, де гори зустрічаються з небом, сусідні пікселі можуть мати абсолютно різні кольори. Таким чином, окремі пікселі не є абсолютно незалежними. Можна сказати, що найближчі пікселі зображення корелюють між собою. З цим видом надлишковості можна боротися різними способами.

Незалежно від методу, яким стискається зображення, ефективність його компресії визначається насамперед кількістю надлишковості, що містяться в ньому. Граничний випадок - це однотонне зображення. Воно має максимальну надлишковість, тому що сусідні пікселі тотожні. Зрозуміло, таке зображення не цікаве з практичної точки зору, воно, якщо зустрічається, то вкрай рідко. Тим не менше, воно буде дуже добре стискатися будь-яким методом компресії. Іншим екстремальним випадком є зображення з некорельованими, тобто, випадковими пікселями. У такому зображенні сусідні пікселі, як правило, дуже різняться за своїм кольором, а надмірність цього зображення дорівнює нулю. Його неможливо стиснути ніяким методом. Воно виглядає як випадкова мішанина забарвлених точок, і тому не є цікавим. Нам навряд чи знадобиться зберігати і обробляти подібні зображення, і немає сенсу намагатися їх стискати.

Не існує методів і алгоритмів, здатних ефективно стискати будь-які файли, або навіть істотну частину їх. Для того, щоб стиснути файл, алгоритм компресії повинен спочатку вивчити його, знайти в ньому надлишковість, а потім спробувати видалити її. Оскільки надлишковість залежить від типу даних (текст, графіка, звук і т.д.), методи компресії повинні розроблятися з урахуванням цього типу. Алгоритм буде найкраще працювати саме зі своїми даними. У цій області не існує універсальних методів і рішень.

3. Стиснення даних в різних інформаційних системах

Мета стиснення даних - забезпечити компактне представлення даних, що виробляються джерелом, для їх більш економного збереження і передачі по каналах зв'язку.

Умовна структура системи стиснення даних виглядає наступним чином:

Дані джерела - Кодер - Стислі дані - Декодер - Відновлені дані

У цій схемі дані, що виробляються джерелом визначимо як дані джерела, а їх компактне уявлення - як стислі дані. Система стиснення даних складається з кодера і декодера джерела. Кодер перетворює дані джерела в стислі дані, а декодер призначений для відновлення даних джерела зі стислих даних. Відновлені дані, що виробляються декодером, можуть або абсолютно точно збігатися з вихідними даними джерела, або незначно відрізнятися від них.

Компресор або кодер - програма, яка стискає вхідний файл і створює на виході файл із стисненими даними, в яких мало надлишковості. Декомпресор або декодер працює в зворотному напрямку. Відзначимо, що поняття кодера є вельми загальними і має багато значень, але оскільки ми обговорюємо стиснення даних, то у нас слово кодер буде означати компресор. Термін кодек іноді використовується для об'єднання кодера і декодера.

3.1 Стиснення даних в системах передачі інформації

Особливо важливу роль відіграє стиснення при передачі даних. В загальній структурній схемі передачі даних стиснення даних входить у задачі кодера джерела. Розглянемо детальніше місце процедури стиснення серед інших перетворень інформації в передавачі та їх взаємозв'язок.

В широкому розумінні стиснення даних може відбуватись вже на стадії аналого-цифрового перетворення – за рахунок вибору параметрів перетворення (частота дискретизації, інтервал квантування або характеристика квантувача) таким чином, щоб зменшити кількість інформації до мінімально необхідної для забезпечення заданого критерію якості.

Стиснення інформації може підвищувати стійкість систем шифрування (криптосистем), оскільки внаслідок зменшення надлишковості інформація після стиснення носить псевдовипадковий характер. Хоча хороші криптоалгоритми також дають на виході псевдовипадкову інформацію (тому шифрування і виконують після стиснення, а не навпаки), однак тим «випадковішими» є характеристики даних на вході криптосистеми, тим менше додаткової інформації надається криптоаналітику для зламування шифру. Наприклад, якщо використовується простий підстановочний шифр (кожна буква заміняється іншою буквою або символом) і кодується текст на природній мові, то криптоаналітику достатньо підрахувати ймовірності появи окремих символів для даної мови (приклад такого криптоаналізу приведено в оповіданні Конан Дойля «Танцюючі чоловічки»). Якщо ж дані попередньо стиснути, то ймовірності появи символів будуть приблизно однаковими, і дешифрування можна буде здійснити тільки шляхом прямого перебору всіх можливих

варіантів, причому тільки тоді, коли відомий алгоритм стиснення. З останнього випливає, що стиснення може навіть відігравати роль шифрування – за умови, що алгоритм стиснення гарантовано не стане відомий криптоаналітику, або ж має достатньо велику кількість параметрів, сукупність яких може використовуватись в якості ключа шифру.

Кодер джерела і кодер каналу з точки зору зміни об'єму інформації виконують протилежні функції – кодер джерела усуває надлишковість (для зменшення об'єму даних), а кодер каналу, навпаки, вводить надлишковість (для забезпечення завадозахищеної передачі). Необхідність двох окремих процедур пояснюється тим, що надлишковість, яка вноситься кодером каналу, є мінімально необхідною для гарантованої передач даних із заданим рівнем достовірності з врахуванням характеристик каналу зв'язку (рівень завад, ймовірність помилки); надлишковість же джерела інформації може бути «надмірною» або «недостатньою» для конкретного каналу зв'язку.

На етапі реалізації множинного доступу стиснення даних можна досягти в тому випадку, якщо дані різних користувачів є корельованими (наприклад, інформація з різних об'єктів, що підлягають впливу схожих зовнішніх факторів – температура в просторово близьких точках та ін., чи є структурованою за однаковим алгоритмом) або якщо сукупність даних всіх користувачів має нерівномірний характер розподілу символів.

3.2 Стиснення даних в інформаційно-вимірювальних та автоматизованих управляючих системах

В системах цифрового зв'язку та при зберіганні великих масивів інформації переваги використання стиснення даних очевидні. Для систем же вимірювання та управління доцільність застосування стиснення визначається виходячи із особливостей структури системи в цілому. Наприклад, при наявності інтерфейсу з достатньою пропускною здатністю та (або) незначних об'ємів даних, стиснення може бути не обов'язковим, особливо якщо його застосування вимагає значних додаткових обчислювальних ресурсів та збільшує витрати на розробку системи.

Основна задача стиснення – зменшення надлишковості – може бути частково вирішена вже на етапі проектування системи шляхом належного вибору:

- ✓ виду і кількості параметрів, що контролюються (на підставі системного аналізу проблеми);
- ✓ частоти знімання параметрів датчиків (виходячи із припущень про ширину частотної смуги вимірюваних сигналів);
- ✓ точності вимірювань, обрахунків та представлення результатів (з врахуванням точності датчиків, розрядності та швидкодії обчислювальних засобів, вимог до точності при подальшій обробці).

Наприклад, якщо необхідно вимірювати температуру в кількох близько розташованих точках, результати вимірювань виявлятимуть сильну кореляцію. В цьому випадку можливі наступні способи скорочення об'єму даних:

1) кодувати результати сумісно, тобто, наприклад, абсолютні значення зберігати лише для однієї опорної точки, а для всіх інших – різницю відносно значення в опорній точці; розрядність різниць може бути значно меншою, ніж абсолютних значень, за рахунок чого і досягається стиснення;
2) проаналізувавши на основі експериментальних даних кореляційні зв'язки між температурою в усіх точках та врахувавши інші можливі варіанти розподілу температурного поля, скоротити кількість точок до мінімально необхідної, що забезпечує задану достовірність контролю температури об'єкта.

Надалі розглядатимемо переважно методи власне стиснення (перший варіант) безвідносно до питань системотехнічного характеру (другий варіант), тобто методи задача яких – зменшення надлишковості інформації, а не надлишковості структури інформаційної системи.

В інформаційно-вимірювальних та управляючих системах джерело інформації в більшості випадків є аналоговим. Тому важливого значення набуває правильний вибір кроку дискретизації і квантування, що зменшує обсяг інформації до мінімально необхідного для опису процесу із заданою точністю. У вищенаведеному прикладі, оскільки температура – повільно-змінний параметр, доцільно прийняти частоту дискретизації порівняно низькою (порядку кількох Герц), або одразу усереднювати дані, вимірювані з високою частотою дискретизації, протягом деякого інтервалу часу, а вже потім записувати усереднене значення в запам'ятовуючий пристрій чи передавати по каналах зв'язку. Однак в загальному випадку вирішити цю задачу на етапі схемотехнічного проектування достатньо ефективно можна переважно тільки для стаціонарних процесів. Якщо ж дані на виході аналогового джерела носять нестаціонарний характер, тобто їх статистичні характеристики змінюються в часі, більш доцільним може бути застосування адаптивних методів дискретизації і квантування, що забезпечують зміну параметрів аналого-цифрового перетворення в залежності від статистичних характеристик процесу і в певному розумінні можуть бути віднесені до методів стиснення.

4. Класифікація методів стиснення

Деякі методи стиснення допускають втрати. Вони краще працюють, якщо частина інформації буде втрачена. Коли такий декодер відновлює дані, результат може не бути тотожним вихідному файлу. Такі методи дозволені, коли стискаються графіка, звук або відео. Якщо втрати невеликі, то буває неможливо виявити різницю. А текстові дані, наприклад, текст комп'ютерної програми, можуть стати зовсім непридатними, якщо загубиться чи зміниться хоч один біт інформації. Такі файли можна стискати тільки методами без втрати інформації.

Розглянемо **класифікацію** систем стиснення за різними критеріями.

1. Існують два типи систем стиснення даних:

- ✓ системи стиснення без втрат інформації (неруйнуюче стиснення);
- ✓ системи стиснення з втратами інформації (руйнуюче стиснення).

Стиснення без втрат інформації

У системах стиснення без втрат декодер відновлює дані джерела абсолютно точно, таким чином, структура системи стиснення виглядає наступним чином:

Вектор даних X - Кодер - B (X) - Декодер - X

При стисненні без втрат дані після розпакування посимвольно співпадають із первинними даними, що підлягали стисненню. Ці методи застосовуються насамперед для стиснення довільних цифрових даних в програмах-архіваторах, а також у деяких графічних форматах, при передачі зображень по факсу і т.п.

Стиснення з втратою інформації

В системі стиснення з втратами (або з руйнуванням) кодування проводиться таким чином, що декодер не в змозі відновити дані джерела в первісному вигляді. Структурна схема системи стиснення з руйнуванням виглядає наступним чином:

X - Квантувач - X_q - Неруйнівний кодер - B (X_q) - Декодер - X*

Як і в попередній схемі, $X = (x_1, x_2, ... x_n)$ - вектор даних, що підлягають стисненню. Відновлений вектор позначимо як $X^* = (x_1, x_2, ... x_n)$. Відзначимо наявність в цій схемі стиснення елемента, який був відсутній при неруйнівному стисненні – квантувача.

Квантувач стосовно вектору вхідних даних X формує вектор X_q, досить близький до X в сенсі середньоквадратичної відстані. Робота квантувача заснована на зниженні розміру алфавіту (найпростіший квантувач виконує округлення даних до найближчого цілого числа).

Методи стиснення з втратами використовуються переважно для звуку і зображень, тобто інформації, що сприймається органами чуття. При цьому розпакована інформація відтворює вихідну із спотвореннями, які або не відчуваються взагалі, оскільки лежать нижче порогу сприйняття, або можуть бути більш чи менш помітними в залежності від ступеня стиснення. При цьому максимально можливий ступінь стиснення – величина, суб'єктивно обумовлена вимогами отримувач інформації. Наприклад, для звукових файлів формату MP3 деяким не особливо вибагливим слухачам важко помітити різницю між бітрейтами 256 і 128 кбіт/с, хоча розмір файлу (відповідно і ступінь стиснення) може відрізнятись в 1,5-2 рази. Формально можна обмежити допустиме стиснення умовою виконання деякого критерію якості, наприклад, щоб ступінь спотворення не перевищувала деякої заданої величини.

Отже, вибір системи неруйнівного або руйнівного стиснення залежить від типу даних, що підлягають стисненню. При стисненні текстових даних, комп'ютерних програм, документів, креслень і т.п. цілком очевидно, що потрібно застосовувати неруйнівні методи, оскільки необхідно абсолютно точне відновлення вихідної інформації після її стиснення. При стисненні мови, музичних даних і зображень, навпаки, частіше використовується руйнівне

стиснення, оскільки при практично непомітних спотвореннях воно забезпечує, як правило, істотно вищі коефіцієнти стиснення , ніж неруйнуюче.

2. За ступенем налаштування розрізняють методи:

- ✓ адаптивного стиснення
- ✓ напівадаптивного стиснення
- ✓ локально-адаптивного стиснення

Метод адаптивного стиснення має на увазі здатність алгоритму міняти свої операції, параметри і настройки в залежності від стискаємих даних. Адаптивні методи спочатку тестують «сирі» вихідні дані, а потім підлаштовують свої параметри і / або операції відповідно до результату перевірки. Прикладом такого алгоритму може служити кодування Хаффмана. Деякі методи компресії використовують двопрохідні алгоритми, коли на першому проході по файлу збирається деяка статистика стискаються даних, а на другому проході відбувається безпосередньо стиснення з використанням параметрів, обчислених на першій стадії. Такі методи можна назвати напівадаптивними. Методи компресії можуть бути також локально-адаптивними, що означає здатність алгоритму налаштовувати свої параметри виходячи з локальних особливостей файлу і змінювати їх, переміщаючись від області до області вхідних даних.

3. За алгоритмами, що використовують кодер і декодер розрізняють методи:

- ✓ симетричного стиснення
- ✓ асиметричного стиснення

Симетричне стиснення - це коли і кодер, і декодер використовують один і той же базовий алгоритм, але використовують його в протилежних напрямках. Такий метод є доцільним, якщо постійно стискається і розкодовується одне і те ж число файлів. При асиметричному методі або компресор або декомпресор повинні виконати істотно більшу роботу. Таким методам теж знаходиться місце під сонцем, вони зовсім не погані. Метод, коли компресія робиться довго і ретельно за допомогою складного алгоритму, а декомпресія робиться швидко і просто, цілком виправданий при роботі з архівами, коли доводиться часто добувати дані. Те ж відбувається і при створенні і прослуховуванні аудіофайлів формату MP3. Зворотний випадок, коли зовнішні файли часто змінюються і зберігаються як резервні копії. Оскільки ймовірність вилучення резервних даних невелика, то декодер може працювати значно повільніше свого кодера.

5. Продуктивність стиснення, основні характеристики методів

При виборі методів стиснення даних для застосування в конкретній системі необхідно зважати на наступні характеристики:

- ✓ коефіцієнт стисненя;
- ✓ ступінь стиснення (фактор стиснення);
- ✓ швидкість упаковки та розпаковки;
- ✓ необхідний об’єм пам’яті (при упаковці та розпаковці);

- ✓ складність реалізації;
- ✓ тип даних, для яких стиснення є максимальним;
- ✓ можливість роботи в потоці – для систем передачі;
- ✓ ступінь спотворення – для стискування з втратами.

Коефіцієнт стиснення визначається за формулою

$$К_с=Р_{вих}/Р_{вх}$$

Коефіцієнт 0.6 означає, що стиснуті дані займають 60% від початкового розміру. Значення більше 1 говорять про те, що вихідний файл більше вхідного (негативне стиснення). Коефіцієнт стиснення прийнято вимірювати в bpb (bit per bit, біт на біт), так як він показує, скільки в середньому знадобиться біт стисненого файлу для представлення одного біта файлу на вході. При стисненні графічних зображень аналогічно визначається величина bpp (bit per pixel, біт на піксель). У сучасних ефективних алгоритмах стиснення текстової інформації має сенс говорити про схожу величиною bpc (bit per character, біт на символ), тобто, скільки в середньому потрібно біт для зберігання однієї літери тексту.

Слід згадати ще два поняття, пов'язані з коефіцієнтом стиснення. Термін бітова швидкість (bitrate) є більш загальним, ніж bpb і bpc. Метою компресії інформації є подання даних з найменшою бітовою швидкістю. Бітовий бюджет (bit budget) означає певний довісок до кожного біту в стислому файлі. Уявіть собі стислий файл, в якому 90% розміру займають коди змінної довжини, що відповідають конкретним символам вихідного файлу, а 10%, що залишилися, використовуються для зберігання деяких таблиць, які будуть використовуватися декодером при декомпресії. В цьому випадку бітовий бюджет дорівнює 10%.

Величина, зворотня коефіцієнту стиснення, називається ступенем або фактором стиснення:

$$С_с=Ф_с=Р_{вх}/Р_{вих}$$

У цьому випадку значення більше 1 означають стиснення, а менше 1 - розширення. Цей показник для більшості людей є більш природним: чим більше ступінь (фактор) стиснення, тим краще компресія.

В залежності від області застосування можуть висуватися різні вимоги до таких характеристик, як швидкодія та необхідний об’єм памяті при упаковці та розпаковці. Наприклад, для форматів зберігання графічних чи звукових файлів ці характеристики бажано мати мінімальними при розпаковці (для реалізації швидкого перегляду та відтворення), а при упаковці вони можуть бути і значними, якщо більше значення має якість стиснення, а не його швидкість. При застосуванні в системах передачі параметри швидкодії є критичними насамперед при стисненні (оскільки якщо швидкість генерування інформації джерелом буде вищою, ніж швидкість на виході ефективного кодера, інформація буде безповоротно втрачена), а якщо отримувачем інформації на

стороні приймача є людина (наприклад, при телефонній розмові), то і при розпакуванні.

Складність реалізації алгоритму доцільно враховувати, якщо критичними параметрами є час та вартість розробки системи (а також кваліфікація програміста), особливо якщо ступінь стиснення не набагато вищиц ніж у методу, простішого в реалізації. Інколи доцільно розглянути і варіант використання існуючих програм чич бібліотек функцій архівування, якщо логіка побудови системи допускає таку можливість. Завжди існує і вказується, таклж обов`язково враховується тип даних, для яких стиснення даним алгоритмом є максимальним.

Для систем передачі важливою є також можливість ефективного кодера видавати інформацію на вихід до закінчення обробки всього повідомлення: для дискретних джерел інформації – після надходження кожного наступного символу, для аналогових – після оцифровування кожного наступного значення або після обробки порівняно невеликого блоку інформації (тривалість якого є меншою, ніж допустима затримка передачі).

Ступінь спотворення даних при стисненні з втратами може бути оцінений, наприклад, як середньоквадратичне відхилення між вихідними даними X та даними, отриманими після розпаковки X^*:

$$D = \frac{1}{N}\sum_{i=1}^{N}(X_i - X_i^*)^2 .$$

6. Перспективи розвитку методів стиснення

Як відомо, основоположником науки про стиснення інформації прийнято вважати Клода Шеннона. Його теорема про оптимальне кодування показує, до чого потрібно прагнути при кодуванні інформації та на скільки та чи інша інформація при цьому стиснеться. Крім того, ним були проведені досліди за емпіричною оцінкою надмірності англійського тексту. Він пропонував людям вгадувати наступну букву і оцінював вірогідність правильного вгадування. На основі ряду дослідів він прийшов до висновку, що кількість інформації в англійському тексті коливається в межах 0.6 - 1.3 біта на символ. Незважаючи на те, що результати досліджень Шеннона були по-справжньому затребувані лише десятиліття опісля, важко переоцінити їх значення.

Перші алгоритми стиснення були примітивними в зв'язку з тим, що була примітивною обчислювальна техніка. З розвитком потужностей комп'ютерів стали можливими все більш потужні алгоритми. Справжнім проривом був винахід Лемпелем і Зівом в 1977 р словникових алгоритмів. До цього моменту стиснення зводилося до примітивного кодування символів. Словникові алгоритми дозволяли кодувати повторювані рядки символів, що дозволило різко підвищити ступінь стиснення. Важливу роль зіграло винахід приблизно в цей же час арифметичного кодування, що дозволив втілити в життя ідею Шеннона про оптимальне кодування. Наступним проривом був винахід в 1984

р алгоритму PPM. Слід зазначити, що цей винахід довго залишався непоміченим. Справа в тому, що алгоритм складний і вимагає великих ресурсів, в першу чергу великих обсягів пам'яті, що було серйозною проблемою в той час. Винайдений в тому ж 1984 р алгоритм LZW був надзвичайно популярний завдяки своїй простоті, хорошій рекламі і невимогливості до ресурсів, не дивлячись на відносно низький ступінь стиснення. На сьогоднішній день алгоритм PPM є найкращим алгоритмом для стиснення текстової інформації, а LZW давно вже не вбудовується в нові додатки (проте широко використовується в старих).

Майбутнє алгоритмів стиснення тісно пов'язане з майбутнім комп`ютерних технологій. Сучасні алгоритми вже впритул наблизились до Шенноновської оцінки 1.3 біта на символ, але вчені не бачать причин, за якими комп'ютер не може передбачати краще, ніж людина. Для досягнення високих ступенів стиснення доводиться використовувати більш складні алгоритми. Однак існуюче у свій час упередження, що складні алгоритми з більш високим ступенем стиснення завжди більш повільні, не має сенсу. Так, існують вкрай швидкі реалізації алгоритмів PPM для текстової інформації, що мають дуже високий ступінь стиснення.

Таким чином, майбутнє за новими алгоритмами з високими вимогами до ресурсів і більш високим ступенем стиснення.

Застарівають не тільки алгоритми, але і типи інформації, на які вони орієнтовані. Так, на зміну графіці з малим числом кольорів і звичайному тексту прийшли високоякісні зображення і електронні документи в різних форматах. Відомі алгоритми не завжди ефективні на нових типах даних. Це робить вкрай актуальною проблему синтезу нових алгоритмів.

Кількість потрібної людині інформації неухильно зростає. Обсяги пристроїв для зберігання даних і пропускна спроможність ліній зв'язку також ростуть. Однак кількість інформації зростає швидше. У цієї проблеми є три рішення. Перше - обмеження кількості інформації. На жаль, воно не завжди прийнятно. Наприклад, для зображень це означає зменшення роздільної здатності, що призведе до втрати дрібних деталей і може зробити зображення взагалі не корисними (наприклад, для медичних або космічних зображень). Друге - збільшення обсягу носіїв інформації і пропускної здатності каналів зв'язку. Це рішення пов'язане з матеріальними витратами, причому іноді досить значними. Третє рішення - використання стиснення інформації. Це рішення дозволяє в кілька разів скоротити вимоги до обсягу пристроїв зберігання даних і пропускної здатності каналів зв'язку без додаткових витрат (за винятком витрат на реалізацію алгоритмів стиснення). Умовами його застосування є збитковість інформації і можливість установки спеціального програмного забезпечення або апаратури як поблизу джерела, так і поблизу приймача інформації. Як правило, обидві ці умови задовольняються.

Саме завдяки необхідності використання стиснення інформації методи стиснення досить широко поширені. Однак існують дві серйозні проблеми. По-перше, широко використовувані методи стиснення, як правило, застаріли і не

забезпечують в достатній мірі стиснення. У той же час вони вбудовані в велику кількість програмних продуктів і бібліотек і тому будуть використовуватися ще досить довгий час. Другою проблемою є часте застосування методів стиснення, що не відповідають характеру даних. Наприклад, для стиснення графіки широко використовується алгоритм LZW, орієнтований на стиснення одновимірної інформації, наприклад тексту. Вирішення цих проблем дозволяє різко підвищити ефективність застосування алгоритмів стиснення.

Таким чином, розробка і впровадження нових алгоритмів стиснення, а також правильне використання існуючих дозволить значно скоротити витрати на апаратне забезпечення обчислювальних систем.

Контрольні питання

1. Міра невизначеності для зняття якої необхідно повідомити певну кількість інформації.
2. Які способи оцінки кількості інформації Ви знаєте?
3. Охарактеризуйте умовну структуру системи стиснення даних.
4. Охарактеризуйте умовну структуру системи стиснення даних без втрат.
5. Охарактеризуйте умовну структуру системи стиснення даних з руйнуванням.
6. Методи стиснення, алгоритми яких не змінюють свої операції, параметри і настройки в залежності від типу даних.
7. Методи стиснення, алгоритми яких тестують початкові дані та налаштовують свої параметри і операції в залежності від типу даних.
8. Методи компресії, що використовують двохпрохідні алгоритми, які на першому проході збирають статистику, а на другому – виконують стиснення.
9. Методи стиснення, алгоритми яких здатні налаштовувати свої параметри виходячи із певних особливостей файлу і можуть динамічно змінювати її.
10. Відношення розміру стиснених даних до розміру даних джерела.
11. Що таке коефіцієнт стиснення?
12. Для даних яких типів можна використовувати методи руйнівного стиснення?
13. Охарактеризуйте стандарт ASCII та стандарт Unicode.

ЛЕКЦІЯ 4

на тему: Кодування джерел без пам'яті. Методи стиснення без втрат (частина 1)

У лекції приводяться основні відомості про нерівномірні префіксні коди та методи стиснення без втрат, розглядаються канонічний алгоритм Хаффмана, алгоритм Шеннона-Фано, їх переваги і недоліки.

План:

1. Нерівномірні префіксні коди.
2. Загальні відомості про методи стиснення без втрат.
3. Оптимальні нерівномірні префіксні коди.

1. Нерівномірні префіксні коди

Нерівномірні коди реалізують простий принцип стиснення: символи, що часто зустрічаються в повідомленні, кодуються короткими кодовими словами, а символи, що зустрічаються рідко – довгими. За рахунок цього середня довжина кодового слова зменшується, і після кодування дані займають менший об'єм.

Для практичного використання нерівномірний код повинен бути однозначно декодованим. Код є однозначно декодованим, якщо будь-яка послідовність його кодових слів допускає тільки один спосіб розбиття на кодові слова.

Достатньою умовою однозначної декодованості є префіксність. Код називається префікс ним, якщо жодне його кодове слово не є початком іншого кодового слова. Всі префіксні коди є однозначно декодованими, але не навпаки.

Будь-який код можна представити у вигляді дерева, кожне ребро якого відповідає символу алфавіта джерела, кодові слова утворюються при проходженні дерева до деякого вузла. Якщо код є префікс ним, то його кодові слова відповідають тільки листкам (кінцевим вузлам) дерева.

На рис. 4.1 наведено приклади кодів: коди а) і б) є однозначно декодованими, код в) таким не є. Код в) – не префікс ний, оскільки комбінації 0 та01 відповідають проміжним вузлам (вони є початком іншої кодової комбінації – 011). Для прикладу, послідовність 00110100 для коду в) має два варіанти декодування: 0, 01, 10, 10, 0 та 0, 011, 0, 10, 0, а для коду а) ця є послідовність однозначно розбивається на 0, 0, 11, 0, 100. Хоча код б) також не є префікс ним, однак він є однозначно декодованим, оскільки кожна кодова комбінація містить тільки один нуль, що може слугувати роздіюючим стмволом для кодових комбінацій. Наприклад, послідовність 0100011101 для коду б) однозначно розбивається на 01, 0, 0, 011, 01.

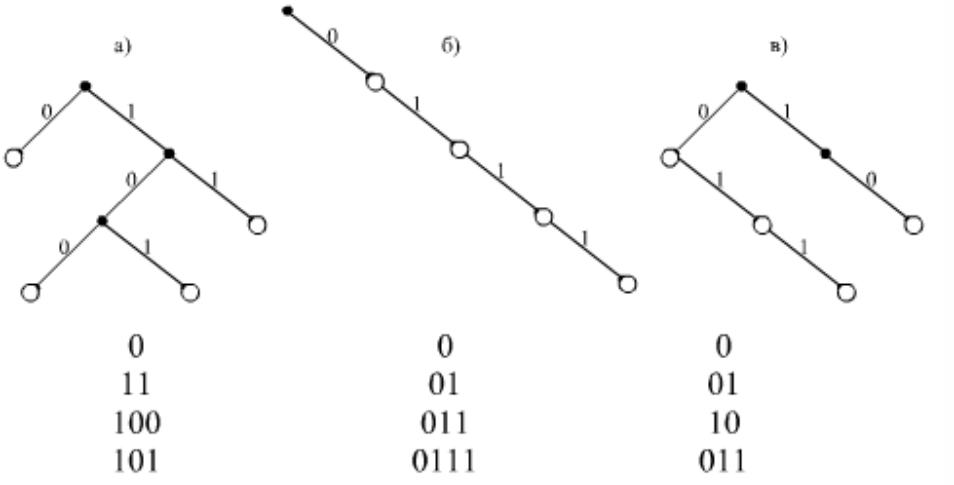

Рисунок 4.1– Приклади кодів

Довжини кодових слів λ_i у префіксному коді задовольняють нерівність Крафта:

$$\sum_{i=0}^{K-1} 2^{-\lambda_i} \le 1.$$

2. Загальні відомості про методи стиснення без втрат

У основі всіх методів стиснення лежить проста ідея: якщо представляти елементи, що часто використовуються, короткими кодами, а ті, що рідко використовуються - довгими кодами, то для зберігання блоку даних потрібен менший об'єм пам'яті, ніж якби всі елементи представлялися кодами однакової довжини. Даний факт відомий давно: пригадаємо, наприклад, азбуку Морзе, в якій символам, що часто використовуються, поставлені у відповідність короткі послідовності крапок і тире, а тим, що зустрічається рідко - довгі.

Точний зв'язок між ймовірністю і кодами встановлений в теоремі Шеннона про кодування джерела, яка свідчить, що елемент s_i, ймовірність появи якого дорівнює $p(s_i)$, вигідніше всього представляти $-\log_2 p(s_i)$ бітами. Якщо при кодуванні розмір кодів завжди в точності виходить рівним $-\log_2 p(s_i)$ бітам, то в цьому випадку довжина закодованої послідовності буде мінімальною для всіх можливих способів кодування. Якщо розподіл ймовірності $F = \{p(s_i)\}$ незмінний, і ймовірність появи елементів незалежна, то ми можемо знайти середню довжину кодів як середнє зважене

$$H = -\sum_i p(s_i) \cdot \log_2 p(s_i)$$

Це значення також називається ентропією розподілу ймовірності F або ентропією джерела в заданий момент часу.

Зазвичай ймовірність появи елементу є умовною, тобто залежить від якоїсь події. В цьому випадку при кодуванні чергового елементу s_i розподіл ймовірності F приймає одне з можливих значень F_k, тобто $F = F_k$, і, відповідно $H = H_k$. Можна сказати, що джерело знаходиться в стані k, якому відповідає набір ймовірностей $p_k(s_i)$ генерації всіх можливих елементів s_i. Тому середню довжину кодів можна обчислити за формулою

$$H = -\sum_k P_k \cdot H_k = -\sum_{k,i} P_k \cdot p_k(s_i) \log_2 p_k(s_i),$$

де p_k - ймовірність того, що F прийме k значення, або, інакше, ймовірність знаходження джерела в стані k.

Отже, якщо нам відомий розподіл ймовірності елементів, що генеруються джерелом, то ми можемо представити дані найкомпактнішим чином, при цьому середня довжина кодів може бути обчислена по формулі (1а).

Але в переважній більшості випадків істинна структура джерела нам не відома, тому необхідно будувати модель джерела, яка дозволила б нам в кожній позиції вхідної послідовності оцінити ймовірність $p(s_i)$ появи кожного елементу s_i алфавіту вхідної послідовності. В цьому випадку ми оперуємо оцінкою $q(s_i)$ ймовірності елементу s_i.

Методи стиснення можуть будувати модель джерела адаптивно у міру обробки потоку даних або використовувати фіксовану модель, створену на основі апріорних уявлень про природу типових даних, що вимагають стиснення.

Процес моделювання може бути або явним, або прихованим. Ймовірність елементів може використовуватися в методі як явним, так і неявним чином. Але завжди стиснення досягається за рахунок усунення статистичної надмірності в представленні інформації.

Жоден компресор не може стиснути будь-який файл. Після обробки будь-яким компресором розмір частини файлів зменшиться, а частини, що залишилася, - збільшиться або залишиться незмінним. Даний факт можна довести виходячи з нерівномірності кодування, тобто різної довжини кодів, що використовуються, але найбільш простий для розуміння наступний комбінаторний аргумент.

Існує 2^n різних файлів довжини n бітів, де $n = 0, 1, 2, \ldots$ Якщо розмір кожного такого файлу в результаті обробки зменшується хоча б на 1 біт, то 2^n початковим файлам відповідатиме саме більше 2^{n-1} архівних файлів. Тоді принаймні одному архівному файлу відповідатимуть декілька початкових і, отже, його декодування без втрат інформації неможливе у принципі.

Вищесказане припускає, що файл відображається в один файл, і об'єм даних вказується в самих даних. Якщо це не так, то слід враховувати не тільки сумарний розмір архівних файлів, але і об'єм інформації, необхідної для опису декількох взаємозв'язаних архівних файлів і/або розміру початкового файлу. Спільність доказу при цьому зберігається.

Тому неможливий «вічний» архіватор, який здатний нескінченне число раз стискати свої ж архіви. «Найкращим» архіватором є програма копіювання, оскільки в цьому випадку ми можемо бути завжди упевнені в тому, що об'єм даних в результаті обробки не збільшиться.

Заяви, що регулярно з'являються, про створення алгоритмів стиснення, що "забезпечують стиснення в десятки разів краще, ніж у звичайних архіваторів", є або помилковими чутками, породженими неуцтвом і гонитвою за сенсаціями, або рекламою аферистів. В області стиснення без втрат, тобто власне стиснення, такі революції неможливі. Безумовно, ступінь стиснення компресорами типових даних неухильно ростиме, але поліпшення складуть в середньому десятки або навіть одиниці відсотків, при цьому кожний подальший етап еволюції обходитиметься значно дорожче попереднього. З другого боку, у сфері стиснення з втратами, в першу чергу компресії відеоданих, все ще можливе багатократне поліпшення стиснення при збереженні суб'єктивної повноти одержуваної інформації.

3. Оптимальні нерівномірні префіксні коди

3.1 Канонічний алгоритм Хаффмана

Один з класичних алгоритмів, відомих з 60-х років. Використовує тільки частоту появи однакових байт у вхідному блоці даних. Зіставляє символам вхідного потоку, які зустрічаються частіше, ланцюжок бітів меншої довжини. І, навпаки, тим, що зустрічається рідко - ланцюжок більшої довжини. Для збору статистики вимагає двох проходів по вхідному блоку (також існують однопрохідні адаптивні варіанти алгоритму).

Спершу введемо декілька визначень.

Визначення. Нехай заданий алфавіт $\Psi = \{a_1, ..., a_r\}$, що складається з кінцевого числа букв. Кінцеву послідовність символів з ψ

$A = a_{i_1} a_{i_2} ... a_{i_n}$

називатимемо словом в алфавіті ψ, а число n - довжиною слова A. Довжина слова позначається як $l(A)$

Нехай заданий алфавіт $\Omega = \{b_1, ..., b_q\}$. Через B позначимо слово в алфавіті Ωі через $S(\Omega)$- множину всіх непорожніх слів в алфавіті Ω.

Нехай $S = S(\Psi)$- множина всіх непорожніх слів в алфавіті ψ, і S' - деяка підмножина множини S. Хай також задано відображення F, яке кожному слову A, $A \in S(\Psi)$, ставить у відповідність слово

$B = S(A)$, $B \in S(\Omega)$.

Слово Bбудемо назвати кодом повідомлення A, а перехід від слова A до його коду - кодуванням.

Визначення. Розглянемо відповідність між буквами алфавіту ψ і деякими словами алфавіту Ω:

$a_1 - B_1$
$a_2 - B_2$
...
$a_r - B_r$

Цю відповідність називають схемою і позначають через $\sum$. Воно визначає кодування таким чином: кожному слову $A = a_{i_1} a_{i_2} ... a_{i_n}$ з $S'(\Omega) = S(\Omega)$ ставиться у відповідність слово $B = B_{i_1} B_{i_2} ... B_{i_n}$, зване кодом слова A. Слова $B_1 ... B_r$ називаються елементарними кодами. Даний вид кодування називають алфавітним кодуванням.

Визначення. Нехай слово B має вигляд

$B = B' \cdot B''$

Тоді слово B' називається початком або префіксом слова B, а B'' - кінцем слова B. При цьому порожнє слово Λ і саме слово B вважаються початками і кінцями слова B.

Визначення. Схема $\sum$ володіє властивістю префікса, якщо для будь-яких i і j (($1 \le i, j \le r, i \ne j$) слово B_i не є префіксом слова B_j.

Теорема 1. Якщо схема $\sum$ володіє властивістю префікса, то алфавітне кодування буде взаємно однозначним.

Припустимо, що заданий алфавіт $\psi = \{a_1, ... a_r\}, r > 1$ і набір ймовірностей $p_1, ..., p_r (\sum_{i=1}^{r} p_i = 1)$ появи символів $a_1, ..., a_r$. Нехай, далі, заданий алфавіт Ω, $\Omega = \{b_1, ..., b_q\}, (q > 1)$. Тоді можна побудувати цілий ряд схем $\sum$ алфавітного кодування, володіючих властивістю взаємної однозначності.

$a_1 - B_1$
$a_2 - B_2$
...
$a_r - B_r$

Для кожної схеми можна ввести середню довжину l_{cp}, що визначається як математичне очікування довжини елементарного коду:

$l_{cp} = \sum_{i=1}^{r} l_i p_i, l_i = l(B_i)$ - довжини слів.

Довжина l_{cp} показує, в скільки разів збільшується середня довжина слова при кодуванні за допомогою схеми $\sum$.

Визначення. Коди, що визначаються схемою $\sum$ $l_{cp} = l_*$, називаються кодами з мінімальною надмірністю, або кодами Хаффмана.

Коди з мінімальною надмірністю дають в середньому мінімальне збільшення довжин слів при відповідному кодуванні.

У нашому випадку, алфавіт $\Psi = \{a_1, ..., a_r\}$ задає символи вхідного потоку, а алфавіт $\Omega = \{0, 1\}$ тобто складається всього з нуля і одиниці.

Алгоритм побудови схеми $\sum$ можна представити таким чином:

Крок 1. Упорядковуємо всі букви вхідного алфавіту в порядку спадання ймовірностей. Рахуємо всі відповідні слова B_i з алфавіту $\Omega = \{0, 1\}$ порожніми.

Крок 2. Об'єднуємо два символи $a_{i_{r-1}}$ і a_{i_r} з найменшими ймовірностями $p_{i_{r-1}}$ і p_{i_r} у псевдосимвол $a'\{a_{i_{r-1}} a_{i_r}\}$ з ймовірністю $p_{i_{r-1}} + p_{i_r}$. Дописуємо 0 в початок слова $B_{i_{r-1}} (B_{i_{r-1}} = 0B_{i_{r-1}})$, і 1 в початок слова $B_{i_r} (B_{i_r} = 1B_{i_r})$.

Крок 3. Видаляємо із списку впорядкованих символів $a_{i_{r-1}}$ і a_{i_r}, заносимо туди псевдосимвол $a'\{a_{i_{r-1}} a_{i_r}\}$. Проводимо крок 2, додаючи при необхідності 1 або 0 для всіх слів B_i, відповідних псевдосимволам, до тих пір, поки в списку не залишиться один псевдосимвол.

Приклад: Нехай у нас є 4 букви в алфавіті $\Psi = \{a_1, ..., a_4\} (r = 4)$, $p_1 = 0.5, p_2 = 0.24, p_3 = 0.15, p_4 = 0.11 \left(\sum_{i=1}^{4} p_i = 1\right)$. Процес побудови схеми представлений на рис. 4.2:

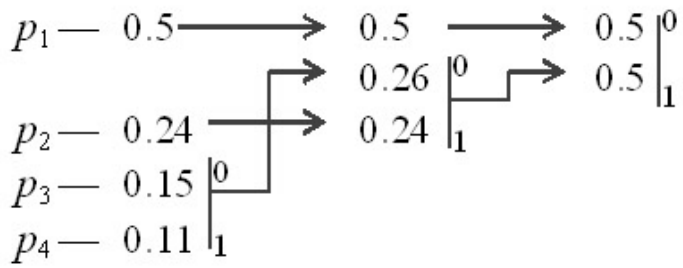

Рисунок 4.2 – Побудова схеми

Проводячи дії, відповідні 2-му кроку, ми одержуємо псевдосимвол з ймовірністю 0.26 (і приписуємо 0 і 1 відповідним словам). Повторюючи ж ці дії для зміненого списку, ми одержуємо псевдосимвол з ймовірністю 0.5. І, нарешті, на останньому етапі ми одержуємо сумарну ймовірність 1.

Результуючий код можна отримати, проходячи дерево від кореня до кожного із символів. Так, для символу з ймовірністю p_4 одержимо $B_4 = 101$, для p_3 одержимо $B_3 = 100$, для p_2 одержимо $B_2 = 11$, для p_1 одержимо $B_1 = 0$. Що відповідає схемі:

$a_1 - 0a_2 - 11a_3 - 100a_4 - 101$

Ця схема є префіксним кодом, що є кодом Хаффмана. Символ, що найчастіше зустрічається в потоці a_1 ми кодуватимемо найкоротшим словом 0, а самий рідко зустрічаємий a_4- довгим словом 101.

Для послідовності з 100 символів, в якій символ a_1 зустрінеться 50 разів, символ a_2- 24 рази, символ a_3- 15 разів, а символ a_4- 11 разів, даний код дозволить одержати послідовність з 176 бітів $(100 \cdot l_{cp} = \sum_{i=1}^{4} p_i l_i)$. Тобто в середньому ми витратимо 1.76 біти на символ потоку.

Як зрозуміло з викладеного вище, канонічний алгоритм Хаффмана вимагає вкладення у файл із стислими даними таблиці відповідності кодованих символів і кодуючих ланцюжків.

На практиці використовуються його різновиди. Так, в деяких випадках резонно або використовувати постійну таблицю, або будувати її адаптивно, тобто в процесі архівації/розархівування. Ці прийоми позбавляють нас від двох проходів по вхідному блоку і необхідності зберігання таблиці разом з файлом. Кодування з фіксованою таблицею застосовується як останній етап архівації в JPEG і в алгоритмі CCITT Group.

Характеристики канонічного алгоритму Хаффмана:

Ступень стиснення: 8, 1.5, 1 (краща, середня, гірша ступінь).

Симетричність за часом: 2:1 (за рахунок того, що вимагає двох проходів по масиву даних, що стискаються).

Характерні особливості: Один з небагатьох алгоритмів, який не збільшує розміру початкових даних у гіршому разі (якщо не враховувати необхідності зберігати таблицю перекодування разом з файлом).

3.2 Алгоритм Шеннона-Фано

Алгоритм побудови коду:

- ✓ впорядкувати символи по спаданню ймовірностей;
- ✓ розділити множину символів на дві підмножини так, щоб сума ймовірностей символів в кожній з них була приблизно однаковою;
- ✓ до результуючих кодів символів однієї підмножини дописати справа «0», до кодів іншої – «1»;
- ✓ для кожної з підмножин повторювати даний алгоритм до тих пір, поки всі підмножини не будуть містити по одному символу.

Приклад.

Нехай для джерела з алфавітом X={A,B,C,D,E} ймовірності символів складають: p(A)=0.4, p(B)=0.2, p(C)=0.15, p(D)=0.15, p(E)=0.1. Утворимо код Шеннона-Фано:

Символ	Імовірність				Результуючий код
A	0.4	0			0
B	0.2	1	0	0	100
C	0.15			1	101
D	0.15		1	0	110
E	0.1			1	111

Для коду Шеннона-Фано може існувати декілька способів розбиття символів на підмножини, тобто для одного і того ж набору ймовірностей можна отримати різні коди. Так, для розглянутого прикладу можна було утворити код і наступним чином:

Символ	Імовірність				Результуючий код
A	0.4	0	0		00
B	0.2		1		01
C	0.15	1	0		10
D	0.15		1	0	110
E	0.1			1	111

На перший погляд цей код є кращим, оскільки в ньому більша кількість коротких кодових слів. Однак в дійсності для першого випадку середня довжина кодового слова складає

$\lambda_{сер}=1*0.4+3(0.2+0.15+0.15+0.1)=2.2$ біт/символ,

а в другому

$\lambda_{сер}=2(0.4+0.2+0.15)+3(0.15+0.1)=2.25$ біт/символ,

тобто більш оптимальним є перший код.

Ентропія джерела при цьому складає

$H(X)=0.4*\log_2 0.4+0.2*\log_2 0.2+2*0.15*\log_2 0.15+0.1*\log_2 0.1=2.146$ біт/символ.

В даному випадку середня довжина кодового слова більша за ентропію, оскільки ймовірності символів не є цілими від'ємними ступенями двійки, відповідно довжини кодових слів не співпадають із значеннями власної інформації для кожного символу. Можна показати, що середня довжина кожного слова для коду Шеннона-Фано знаходиться в межах

$$H(X) \le \lambda_{сер} \le H(X)+1.$$

Характеристики алгоритму Шеннона-Фано:

Ступень стиснення: 8, 1.5, 1 (краща, середня, гірша ступінь).

Характерні особливості: Методика Шеннона-Фано не завжди призводить до однозначної побудови коду, оскільки при розбитті на частини можна зробити більшою за ймовірністю як верхню, так і нижню частини. Крім того, методика не забезпечує відшукання оптимальної множини кодових слів для кодування даної множини повідомлень.

Переваги і недоліки: Перевагою даного методу є його очевидна простота реалізації і, як наслідок цього, висока швидкість кодування / декодування. Основним недоліком є його не оптимальність в загальному випадку.

Таким чином, коди Хаффмана тп Шеннона-Фано забезпечують середню довжину кодового слова, рівну ентропії тільки тоді, коли ймовірності всіх символів є цілими ступенями двійки. В цьому випадку довжини кодових слів дорівнюють для кожного символу його власній інформації, що є цілим числом. В загальному випадку власна інформація може бути і дробовою, і значно меншою від 1, в той час як довжини кодових слів не можуть бути нецілими, що не дозволяє усунути надлишковість повністю.

Оскільки середня довжина кодового слова перевищуватиме ентропію не більше ніж на одиницю, для великих значень ентропії джерела це не вплине суттєво на ефективність стиснення, в той же час для малих значень ентропії (тобто даних з великою надлишковістю), наприклад, порядку 1 біт/символ, втрати в ефективності можуть бути суттєвими.

Коди Хаффмана та Шеннона-Фано є найбільш ефективними, якщо наперед відомі статистичні характеристики даних, що стискаються (або принаймні їх тип – наприклад, «текст українською мовою»). В цьому випадку можна використовувати заздалегідь створені кодові дерева і разом із стиснутими даними передавати тільки специфікатор типу даних (чи номер кодового дерева). Якщо ж розподіл ймовірностей символів не відомий наперед, то для кожного повідомлення слід шукати свій оптимальний код, і разом із стиснутим повідомленням необхідно передавати його кодову таблицю (кодове дерево), що може суттєво зменшити коефіцієнт стиснення, особливо для коротких повідомлень. Тому важливою є також проблема ефективної передачі кодового дерева. Найпростіший і найменш ефективний спосіб – передати послідовність пар (довжина коду, код символу) для всіх символів в алфавітному порядку. Довжини можуть кодуватися цілими змінної довжини або слід на початку повідомлення передати кількість біт, що буде використовуватись для кодування довжин.

Більш економною є передача структури кодового дерева та відповідності між його листками та символами алфавіту, оскільки при відомій структурі дерева коди всіх символів можна відновити однозначно. Структуру дерева можна передати, наприклад, наступним чином: починаючи від кореня дерева і проходячи його поярусно, кожна проміжна вершина кодується бітом 0, кінцева вершина – бітом 1. При декодуванні цієї послідовності розбиття дерева на яруси (і розбиття послідовності на фрагменти) відбувається однозначно, оскільки кількість вузлів на кожному ярусі дорівнює кількості проміжних вузлів на попередньому ярусі, помножений на два (тобто, наприклад, прийнявши перший нуль, ми дізнаємося, що на другому ярусі два вузли, і наступні два біти кодують другий ярус; з цих бітів один нульовий, тому на наступному ярусі також два вузли і т.д.). Якщо ж спочатку впорядкувати дерево таким чином, щоб при його обході довжини кодів

монотонно зростали, достатньо закодувати кількість кінцевих вузлів на кожному ярусі.

Дещо підвищити ефективність стиснення нерівномірними кодами можна шляхом кодування не окремих символів, а блоків (комбінацій) символів. За рахунок того, що кількість таких комбінацій є більшою, ніж кількість символів, а розподіл ймовірностей носить більш рівномірний характер, таке кодування є більш ефективним. Зокрема, якщо у первинному алфавіті деякий символ мав ймовірність, суттєво більшу за 0.5, тобто оптимальна кількість біт для його кодування була меншою за 1, в отриманому алфавіті комбінацій символів максимальна ймовірність буде нижчою, і, отже, нижчою буде надлишковість коду (вище було вказано, що, наприклад, надлишковість коду Хаффмана є тим більшою, чим більша ймовірність найбільш ймовірного символу).

Приклад.

Нехай для джерела з алфавітом X={A,B,C} ймовірності символів складають: p(A)=0.8, p(B)=0.15, p(C)=0.05. Утворимо код Хаффмана:

A	0
B	10
C	11

Середня довжина кодового слова складає

$\lambda_{сер}$=1*0.8+2(0.15+0.05)=1.2 біт/символ.

Утворимо новий алфавіт X’ із символів x_0=AA, x_1=AB, x_2=BA, x_3=AC, x_4=CA, x_5=BB, x_6=BC, x_7=CB, x_8=CC. Підрахуємо ймовірності цих символів (оскільки вважаємо джерело інформації джерелом без пам’яті, вони будуть рівні добуткам ймовірностей відповідних символів алфавіту X), та побудуємо для цього алфавіту код Хаффмана.

Символ	Імовірність	Код Хаффмена
x_0	0.64	0
x_1	0.12	10
x_2	0.12	110
x_3	0.04	1110
x_4	0.04	11110
x_5	0.0225	111110
x_6	0.0075	1111110
x_7	0.0075	11111110
x_8	0.0025	11111111

Середня довжина кодового слова складає

$\lambda_{сер}$=1*0.64+0.12(2+3)+0.04(4+5)+0,0225*6+0.0075(7+8)+0.0025*8=1.868 біт/символ

Оскільки кожний символ є складеним із двох, на кожен символ первинного алфавіту припадає 1.868/2=0.934 біт – суттєво менше, ніж при кодуванні

одиничних символів. При збільшенні довжини блоку символів ця величина буде ще меншою.

Для оптимального префіксного коду, що кодує блоки довжиною *n* символів, середня довжина кодового слова

$$H_n(X) \le \lambda_{сер} \le H_n(X) + \frac{1}{n},$$

тобто надлишковість після стиснення буде тим меншою, чим більша довжина блоку. Однак при цьому значно зростає алфавіт символів. Так, якщо первинний алфавіт складається з 256 символів, то після об'єднання символів у блоки довжиною лише 2 символи розмір алфавіту зросте до 256^2=65536. Це збільшує час побудови кодового дерева і зменшує ефективність стиснення при необхідності передачі цього дерева разом із стиснутими даними.

Слід зауважити, що при блоковому кодуванні з описаним алгоритмом вважається, що джерело інформації є джерелом без пам'яті, тобто кореляція між символами не враховується, хоча і розглядаються ймовірності комбінацій символів, а не окремих символів.

Контрольні питання

1. Охарактеризуйте нерівномірні префіксні коди.
2. Наведіть загальні відомості про методи стиснення без втрат.
3. Дайте характеристику оптимальним нерівномірним префіксним кодам.
4. Охарактеризуйте Канонічний алгоритм Хаффмана.
5. Охарактеризуйте Алгоритм Шеннона-Фано.
6. Алгоритм, що використовує тільки частоту появи однакових байт у вхідному потоці даних та співставляє символам, що зустрічаються частіше ланцюг бітів меншої довжини.
7. За рахунок чого здійснюється стиснення по Хаффману?
8. Охарактеризуйте неруйнуючий алгоритм стиснення, що не завжди приводить до однозначної побудови кода.
9. Скільки разів класичний алгоритм Хафмана проходить по файлу?

ЛЕКЦІЯ 5

на тему: Кодування джерел без пам'яті. Методи стиснення без втрат (частина 2)

У лекції приводяться основні відомості про методи стиснення без втрат, розглядаються алгоритм арифметичного стиснення, а також алгоритм інтервального кодування, їх переваги і недоліки.

План:

1. Арифметичне стиснення.
2. Адаптивні методи статистичного кодування

1. Арифметичне стиснення

1.1 Класичний варіант алгоритму

Стиснення по методу Хаффмана поступово витісняється арифметичним стисненням. Свою роль в цьому зіграло те, що закінчилися терміни дії патентів, що обмежують використовування арифметичного стиснення. Крім того, алгоритм Хаффмана наближає відносні частоти появи символів в потоці частотами кратними ступеня двійки (наприклад, для символів a, b, c, d з вірогідністю 1/2, 1/4, 1/8, 1/8 будуть використані коди 0, 10, 110, 111), а арифметичне стиснення дає кращий ступінь наближення частоти. За теоремою Шеннона, найкраще стиснення в двійковій арифметиці ми одержимо, якщо кодуватимемо символ з відносною частотою f з допомогою $-\log_2(f)$ бітів.

На графіку, показаному на рис. 5.1 приводиться порівняння оптимального кодування і кодування по методу Хаффмана. Добре видно, що за ситуації, коли відносні частоти не є ступенями двійки, стиснення стає менш ефективним (ми витрачаємо більше бітів, ніж це необхідне). Наприклад, якщо у нас два символи a і b з вірогідністю 253/256 і 3/256, то в ідеалі ми повинні витратити на ланцюжок з 256 байт $-\log_2(253/256)\cdot 253 - \log_2(3/256)\cdot 3 = 23.546$, тобто 24 біти. При кодуванні по Хаффману ми закодуємо a і b як 0 і 1, і нам доведеться витратити 1·253+1·3=256 бітів, тобто в 10 разів більше. Розглянемо алгоритм, що дає результат, близький до оптимального.

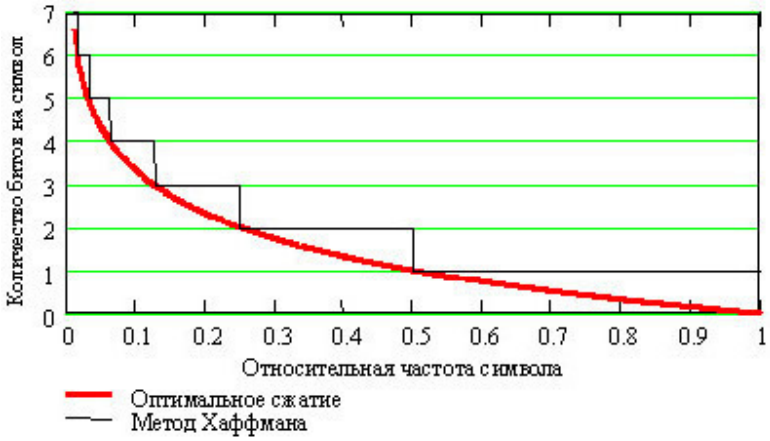

Рисунок 5.1 – Порівняння оптимального кодування і кодування по методу Хаффмана

Арифметичне стиснення - достатньо витончений метод, в основі якого лежить дуже проста ідея. Ми представляємо кодований текст у вигляді дробу, при цьому будуємо дріб так, щоб наш текст був представлений якомога компактніше.

На відміну від префікс них кодів, при арифметичному кодуванні можна досягти степені стиснення, як завгодно близької до теоретичного максимуму, визначеного теоремою Шеннона. Це пояснюється тим, що результуючий код на виході арифметичного кодера ставиться у відповідність цілому повідомленню, тобто у результуючому коді неможливо виділити фрагменти, що відповідають окремим символам вхідного повідомлення. При префіксному ж кодуванні кожному символу ставиться у відповідність певне кодове слово, довжина якого не може бути нецілою, що і зумовлює деяку втрату ефективності стиснення.

При арифметичному кодуванні повідомлення представляється дійсним додатнім числом із інтервалу [0,1]. По мірі кодування повідомлення інтервал його представлення звужується, а кількість двійкових розрядів збільшується. Кожен символ повідомлення звужує цей інтервал на величину, що залежить від ймовірності появи символу: чим більш ймовірний символ, тим ширшим є відповідний йому підінтервал і тим менше біт додається до вихідного коду для кодування цього підінтервалу; чим менш ймовірний символ, тим вужчий під інтервал, тим точніше необхідно задавати його межі і тим більше біт необхідно для цього.

До початку кодування робочим інтервалом є [0,1). Він розбивається на підінтервали, довжини яких пропорційні ймовірностям символів. Після надходження першого символу на вхід кодера із робочого інтервалу виділяється підінтервал, що відповідає цьому символу. Цей під інтервал використовується в якості робочого на наступному кроці, тобто він знову розбивається на підінтервали, довжини яких пропорційні ймовірностям символів, і далі процес повторюється до закінчення символів повідомлення.

Алгоритм оновлення робочого інтервалу на кожному кроці можна представити так:

довжина інтервалу

$L_i=L_{i-1}*p(x^{(i)})$,

початок інтервалу

$a_i=a_{i-1}+L_{i-1}*q(x^{(i)})$,

кінець інтервалу

$b_i=a_i+L_i$.

Тут $x^{(i)}$ – символ на виході джерела в і-й момент часу, $q(x_j)=\sum_{k=0}^{j-1} p(x_k)$ – кумулятивна ймовірність поточного символу, тобто сума всіх символів, що знаходяться в алфавіті перед цим символом.

Остаточним результатом кодування є деякий інтервал, тобто два числа. Однак для однозначного відновлення повідомлення достатньо зберігати одне число, що міститься в середині цього інтервалу, або є його початком. Достатньо зберігати стільки бітів результату, щоб похибка представлення була не більшою за довжину інтервалу.

На рис. 5.2 наведено приклад роботи арифметичного кодера для джерела з алфавітом X={A,B,C} та ймовірностями символів p(A)=1/2, p(B)=1/4, p(C)=1/4 для послідовності символів «ABAC».

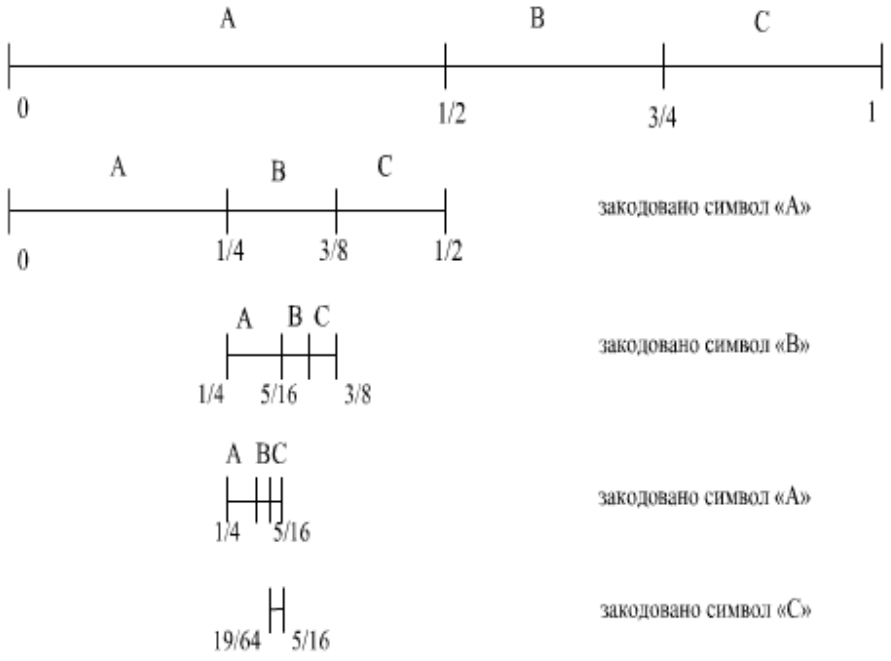

Рисунок 5.2 – Схема роботи арифметичного кодера

Результатом кодування в цьому випадку є інтервал (19/64…5/16) довжиною 1/64. В двійковому представленні 19/64=0,010011, 5/16=0,010100. Ці результати відрізняються на 0,000001, отже зберігати необхідно 6 біт (або простіше,

$\log_2(1/64)=6$), тому результатом кодування є 010011. Ціла частина завжди нульова, тому зберігати її немає необхідності.

Декодер працює наступним чином. Після отримання результату кодування визначається, до якого інтервалу належить прийняте значення, тобто який символ є першим у повідомленні. На наступному кроці, як і в алгоритмі кодування, цей інтервал розглядається в якості робочого, тобто ділиться на під інтервали відповідно до ймовірностей символів, знову визначається , якому з під інтервалів належить результат кодування і т.д. Процес декодування продовжується до тих пір, поки ширина робочого інтервалу не стане меншою за 2^{-n}, n – кількість біт у стиснутому повідомленні.

В розглянутому прикладі, отримавши результат 0,010011 (19/64), на першому кроці визначаємо, що він належить до інтервалу (0,1/2), який відповідає символу «A» - отже, першим символом повідомлення є «A». Поділивши інтервал (0,1/2) так само, якна другому кроці кодування (рис. 2.3), далі визначаємо, що 19/64 належить інтервалу (1/4, 3/8), що відповідає символу «B», і т.д.

Із алгоритму кодування випливає, що остаточний результат стиснення стає відомим тільки в кінці процесу кодування, тому не можна починати передачу стиснутого повідомлення, поки не буде закодований його останній символ. Однак в дійсності по мірі кодування інтервал представлення результату звужується, і його старші біти перестають змінюватися, тому їх можна починати передавати із відносно невеликою затримкою.

Точність представлення інтервалу залежить від довжини повідомлення, тобто для довгих повідомлень необхідна точність, яка може перевищувати точність стандартних машинних типів чисел з плаваючою комою. Ця проблема вирішується шляхом використання цілочисельної арифметики та масштабування робочого інтервалу кожен раз після видачі на вихід кодера старших розрядів результату. Такий підхід використовується, зокрема, в модифікації арифметичного кодування, що має назву RANGE-кодування. При цьому розглядається не інтервал дійсних чисел [0,1], а інтервал цілих чисел $0\ldots2^{N-1}$, де N – це, як правило, розрядність одного із стандартних цілих типів даних. Якщо в процесі кодування поточний інтервал звужується до значення, що менше за деякий поріг, наприклад, коли розрядність коду стає менше половини довжини регістра, то вже закодовані біти результату передаються на вихід кодера, а інтервал знову розширюється шляхом зсуву вліво на відповідну кількість біт із заповненням молодших біті нулями (для нижньої межи) та одиницями (для верхньої межі). Процес повторюється до закінчення повідомлення, що дозволяє представити результуючий інтервал із як завгодно великою точністю.

Декодування також можна за тим же принципом. Декодер на початку процесу декодування повинен зчитати із вхідного потоку фрагмент довжиною N біт. Робочий інтервал $0\ldots2^{N-1}$ ділиться на підінтервали, пропорційні ймовірностям символів, визначається, якому з під інтервалів належить число, утворене поточним

фрагментом, і відповідний символ видається на вихід декодера. Якщо довжина робочого інтервалу досягає заданої межі в K розрядів, то K старших біт усувається з поточного фрагменту (ці біти співпадають із старшими бітами маж інтервалу, що надалі не будуть змінюватись), фрагмент доповнюється наступними K бітами із вхідного потоку, а робочий інтервал розширюється до N розрядів. Процес продовжується до тих пір поки не будуть зчитані всі біти із вхідного потоку. Після зчитування останніх K біт декодування продовжується, поки довжина робочого інтервалу не досягне K розрядів.

Отже, реальний алгоритм працює з цілими числами і оперує з дробами, чисельник і знаменник яких є цілими числами. При цьому з втратою точності можна боротися, відстежуючи зближення l_i і h_i і помножуючи чисельник і знаменник їх дробу на якесь число (зручно на 2).

Процедура зміни значень l_i і h називається нормалізацією, а виведення відповідних бітів - перенесенням.

На символ з меншою ймовірністю у нас витрачається в цілому більше число бітів, ніж на символ з більшою ймовірністю.

Як видно, з неточностями арифметики ми боремося, виконуючи окремі операції над l_i і h_i синхронно в компресорі і декомпресорі.

Незначні втрати точності (частки відсотка при достатньо великому файлі) і, відповідно, зменшення ступеня стиснення в порівнянні з ідеальним алгоритмом відбуваються під час операції розподілу, при округленні відносних частот до цілого, при записі останніх бітів у файл. Алгоритм можна прискорити, якщо представляти відносні частоти так, щоб дільник був ступенем двійки (тобто замінити розподіл операцією побітового зсуву).

Для того, щоб оцінити ступінь стиснення арифметичним алгоритмом конкретного рядка, потрібно знайти мінімальне число N, таке, що довжина робочого інтервалу при стисненні останнього символу ланцюжка була б менше $1/2^N$. Цей критерій означає, що усередині нашого інтервалу явно знайдеться хоча б одне число, в двійковому представленні якого після N-го знаку буде тільки 0. Довжину ж інтервалу порахувати просто, оскільки вона рівна добутку ймовірності всіх символів.

Розглянемо приклад рядка, що наводився раніше, з двох символів a і b з ймовірністю 253/256 і 3/256. Довжина останнього робочого інтервалу для ланцюжка з 256 символів a і b з вказаною ймовірністю дорівнює:

$$h_{256} - l_{256} = \left(\frac{253}{256}\right)^{253} \cdot \left(\frac{3}{256}\right)^{3} = \frac{253^{253} \cdot 9}{2^{2048}} \approx 8.15501 \cdot 10^{-8}$$

Легко підрахувати, що $N = 24$(($1/2^{24} \approx 5.96 \cdot 10^{-8}$), оскільки 23 дає дуже великий інтервал (в 2 рази ширше), а 25 не є мінімальним числом, що задовольняє критерію. Вище було показано, що алгоритм Хаффмана кодує даний ланцюжок в 256 бітів. Тобто для розглянутого прикладу арифметичний алгоритм дає

десятиразову перевагу перед алгоритмом Хаффмана і вимагає менше 0.1 бітів на символ.

Характеристики арифметичного алгоритму:

Ступінь стиснення: Краща > 8 (можливе кодування менш біта на символ), гірша - 1.

Плюси алгоритму: Забезпечує кращий ступінь стиснення, ніж алгоритм Хаффмана (на типових даних на 1-10%).

Характерні особливості: Також як кодування по Хаффману, не збільшує розміру початкових даних у гіршому разі.

1.2 Інтервальне кодування

На відміну від класичного алгоритму, інтервальне кодування припускає, що ми маємо справу з цілими дискретними величинами, які можуть приймати обмежене число значень. Як вже було відзначено, початковий інтервал в цілочисельній арифметиці записується у вигляді $[0, N)$ або $[0, N-1]$, де N - число можливих значень змінної, що використовується для зберігання меж інтервалу.

Щоб найбільш ефективно стиснути дані, ми повинні закодувати кожний символ s за допомогою $-\log_2(f_s)$ бітів, де f_s - частота символу s. Звичайно, на практиці така точність недосяжна, але ми можемо для кожного символу s відвести в інтервалі діапазон значень $[N(F_s), N(F_s + f_s))$, де F_s - накопичена частота символів, передуючих символу s у алфавіті $N(f)$ - значення, що відповідає частоті f у інтервалі з N можливих значень. І, чим більше буде $N(f_s)$, тим точніше буде представлений символ s у інтервалі. Слід зазначити, що для всіх символів алфавіту повинна дотримуватися нерівність $f_s > 0$.

Задачу збільшення розміру інтервалу виконує процедура, що називається нормалізацією. Практика показує, що можна відкласти виконання нормалізації на деякий час, поки розмір інтервалу забезпечує прийнятну точність. Мікаель Шиндлер (Michael Schindler) запропонував розглядати вихідний потік як послідовність байтів, а не бітів, що позбавило від бітових операцій і дозволило проводити нормалізацію помітно рідше. І частіше за все нормалізація обходиться без виконання перенесення, що виникає при складанні значень нижньої межі інтервалу і розміру інтервалу. В результаті швидкість кодування зросла в півтора разу при незначній втраті в ступені стиснення (розмір стислого файлу звичайно збільшується лише на соті частки відсотка).

Вихідні дані кодера можна представити у вигляді чотирьох складових:

1. Складова, записана у вихідний файл, яка вже не може змінитися.
2. Один елемент (битий або байт), який може бути змінений перенесенням, якщо останній виникне при складанні значень нижньої межі інтервалу і розміру інтервалу.

3. Блок елементів, що мають максимальне значення, через які по ланцюжку може пройти перенесення.

4. Поточний стан кодера, представлений нижньою межею інтервалу.

При виконанні нормалізації можливі наступні дії:

1. Якщо інтервал має прийнятний для забезпечення заданої точності розмір, нормалізація не потрібна.

2. Якщо при складанні значень нижньої межі інтервалу і розміру інтервалу не виникає перенесення, складові 2 і 3 можуть бути записані у вихідний файл без змін.

3. У разі виникнення перенесення він виконується в складових 2 і 3, після чого вони також записуються у вихідний файл.

4. Якщо елемент, що претендує на запис у вихідний файл, має максимальне значення (у разі біта - 1, у разі байта - 0xFF), то він може вплинути на попередній при виникненні перенесення. Тому цей елемент записується в блок, відповідний третій складовій.

Як вже було відзначено, частіше за все при нормалізації не відбувається перенесення. Виходячи з цього, Дмитро Субботін запропонував відмовитися від перенесення зовсім. Виявилося, що втрати в стисненні зовсім незначні, порядка декілька байт. Втім, виграш по швидкості теж виявився не дуже помітний. Головна перевага такого підходу - в простоті і компактності коду.

Можна помітити, що уникнути перенесення нам дозволяє своєчасне примусове зменшення значення розміру інтервалу.

2. Адаптивні методи статистичного кодування

Суть адаптивних алгоритмів полягає у перебудові коду після надходження кожного наступного символа повідомлення. При цьому перебудова коду здійснюється тільки після кодування поточного символа за допомогою коду, створеного на попередньому кроці. Таким чином підтримується синхронна робота кодера і декодера, що дозволяє реалізувати однопрохідний алгоритм та не передавати кодове дерево чи маси ймовірностей символів разом із стиснутими даними (оскільки код перебудовується декодером після отримання кожного наступного символа, тобто аналогічним чином як у кодера).

Для оцінки ймовірностей символів в адаптивних алгоритмах використовуються лічильники частоти символів, що збільшуються на 1 при надходженні відповідного символа на вхід кодера.

2.1 Адаптивний алгоритм Хаффмана

В адаптивному алгоритмі Хаффмана на кожному кроці здійснюється перебудова кодового дерева. До початку кодування кодове дерево містить тільки один спеціальний символ (escape-символ, ESC), який використовується при

додаванні в дерево нових символів. При появі у повідомленні символу, що раніше не зустрічався, на вихід кодера подається код ESC та незакодоване значення символу, після чого цей симол додається у дерево. Для декодера символ ESC є вказівкою на те, що наступні К біт позначають символ первинного алфавіту, при цьому $K=\log_2 N$, N – кількість символів в первинному алфавіті.

При надходженні кожного наступного символа лічильник його частоти збільшується на одиницю, і дерево перебудовується таким чином, щоб частоти вузлів були впорядковані по зростанню від листків до кореня дерева і зліва направо – на кожному ярусі дерева. Якщо символ раніше зустрічався в повідомленні, то він присутній у кодовому дереві, і його код формується шляхом проходження дерева від кореня до вузла, який відповідає даному символу.

До заповнення дерева всіма можливими символами ефективність стиснення може бути дещо нижчою, ніж у не адаптивного алгоритму Хаффмана. Однак при значній довжині повідомлення ступінь стиснення буде наближатися до показників не адаптивного алгоритму, оскільки розподіл значень лічильників символів буде наближатись до розподілу ймовірностей на виході джерела інформації. Крім того, якщо статистичні характеристики даних в межах повідомлення змінюються, адаптивний алгоритм забезпечуватиме повільну адаптацію до цих змін.

2.2 Адаптивне арифметичне кодування

Ідея адаптивного алгоритму арифметичного стиснення полягає в тому, щоб перебудовувати таблицю ймовірності $b[j]$ по ходу упаковки і розпаковування безпосередньо при отриманні чергового символу. Такий алгоритм не вимагає збереження значень ймовірності символів у вихідний файл і, як правило, дає великий ступінь стиснення.

До початку кодування лічильникам частот усіх символів присвоюється значення 1. На кожному кроці поточний інтервал представлення повідомлення ділиться на під інтервали, довжини яких пропорційні значенням лічильників символів, вибирається підінтервал, що відповідає наступному символу на вході кодера, після чого до лічильника частот цього символу додається 1, і на наступному кроці підінтервали будуються уже із врахуванням цієї нової частоти.

Якщо заздалегідь хоча б приблизно відомі статистичні характеристики джерела інформації, можна ініціалізувати лічильники символів значеннями, пропорційними оцінкам ймовірностей. Це прискорить процес адаптації до характеристик джерела і відповідно дещо підвищить ефективність стиснення, однак аналогічні оцінки ймовірностей повинні бути відомі декодеру.

Контрольні питання

1. Охарактеризуйте класичний варіант алгоритму арифметичного стиснення.
2. Охарактеризуйте варіант алгоритму арифметичного стиснення – інтервальне кодування.
3. Алгоритм, що є наближеним до оптимального кодування, представляє кодуємий текст у вигляді дробу і використовує діапазони відповідних символів.
4. Алгоритм, що є наближеним до оптимального кодування, передбачає і використовує цілі дискретні величини, які можуть приймати обмежене число значень
5. Охарактеризуйте адаптивний метод статистичного кодування – адаптивний алгоритм Хаффмана.
6. Охарактеризуйте адаптивний метод статистичного кодування – адаптивний арифметичне кодування.
7. На чому засноване ймовірнісне стиснення?

ЛЕКЦІЯ 6

на тему: Словникові методи стиснення даних

У лекції дається опис основної ідеї словникових методів стиснення даних, розглядаються класичні алгоритми Зіва-Лемпеля, такі як LZ77 та LZ78, приводиться приклади реалізації алгоритмів.

План:

1. Ідея словникових методів.
2. Класичні алгоритми Зіва-Лемпеля. Загальна характеристика.
3. Алгоритм LZ77 та його модифікації.
4. Алгоритм LZ78 та його модифікації.

1. Ідея словникових методів

Вхідну послідовність символів можна розглядати як послідовність рядків, що містять довільну кількість символів. Ідея словникових методів полягає в заміні рядків символів на такі коди, що їх можна розглядати як індекси рядків деякого словника. Рядки, що утворюють словник далі називатимемо **фразами**. При декодуванні здійснюється зворотна заміна індексу на відповідну йому фразу словника.

Можна сказати, що ми намагаємося перетворити початкову послідовність шляхом її представлення в такому алфавіті, що його "букви" є фразами словника, які складаються, в загальному випадку, з довільної кількості символів вхідної послідовності.

Словник - це набір таких фраз, які, як ми вважаємо, зустрічатимуться в оброблюваній послідовності. Індекси фраз повинні бути побудовані так, щоб в середньому їх представлення займало менше місця, ніж вимагають рядки, що заміщаються. За рахунок цього і відбувається стиснення.

Зменшення розміру можливе в першу чергу за рахунок того, що звичайно в даних, що стискаються, зустрічається лише мала кількість всіх можливих рядків довжини n, тому для представлення індексу фрази потрібне, як правило, менше число бітів, ніж для представлення початкового рядка.

Наприклад, розглянемо – (див. табл. 6.1) кількість взаємно різних рядків довжини від 1 до 5 в тексті (роман Ф.М. Достоєвского «Біси», звичайний неформатований текст, розмір близько 1.3 Мбайт).

Таблиця 6.1 – Кількість взаємно різних рядків довжини від 1 до 5 в тексті

Довжина рядка	Кількість різних рядків	Використано комбінацій, % від усіх можливих
5	196969	0.0004
4	72882	0.0213
3	17481	0.6949
2	2536	13.7111
1	136	100.0000

Інакше кажучи, розмір (потужність) алфавіту рівний 136 символам, але реально використовується тільки $\frac{2536}{136 \cdot 136} \cdot 100\% \approx 13.7\%$ від всіх можливих двосимвольних рядків, і т.д.

Далі, якщо у нас є заслуговуючі довіри гіпотези про частоту використовування тих або інших фраз, або проводився якийсь частотний аналіз оброблюваних даних, то ми можемо призначити більш вірогідним фразам коди меншої довжини. Наприклад, для тієї ж електронної версії романа "Біси" статистика зустрічаємості рядків довжини 5 преставлена в табл. 6.2.

Таблиця 6.2 – Статистика зустрічаємості рядків довжини 5

N	Кількість рядків довжини 5, що зустрілися рівно N раз	Кількість відносно загального числа всіх різних рядків довжини 5, %
1	91227	46.3%
2	30650	15.6%
3	16483	8.4%
4	10391	5.3%
5	7224	3.7%
≥6	40994	20.7%
Всього	196969	100.0%

Зауважимо, що зі всіх 197 тисяч різних рядків довжини 5 майже половина зустрілася лише один раз, тому вони взагалі не будуть використані як фрази при словарному кодуванні в тому випадку, якщо словник будується тільки з рядків обробленої частини потоку. Спостережувані частоти частини рядків, що залишилася, швидко зменшуються із збільшенням N, що указує на вигідність використання статистичного кодування, коли фразам, що часто використовуються, ставляться у відповідність коди меншої довжини.

Зазвичай просто передбачається, що короткі фрази використовуються частіше довгих. Тому в більшості випадків індекси будуються так, щоб довжина індексу короткої фрази була менше довжини індексу довгої фрази. Такий прийом сприяє поліпшенню стиснення.

Словникові методи є універсальними і підходять для цифрових даних будь-якого типу. В порівнянні з іншими універсальними методами стиснення вони є найменш вимогливими до обчислювальних ресурсів (швидкості стиснення / розпаковки, необхідного об'єму пам`яті). Саме тому словникові методи використовуються в програмах-архіваторах (WinRar, WinZip).

Словникові методи орієнтовані на стиснення якісних даних, причому ефективність використання досягається у тому випадку, коли статистичні характеристики оброблюваних даних відповідають моделі джерела з пам'яттю.

2. Класичні алгоритми Зіва-Лемпеля. Загальна характеристика

Алгоритми словникового стиснення Зіва-Лемпеля з'явилися в другій половині 1970-х років. Це були так звані алгоритми LZ77 і LZ78, розроблені сумісно Зівом (Ziv) і Лемпельом (Lempel).

Практично всі варіанти словникових методів ґрунтуються на працях А. Лемпеля та Я. Зіва 1977 та 1978 рр. Відповідно до розглянутих у цих статтях двох підходів можна виділити методи групи LZ77, які використовують в якості словника уже переглянутий фрагмент повідомлення, та методів групи LZ78, які формують словник із підстрок проглянутої частини повідомлення. Різні модифікації методів відрізняються способами організації словника та способами посилання на словник.

Надалі первинні схеми піддавалися множинним змінам, внаслідок чого ми сьогодні маємо десятки достатньо самостійних алгоритмів і незлічену кількість модифікацій.

LZ77 і LZ78 є універсальними алгоритмами стиснення, в яких словник формується на підставі вже обробленої частини вхідного потоку, тобто адаптивно. Принциповою відмінністю є лише спосіб формування фраз. Тому словарні алгоритми Зіва-Лемпеля розділяють на два сімейства - алгоритми типу LZ77 і алгоритми типу LZ78. Іноді також говорять про словарні методи LZ1 і LZ2.

Стиснення при використанні словникових методів досягається за рахунок того, що покажчики на елементи словника займають менше місця, ніж строки, на які вони посилаються. Оскільки ймовірність появи різних строк не однакова, розподіл ймовірностей покажчиків буде нерівномірним, тому додаткового стиснення модна досягти шляхом застосування до результату кодування одного із статистичних методів. Наприклад, оскільки коротші строки зустрічаються частіше, ніж довгі, їх індекси можна кодувати коротшими кодами.

Чим більшим є словник, тим більша ймовірність, що у ньому буде знайдена потрібна строка. Однак із збільшенням словника зростає час пошуку, а також кількість біт, якими кодується посилання на словник (N=$\log_2 L_{СЛ}$, де $L_{СЛ}$ – довжина словника).

Публікації Зіва і Лемпеля носили чисто теоретичний характер, оскільки ці дослідники насправді займалися проблемою вимірювання "складності" рядка, і використання створених алгоритмів до стиснення даних було, швидше, лише приватним результатом. Потрібен був якийсь час, щоб ідея організації словника, часто в перекладенні вже інших людей, досягла розробників програмного і апаратного забезпечення. Тому практичне використовування алгоритмів почалося через пару років.

З тих пір методи даного сімейства незмінно є найпопулярнішими серед всіх методів стиснення даних, хоча останнім часом ситуація початку мінятися на користь BWT і PPM, як забезпечуючих краще стиснення.

Необхідно сказати декілька слів про найменування алгоритмів і методів. При позначенні сімейства загальноприйнятою є абревіатура "LZ", але розшифровуватися вона повинна як "Ziv-Lempel", тому і алгоритми "Зіва-Лемпеля", а не "Лемпеля-Зіва". Згідно загальноприйнятому поясненню цього курйозу, Якоб Зів вніс більший внесок у відкриття відповідних словникових схем і дослідження їх властивостей і таким чином заслужив, щоб першим стояло його прізвище. Але випадково була допущена помилка, і прикріпилося скорочення "LZ" (букви впорядковані в алфавітному порядку). Іноді, до речі, зустрічається і позначення "ZL" (порядок букв відповідає порядку прізвищ авторів в публікаціях). Надалі, якщо якийсь дослідник істотно змінював якийсь алгоритм, що відноситься до сімейства LZ, то в назві одержаної модифікації до рядка "LZ" зазвичай дописувалася перша буква його прізвища, наприклад: алгоритм LZB, автор Белл (Bell).

Підкреслимо також наявність великої плутанини з класифікацією алгоритмів. Звичайно вона виявляється в небажанні визнавати існування двох самостійних сімейств LZ, а також в неправильному віднесенні алгоритмів до конкретного сімейства. Безладдю часто сприяють самі розробники: багато кому невигідно розкривати, на основі якого алгоритму створена дана модифікація через комерційні, патентні або інші меркантильні міркування. Наприклад, у разі комерційного програмного забезпечення загальноприйнятою є практика класифікації алгоритму стиснення, що використовується, як "модифікації LZ77". І в цьому немає нічого дивного, адже алгоритм LZ77 не запатентований.

В таблиці 6.3 наведені основні варіанти LZ-методу та відомості про їх основні відмінності в реалізаціях.

Таблиця 6.3 – Основні варіанти LZ-схеми

Назва	Рік	Автори	Відмінності
LZ77	1977	Ziv and Lempel	Вказівники і символи чергуються. Вказівники адресують рядок серед попередніх N символів.
LZR	1981	Roden et al.	Вказівники і символи чергуються. Вказівники адресують рядок серед усіх попередніх символів.
LZSS	1986	Bell	Вказівники і символи розрізняються прапорцем-бітом. Вказівники адресують рядок серед попередніх N символів.
LZB	1987	Bell	Аналогічно LZSS, але для вказівників застосовується різне кодування.
LZH	1987	Brent	Аналогічно LZSS, але на другому кроці для вказівників застосовується кодування Хаффмана.
LZ78	1978	Ziv and Lempel	Вказівники і символи чергуються. Вказівники адресують раніше розібраний рядок. Вказівники мають фіксовану довжину.
LZW	1984	Welch	Виведення вміщує тільки вказівники. Вказівники адресують раніше розібраний рядок. Вказівники мають фіксовану довжину.
LZC	1985	Tomas et al.	Виведення вміщує тільки вказівники. Вказівники адресують раніше розібраний рядок.
LZT	1987	Tischer	Аналогічно LZC, але фрази поміщаються в LRU-список.
LZMW	1984	Miller and Wegman	Аналогічно LZT, але фрази будуються конкатенацією двох попередніх фраз.
LZJ	1985	Jakobsson	Виведення вміщує тільки вказівники. Вказівники адресують рядок серед усіх попередніх символів.
LZFG	1989	Fiala and Greene	Вказівник обирає вузол в дереві цифрового пошуку. Рядки в дереві беруться із ковзаю чого вікна.

В таблиці 6.4 наведені деякі типові розширення, а також програми-архіватори та методи стиснення даних, що їм відповідають.

Таблиця 6.4 – Методи стиснення та їх використання в програмах-архіваторах

Розширення	Програми	Тип кодування
Z	compress	LZW
arc	arc, pkarc	LZW, Хаффмана
zip	zip, unzip, pkzip, pkunzip	LZW, LZ77, Хаффмана, Шеннона-Фано
gz	gzip	LZ77, Хаффмана
bz2	bzip2	Барроуза-Уіллера, Хаффмана
arj	arj	LZW, Хаффмана
ice, lzh	lha, lharc	LZW, Хаффмана
pak	pak	LZW

Практично всі формати файлів для зберігання графічної інформації використовують стиснення даних. Формат графічного файла також, як правило, ідентифікується розширенням імені файла. В таблиці 6.5 наведені деякі типові розширення графічних файлів і методи стиснення даних, що їм відповідають.

Таблиця 6.5 – Методи стиснення в форматах графічної інформації

Розширення	Тип кодування
gif	LZW
png	LZW, Хаффмана
jpeg, ipg	стиснення з втратами, Хаффмана або арифметичне стиснення
bmp, pcx	RLE
tiff, tif	CCITT/3 для факсів, LZW або інші

3. Алгоритм LZ77 та його модифікації

3.1 Алгоритм LZ77

Цей словарний алгоритм стиснення є найстарішим серед методів LZ. Опис був опублікований в 1977 році, але сам алгоритм розроблений не пізніше за 1975 рік.

Алгоритм LZ77 є "родоначальником" цілого сімейства словникових схем - так званих алгоритмів з ковзаючим словником, або ковзаючим вікном. Дійсно, в LZ77 як словник використовується блок вже закодованої послідовності. Як правило, у міру виконання обробки положення цього блоку щодо початку послідовності постійно змінюється, словник "ковзає" по вхідному потоку даних.

Ковзаюче вікно має довжину N, тобто в нього поміщається N символів, і складається з 2 частин:

- послідовності довжини W = N-n вже закодованих символів, яка і є словником;
- попереджуючого буфера, або буфера попереднього перегляду (lookahead), довжини n; звичайно n на порядок менше W.

Одержана в результаті пошуку фраза кодується за допомогою двох чисел:

- зсуву (offset) від початку буфера, i;
- довжини відповідності, або збігу (match length), j.

Зсув і довжина відповідності грають роль покажчика (посилання), однозначно визначаючого фразу. Додатково у вихідний потік записується символ s, безпосередньо наступний за рядком буфера, що співпав.

Таким чином, на кожному кроці кодер видає опис трьох об'єктів: зсуву і довжини відповідності, утворюючих код фрази, що складається з обробленого рядка буфера, і одного символу s (літерала).

Потім вікно зміщується на j+1 символів вправо і здійснюється перехід до нового циклу кодування. Величина зсуву пояснюється тим, що ми реально

закодували саме j+1 символів: j за допомогою покажчика на фразу в словнику, і 1 за допомогою тривіального копіювання. Передача одного символу в явному вигляді дозволяє вирішити проблему обробки ще жодного разу не бачених символів, але істотно збільшує розмір стислого блоку.

В методі LZ77 в якості словника використовується переглянутий фрагмент повідомлення, а посилання на словник містить відстань до строки у словнику від поточної позиції та довжину цієї строки.

Базова модифікація LZ77 передбачає використання в якості словника ковзаю чого вікна – фрагмента повідомлення довжиною $L_{СЛ}$ до поточної позиції, та буферу перегляду – фрагмента повідомлення довжиною $L_{БУФ}$ після поточної позиції. На кожному кроці у словнику шукається найдовший рядок, що співпадає із вмістом буфера, починаючи від поточної позиції. Якщо такий рядок знайдений, то на вихід кодера видається:

- зміщення знайденого у словнику рядка відносно поточної позиції;
- довжина знайденого рядка;
- наступний символ з буфера після закодованого рядка.

Якщо рядок не знайдений, на вихід кодера видається (0,0, наступний символ з буфера).

Для кодування кожної трійки значень використовується $\log_2 L_{СЛ} + \log_2 L_{БУФ} +$ $+ \log_2 L_A$ біт, де L_A – довжина алфавіту символів повідомлення. Наприклад, якщо довжина словника 4096, довжина буфера 256 та використовується алфавіт з 256 символів, то для кодування відстані до елемента словника використовується 12 біт, для кодування довжини строки – 8 біт, всього 28 біт.

Довжина рядка, що може бути закодований шляхом посилання на словник, у методі LZ77 обмежується довжиною буфера, а максимальна відстань між повторними входженнями строк, що можуть бути закодовані шляхом посилання на словник – довжиною словника. При малих розмірах словника і буфера це може в окремих випадках дещо знизити ефективність методу, а використання надто довгого словника і буфера є небажаним, оскільки збільшуються витрати на кодування. Типове значення довжини словника складає 2048…16384 символів, буфера – 128…2048 символів.

Для досягнення максимально можливого стиснення в процесі пошуку рядка у словнику необхідно переглянути весь словник, знайти всі рядки, які починаються так само, як поточний рядок у буфері, та вибрати з них строку з максимальною довжиною. На практиці часто використовують спрощений підхід: пошук продовжується до тих пір, поки не буде знайдена перший рядок, що частково співпадає із вмістом буфера. Це може дещо зменшити ефективність стиснення, однак суттєво пришвидшує роботу алгоритму.

Приклад

Спробуємо стиснути рядок "кот_ломом_колол_слона" завдовжки 21 символ. Хай довжина буфера рівна 7 символам, а розмір словника більше довжини рядка, що стискається. Домовимося також, що:

- нульовий зсув зарезервували для позначення кінця кодування;
- символ s_t відповідає одиничному зсуву щодо символу s_{t+1}, з якого починається буфер;
- якщо є декілька фраз з однаковою довжиною збігу, то вибираємо найближчу до буфера;
- у невизначених ситуаціях - коли довжина збігу нульова - зсуву привласнюємо одиничне значення.

Нехай для кодування і нам достатньо 5 бітів, для j потрібно 3 біти, і символи s вимагають 1 байта для свого представлення. Тоді всього ми витратимо 12·(5+3+8)= 192 біти. Початково рядок займав 21·8 = 168 бітів, тобто LZ77 кодує наш рядок ще більш марнотратним чином. Не слід також забувати, що ми опустили крок кодування кінця послідовності, який зажадав би ще як мінімум 5 бітів (розмір поля i = 5 біт) (див. табл. 6.6).

Таблиця 6.6 – Приклад кодування (алгоритм LZ77)

Крок	Ковзаюче вікно		Співпадаюча фраза	Закодовані дані		
	Словник	Буфер		i	j	s
1	-	кот_лом	-	1	0	'к'
2	к	от_ломо	-	1	0	'о'
3	ко	т_ломом	-	1	0	'т'
4	кот	_ломом_	-	1	0	'_'
5	кот_	ломом_к	-	1	0	'л'
6	кот_л	омом_ко	о	2	1	'м'
7	кот_лом	ом_коло	ом	3	2	'_'
8	кот_ломом_	колол_с	ко	3	2	'л'
9	кот_ломом_кол	ол_слон	ол	3	2	'_'
10	..._ломом_колол_	слона	-	1	0	'с'
11	...ломом_колол_с	лона	ло	3	2	'н'
12	...ом_колол_слон	а	-	1	0	'а'

Приклад підтвердив, що спосіб формування кодів в LZ77 неефективний і дозволяє стискати тільки порівняно довгі послідовності. До деякої міри стиснення невеликих файлів можна поліпшити, використовуючи коди змінної довжини для зсуву i. Дійсно, навіть якщо ми використовуємо словник в 32 Кбайт, але закодували ще тільки 3 Кбайт, то зсув реально вимагає не 15, а 12 бітів.

Крім того, відбувається істотний програш через використовування кодів однакової довжини при вказівці довжин збігу j. Наприклад, для вже згадуваної електронної версії романа "Біси" були одержані наступні частоти використовування довжин збігу (див. табл. 6.7).

З таблиці виходить, що в цілях мінімізації закодованого представлення для j=6 слід використовувати код якнайменшої довжини, оскільки ця довжина збігу зустрічається частіше за все.

Таблиця 6.7 – Частоти використовування довжин збігу

Довжина збігу, j	Кількість раз, коли максимальна довжина співпадіння дорівнювала j
0	136
1	1593
2	4675
3	11165
4	20047
5	26939
6	28653
7	24725
8	19702
9	14767
10	10820
≥11	27903

Хоча автори алгоритму і довели, що LZ77 може стиснути дані не гірше, ніж будь-який спеціально на них налаштований напівадаптивний словниковий метод, через вказані недоліки це виконується тільки для послідовностей достатньо великого розміру.

Що стосується декодування стислих даних, то воно здійснюється шляхом простої заміни коду на блок символів, що складається з фрази словника і явного символу. Декодер повинен виконувати ті ж дії по зміні вікна, що і кодер. Фраза словника елементарно визначається по зсуву і довжині, тому важливою властивістю LZ77 і інших алгоритмів з ковзаючим вікном є дуже швидка робота декодера.

Декодер LZ77 працює значно швидше, ніж кодер, оскільки процес декодування зводиться до копіювання рядків із уже декодованої частини повідомлення у вихідний потік. Кожну трійку значень (M,N,P) декодер інтерпретує як «скопіювати N символів з позиції, віддаленої на M символів від поточної, та записати у вихідний потік символ P». Зокрема, якщо M=N=0, у вихідний потік копіюється тільки символ P.

Недоліком алгоритму LZ77 є неефективне кодування одиночних символів: якщо у словнику не був знайдений рядок, що співпадає з поточним вмістом буфера, на вихід, окрім поточного символу, передається два «зайвих» нулі.

3.2 Алгоритм LZSS

Алгоритм LZSS дозволяє достатньо гнучко поєднувати у вихідній послідовності символи і покажчики (коди фраз), що до деякої міри усуває властиве LZ77 марнотратство, яке виявляється в регулярній передачі одного символу в прямому вигляді. Ця модифікація LZ77 була запропонована в 1982 році Сторером (Storer) і Жиманські (Szymanski).

Ідея алгоритму полягає в додаванні до кожного покажчика і символу однобітового префікса f, що дозволяє розрізняти ці об'єкти. Інакше кажучи, однобітовий прапор f вказує тип і, відповідно, довжину безпосередньо наступних за ним даних. Це дозволяє:

- записувати символи в явному вигляді, коли відповідний їм код має велику довжину, і, отже, словарне кодування тільки шкодить;
- обробляти символи, що жодного разу не зустрілися до теперішнього моменту.

Приклад

Закодуємо рядок "кот_ломом_колол_слона" з попереднього прикладу і порівняємо коефіцієнт стиснення для LZ77 і LZSS.

Нехай ми переписуємо символ в явному вигляді, якщо поточна довжина максимального збігу буфера і якоїсь фрази словника менше або дорівнює 1. Якщо ми записуємо символ, то перед ним видаємо прапор із значенням 0, якщо покажчик - то із значенням 1. Якщо є декілька співпадаючих фраз однакової довжини, то вибираємо найближчу до буфера. Процес кодування представлений в табл. 6.8.

Таким чином, для кодування рядка по алгоритму LZSS нам було потрібно 17 кроків: 13 разів символи були передані в явному вигляді, і 4 рази ми застосували покажчики. Зауважемо, що при роботі по алгоритму LZ77 нам було потрібно всього лише 12 кроків. З другого боку, якщо задатися тими ж довжинами для i і j, то розмір закодованих після LZSS даних рівний $13\cdot(1+8)+ 4\cdot(1+5+3)= 153$ бітам. Це означає, що рядок був дійсно стиснутий, оскільки його початковий розмір 168 бітів.

Таблиця 6.8 – Приклад кодування (алгоритм LZSS)

Крок	Ковзаюче вікно		Співпадаюча фраза	Закодовані дані		
	Словник	Буфер		f	j	s
1	-	кот_лом	-	0	-	'к'
2	к	от_ломо	-	0	-	'о'
3	ко	т_ломом	-	0	-	'т'
4	кот	_ломом_	-	0	-	' _'
5	кот_	ломом_к	-	0	-	'л'
6	кот_л	омом_ко	о	0	-	'о'
7	кот_ло	мом_кол	-	0	-	'м'
8	кот_лом	ом_коло	ом	1	2	-
9	кот_ломом	_колол_	_	0	2	'_'
10	кот_ломом_	колол_с	ко	1	0	-
11	кот_ломом_ко	лол_сло	ло	1	2	-
12	...от_ломом_коло	л_слона	л	0	0	'л'
13	...т_ломом_колол	_слона	_	0	0	'_'
14	..._ломом_колол_	слона	-	0	0	'с'
15	...ломом_колол_с	лона	ло	1	0	-
16	...мом_колол_сло	на	-	0	0	'н'
17	...ом_колол_слон	а	-	0	0	'а'

4. Алгоритм LZ78 та його модифікації

4.1 Алгоритм LZ78

Алгоритм LZ78 був опублікований в 1978 році, і згодом став "батьком" сімейства словникових методів LZ78.

Алгоритми цієї групи не використовують ковзаючого вікна і в словник поміщають не всі рядки, що зустрічаються при кодуванні, а лише "перспективні" з погляду ймовірності подальшого використання. На кожному кроці в словник вставляється нова фраза, яка є зчепленням (конкатенацію) однієї з фраз S словника, має найдовший збіг з рядком буфера, і символу s. Символ s є символом, наступним за рядком буфера, для якого знайдена співпадаюча фраза S. На відміну від сімейства LZ77, в словнику не може бути однакових фраз.

Кодер породжує тільки послідовність кодів фраз. Кожний код складається з номера (індексу) n "батьківської" фрази S, або префікса, і символу s.

На початку обробки словник порожній. Далі, теоретично, словник може рости нескінченно, тобто на його зростання сам алгоритм не накладає обмежень. На практиці досягши певного об'єму займаної пам'яті словник повинен очищатися повністю або частково.

Алгоритм не має формального обмеження на довжину рядка та відстань між однаковими рядками, як LZ77. Однак на практиці обмежують розмір словника, виходячи із вимог до пам`яті та швидкодії. При цьому виникає проблема переповнення словника, що може бути вирішена наступними способами:

- повністю очищувати словник після його заповнення;
- не додавати нові елементи у словник;
- заміняти новими елементами ті, які найрідше або найдавніше використовувались.

В базовій модифікації використовується перший підхід. Він є ефективним, якщо статистичні характеристики повідомлення періодично змінюються, в той час як другий підхід може бути більш ефективним, якщо статистичні характеристики лишаються незмінними. Третій підхід вимагає більших затрат на реалізацію, однак теоретично забезпечує кращу адаптацію до характеристик повідомлення порівняно з першими двома.

Приклад

І ще раз закодуємо рядок "кот_ломом_колол_слона" завдовжки 21 символ. Для LZ78 буфер, у принципі, не потрібен, оскільки достатньо легко так реалізувати пошук співпадаючої фрази максимальної довжини, що послідовність незакодованих символів буде видимою тільки один раз. Тому буфер показаний тільки з метою більшої наочності прикладу. Фразу з номером 0 зарезервуємо для позначення кінця стислого рядка, номером 1 задаватимемо порожню фразу словника.

Рядок вдалося закодувати за 13 кроків. Оскільки на кожному кроці видавався один код, стисла послідовність складається з 13 кодів. Можливе використовування 15 номерів фраз (від 0 до 14), тому для представлення n за допомогою кодів фіксованої довжини нам буде потрібно 4 біти. Тоді розмір стислого рядка рівний $13 \cdot (4+8) = 156$ бітам.

До прикладу, процес пошуку в словнику може відбуватись наступним чином: з буфера зчитується перший символ, і шукається елемент словника, що дорівнює цьому символу. Якщо такий елемент знайдений, то запам’ятовується його номер, з буфера зчитується наступний символ і далі в словнику шукається елемент з цих двох символів. Далі процес повторюється (додавання символу з буфера – пошук елемента – запам’ятовування його номера), поки пошук не закінчиться невдачею. Тоді поточний рядок, що шукався в словнику і не був знайдений, додається в словник, а на вихід кодера видається номер елемента, знайденого на попередньому кроці (тобто цієї самої строки, коротшої на один символ), та останній символ.

4.2 Алгоритм LZW

Базова модифікація LZ78, як і LZ77, передбачає видачу у вихідний потік наступного символа повідомлення незалежно від того, чи був знайдений рядок у словнику. Вдосконаленням LZ78, що усуває цей недолік, є метод LZW (W – від прізвища Welch). Результат кодування методом LZW містить лише посилання на словник і не містить символів вхідного потоку. Для цього перед початком кодування словник ініціалізується всіма можливими одно символьними рядками – тобто всіма символами алфавіту джерела, тому неможливою є ситуація, коли символу немає в словнику, яка спеціальним чином кодувалась в LZ78. Як і в базовій модифікації, на кожному кроці алгоритму в словнику шукається рядок максимальної довжини, що співпадає із поточним вмістом буферу, і на вихід передається її номер у словнику, після чого у словник додається новий рядок, утворений шляхом додавання до знайденого рядка останнього поточного символа. Далі початок буфера зміщується саме до цього символа, а не до наступного за ним, як в LZ78.

Декодер, отримавши номер елемента словника, копіює цей елемент у вихідний потік. Новий елемент у словник додається лише після того, як із вхідного потоку буде зчитаний наступний номер, оскільки цей новий елемент утворюється шляхом додавання до існуючого елемента словника першого символа наступного елемента. Таким чином, оновлення словника декодера відбувається аналогічним чином, як у кодера, але із запізненням на один крок.

4.3 Інші модифікації

Інші модифікації методу LZ78 відрізняються головним чином стратегією оновлення словника та способом його організації. Так в модифікації LZT (T – від Tischer) при заповненні словника нові елементи додаються замість тих, що найменше використовувались останнім часом, а в LZC (C – від Compress – назви утіліти ОС Unix, в якій було використано даний алгоритм) в процесі роботи алгоритму відбувається відслідковування поточного коефіцієнту стиснення, і оновлення словника відбувається тільки після того, як коефіцієнт починає знижуватись.

Модифікація LZFG (FG – від прізвищ Fiala та Greene) більше нагадує LZ77, оскільки посилання на словник являють собою сукупність довжин елементів і відстаней до них, причому елементи шукаються у щойно проглянутому фрагменті тексту (елементи видаляються із словника, коли відстань до них від поточної позиції перевищує задане значення, що в певному сенсі зближує його із алгоритмом LZ77 з його ковзаючим вікном).

Словник для методів LZ78 найбільш доцільно організовувати у вигляді дерева цифрового пошуку, де вузлам відповідають символи елементів словника, а кожен елемент утворюється при проходженні дерева від кореня до вузла, номер якого і є номером цього елемента у словнику. Це дозволяє представити словник

більш компактно, ніж при використанні простого списку усіх елементів, оскільки більшість елементів утворюються із уже існуючих шляхом додавання одного символу. При цьому також суттєво пришвидшується пошук: якщо при пошуку у списку необхідно проглядати всі його елементи або сортувати список після включення кожного нового елемента, то при пошуку у дереві проходяться лише ті вузли, що відповідають символам, наявним в даному елементі.

Словникові методи є однопрохідними, що сприяє їх застосуванню у системах передачі даних. Зокрема, LZW використовується в стандартах стиснення для модемів V.42bis та V.44.

Очевидно, що швидкість розкодування для алгоритмів сімейства LZ78 потенційно завжди менше швидкості для алгоритмів з ковзаючим вікном, оскільки них витрати по підтримці словника в правильному стані мінімальні. З іншого боку, для LZ78 і його нащадків, наприклад LZW, існують ефективні реалізації процедур пошуку і додавання фраз в словник, що забезпечує значну перевагу над алгоритмами сімейства LZ77 в швидкості стиснення.

Не дивлячись на відносну швидкість кодування LZ78, при грамотній реалізації алгоритму воно все ж таки повільніше за декодування, співвідношення швидкостей зазвичай складає 3:2.

Список архіваторів і компресорів

1. Info-ZIP group. Info-ZIP's portable Zip - 3 sources. http://www.infozip.org
2. Jung R. ARJ archiver. http://www.arjsoftware.com
3. Jung R. JAR archiver. ftp://ftp.elf.stuba.sk/pub/pc/pack/jar102x.exe
4. Lemke M. ACI archiver. http://www.winace.com
5. Microlog Cabinet Manager 2001 for Win9x/NT - Compression tool for .cab files. ftp://ftp.elf.stuba.sk/pub/pc/pack/cab2001.zip
6. Microsoft Corporation. Cabinet Software Development Tool. http://msdn.microsoft.com/library/en-us/dnsamples/cab-sdk.exe
7. Nico Mak Computing. WinZip archiver. http://www.winzip.com
8. Pavlov I. 7-Zip archiver. http://www.7-zip.org
9. PKWARE Inc. PKZIP archiver. ftp://ftp.elf.stuba.sk/pub/pc/pack/pk250dos.exe
10. Roshal E. RAR for Windows. http://www.rarsoft.com
11. Technelysium Pty Ltd. IMP archiver. ftp://ftp.elf.stuba.sk/pub/pc/pack/imp112.exe
12. Ziganshin B. ARJZ archiver. ftp://ftp.elf.stuba.sk/pub/pc/pack/arjz015.zip

Контрольні питання

1. Розкрийте ідею словникових методів.
2. Дайте загальну характеристику класичним алгоритмам Зіва-Лемпеля.
3. Які методи стиснення даних відносяться до словникових?
4. Охарактеризуйте алгоритм LZ77 та його модифікації.
5. Охарактеризуйте алгоритм LZ78 та його модифікації.
6. Якому методу належить ідея заміни строк символів на індекси строк деякого словника?
7. Назвіть словникові алгоритми, які використовують словники, що ковзають.
8. Назвіть словникові алгоритми, які не використовують словники, що ковзають, і в словник додають лише перспективні строки.
9. Назвіть словниковий алгоритм, який додає до кожного вказівника і символу однобітовий префікс.
10. Назвіть мловниковий алгоритм, який використовує ефективні процедури пошуку і додавання фраз до словника.
11. Назвіть словниковий алгоритм, в якому при заповненні словника нові елементи додаються замість тих, що найменше використовувалися останнім часом.
12. Назвіть словниковий алгоритм, в процесі роботи якого відслідковування поточного коефіцієнта стиснення і оновлення словника відбувається тільки після того, як коефіцієнт починає знижуватися.
13. Назвіть словниковий алгоритм, в якому посилання на словник являють собою сукупність довжин елементів і відстаней до них, причому елементи шукаються у щойно проглянутому фрагменті тексту.

ЛЕКЦІЯ 7

на тему: Стиснення даних із послідовностями однакових символів і сортуючі перетворення

У лекції розглядаються методи стиснення даних із послідовностями однакових символів: кодування довжин повторів (RLE), методи усунення констант (алгоритм заголовочного стиснення і метод BAP) та методи сортуючих перетворень: перетворення Барроуза – Уілера (BWT), частинні сортуючі перетворення (ST), метод стопки книг (MTF), сортування паралельних блоків (PBS) та ін.

План:

1. Методи стиснення даних із послідовностями однакових символів.
2. Методи сортуючих перетворень.

Для стиснення даних придумано безліч технік. Більшість з них комбінують декілька принципів стиснення для створення повноцінного алгоритму. Навіть хороші принципи, будучи скомбіновані разом, дають кращий результат. Більшість технік використовують принцип ентропійного кодування, але часто зустрічаються й інші — стиснення даних із послідовностями однакових символів, наприклад, кодування довжин серій (Run-Length Encoding) і сортуючі перетворення, наприклад, перетворення Барроуза-Уїлера (Burrows-Wheeler Transform).

1. Методи стиснення даних із послідовностями однакових символів

Типовим прикладом даних, у яких зустрічаються довгі послідовності однакових символів, є графічні файли в растровому форматі, особливо чорно-білі. Крім того, цей вид надлишковості може з’явитися внаслідок застосування методів стиснення, орієнтованих на деко реляцію символів повідомлення. Для усунення цього виду надлишковості існує ряд специфічних методів, застосування яких до даних, що не містять надлишковості даного виду, як правило, призводить не до стиснення, а до збільшення їх об’єму.

Методи стиснення даних із послідовностями однакових символів:

- ✓ кодування довжин повторів (серій);
- ✓ усунення констант:
  - упаковка бітів (класичний алгоритм),
  - алгоритм заголовочного стиснення,
  - метод BAP.

Переваги:

- ✓ ефективність досягається за рахунок економного кодування часто повторюваних значень;
- ✓ незалежність від обсягу даних;

- ✓ відрізняється простотою і високою швидкістю роботи.

Недоліки:

- ✓ повільне декодування;
- ✓ у середньому забезпечує недостатнє стиснення.

1.1 Кодування довжин повторів (RLE – run length encoding)

Це дуже простий алгоритм. Він замінює серії з двох або більше однакових символів числом, що позначає довжину серії, за яким йде сам символ. Корисний для сильно надлишкових даних, типу картинок з великою кількістю однакових пікселів, або в комбінації з алгоритмами типу BWT.

Приклад 1

На вході: AAABBCCCCDEEEEEEAAAAAAAAAAAAAAAAAAA

На виході: 3A2B4C1D6E20A

Суть методу полягає у заміні кожної послідовності однакових символів (серії) на код довжини цієї послідовності.

Якщо алфавіт джерела містить тільки два символи (наприклад, для чорно-білого зображення), то на виході кодера достатньо формувати тільки довжини послідовностей, що чергуються.

Приклад 2

Потрібно закодувати зображення, приведене на рис. 7.1.

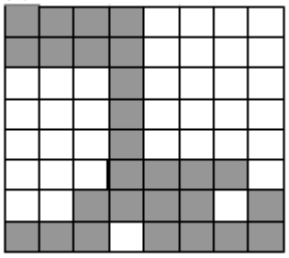

Рисунок 7.1 – Приклад чорно-білого зображення для кодування

Без застосування стиснення для кодування даного зображення необхідно 64 біти:

1111000011110000001000000010000000111100011110111101111.

Кодуючи довжини послідовностей нулів та одиниць, отримуємо:

4, 4, 4, 7, 1, 7, 1, 7, 1, 7, 4, 3, 4, 1, 4, 1, 4.

Навіть при кодуванні рівномірним кодом (три біти на символ) довжина стиснутого повідомлення складатиме 3*17=51 біт, що менше довжини вихідного повідомлення приблизно на 20%. Якщо ж застосувати після RLE статистичне кодування, наприклад, Хаффмана, то результуючу довжину повідомлення можна зменшити вдвічі порівняно з первинним.

Якщо алфавіт містить більше двох символів, то можливі наступні варіанти в залежності від характеру розподілу символів у повідомленні.

1. Якщо одиночні символи та послідовності різних символів зустрічаються дуже рідко, то кожна серія заміняється на два значення – довжина серії та символ, що повторюється (кожен одиничний символ також кодується як серія довжини 1).

Приклад 3

Повідомлення:

4 4 4 4 4 4 4 4 4 8 6 6 6 6 6 6 6 2 2 2 7 5 5 5

замінюється на

9 4 1 8 7 6 4 2 1 7 3 5.

Якщо одиночні символи зустрічаються порівняно часто, це може призвести до зниження ефективності кодування, оскільки кожен такий символ заміняється двома. Застосувавши такий метод до даних, які не містять серій, можна отримати замість стиснення збільшення об'єму даних – в найгіршому випадку в 2 рази.

2. Якщо порівняно часто зустрічаються послідовності різних символів, то серія заміняється сукупністю трьох значень:

- префікс, що вказує на необхідність розшифровки наступних двох символів як коду серії;
- довжина серії;
- символ, що повторюється.

Решта символів передаються із входу на вихід кодера без додаткового кодування. Якщо довжина серії менша 4, її недоцільно кодувати в префіксному вигляді, оскільки це не дає виграшу в об'ємі.

В якості префікса слід використовувати символ, що ніколи не може з`явитися у повідомленні. Якщо таких символів немає, то префіксом може бути символ, що має найменшу ймовірність. У випадку появи цього символа в повідомленні він кодується у префіксному вигляді.

Приклад 4

Нехай префікс – число 255, тоді

4 4 4 4 4 4 4 4 4 4 4 4 3 6 4 7 8 4 6 5 3 3 3 3 3 3 3 3 3 6 2 2 2 255 8 8 8 8

заміняється на

255 12 4 3 6 4 7 8 4 6 5 255 10 3 6 2 2 2 255 1 255 255 4 8

3.Якщо одиночні символи зустрічаються рідко, а послідовності різних символів зустрічаються рідше ніж серії, але можуть мати відносно велику довжину (що може призвести до збільшення надлишковості при застосуванні першого з описаних методів), у префіксному вигляді можна кодувати саме ці послідовності. Після префікса слід вказати довжину послідовності різних символів. В якості префікса доцільно використовувати 0, оскільки для серії довжина 0 неможлива.

Приклад 5

Нехай префікс – число 0, тоді

6666666654694625439855555555555444444446543234567 8

заміняється на

8 6 0 12 5 4 6 9 4 6 2 5 4 3 9 8 11 5 9 4 0 11 6 5 4 3 2 3 4 5 6 7 8

4. Якщо часто зустрічаються як довгі серії, так і довгі послідовності різних символів, то при досягненні довжиною серії деякого порогу всі наступні символи серії заміняються кодом їх кількості (причому якщо довжина дорівнює порогу, обов'язково передається код 0 для забезпечення правильного декодування).

Приклад 6

Нехай поріг складає 3 символи, тоді

66666666546946255555555555577744444444465432345678

заміняється на

6 6 6 5 5 4 6 9 4 6 2 5 5 5 9 7 7 7 0 4 4 4 6 6 5 4 3 2 3 4 5 6 7 8

В цьому прикладі, якщо після послідовності «7 7 7» не передати 0, декодер інтерпретував би наступний символ «4» як кількість додаткових символів «7», що призвело би до неправильного кодування.

Метод RLE застосовується, зокрема у форматах графічних файлів BMP, PCX та TIFF, як опція одного із режимів роботи (ECP) паралельного порта EOM (LPT), також, в поєднанні з кодуванням Хаффмана – в стандарті передачі факсимільних повідомлень CCITT Group 3.

1.2 Методи усунення констант

У загальному випадку, основний виграш при кодуванні числових даних може бути отриманий за рахунок стиснення нулів і економного кодування часто повторюваних значень, якими зазвичай є нуль і відсутнє значення (NULL). Такі часто повторювані значення називаються в цьому підпункті константами. Стиснення нулів часто виконується за допомогою упаковки бітів (bit paccing).

Послідовності констант при кодуванні довжин серій замінюються на трійки <прапор, константа, довжина серії>. Існують інші модифікації методу, але загальним недоліком є повільне декодування.

Усунення певної константи може бути реалізовано за допомогою бітової карти. При цьому фізично зберігаються тільки неконстантні значення. Місце знаходження констант визначається бітової картою, кожен розряд якої дорівнює, наприклад, 1, якщо у відповідній позиції знаходиться звичайне значення, і 0, якщо константа.

Була запропонована модифікація методу бітової карти, що забезпечує в середньому більш швидкий пошук в стислих даних. У цьому методі, названому "заголовочне стиснення", бітова карта перетворена в заголовочний вектор, в непарних позиціях якого записується число нестиснутих значень з накопиченням, а в парних - число пропущених констант з накопиченням.

Існує більш складна модифікація методу бітової карти, що дозволяє скоротити число звернень до зовнішньої пам'яті при декодуванні. Метод

названий ВАР, оскільки при стисканні використовується три об'єкти: бітовий вектор (bit vector, *BV*), адресний вектор (address vector, *AV*) і так званий фізичний вектор (physical vector, *PV*). У позиції бітового вектора записується 0, якщо поточний символ дорівнює константі, і 1, якщо навпаки. У фізичному векторі перераховуються всі неконстантні значення. Бітовий вектор розбивається на підвектори, кожен з яких незалежно стискається за допомогою коду Голомба і зберігається в окремому блоці або послідовності блоків пристрою зовнішньої пам'яті. В чарунці адресного вектора *AV(i)* зберігається відносна позиція у фізичному векторі останнього неконстантного елемента для підвектора з номером (*i*–1) . Перший елемент адресного вектора дорівнює нулю. Якщо в попередньому підвекторі одні константи, то в адресний вектор записується значення його попереднього елемента.

Адресний вектор також може бути стиснутий за допомогою диференціального кодування. При кодуванні і декодуванні на підставі адресного вектора визначається, який підвектор треба розпакувати, і декодується тільки він. Якщо адресний вектор досить малий, щоб поміститися в оперативну пам'ять, то під час декодування потрібно лише одне звернення до зовнішньої пам'яті.

Приклад 7

Якщо розмір підвектора дорівнює 5, і константа є "0", то для рядка

$S = \{1,0,0,4,0,\ 0,0,0,0,0,\ 0,0,8,0,0,\ 0,12,0,0,0,\ 0,17,0,0,20\}$

вміст векторів буде таким:

$BV = \{1,0,0,1,0,\ 0,0,0,0,0,\ 0,0,1,0,0,\ 0,1,0,0,0,\ 0,1,0,0,1\}$,

$PV = \{1,4,8,12,17,20\}$,

$AV = \{0,2,2,3,4\}$.

AV(5)=4, оскільки останній неконстатний елемент 4-го під вектора міститься в *PV(4)*. Це число 12.

Варіюючи параметри, можна задавати співвідношення між числом звернень, розміром адресного вектора, розміром блоку, максимальним ступенем стиснення і максимальним розміром даних.

2. Методи сортуючих перетворень

Це ще одна група універсальних методів стисненні даних, що містять надлишковість, зумовлену статистичними зв’язками між близько розташованими символами. Можна провести аналогію між сортуючими перетвореннями для символьних даних та ортогональними перетвореннями для даних аналогового походження: задача обох перетворень – усунення кореляції між елементами даних. Якщо в результаті ортогонального перетворення з наступним квантуванням можна отримати довгі послідовності нульових або однакових коефіцієнтів перетворення, то в результаті сортуючого перетворення отримуються довгі послідовності однакових символів (інформація при цьому не

втрачається). Такі послідовності легко стиснути методами, розглянутими раніше. Самі ж по собі перетворення не призводять до стиснення даних, а лише перетворюють їх до вигляду, зручного для стиснення.

Сортуючі перетворення:

- ✓ перетворення Барроуза-Уілера (BWT);
- ✓ частинні сортуючі перетворення (ST);
- ✓ метод стопки книг (MTF);
- ✓ сортування паралельних блоків (PBS);
- ✓ алгоритм знаходження оптимальних груп.

Переваги:

- ✓ як попередня операція дозволяє застосувати до отриманих даних прості методи стиснення;
- ✓ сортування дозволяє значно підвищити ступінь стиснення, особливо на текстових даних;
- ✓ декомпресія в 3-4 рази швидше стиснення.

Недоліки:

- ✓ ступінь стиснення сильно залежить від типу даних;
- ✓ вимагає досить багато пам'яті;
- ✓ для однозначного відновлення сортування структура повинна бути стійка, зі збереженням порядку однакових елементів;
- ✓ швидкість стиснення - відносно невисока;
- ✓ алгоритми знаходження оптимального угруповання працюють порівняно повільно.

2.1 Перетворення Барроуза-Уілера (BWT – Burrows-Wheeler Transform)

BWT-перетворення (Burrows-Wheeler Transform) - алгоритм для стиснення інформації (особливо текстів), заснований на перетворенні, відкритому в 1983 р. BWT є дивним алгоритмом. По-перше, незвичайно саме перетворення, відкрите в науковій області, далекій від архіваторів. По-друге, навіть знаючи BWT, не зовсім ясно, як його застосувати до стиснення інформації. По-третє, BWT перетворення надзвичайно просто. І, нарешті, сам BWT компресор складається з послідовності декількох розглянутих раніше алгоритмів і вимагає, тому, для своєї реалізації найрізноманітніших програмних навичок.

Вперше алгоритм цього перетворення та спосіб його застосування при стисненні даних було описано М. Барроузом та Д. Уілером у 1994 р. На той час алгоритми PPM мали обмежене практичне застосування, оскільки вимагали значних обчислювальних ресурсів, том уданий метод, який був не набагато повільніший за словникові методи, але забезпечував краще стиснення, і одночасно був значно швидшим за PPM, забезпечуючи не набагато гірше стиснення, став досить популярним.

Алгоритм BWT не стискує дані, але перетворює блок даних в формат, виключно відповідний для компресії.

Перш за все, слід відзначити одну з його особливостей. BWT оперує відразу цілим блоком даних. Тобто, йому заздалегідь відомі відразу всі елементи вхідного потоку або, принаймні, досить великого блоку. Це робить скрутним використання алгоритму в тих областях застосування, де потрібне стиснення даних "на льоту", символ за символом. В цьому відношенні BWT навіть більш вимогливий, ніж методи сімейства LZ, що використовують для стиснення ковзаюче вікно.

Слід зазначити, що можлива реалізація стиснення даних на основі BWT, оброблюючи дані послідовно по символам, а не по блокам. Але швидкісні характеристики програм, що використовують таку реалізацію, будуть дуже далекі від досконалості.

Перетворення Барроуза-Уілера пре впорядковує символи у блоці даних таким чином, що поряд опиняються символи, які зустрічаються в одному і тому ж контексті. Результат перетворення часто містить довгі послідовності однакових символів, які легко стиснути спеціальними методами (MTF або DC). Після цього доцільним є застосування статистичного методу стиснення, що усуває рештки надлишковості.

Алгоритм BWT містить наступні кроки:

1. Із повідомлення виділяється блок певної довжини (довжина блоку є параметром алгоритму і вибирається виходячи з вимог до пам’яті та швидкодії або на підставі визначення інтервалу стабільності статистичних характеристик повідомлення).

2. З отриманого блоку утворюється матриця всіх його циклічних перестановок.

3. Строки матриці сортуються в лексикографічному порядку (за алфавітом).

4. Виписуються символи останнього стовпчика матриці та запам’ятовується номер рядка, в якому міститься вихідне повідомлення.

Отже, перша фаза перетворення - виділення з безперервного потоку блоку даних. Далі, потрібно з отриманого блоку даних створити матрицю всіх можливих його циклічних перестановок. Першим рядком матриці буде початкова послідовність, другим рядком - вона ж, зрушена циклічно на один символ вліво і т.д.

Приклад 8

Розглянемо блок «ABABAC», тоді матриця буде мати вигляд:

ABABAC
BABACA
ABACAB
BACABA
ACABAB
CABABA

Після сортування отримується матриця:

ABABAC

ABACAB
ACABAB
BABACA
BACABA
CABABA
На наступному кроці отримаємо результат
CBBAAA, 0

Приклад 9

Розглянемо перетворення на прикладі рядка "абракадабра". Отримаємо наступну матрицю:

абракадабра
бракадабраа
ракадабрааб
акадабраабр
кадабраабра
адабраабрак
дабраабрака
абраабракад
браабракада
раабракадаб
аабракадабр

Помітимо в цій матриці вихідний рядок і відсортуємо всі рядки відповідно до лексикографічних порядків символів. Будемо вважати, що один рядок повинен знаходитися в матриці вище іншого в тому випадку, якщо в самій лівій з позицій, починаючи з якої рядки відрізняються, в цьому рядку знаходиться символ лексикографічно менший, ніж у іншого рядка. Іншими словами, слід впорядкувати рядки спочатку по першому символу, потім рядки, у яких перші символи рівні - по другому і т. д.

0 аабракадабр
1 абраабракад
2 абракадабра - вихідний рядок (повідомлення)
3 адабраабрак
4 акадабраабр
5 браабракада
6 бракадабраа
7 дабраабрака
8 кадабраабра
9 раабракадаб
10 ракадабрааб

Тепер залишився останній крок - виписати символи останнього стовпця і запам'ятати номер початкового рядка серед відсортованих. У нашому прикладі "рдакрааааббб", 2 - це результат, отриманий в результаті перетворення Барроуза-Уілера.

Пертворення Барроуза-Уілера є оберненим, тому із результату перетворення можна однозначно відновити вихідне повідомлення. Розглянемо алгоритм зворотнього перетворення для наведеного раніше прикладу 8.

Приклад 10

Оскільки відомо, що рядки матриці були відсортовані по алфавіту, то, відсортувавши по алфавіту рядок-результат, отримаємо перший стовпчик матриці:

A….C
A….B
A….B
B….A
B….A
C….A

Оскільки рядки матриці були отримані в результаті циклічного зсуву, можна однозначно відновити другий стовпчик матриці, сортуючи рядки, утворені останнім і першим символом (тобто рядки CA, BA, BA, AB, AB, AC). Отримуємо:

AB…C
AB…B
AC…B
BA…A
BA…A
CA…A

Далі сортуємо рядки, утворені останнім і першими двома символами (CAB, BAB, BAC, ABA, ABA, ACA):

ABA..C
ABA..B
ACA..B
BAB..A
BAC..A
CAB..A

Ще два аналогічних кроки дозволяють повністю відновити матрицю:

ABAB.C
ABAC.B
ACAB.B
BABA.A
BACA.A
CABA.A

та

ABABAC
ABACAB
ACABAB
BABACA

BACABA
CABABA

Залишилось вибрати із матриці строку, номер якої міститься в результаті перетворення.

Слід зауважити, що при практичній реалізації даного алгоритму зовсім немає необхідності зберігати в пам'яті всю матрицю перестановок (тоді наприклад, для перетворення 1-мегабайтного блоку знадобився б 1 Тбайт пам'яті). Достатньо визначити вектор номерів рядків після першого впорядкування.

На кожному наступному кроці буде отриманий аналогічний вектор. Тому для отримання вихідного рядка достатньо визначити порядок отримання символів першого стовпчика із символів останнього.

Як уже було сказано, перетворення Барроуза-Уілера призводить лише до пере впорядкування символів тексту без зміни його довжини. Для отримання власне стиснення необхідно застосовувати наступну послідовність методів:

1) кодування довжин повторів (необов'язково)
2) перетворення Барроуза-Уілера
3) усунення довгих послідовностей однакових символів методом MTF або DC
4) кодування довжин повторів (необов'язково)
5) арифметичне кодування або кодування Хаффмана.

Коефіцієнт стиснення при цьому наближається до значень, що отримуються для методів PPM.

2.2 Частинні сортуючі перетворення (ST)

Крім перетворення Барроуза-Уілера у вищенаведеній схемі можуть бути використані і т.зв. частинні сортуючі перетворення. Від BWT вони відрізняються лише тим, що сортування матриці циклічних перестановок здійснюється не по всій довжині рядка матриці, а по фіксованій кількості початкових символів. Ця кількість називається порядком сортування.

Приклад 11

Розглянемо приклад застосування сортуючих перетворень різних порядків повідомлення «BCBACB».

Таблиця 4.1 - Застосування сортуючих перетворень

Матриця циклічних перестановок	ST(1)	ST(2)	ST(3)
BCBACB	**A**CBBCB	**AC**BBCB	**ACB**BCB
BBCBAC	**B**CBACB	**BA**CBBC	**BAC**BBC
CBBCBA	**B**BCBAC	**BB**CBAC	**BBC**BAC
ACBBCB	**B**ACBBC	**BC**BACB	**BCB**ACB
BACBBC	**C**BBCBA	**CB**BCBA	**CBA**CBB
CBACBB	**C**BACBB	**CB**ACBB	**CBB**CBA

При перетворенні першого порядку сортування здійснюється тільки за першим символом. Рядки з однаковим першим символом у результаті перетворення ідуть в тому ж порядку, що у матриці циклічних перестановок, тому другі, треті і т.д. символи не впорядковані за алфавітом. Фактично це означає, що для сортуючого перетворення 1-го порядку враховуються головним чином контексти довжини 1, для 2-го порядку – контексти довжини 2 і т.д. Тому в порівнянні з перетворенням Барроуза-Уілера ефективність сортуючих перетворень буде меншою лише в тому випадку, коли у даних, що стискаються, наявні повторювані фрагменти довжиною більше порядку сортування.

Частинні сортуючі перетворення виконуються швидше, ніж перетворення Барроуза-Уілера, однак зворотнє перетворення виконується дещо довше, особливо для невисоких порядків сортування, оскільки з'являється велика кількість однакових контекстів і потрібні додаткові затрати часу на наліз і усунення неоднозначності при декодування.

Легко помітити, що відмінність між частинним сортуючим перетворенням і перетворенням Барроуза - Уілера можна побачити тільки при сортуванні стійких контекстів довше порядку перетворення ST. У методі BWT в цьому випадку триває процес порівняння символів, а в ST- порівняння символів припиняється і вище розташовується той рядок, перший символ якої зустрівся у вхідному блоці раніше. Отже, ті дані, в яких зустрічаються довгі повтори, більш ефективно стискуються перетворенням Барроуза - Уілера. Наприклад, типові текстові файли англійською мовою втрачають в стисненні близько 5% при виконанні сортування за чотирма символами.

З іншого боку, частинне сортуюче перетворення не вимагає повного сортування всіх рядків матриці і вільно від тих проблем, які виникають при перетворенні дуже надлишкових даних з використанням повного сортування. Як правило, дане перетворення виконується швидше, ніж BWT.

Але при виконанні зворотного перетворення спостерігається інша картина. Якщо в разі BWT воно виконується легко і просто, то для відновлення вихідних даних після частинного сортуючого перетворення необхідно докласти додаткових зусиль. А саме вести облік кількості однакових контекстів. І чим порядок перетворення більше, тим потрібно більше часу на підрахунок.

2.3 Метод стопки книг (MTF – move to front)

Саме по собі це перетворення не призводить до стиснення даних, а лише перетворює надлишковість довгих послідовностей символів у надлишковість різноймовірних символів, яка в подальшому легко може бути усунена методами статистичного кодування.

Суть алгоритму полягає в постійному пере впорядкуванні алфавіту символів, що може здійснюватись синхронно кодером і декодером (цим алгоритм MTF нагадує адаптивні статистичні методи стиснення). При надходженні наступного символа на вихід кодера передається номер символу в поточному алфавіті, після чого цей символ переміщується на початок алфавіту, а решта символів в алфавіті відповідно зсувається.

Назва методу пояснюється аналогією із стопкою книг, із якої дістаються потрібні книги і після використання кладуться на верх стопки. Таким чином, через деякий час книги, що найчастіше використовуються, опиняються вгорі.

В застосуванні до символів повідомлення це означає, що на вихід частіше будуть передаватись менші номери, відповідно їх можна кодувати меншою кількістю біт. Якщо до початку кодування однозначно пов'язати номер символа в алфавіті з деяким нерівномірним кодом і передати на вихід кодера одразу цей код, це призведе до стиснення даних.

Приклад 12

Нехай алфавіт повідомлення містить 4 символи: *X={A,B,C,D}* та необхідно закодувати повідомлення «CCCCBBBADDDDAAAB». Всі символи в цьому повідомленні мають однакову ймовірність, тому кодування з врахуванням лише ймовірностей появи символів означає кодування рівномірним кодом (2 біт/символ) і призведе до довжини повідомлення L=2*16=32 біт.

Застосуємо алгоритм MTF (див. табл. 4.2).

Таблиця 4.2 - Застосування алгоритму MTF

Символ повідомлення	Поточний алфавіт				Вихідний символ
	0	1	2	3	
c	A	B	C	D	2
c	C	A	B	D	0
c	C	A	B	D	0
c	C	A	B	D	0
b	C	A	B	D	2
b	B	C	A	D	0
b	B	C	A	D	0
a	B	C	A	D	2
d	A	B	C	D	3
d	D	A	B	C	0
d	D	A	B	C	0
d	D	A	B	C	0
a	D	A	B	C	1
a	A	D	B	C	0
a	A	D	B	C	0
B	A	D	B	C	2

У вихідному повідомленні «20002002300010002» ймовірності символів розділені нерівномірно. Якщо використати для їх кодування код Хаффмана 0 – 0, 2 – 10, 1 – 110, 3 – 111, то результуюче повідомлення «10000100010111000110010» має довжину L_{CT}=24 біт, тобто коефіцієнт стиснення складає $К_{С}Т$=1.25.

Спочатку алгоритм MTF був запропонований як самостійний алгоритм стиснення, однак зараз він переважно використовується як один із етапів стиснення з використанням сортуючих перетворень. Сам по собі цей алгоритм

не усуває послідовності однакових символів (кожна така послідовність перетворюється на послідовність нулів), тому обов'язковим є застосування після нього статистичних методів стиснення.

2.4 Інші методи сортуючих перетворень

До інших методів сортуючих перетворень відносяться:

- ✓ сортування паралельних блоків (PBS);
- ✓ алгоритм знаходження оптимальних груп.

Відомо, що перестановка стовпців може суттєво покращити стиснення при порядковій обробці. Для знаходження оптимальних груп запропоновано кодувати невеликий навчальний фрагмент даних і використовувати статистику для рішення задачі з допомогою динамічного планування (програмування). Для прискорення оптимізації n стовпців розбивається на k попарних груп, що не перетинаються. Це дозволяє отримати оптимальний результат при часовій складності $O(nk)$ замість $O(n^2)$. Втрати в результуючому ступені стиснення дуже малі в тому випадку, коли n і k відрізняються тільки в рази, тобто $n/k<10$.

Контрольні питання

1. Назвіть метод, що замінює кожну послідовність однакових символів (серій) на код довжини цієї послідовності
2. За рахунок чого досягається стиснення в RLE?
3. Назвіть метод, що фізично зберігає тільки неконстантні значення, а місце знаходження констант визначається бітовою картою.
4. Назвіть метод, що використовує три об'єкти: бітовий, адресний та фізичний вектори.
5. Згідно алгоритму кодування довжин повторів (RLE), якщо одиночні символи та послідовності різних символів зустрічаються дуже рідко, то на що замінюється кожна серія?
6. Згідно алгоритму кодування довжин повторів (RLE), якщо порівняно часто зустрічаються послідовності різних символів, то на що замінюється серія?
7. Що вказується, згідно алгоритму кодування довжин повторів (RLE), якщо одиночні символи зустрічаються рідко, а послідовності різних символів зустрічаються рідше ніж серії, але можуть мати відносно велику довжину?
8. Згідно алгоритму кодування довжин повторів (RLE), якщо часто зустрічаються як довгі серії, так і довгі послідовності різних символів, у випадку досягнення довжиною серії деякого порогу на що замінюються всі наступні символи?
9. Що записується у позиції бітового вектора, згідно методу BAP?
10. Що записується у позиції фізичного вектора, згідно методу BAP?
11. Що записується у позиції адресного вектора, згідно методу BAP?
12. Назвіть перетворення, що перевпорядковує символи у блоці даних таким чином, що поряд опиняються символи, які зустрічаються в одному і тому ж контексті

13. Назвіть перетворення, в якому сортування матриці циклічних перестановок здійснюється не по всій довжині рядка матриці, а по фіксованій кількості початкових символів

14. Назвіть перетворення, суть якого полягає в постійному перевпорядкуванні алфавіту символів (при надходженні наступного символу в поточному алфавіті, після чого цей символ переміщується на початок алфавіту, а решта символів в алфавіті відповідно зсувається).

15. Охарактеризуйте перетворення Барроуза-Уілера (BWT**).**

16. Назвіть перетворення, суть якого полягає в сортуванні елементів вхідного блоку і розкладає їх в декілька вихідних.

ЛЕКЦІЯ 8

на тему: Методи контекстного моделювання (частина 1)

У лекції розглядаються основні терміни методів контекстного моделювання, приводяться класифікація і короткий зміст стратегій моделювання, описуються основні поняття, види і механізми контекстного моделювання.

План:

1. Концепція універсального моделювання та кодування
2. Класифікація стратегій моделювання
3. Основні терміни методів контекстного моделювання
4. Види контекстного моделювання

1. Концепція універсального моделювання та кодування

Використання методів контекстного моделювання для стиснення даних спирається на парадигму стиснення за допомогою "універсальних моделювання і кодування" (universal modelling and coding), запропоновану Ріссаненом (Rissanen) і Ленгдоном (Langdon) в 1981 році. Відповідно до даної ідеї процес стиснення складається з двох самостійних частин:

- моделювання;
- кодування.

Під моделюванням розуміється побудова моделі інформаційного джерела, що породило дані, що стискаються, а під кодуванням - відображення оброблюваних даних в стислу форму представлення на підставі результатів моделювання. "**Кодувальник"** створює вихідний потік, що є компактною формою представлення оброблюваної послідовності, на підставі інформації, що поставляється йому "**моделювальник"**.

Схема процесу стиснення даних відповідно до концепції універсальних моделювання і кодування представлена на рис. 8.1.

Слід замітити, що поняття "кодування" часто використовують в широкому значенні для позначення всього процесу стиснення, тобто включаючи моделювання в даному нами визначенні. Таким чином, необхідно розрізняти поняття кодування в широкому значенні (весь процес) і у вузькому (генерація потоку кодів на підставі інформації моделі). Поняття "статистичне кодування" також використовується, часто з сумнівною коректністю, для позначення того або іншого рівня кодування. Щоб уникнути плутанини ряд авторів застосовує термін "ентропійне кодування" для кодування у вузькому значенні. Це найменування далеко від досконалості і зустрічає цілком обгрунтовану критику. Далі в цій темі

процес кодування в широкому значенні іменуватимемо "кодуванням", а у вузькому значенні - "статистичним кодуванням", або "власне кодуванням".

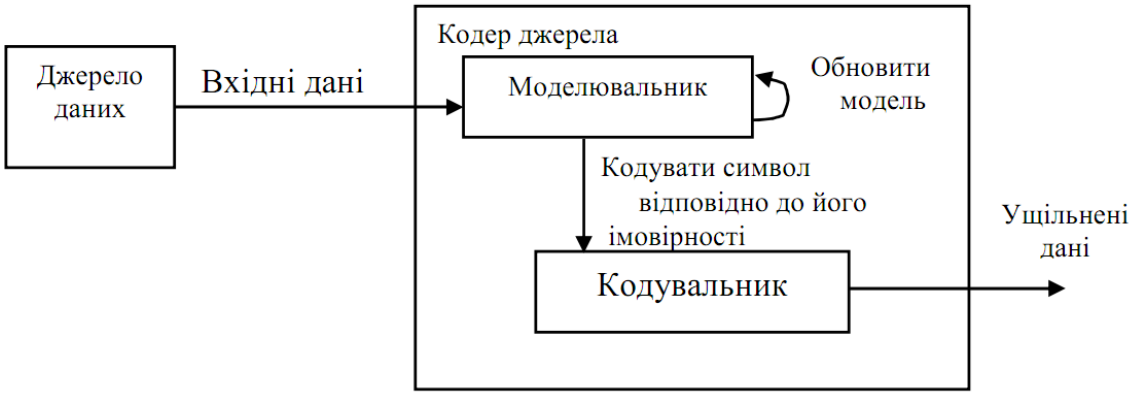

Рисунок 8.1 – Схема процесу стиснення даних відповідно до концепції універсальних моделювання і кодування

З теореми Шеннона про кодування джерела відомо, що символ s_i, ймовірність появи якого дорівнює $p(s_i)$, вигідніше всього представляти $-\log_2 p(s_i)$ бітами, при цьому середня довжина кодів може бути обчислена по формулі, що приводилася раніше. Практично завжди істинна структура джерела прихована, тому необхідно будувати **модель джерела**, яка дозволила б нам в кожній позиції вхідної послідовності знайти оцінку $q(s_i)$ ймовірності появи кожного символу s_i алфавіту вхідної послідовності.

Оцінка ймовірності символів при моделюванні проводиться на підставі відомої статистики і, можливо, апріорних припущень, тому часто говорять про задачу статистичного моделювання. Можна сказати, що моделювальник передбачає ймовірність появи кожного символу в кожній позиції вхідного рядка, звідси ще одне найменування цього компоненту - "провісник", або "предиктор" (від "predictor"). На етапі статистичного кодування виконується заміщення символу s_i з оцінкою ймовірності появи кодом завдовжки $-\log_2 q(s_i)$ бітів.

Приклад 1.

Припустимо, що ми стискаємо послідовність символів алфавіту {'0','1'}, породжену джерелом без пам'яті, і ймовірність генерації символів наступні: p('0')= 0.4, p('1')= 0.6. Хай наша модель дає такі оцінки ймовірності: q('0')= 0.35, q('1')= 0.65. Ентропія H джерела дорівнює

$$– p(‘0’) \log_2 p(‘0’) – p(‘1’) \log_2 p(‘1’) = –0.4 \log_2 0.4 –0.6 \log_2 0.6 = 0.971 \text{ біта.}$$

Якщо підходити формально, то "ентропія" моделі виходить рівною

$$– q(‘0’) \log_2 q(‘0’) – q(‘1’) \log_2 q(‘1’) = –0.35 \log_2 0.35 –0.65 \log_2 0.65 = 0.934 \text{ біта.}$$

Здавалося б, що модель забезпечує краще стиснення, ніж це дозволяє формула Шеннона. Але істинна ймовірність появи символів не змінилася! Якщо виходити з ймовірності p, то '0' слід кодувати $– \log_2 0.4 \approx 1.332$, а для '1' потрібно

відводити $-\log_2 0.6 \approx 0.737$ біта. Для оцінок ймовірності q ми маємо $-\log_2 0.35 \approx$ ≈ 1.515 біта і $-\log_2 0.65 \approx 0.621$ біта відповідно. При кожному кодуванні на підставі інформації моделі у разі '0' ми втрачатимемо 1.515 – 1.322 = 0.193 біти, а у разі '1' вигравтимемо 0.737 – 0.621 = 0.116 біти. З урахуванням ймовірності появи символів середній програш при кожному кодуванні складе 0.4*0.193 – 0.6*0.116 = = 0.008 біти.

Висновок: чим точніше оцінка ймовірності появи символів, тим більше коди відповідають оптимальним, тим краще стиснення.

Правильність декодування забезпечується використовуванням точно такої ж моделі, що була застосована при кодуванні. Отже, при моделюванні для стиснення даних не можна користуватися інформацією, яка невідома декодеру.

Усвідомлення подвійної природи процесу стиснення дозволяє здійснювати декомпозицію задач компресії даних з складною структурою і нетривіальними взаємозалежностями, забезпечувати певну самостійність процедур, зосереджувати більше уваги на деталях реалізації конкретного елементу.

Задача статистичного кодування була в цілому успішно вирішена до початку 1980-х років. Арифметичний кодер дозволяє згенерувати стислу послідовність, довжина якої звичайно всього лише на десяті частки відсотка перевищує теоретичну довжину. Більш того, використання сучасної модифікації арифметичного кодера - інтервального кодера - дозволяє здійснювати власне кодування дуже швидко. Швидкість статистичного кодування складає мільйони символів в секунду на сучасних ПК.

У світлі вищесказаного, підвищення точності моделей є, фактично, єдиним способом істотного поліпшення стиснення.

2. Класифікація стратегій моделювання

Перед розглядом контекстних методів моделювання слід сказати про класифікацію стратегій моделювання джерела даних за способом побудови і оновлення моделі. Виділяють чотири варіанти моделювання:

- статичне;
- напівадаптивне;
- адаптивне (динамічне);
- блоково-адаптивне.

При статичному моделюванні для будь-яких оброблюваних даних використовується одна і та ж модель. Інакше кажучи, не проводиться адаптація моделі до особливостей даних, що стискаються. Опис наперед побудованої моделі зберігається в структурах даних кодера і декодера; таким чином досягається однозначність кодування, з одного боку, і відсутність необхідності в явній передачі моделі, з іншого. Недолік підходу також очевидний: ми можемо одержувати погане стиснення і навіть збільшувати розмір представлення, якщо оброблювані дані не відповідають вибраній моделі. Тому така стратегія

використовується тільки в спеціалізованих додатках, коли тип даних, що стискаються, незмінний і наперед відомий.

Напівадаптивне стиснення є розвитком стратегії статичного моделювання. В цьому випадку для стиснення заданої послідовності вибирається або будується модель на підставі аналізу саме оброблюваних даних. Зрозуміло, що кодер повинен передавати декодеру не тільки закодовані дані, але і опис використаної моделі. Якщо модель вибирається з наперед створених і відомих як кодеру, так і декодеру, то це просто порядковий номер моделі. Інакше, якщо модель була набудована або побудована при кодуванні, то необхідно передавати або значення параметрів настройки, або модель повністю. В загальному випадку напівадаптивний підхід дає краще стиснення, ніж статичний, оскільки забезпечує пристосування до природи оброблюваних даних, зменшуючи ймовірність значної різниці між прогнозами моделі і реальною поведінкою потоку даних.

Адаптивне моделювання є природною протилежністю статичної стратегії. У міру кодування модель змінюється по заданому алгоритму після стиснення кожного символу. Однозначність декодування досягається тим, що, по-перше, спочатку кодер і декодер мають ідентичну і звичайно дуже просту модель і, по-друге, модифікація моделі при стисненні і розтисненні здійснюється однаковим чином. Досвід використовування моделей різних типів показує, що адаптивне моделювання є не тільки елегантною технікою, але і забезпечує, принаймні, не гірше стиснення, ніж напівадаптивне моделювання. Зрозуміло, що якщо стоїть задача створення "універсального" компресора для стиснення даних несхожих типів, то адаптивний підхід є природним вибором розробника.

Блоково-адаптивне моделювання можна розглядати як окремий випадок адаптивної стратегії (або навпаки, що суті справи не міняє).

Залежно від конкретного алгоритму оновлення моделі, оцінки ймовірності символів, методу статистичного кодування і самих даних зміна моделі після обробки кожного символу може бути пов'язана з наступними неприємностями:

- втрата стійкості оцінок, якщо дані "зашумлені", або є значні локальні зміни статистичних взаємозв'язків між символами оброблюваного потоку; інакше кажучи, занадто швидка, "агресивна" адаптація моделі може приводити до погіршення точності оцінок;
- великі обчислювальні витрати на оновлення моделі (як приклад - у разі адаптивного кодування по алгоритму Хаффмана);
- великі витрати пам'яті для зберігання структур даних, що забезпечують швидку модифікацію моделі.

Тому оновлення моделі може виконуватися після обробки цілого блоку символів, в загальному випадку змінної довжини. Для забезпечення правильності розтиснення декодер повинен виконувати таку ж послідовність дій по оновленню моделі, що і кодер, або кодеру необхідно передавати разом із стислими даними інструкції по модифікації моделі. Останній варіант достатньо часто

використовується при блоково-адаптивному моделюванні для прискорення процесу декодування в збиток коефіцієнту стиснення.

Зрозуміло, що приведена класифікація є до деякої міри абстрактною, і на практиці часто використовують гібридні схеми.

3. Основні терміни методів контекстного моделювання

Отже, нам необхідно вирішити задачу оцінки ймовірності появи символів в кожній позиції оброблюваної послідовності. Для того, щоб декодування відбулося без втрат, ми можемо користуватися тільки тією інформацією, яка повною мірою відома як кодеру, так і декодеру. Звичайно це означає, що оцінка ймовірності чергового символу повинна залежати тільки від властивостей вже обробленого блоку даних.

Найпростіший спосіб оцінки реалізується за допомогою напівадаптивного моделювання і полягає в попередньому підрахунку безумовної частоти появи символів в блоці, що стискається. Одержаний розподіл ймовірності використовується для статистичного кодування всіх символів блоку. Якщо, наприклад, таку модель застосувати для стиснення тексту українською або російською мовами, то в середньому на кодування кожного символу буде витрачено приблизно 4.5 біти. Це значення є середньою довжиною кодів для моделі, що базується на використовуванні безумовного розподілу ймовірності букв в тексті. Помітимо, що вже в цьому простому випадку досягається ступінь стиснення 1.5 по відношенню до тривіального кодування, коли всім символам призначаються коди однакової довжини. Дійсно, розмір алфавіту і українського і російського тексту перевищує 64, але менше 128 знаків (рядкові і заголовні букви, розділові знаки, пропуск), що вимагає 7-бітових кодів.

Аналіз поширених типів даних - наприклад, тих же текстів на природних мовах, - виявляє сильну залежність ймовірності появи символів від безпосередньо ним передуючих. Інакше кажучи, велика частина даних, з якими ми стикаємося, породжується джерелами з пам'яттю. Припустимо, нам відомо, що блок, що стискається, є текстом українською мовою. У разі аналізу відразу декількох слів, якщо попередній рядок рівний "Ще не вмерла України", то продовженням явно буде "ні слава, ні воля". Отже, облік залежності частоти появи символу (в загальному випадку - блоку символів) від попередніх повинен давати більш точні оцінки і, кінець кінцем, краще стиснення. Дійсно, у разі посимвольного кодування при використовуванні інформації про один безпосередньо попередній символ досягається середня довжина кодів в 3.6 біти для українських і російських текстів, при обліку двох останніх - вже порядка 3.2 бітів. В першому випадку моделюються умовні розподіли ймовірності символів, залежні від значення рядка з одного попереднього символу, в другому - залежні від рядка з двох попередніх символів.

Цікаво, що моделі, що оперують безумовними частотами і частотами залежно від одного попереднього символу, дають приблизно однакові результати для всіх європейських мов (за виключенням, мабуть, самих екзотичних) - 4.5 і 3.6 біти відповідно.

Поліпшення стиснення при обліку попередніх елементів (пікселів, семплів, відліків, чисел) наголошується і при обробці даних інших поширених типів: об'єктних файлів, зображень, аудіозаписів, таблиць чисел.

Під **контекстним моделюванням** розумітимемо оцінку ймовірності появи символу (елементу, пікселя, семпла і навіть набору якісно різних об'єктів) залежно від безпосередньо йому передуючих, або контексту.

Помітимо, що в побуті поняття "контекст" звичайно використовується в глобальному значенні - як сукупність символів (елементів), що оточують поточний оброблюваний. Це контекст в широкому значенні. Виділяють також "лівосторонні" і "правосторонні" контексти, тобто послідовності символів, що безпосередньо примикають до поточного символу зліва і справа відповідно. Тут і далі під контекстом розумітимемо саме класичний лівобічний: так, наприклад, для останнього символу 'о' послідовності "...молоко..." контекстом є "...молок".

Якщо довжина контексту обмежена, то такий підхід називатимемо контекстним моделюванням обмеженого порядку (finite-context modeling), при цьому під порядком розуміється максимальна довжина контекстів N, що використовуються. Наприклад, при моделюванні порядка 3 для останнього символу 'о' в послідовності "...молоко..." контекстом максимальної довжини 3 є рядок "лок". При стисненні цього символу під "поточними контекстами" можуть розумітися "лок", "ок", "к", а також порожній рядок "". Всі ці контексти довжини від N до 0 назвемо активними контекстами в тому значенні, що при оцінці символу може бути використана накопичена для них статистика.

Через об'єктивні причини - обмеженість обчислювальних ресурсів - техніка контекстного моделювання саме обмеженого порядку одержала найбільший розвиток і розповсюдження, тому далі під контекстним моделюванням розумітимемо саме її. Подальший виклад також враховує специфіку того, що контекстне моделювання практично завжди застосовується як адаптивне.

Оцінки ймовірності при контекстному моделюванні будуються на підставі звичайних лічильників частот, пов'язаних з поточним контекстом. Якщо ми обробили рядок "абсабвбабс", то для контексту "аб" лічильник символу 'с' рівний двом (говорять, що символ 'с' з'явився в контексті "аб" двічі), символу 'в' - одиниці. На підставі цієї статистики можна стверджувати, що ймовірність появи 'с' після "аб" дорівнює 2/3, а ймовірність появи 'в' - 1/3, тобто оцінки формуються на основі вже проглянутої частини потоку.

У загальному випадку для кожного контексту кінцевої довжини $o \leq N$, що зустрічається в оброблюваній послідовності, створюється контекстна модель (КМ). Будь-яка КМ включає лічильники всіх символів, які зустрілися у

відповідному їй контексті, тобто відразу після рядка контексту. Після кожної появи символу s в даному контексті проводиться збільшення значення лічильника символу s у відповідній контексту КМ. Звичайно лічильники ініціалізувалися нулями. На практиці лічильники створюються у міру появи в заданому контексті нових символів, тобто лічильників жодного разу не бачених в заданому контексті символів просто не існує.

Під порядком КМ розумітимемо довжину відповідного їй контексту. Якщо порядок КМ рівний 0, то позначатимемо таку КМ як "КМ(0)".

Окрім звичайних КМ, часто використовують контекстну модель мінус першого порядку КМ(-1), привласнюючу однакову ймовірність всім символам алфавіту потоку, що стискається.

Зрозуміло, що для нульового і мінус першого порядку контекстна модель одна, а КМ більшого порядку може бути декілька, аж до q^N, де q - розмір алфавіту оброблюваної послідовності. КМ(0) і КМ(-1) завжди активні.

Часто говорять про "батьківські" і "дочірні" контексти. Для контексту "к" дочірніми є "ок" і "лк", оскільки вони утворені зчепленням (конкатенацією) одного символу і контексту "к". Аналогічно, для контексту "лок" батьківським є контекст "ок", а контекстами-предками - "ок", "к", "". Очевидно, що "порожній" контекст "" є предком для всіх.

Сукупність КМ утворює модель джерела даних. Під порядком моделі розуміється максимальний порядок КМ, що використовуються.

4. Види контекстного моделювання

Приклад обробки рядка "абсабвбабс" ілюструє відразу дві проблеми контекстного моделювання:

- як вибирати відповідний контекст (або контексти) серед активних з метою отримання більш точної оцінки, адже поточний символ може краще передбачатися не контекстом другого порядку "аб", а контекстом першого порядку "б";
- як оцінювати ймовірність символів, що мають нульову частоту (наприклад, 'г').

Вище були приведені цифри, відповідно до яких при збільшенні довжини контексту, що використовується, стиснення даних поліпшується. На жаль, при кодуванні блоків типової довжини - одиниці мегабайтів і менше - це справедливо тільки для невеликих порядків моделі, оскільки статистика для довгих контекстів нагромаджується повільно. При цьому також слід враховувати, що більшість реальних даних характеризується неоднорідністю, нестабільністю сили і виду статистичних взаємозв'язків, тому "стара" статистика контекстно-залежних частот появи символів малокорисна або навіть шкідлива. Тому моделі, що будують оцінку тільки на підставі інформації КМ максимального порядку N, забезпечують

порівняно низьку точність прогнозу. Крім того, зберігання моделі великого порядку вимагає багато пам'яті.

Якщо в моделі використовуються для оцінки тільки КМ(N), то іноді такий підхід називають "чистим" (pure) контекстним моделюванням порядку N. Через вищезгаданий недолік "чисті" моделі представляють звичайно тільки науковий інтерес.

Дійсно, файли, що реально використовуються, звичайно мають порівняно невеликий розмір, тому для поліпшення їх стиснення необхідно враховувати оцінки ймовірності, одержувані на підставі статистики контекстів різних довжин. Техніка об'єднання оцінок ймовірності, відповідних окремим активним контекстам, в одну оцінку називається змішуванням (blending). Відомо декілька способів виконання змішування.

Розрізняють моделі з повним змішуванням (fully blended), коли прогноз визначається статистикою КМ всіх порядків, що використовуються, і з частковим змішуванням (partially blended) – в іншому випадку.

Приклад 2.

Розглянемо процес оцінки відзначеного на рис. 8.2 стрілкою символу 'л', що зустрівся в блоці "молочное_молоко". Вважаємо, що модель працює на рівні символів.

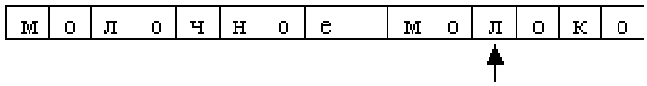

Рисунок 8.2 – Процес оцінки символу 'л'

Хай ми використовуємо контекстне моделювання порядка 2 і робимо повне змішування оцінок розподілів ймовірності в КМ другого, першого і нульового порядків з вагами 0.6, 0.3 і 0.1. На початку кодування в КМ(0) створюються лічильники для всіх символів алфавіту {'м', 'о', 'л', 'ч', 'н', 'е', '_', 'к'} і ініціалізуються одиницею; лічильник символу після його обробки збільшується на 1.

Для поточного символу 'л' є контексти "мо", "о" і порожній (нульового порядку). На даний момент для них накопичена певна статистика, наведена в таблиці 8.1.

Таблиця 8.1 – Накопичена статистика для символу 'л'

Символи	**Частоти**	**'м'**	**'о'**	**'л'**	**'ч'**	**'н'**	**'е'**	**' _ '**	**'к'**
КМ порядка 0 (контекст " ")	Частоти	3	5	2	2	2	2	2	1
	Накоплені частоти	3	8	10	12	14	16	18	19
КМ порядка 1 (контекст "о")	Частоти	-	-	1	1	-	1	-	-
	Накоплені частоти	-	-	1	2	-	3	-	-
КМ порядка 2("мо")	Частоти	-	-	1	-	-	-	-	-
	Накоплені частоти	-	-	1	-	-	-	-	-

Тоді оцінка ймовірності для символу 'л' буде дорівнювати

$$q('л') = 0.1 \cdot \frac{2}{19} + 0.3 \cdot \frac{1}{3} + 0.6 \cdot \frac{1}{1} = 0,71$$

У загальному випадку, для однозначного кодування символу 'л' таку оцінку необхідно виконати для всіх символів алфавіту. Дійсно, з одного боку, декодер не знає, чому рівний поточний символ, з другого боку, оцінка ймовірності не гарантує унікальність коду, а лише задає його довжину. Тому статистичне кодування виконується на підставі накопиченої частоти. Наприклад, якщо кодувати на підставі статистики тільки нульового порядку, то існує взаємно однозначна відповідність між накопиченими частотами з діапазону (8,10] і символом 'л', що не має місця у випадку просто частоти (частоту 2 мають ще 4 символи). Зрозуміло, що аналогічні властивості залишаються в силі і у разі оцінок, одержуваних частковим змішуванням.

Очевидно, що успіх використання змішування залежить від способу вибору вагів w(o). Простий шлях полягає у використовуванні заданого набору фіксованих вагів КМ різних порядків при кожній оцінці. Альтернативою є адаптація вагів у міру кодування. Це може полягати в додаванні все більшої значущості КМ все більших порядків або, скажімо, спробі вибрати якнайкращу вагу на підставі певних статистичних характеристик останнього обробленого блоку даних. Але так історично склалося, що реальний розвиток одержали методи неявного зважування. Це пояснюється в першу чергу їх меншою обчислювальною складністю.

Техніка неявного зважування пов'язана з введенням допоміжного символу відходу (escape). Символ відходу є квазісимволом і не повинен належати алфавіту послідовності, що стискається. Фактично, він використовується для передачі декодеру вказівок кодера. Ідея полягає в тому, що якщо КМ, що використовується, не дозволяє оцінити поточний символ (його лічильник рівний 0 в цій КМ), то на вихід посилається закодований символ відходу і проводиться спроба оцінити поточний символ в інший КМ, якій відповідає контекст іншої довжини. Звичайно спроба оцінки починається з КМ найбільшого порядку N, потім в певній послідовності здійснюється перехід до контекстних моделей менших порядків.

Статистичне кодування символу відходу виконується на підставі його ймовірності, так званої ймовірності відходу. Очевидно, що символи відходу породжуються не джерелом даних, а моделлю. Отже, їх ймовірність може залежати від характеристик даних, що стискаються, властивостей КМ, з якою проводиться відхід, властивостей КМ, на яку відбувається відхід, і т.д. Як можна оцінити цю ймовірність, маючи на увазі, що кінцевий критерій якості - поліпшення стиснення? Ймовірність відходу - це ймовірність появи в контексті нового символу. Тоді, фактично, необхідно оцінити правдоподібність настання події, що жодного разу не відбувалася. Теоретичного фундаменту для вирішення цієї проблеми, мабуть, не існує, але за час розвитку техніки контекстного моделювання було запропоновано декілька підходів, добре працюючих в більшості реальних ситуацій. Крім того, експерименти показують, що моделі з

неявним зважуванням стійкі методу оцінки ймовірності відходу, що відносно використовується, тобто вибір якогось способу обчислення цієї величини не впливає на коефіцієнт стиснення кардинальним чином.

Контрольні питання

1. Назвіть процес побудови моделі інформаційного джерела, який породжує дані, що стискаються.
2. Охарактеризуйте процес відображення оброблених даних в стиснуту форму представлення на основі результатів моделювання.
3. Наведіть класифікацію стратегій моделювання.
4. При якій стратегії моделювання не проводиться адаптація моделі до властивостей вхідних даних?
5. При якій стратегії моделювання для стиснення заданої послідовності вибирається або будується модель на підставі аналізу оброблюваних даних?
6. При якій стратегії моделювання модель змінюється по заданому алгоритму після стиснення кожного символу?
7. При якій стратегії моделювання оновлення моделі виконується після обробки цілого блоку символів змінної довжини?
8. Охарактеризуйте у відповідності до концепції «універсального моделювання і кодування» схему процесу стиснення.
9. Наведіть основні терміни методів контекстного моделювання.
10. Охарактеризуйте види контекстного моделювання.
11. Охарактеризуйте техніку, що пов’язана з введенням допоміжного символу відходу (escape), який є квазісимволом і використовується для передачі декодеру вказівок кодера.
12. Охарактеризуйте ймовірність появи в контексті нового символу.

ЛЕКЦІЯ 9

на тему: Методи контекстного моделювання (частина 2)

У лекції розглядаються методи контекстного моделювання, алгоритм PPM, зокрема розглядається приклад його роботи, а також приклад реалізації PPM-компресора. Описуються механізми оцінки ймовірності відходу апріорними, адаптивними методами, а також методом Z.

План:

1. Алгоритми PPM (Prediction Partial Matching - передбачення по частковому співпаданню)
2. Оцінка ймовірності відходу

1. Алгоритми сімейства PPM (Prediction Partial Matching - передбачення по частковому співпаданню)

Під контекстним моделюванням розумітимемо оцінку ймовірності появи символу (елементу, пікселя, семпла і навіть набору якісно різних об'єктів) залежно від безпосередньо йому передуючих, або контексту.

До методів контекстного моделювання відносять:

- алгоритм PPM (Prediction Partial Matching);
- моделі станів DMC (Dinamic Markov Compression);
- граматичні моделі (використовуються в алгоритмі SEQUITUR);
- моделі з використанням штучних нейронних мереж для побудови провісника.

Найбільш широке застосування отримала техніка контекстного моделювання PPM (Prediction by Partial Matching) – передбачення за частковим збігом та її модифікації. PPM забезпечує ущільнення в 3 - 4 рази для текстів і в 2 - 3 рази для об’єктних файлів при прийнятних обчислювальних затратах.

Техніка контекстного моделювання Prediction Partial Matching (передбачення по частковому співпаданню), запропонована в 1984 році Клірі (Cleary) і Уіттеном (Witten), є одним з найвідоміших підходів до стиснення якісних даних і вже точно найпопулярнішим серед контекстних методів. Значимість підходу обумовлена і тим фактом, що алгоритми, віднесені до PPM, незмінно забезпечують в середньому найкраще стиснення при кодуванні даних різних типів і служать стандартом, "точкою відліку" при порівнянні універсальних алгоритмів стиснення.

PPM впроваджений в WinRAR (з версії 3 і вище), WinZIP (з версії 10 і вище), 7ZIP та інших програмах стиснення.

Безсумнівною перевагою алгоритмів PPM є можливість отримання гарної статистичної моделі обробленої послідовності якісних даних. Модель, що

дозволяє ефективно передбачати невідомі символи повідомлення, можна застосовувати не тільки для стиснення, але й вирішення задач корекції тексту в системах оптичного розпізнавання символів, розпізнавання мовлення, класифікації типу тексту, семантичного аналізу тексту, шифрування.

Недоліки реалізацій PPM наступні:

- повільне декодування (зазвичай на 5 – 10 % повільніше кодування);
- несумісність кодера і декодера у випадку зміни алгоритму оцінки;
- повільна обробка малозбиткових даних;
- найкраще стиснення різних файлів досягається при порядках моделі 4 – 12 для моделей, що не застосовують техніку наслідування інформації та при порядках моделі 16 – 32 в інакших випадках (тому при виборі певного фіксованого порядка моделі ми можемо втрачати або в ступіні стиснення, або використовувати багато ресурсів комп'ютера;
- в загальному випадку недостатньо гарне стиснення файлів, статистичні характеристики яких схильні до частих змін такого типу, що оцінки розподілу ймовірностей в контекстних моделях швидко застарівають (нестабільність статистик контекстів);
- великі запити пам'яті у випадку використання складних моделей високого порядку в поєднанні з симетричністю алгоритма перешкоджають організації ефективного доступа до стиснутих даних;
- значний програш в ефективності (в порівнянні з алгоритмами LZ77) при стисненні файлів, які мають довгі блоки символів, що повторюються.

Перед власне розглядом алгоритмів PPM необхідно зробити зауваження про коректність термінології, що використовується. Протягом приблизно 10 років - з середини 1980-х років до середини 1990-х - під PPM розумілася група методів з цілком певними характеристиками. Останніми роками, вірогідно через різке збільшення числа всіляких гібридних схем і активного практичного використовування статистичних моделей для стиснення, відбулося змішення понять, і термін "PPM" часто використовується для позначення контекстних методів взагалі.

Як і у разі багатьох інших контекстних методів, для кожного контексту, що зустрічається в оброблюваній послідовності, створюється своя контекстна модель КМ. При цьому під контекстом розуміється послідовність елементів одного типу - символів, пікселів, чисел, але не набір різнорідних об'єктів. Далі замість слова "елемент" ми використовуватимемо "символ". Кожна КМ включає лічильники всіх символів, що зустрілися у відповідному контексті.

PPM відноситься до адаптивних методів моделювання. Початково кодеру і декодеру поставлена у відповідність початкова модель джерела даних. Вважатимемо, що вона складається з КМ(-1), привласнюючої однакову ймовірність всім символам алфавіту вхідної послідовності. Після обробки поточного символу кодер і декодер змінюють свої моделі однаковим чином,

зокрема, нарощуючи величину оцінки ймовірності даного символу. Наступний символ кодується (декодується) на підставі нової, зміненої моделі, після чого модель знову модифікується і т.д. На кожному кроці забезпечується ідентичність моделі кодера і декодера за рахунок використання однакового механізму її оновлення.

У PPM використовується неявне зважування оцінок. Спроба оцінки символу починається з КМ(N), де N є параметром алгоритму і називається порядком PPM-моделі. У разі нульової частоти символу в КМ поточного порядку здійснюється перехід до КМ меншого порядку за рахунок використовування механізму відходів (escape strategy).

Фактично, ймовірність відходу - це сумарна ймовірність всіх символів алфавіту вхідного потоку, що ще жодного разу не з'являлися в контексті. Будь-яка КМ повинна давати відмінну від нуля оцінку ймовірності відходу. Виключення з цього правила можливі тільки в тих випадках, коли значення всіх лічильників КМ для всіх символів алфавіту відмінні від нуля, тобто будь-який символ може бути оцінений в даному контексті. Оцінка ймовірності відходу традиційно є однією з основних проблем алгоритмів з неявним зважуванням, і вона буде спеціально розглянута в окремому пункті цієї теми.

Взагалі кажучи, спосіб моделювання джерела за допомогою класичних алгоритмів PPM спирається на наступні припущення про природу джерела:

- джерело є марківським з порядком N, тобто ймовірність генерації символу залежить від N попередніх символів і лише від них;
- джерело має таку додаткову особливість, що чим ближче розташовується один з символів контексту до поточного символу, тим більше кореляція між ними.

Таким чином, механізм відходів спочатку розглядався лише як допоміжний прийом, що дозволяє розв'язати проблему кодування символів, що жодного разу не зустрічалися в контексті порядку N. В ідеалі, що досягається після обробки достатньо довгого блоку, ніякого звернення до КМ порядку менше N відбуватися не повинне. Інакше кажучи, прилічення класичних алгоритмів PPM до методів, що проводять зважування, хай і неявним чином, є не цілком коректним.

При стисненні чергового символу виконуються наступні дії.

Якщо символ s обробляється з використанням PPM-моделі порядку N, то, як ми вже відзначали, в першу чергу розглядається КМ(N). Якщо вона оцінює ймовірність s числом, відмінним від нуля, то сама і використовується для кодування s. Інакше видається сигнал у вигляді символу відходу, і на основі меншої по порядку КМ(N-1) проводиться чергова спроба оцінити ймовірність s. Кодування відбувається через відхід до КМ менших порядків до тих пір, поки s не буде оцінений. КМ(-1) гарантує, що це врешті-решт відбудеться. Таким чином, кожний символ кодується серією кодів символу відходу, за якими слідує код самого символу. З цього виходить, що ймовірність відходу також можна розглядати як ймовірність переходу до контекстної моделі меншого порядку.

Якщо в процесі оцінки виявляється, що поточний контекст зустрічається вперше, то для нього створюється КМ(N).

При оцінці ймовірності символу в КМ порядку 0 < N можна виключити з розгляду всі символи, які містяться в КМ(0+1), оскільки жоден з них точно не є символом s. Для цього в поточній КМ(0) потрібно замаскувати, тобто тимчасово встановити в нуль, значення лічильників всіх символів, що є в КМ(0+1). Така техніка називається методом виключення (exclusion).

Після власне кодування символу звичайно здійснюється оновлення статистики для всіх КМ, використаних при оцінці його ймовірності, за винятком статичної КМ(-1). Такий підхід називається методом виключення при оновленні.

Розглянемо докладніше роботу алгоритму PPM за допомогою прикладу.

Приклад 1.

Є послідовність символів "абвавабвббббв" алфавіту {'а', 'б', 'в', 'г'}, яка вже була закодована (рис. 9.1).

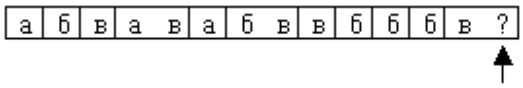

Рисунок 9.1 – Закодована послідовність символів

Хай лічильник символу відходу рівний 1 для всіх КМ, при оновленні моделі лічильники символів збільшуються на 1 у всіх активних КМ, застосовується метод виключення, і максимальна довжина контексту дорівнює 3, тобто N = 3.

Спочатку модель складається з КМ(-1), в якій лічильники всіх чотирьох символів алфавіту мають значення 1. Стан моделі обробки послідовності "абвавабвббббв" представлено на рис. 9.2, де прямокутниками позначені контекстні моделі, при цьому для кожної КМ вказаний курсивом контекст, а також символи і їх частоти, що зустрічалися в контексті.

Хай поточний символ рівний 'г', тобто '?' = 'г', тоді процес його кодування виглядатиме таким чином.

Спочатку розглядається контекст 3-го порядку "ббв". Раніше він не зустрічався, тому кодер, нічого не пославши на вихід, переходить до аналізу статистики для контексту 2-го порядку. В цьому контексті ("бв") зустрічалися символ 'а' і символ 'в', лічильники яких у відповідній КМ дорівнють 1 кожний, тому символ відходу кодується з ймовірністю 1/(2+1), де в знаменнику число 2 - це частота появи контексту "бв", що спостерігалася, 1 - це значення лічильника символу відходу. В контексті 1-го порядку "в" двічі зустрічався символ 'а', який виключається (маскується), 'в', що один раз також виключається, і один раз 'б', тому оцінка ймовірності відходу буде дорівнювати 1/(1+1). В КМ(0) символ 'г' також оцінити не можна, причому всі символи 'а', 'б', 'в', що є в цій КМ, виключаються, оскільки вже зустрічалися нам в КМ більш високого порядку. Тому ймовірність відходу виходить рівною 1. Цикл оцінювання завершується на

рівні КМ(-1), де 'г' до цього часу залишається єдиним символом, що дотепер не зустрічався, тому він одержує ймовірність 1 і кодується за допомогою 0 бітів.

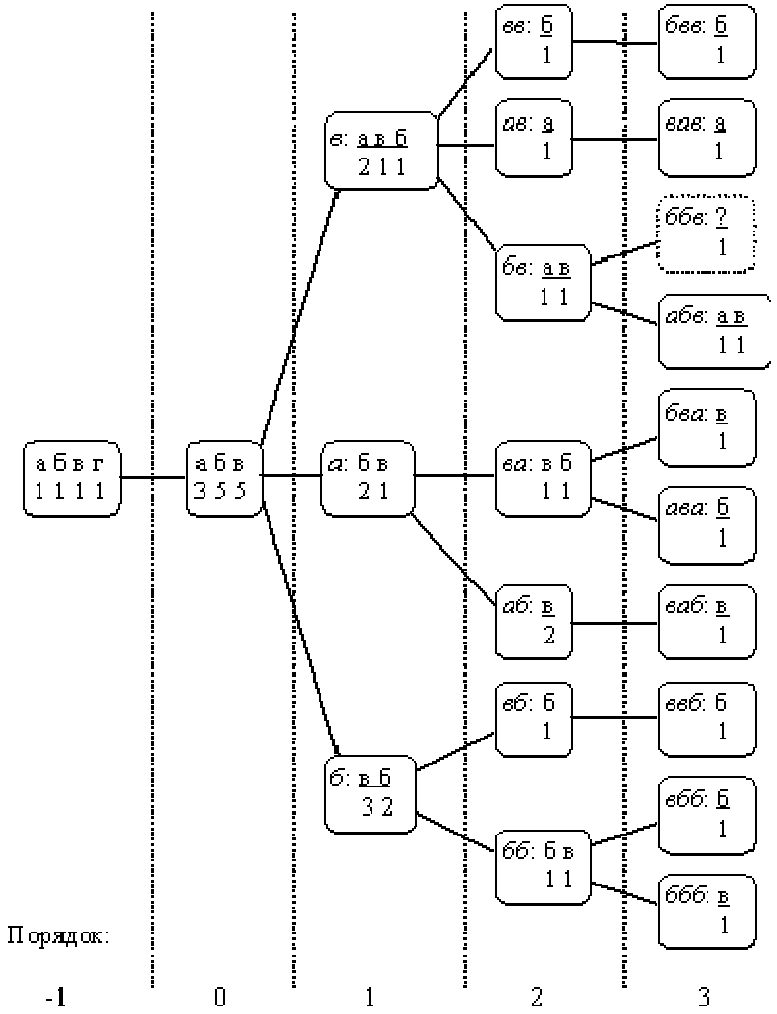

Рисунок 9.2 – Стан моделі після обробки послідовності "абвавабввбббв"

Таким чином, при використовуванні хорошого статистичного кодувальника для представлення 'г' буде потрібно в цілому приблизно 2.6 біти. Перед обробкою наступного символу створюється КМ для рядка "ббв" і проводиться модифікація лічильників символу 'г' в створеній і у всіх проглянутих КМ. В даному випадку потрібна зміна КМ всіх порядків від 0 до N.

Таблиця 9.1 демонструє оцінки ймовірності, які повинні були бути використані при кодуванні символів алфавіту {'а', 'б', 'в', 'г'} в поточній позиції.

Алгоритм декодування абсолютно симетричний алгоритму кодування. Після декодування символу в поточній КМ перевіряється, чи не є він символом відходу, якщо це так, то виконується перехід до КМ нижчого порядку. Інакше вважається, що початковий символ відновлений, він записується в декодований потік і здійснюється перехід до наступного кроку. Зміст процедур оновлення лічильників,

створення нових контекстних моделей, інших допоміжних дій і послідовність їх використання повинні бути строго однаковими при кодуванні і декодуванні. Інакше можлива розсинхронізація копій моделі кодера і декодера, що рано чи пізно приведе до помилкового декодування якогось символу. Починаючи з цієї позиції, вся частина стислої послідовності, що залишилася, буде розтиснута неправильно.

Таблиця 9.1 – Оцінки ймовірності

Символ s	Послідовність оцінок для КМ кожного порядку від 3 до -1					Загальна оцінка ймовірності q(s)	Представлення вимагає бітів
	3	2	1	0	-1		
	"ббв"	"бв"	"в"	" "			
'а'	-	$\frac{1}{2+1}$	-	-	-	$\frac{1}{3}$	1.6
'б'	-	$\frac{1}{2+1}$	$\frac{1}{1+1}$	-	-	$\frac{1}{6}$	2.6
'в'	-	$\frac{1}{2+1}$	-	-	-	$\frac{1}{3}$	1.6
'г'	-	$\frac{1}{2+1}$	$\frac{1}{1+1}$	1	1	$\frac{1}{6}$	2.6

Різниця між кодами символів, оцінки ймовірності яких однакові, досягається за рахунок того, що PPM-провісник передає кодувальнику так звані накопичені частоти (або накопичені ймовірності) оцінюваного символу і його сусідів або кодові простори символів. Так, наприклад, для контексту "бв" з прикладу 3 можна скласти табл. 9.2.

Кодувальник повинен відобразити символ s з оцінкою ймовірності q(s) в код довжини $\log_2 q(s)$, що і забезпечить стиснення всієї оброблюваної послідовності в цілому.

Таблиця 9.2 – Накопичені частоти та кодові простори символів

Символ	Частота	Оцінка ймовірності	Накоплена ймовірність (оцінка)	Кодовий простір
'а'	1	$\frac{1}{3}$	$\frac{1}{3}$	[0 … 0.33)
'б'	0	-	-	-
'в'	1	$\frac{1}{3}$	$\frac{2}{3}$	[0.33 … 0.66)
'г'	0	-	-	-
Відхід	1	$\frac{1}{3}$	1	[0.66 … 1)

2. Оцінка ймовірності відходу

На частку символів відходу звичайно доводиться порядка 30% і більше від всіх оцінок, обчислюваних моделювальником PPM. Це визначило пильну увагу до

проблеми оцінки ймовірності символів з нульовою частотою. Левова частка публікацій, присвячених PPM, прямо торкаються оцінки ймовірності відходу.

Можна виділити два підходи до рішення проблеми оцінки ймовірності відходу: апріорні методи, засновані на припущеннях про природу даних, що стискаються, і адаптивні методи, які намагаються пристосувати оцінку до даних. Зрозуміло, що перші покликані забезпечити хороший коефіцієнт стиснення при обробці типових даних в поєднанні з високою швидкістю обчислень, а другі орієнтовані на забезпечення максимально можливого ступеня стиснення.

2.1 Апріорні методи

Введемо позначення:

C - загальне число проглядань контексту, тобто скільки разів він зустрівся в обробленому блоці даних;

S - кількість різних символів в контексті;

$S^{(i)}$ - кількість таких різних символів, що вони зустрічалися в контексті рівно i раз;

$E^{(x)}$ - значення оцінки ймовірності відходу по методу x.

Винахідники алгоритму PPM запропонували два методи оцінки ймовірності відходу: так звані метод A і метод B. Використовуючі їх алгоритми PPM були названі PPMA і PPMB відповідно.

Надалі було описано ще 5 апріорних підходів до оцінки ймовірності відходу: методи C, D, P, X і XC. По аналогії з PPMA і PPMB, алгоритми PPM, що застосовують методи C і D, одержали назви PPMC і PPMD відповідно.

Ідея методів і їх порівняння представлені в табл. 9.3 і табл. 9.4.

Таблиця 9.3 – Ідея методів оцінки ймовірності відходу

Метод	$E^{(x)} =$
A	$\frac{1}{C+1}$
B	$\frac{S-S^{(1)}}{C}$
C	$\frac{S}{C+S}$
D	$\frac{S}{2C}$
P	$\frac{S^{(1)}}{C} - \frac{S^{(2)}}{C^2} + \frac{S^{(3)}}{C^3} - \dots$
X	$\frac{S^{(1)}}{C}$
XC	$\begin{cases} \frac{S^{(1)}}{C}, & \text{при } 0 < S^{(1)} < C \\ E^{(C)}, & \text{в противном случае} \end{cases}$

При реалізації методу B утримуються від оцінки символів до тих пір, поки вони не з'являться в поточному контексті більше одного разу. Це досягається за рахунок віднімання одиниці з лічильників. Методи P, X, XC базуються на

припущенні про те, що ймовірність появи в оброблюваних даних символу s_i підкоряється закону Пуассона з параметром λ_i .

Таблиця 9.4 – Порівняння методів оцінки ймовірності відходу

Тип файлів	Точність передбачення						
	Краще						Гірше
Тексти	XC	D	P	X	C	B	A
Двійкові файли	C	X	P	XC	D	B	A

Місця в табл. 5.5 дуже умовні. Так, наприклад, при стисненні текстів методи XC, D, P, X показують близькі результати, і багато що залежить від порядку моделі і файлів, що використовуються для порівняння. В більшості випадків істотним є тільки відставання точності оцінки ймовірності відходу за способами A і B від інших методів.

2.2 Адаптивні методи

Щоб поліпшити оцінку ймовірності відходу, необхідно мати таку модель оцінки, яка б адаптувалася до оброблюваних даних. Подібний адаптивний механізм одержав назву Secondary Escape Estimation (SEE), тобто "додаткової оцінки відходу", або "вторинної оцінки відходу". Метод полягає в тривіальному обчисленні ймовірності відходу з поточної КМ через частоту появи нових символів (або символів відходу) в контекстних моделях зі схожими характеристиками:

$$E^{(SEE)}(i) = \frac{f_i(esc)}{n_i}$$

де $f_i(esc)$ - число відходів, що спостерігалися, з контекстних моделей типу i;
n_i - число проглядань контекстних моделей типу i.

Зрозуміло обгрунтування вибору цих характеристик і критеріїв "схожості" за відсутності апріорних знань про характер послідовності, що стискається, дати складно, тому відомі алгоритми адаптивної оцінки базуються на емпіричному аналізі типових даних.

Метод Z

Одна з найраніших спроб реалізації SEE відома як метод Z, а використовуючий його різновид алгоритму PPM - PPMZ. Для точності опису цієї техніки SEE об'єкт "контекст" нижче також іменуватиметься "PPM-контекстом".

Для знаходження оцінку ймовірності відходу будуються так звані контексти відходу (escape contexts) КВ, зформовані з чотирьох полів. В полях КВ міститься інформація про значення наступних величин: останні чотири символи PPM-контексту, порядок PPM-контексту, кількість відходів і кількість успішних оцінок у відповідній КМ. Декільком КМ може відповідати один КВ.

Інформація про фактичну кількість відходів і успішних кодувань у всіх контекстних моделях, що мають загальний КВ, запам'ятовується в лічильниках контекстної моделі відходів КМВ, побудованої для даного КВ. Ця інформація визначає оцінку ймовірності відходу для поточної КМ. Оцінка ймовірності відходу знаходиться шляхом зважування оцінок, які дають три КМВ (КМВ порядка 2, 1 і 0), відповідаючі характеристикам поточної КМ.

При зважуванні статистики КМВ(n) використовуються наступна вага w_n

$$\frac{1}{w_n} = e \cdot \log_2\left(\frac{1}{e}\right) + (1-e) \cdot \log_2\left(\frac{1}{1-e}\right),$$

де e - оцінка ймовірності відходу, яку дає дана зважувана КМВ(n); формується з фактичної кількості відходів і успішних кодувань в контекстних моделях, відповідних цій КМВ, або, інакше, визначається частотою відходу, що спостерігалася з таких КМ.

Остаточна оцінка:

$$E^{(Z)} = \frac{\sum_{n=0}^{2} e_n w_n}{\sum_{n=0}^{2} w_n},$$

Після оцінки ймовірності відходу виконується пошук поточного символу серед тих, що є в КМ. За наслідками пошуку (символ знайдений чи ні) обновляються лічильники відповідних трьох КМВ порядку 0, 1 і 2.

Практично завжди можна підібрати і налаштувати таку PPM-модель, точніше контекстну модель з неявним зважуванням, що вона буде давати краще стиснення, ніж LZ чи BWT. Не дивлячись на це, застосування PPM-компресорів доцільне головним чином для стиснення текстів на природніх мовах або подібних до них даних, оскільки при обробці мало збиткових файлів великі часові витрати.

Контрольні питання

1. Охарактеризуйте алгоритми PPM (Prediction Partial Matching - передбачення по частковому співпаданню).
2. Охарактеризуйте модель, що дозволяє знайти в кожній позиції вхідної послідовності оцінку ймовірності появи кожного символу вхідної послідовності.
3. Назвіть оцінку ймовірності появи символу (елементу, пікселя) залежно від безпосередньо йому передуючих.
4. Охарактеризуйте техніку об’єднання оцінок ймовірності, що відповідає окремим активним контекстам в одну оцінку.
5. Охарактеризуйте апріорні методи оцінки ймовірності відходу.
6. Охарактеризуйте адаптивні методи оцінки ймовірності відходу.

ЛЕКЦІЯ 10

на тему: Аналіз і оцінка методів стиснення даних для сучасних баз даних

У лекції розглядаються характеристики основних методів і алгоритмів стиснення даних, проведена їх порівняльна оцінка за критеріями для визначення найбільш ефективних алгоритмів стиснення при обробці різних типів даних у сучасних базах даних.

План:

1. Актуальність задач стиснення баз даних
2. Аналіз основних методів стиснення даних в СУБД
3. Оцінка методів і алгоритмів стиснення табличних даних БД

1. Актуальність задач стиснення баз даних

В наш час стрімкий розвиток баз даних (БД) породжує необхідність зберігання та обробки все більших обсягів інформації, що вимагає величезних обчислювальних і часових витрат. Збільшенню обсягів даних в БД також сприяє введення інформаційної надлишковості, яка використовується для підвищення надійності функціонування обчислювальних систем. До негативних наслідків введення надмірності відносять підвищення інтенсивності потоку і збільшення ймовірності виникнення помилок, а також зменшення ймовірності їх виправлення.

Для скорочення обсягів БД і прискорення видачі необхідної інформації за запитом, доцільно застосовувати методи стиснення інформації. Питання економного кодування інформації в системах управління базами даних (СУБД) було поставлене в першій половині 1970-х років, але не втратило актуальності й досі. Багато сучасних дослідників відзначають недостатнє теоретичне опрацювання проблеми та неефективність підтримки стиснення даних в промислових СУБД.

Інтерес до задачі стиснення даних в СУБД обумовлений прагненням зменшити фізичний обсяг БД. При належному інтегруванні в СУБД методів неруйнівного стиснення даних досягається значна економія. Невелике збільшення числа використовуваних тактів процесора для кодування-декодування більш ніж компенсується зниженням витрат на підсистему введення-виведення. В даний час інтерес представляє також збільшення продуктивності СУБД за рахунок використання стиснення.

Тому актуальним є проведення системного аналізу і оцінки основних методів стиснення. Для визначення алгоритмів, найбільш ефективних при обробці різних типів даних БД, доцільно провести їх упорядкування і вибір критеріїв зіставлення їх характеристик. Це дозволить провести оцінку методів стиснення, яка базується на аналізі властивостей, принципів роботи та

основних характеристик і визначити найбільш ефективні алгоритми для стиснення різних типів даних БД.

2. Аналіз основних методів стиснення даних в СУБД

Використання стиснення даних в СУБД пов'язано з рішенням безлічі завдань:

- вибір методу стиснення і рівня стискання блоку в ієрархії фізичної структури БД;
- вибір між апаратним та програмним стисненням;
- вибір місця розташування модуля стиснення в архітектурі обчислювальної системи;
- забезпечення оновлення даних і виконання запитів до стиснених даних.

Основна маса досліджень і розробок, які стосуються питання підтримки економного кодування в СУБД, явним чи неявним чином орієнтовані на реляційні системи (РСУБД). Переважний сегмент ринку СУБД зайнятий РСУБД.

Багато методів стиснення і супутні прийоми обробки запитів застосовні до БД різного типу, наприклад, багатовимірного та об'єктного. Виходячи з вищевикладеного, даний аналіз стосується питань ефективної підтримки стиснення реляційних БД (РБД) методами стиснення без втрат.

Різноманіття інформаційної надмірності зумовило різноманіття підходів до стиснення даних в РБД і класифікації цих підходів.

Для табличних даних характерні наступні типи надмірності, які зазвичай проявляються одночасно:

- різниця в частоті окремих символів (елементів);
- наявність послідовностей однакових символів;
- часте повторення певних послідовностей символів;
- часта поява символів або послідовностей символів в певних місцях.

Для табличних даних характерні як вертикальні кореляційні зв'язки (по стовпцю), так і горизонтальні (по рядку). Краще стиснення буде забезпечуватися тими методами, які ефективно експлуатують всі типи надмірності.

Найбільш поширеними типами даних БД є строкові і числові, методи стиснення яких мають специфіку, тому вони розглянуті окремо. У табл. 10.1 наведені основні методи і алгоритми стиснення без втрат для РБД, дано їх опис, розглянуті основні переваги та недоліки.

Таблиця 10.1 – Аналіз основних методів стиснення табличних даних

Метод	Опис	Переваги	Недоліки
Кодування числових даних			
Упаковка бітів ✓ класичний алгоритм (bit packing) ✓ алгоритм стиснення кадра посилань (frame of reference compression)	Полягає в кодуванні значень атрибута бітовими послідовностями фіксованої довжини	✓ Простий спосіб стиснення на рівні значень атрибутів. ✓ Висока швидкість обробки і зберігання впорядкованості.	✓ Обробка появи нових значень атрибута. ✓ Не забезпечує значного ступеня стиснення.
Кодування довжин серій і усунення констант ✓ класичний алгоритм ✓ алгоритм заголовочного стиснення ✓ метод «BAP»	Знаходження послідовностей даних, що повторюються, і заміна їх простою структурою, що складається з кода і коефіцієнта повтора. Усунення константи реалізується з допомогою бітової карти, заголовочного вектора або трьох векторів одночасно.	✓ Ефективність досягається за рахунок економного кодування значень, що часто повторюються. ✓ Незалежність від обсягу даних ✓ Відрізняється простотою і високою швидкістю роботи ✓ Є ефективним для БД з фіксованою довжиною полів	✓ Актуально тільки для сильно розріжених БД ✓ Повільне декодування ✓ В середньому забезпечує недостатнє стиснення
Статистичне кодування ✓ метод Хаффмана ✓ алгоритм Шеннона-Фано ✓ арифметичне стиснення ✓ інтервальне кодування	Явним чином спираються на теорему Шеннона і включають в себе два етапи: оцінку ймовірності кодованих елементів (моделювання) і власне кодування	✓ Може застосовуватися до даних довільного типу ✓ Висока швидкість обробки даних	✓ Достатньо затратне кодування ✓ Є ефективним тільки при великій статистичній збитковості ✓ Класичний алгоритм потребує двох проходів по повідомленню
Диференціальне кодування ✓класичний алгоритм ✓алгоритм «стиснення вікна» (window-based compression)	Значення T[i,j] i-ого рядка j-го стовпця замінюється на різницю T[i-1,j] - T[i,j]. Якщо є тенденція в зміні даних, то результуючі різниці економно представляються простими методами	✓ Один з основних способів стиснення даних в СУБД і часто використовується в якості одного з кроків алгоритму стиснення ✓ Висока швидкість роботи ✓ Високий ступінь стиснення даних	✓ Складність забезпечення випадкового доступу до інформації в БД
Кодування строкових даних і даних довільного типу			
Метод Хаффмана ✓ класичний (статичний) алгоритм ✓ неадаптивний алгоритм з розширенням алфавіту символів ✓ напівадаптивне кодування ✓ адаптивне кодування	Визначає процедуру побудови коду змінної довжини, середня надмірність для якого мінімальна для всіх неблокових кодів. Зіставляє символам вхідного потоку, які зустрічаються більше число раз, ланцюжок біт меншої довжини.	✓Один з основних і найбільш популярних способів економного кодування ✓Простота і висока швидкість декодування ✓Проста апаратна реалізація ✓Унікальний префікс (дозволяє однозначно декодувати коди) ✓Використовується як один з етапів компресії в складних схемах	✓ Малоефективний для великих файлів ✓ Складне декодування ✓ У класичному алгоритмі довжина стиснутого повідомлення збільшується на довжину таблиці частот ✓ Залежність ступеня стиснення від близькості ймовірностей символів до 2 в негативному ступені

Продовження таблиця 10.1

Метод	Опис	Переваги	Недоліки
Арифметичне стиснення ✓ класичний алгоритм ✓ адаптивне кодування ✓ напівадаптивне кодування ✓ інтервальне кодування ✓ адаптивне кодування атрибуту по колонках («COLA»)	Блок елементів представляється дробом як добуток оцінок ймовірності всіх елементів блоку. Стиснення досягається за рахунок економного кодування кінцевих пробілів, що примусово збільшують довжину значення до заданої ширини стовпця.	✓ Високий ступінь стиснення, досягається теоретична межа ступеню стиснення ✓ Зберігається порядок сортування елементів відповідно до їх накопиченої ймовірності ✓ Не вимагає явної перебудови коду при зміні оцінок ймовірності	✓ Складне декодування ✓ Працює повільніше, ніж алгоритм Хаффмана ✓ Арифметичний кодер є двопрохідним і вимагає передачі разом з закодованим текстом таблиці частот символів.
Словникові методи (Зіва-Лемпеля) ✓ LZ77, LZSS ✓ LZ78, LZW ✓ LZW зі збереженням впорядкованості ✓ LZMW (метод Міллера-Уегнама) ✓ LZHAFF, LZARJ	Полягає в заміні послідовностей елементів вхідних даних на ідентифікатори фраз деякого словника, що збігаються з послідовністю, що заміщується.	✓ Використовуються як еталонні при порівнянні схем стиснення ✓ Дозволяють виконувати операції порівняння без декодування даних ✓ Ефективні для масивів великого об'єму	✓ Непрактичні для стиснення на рівні стовпця ✓ Складність забезпечення довільного доступу ✓ Невисока швидкість роботи класичних алгоритмів
Диференціальне кодування ✓ класичний алгоритм ✓ векторне квантування (VQ)	Кодується різниця між попереднім і поточним символами. Метод VQ розбиває безлічі об'єктів на групи, в кожній з яких вибирається зразко-вий об'єкт, до нього прирівнюються всі інші об'єкти	✓ Висока швидкість роботи ✓ Високий ступінь стиснення даних ✓ Ефективно використовується як засіб простого обліку вертикальних взаємозв'язків в даних	✓ Ускладнюється організація ефективного довільного доступу до інформації
Методи контекстного моделювання ✓ статичне моделювання ✓ напівадаптивне моделювання ✓ адаптивне моделю-вання (динамічне) ✓ блоково-адаптивне моделювання	Методи контекстно-залежного моделювання обмеженого порядку, що дозволяють оцінити ймовірність символу в залежності від попередніх символів	✓ Універсальний метод ✓ Хороший ступінь стиснення при високій швидкості ✓ Можливість отримання хорошої статистичної моделі обробленої послідовності даних	✓ Повільне кодування ✓ Несумісність кодера і декодера в разі зміни алгоритму оцінки ✓ Неефективні для малозбиткових даних ✓ Великі запити на пам'ять
Сортування, групування і перетворення стовпців ✓ сортування паралельних блоків (PBS) ✓ перетворення Барроуза-Уілера (BWT) ✓ метод переміщення стопки книг (MTF) ✓ алгоритм знаходження дерева залежності груп колонок ✓ алгоритм знаходження оптимальних груп	Сортування – впорядку-вання записів за значен-нями одного з полів. Групування - утворення однорідних груп по ряду ознак. PBS сортує еле-менти вхідного блоку і розкладає їх в кілька вихідних. BWT змінює порядок символів у вхідному рядку, при цьому підрядки, що повторюються утворю-ють на виході послідов-ності однакових симво-лів, що йдуть підряд.	✓ Як попередня операція дозволяє застосувати до отриманих даних прості методи стиснення ✓ Перестановка стовпців може істотно поліпшити стиснення при порядковій обробці ✓ Сортування дозволяє значно підвищити ступінь стиснення, особливо на текстових даних. ✓ Декомпресія в 3-4 рази швидше стиснення.	✓ Ступінь стиснення сильно залежить від типу даних. ✓ Вимагає досить багато пам'яті ✓ Для однозначного відновлення сортування повинне бути стійким, зі збереженням порядку однакових елементів ✓ Алгоритми групування працюють порівняно повільно ✓ Швидкість стиснення - відносно невисока.

3. Оцінка методів і алгоритмів стиснення табличних даних класичної БД

Для визначення методів і алгоритмів, найбільш ефективних при обробці різних типів даних доцільно провести їх упорядкування і вибір критеріїв зіставлення їх характеристик.

Визначення ефективних алгоритмів базується на аналізі властивостей, принципів роботи та основних характеристиках. **Основними критеріями** для порівняння методів і алгоритмів стиснення в контексті результативності їх роботи є:

- ступінь стиснення - відношення обсягів вхідного і результуючого представлення даних;
- швидкість роботи (кодування і декодування) - час, що витрачається на стиснення і відновлення даних;
- обсяг необхідної пам'яті;
- оберненість алгоритму - можливість відновлення даних (з втратами або без втрат);
- послідовність обробки даних - визначається, однопрохідним або багатопрохідним є конкретний алгоритм;
- можливість зростання надмірності при застосуванні алгоритму стиснення;
- область застосування.

В області стиснення даних діє закон важеля: алгоритми, що використовують більше ресурсів (часу і пам'яті), зазвичай досягають кращого ступеню стиснення, і навпаки, менш ресурсомісткі алгоритми за якістю стиснення, як правило, поступаються більш ресурсоємним. Таким чином, побудова оптимального з практичної точки зору алгоритму стиснення даних представляється складним завданням, оскільки необхідно домогтися досить високої якості стиснення при невеликому обсязі використовуваних ресурсів [6].

Критерії оцінки методів стиснення з практичної точки зору сильно залежать від передбачуваної області застосування. Наприклад, при використанні стиснення в системах реального часу необхідно забезпечити високу швидкість кодування і декодування; для вбудованих систем - обсяг необхідної пам'яті; для систем довгострокового зберігання даних - ступінь стиснення і (або) швидкість стиснення.

Проведемо порівняльний аналіз і оцінимо основні неруйнівні методи і алгоритми стиснення даних, використовуючи перераховані критерії (табл. 10.2).

Таблиця 10.2 – Порівняльний аналіз основних неруйнівних методів і алгоритмів стиснення

Метод стиснення (алгоритми)	Ступінь стиснення	Швидкість роботи	Споживання пам'яті	Кіл-сть проходів	Зростання надмірності	Застосування
Упаковка бітів						
Класичний алгоритм (bit packing)	низький	висока	середнє	1	рідко	Кодування даних числового типу
Кодування довжин серій і усунення констант						
Кодування довжин серій (RLE)	низький	висока	велике	1	можливо	Стиснення зображень, таблиць
Кодування Хаффмана (HAFF)						
статичний алгоритм	середній	достатньо висока	достатньо велике	2	рідко	Універсальні алгоритми, стиснення текстів і бінарної інформації
динамічний алгоритм	середній	вище середньої	вище середнього	1	рідко	
фіксований алгоритм	середній	висока	велике	1	можливо	
Арифметичне кодування (ARIC)						
статичний алгоритм	високий	середня	достатньо велике	2	рідко	Універсальні методи, стиснення текстів
динамічний алгоритм	високий	вище середньої	вище середнього	1	рідко	
Диференційальне кодування (DIFC)						
класичний алгоритм	достатньо високий	достатньо висока	середнє	1	можливо	Кодування даних в БД, стиснення зображень і звука
Словникові методи (LZ)						
LZ77, LZSS	достатньо високий	нижче середньої	невелике	1	рідко	Універсальні методи, стиснення текстів, стиснення зображень
LZHAFF, LZARJ	високий	низька	невелике	2	рідко	
LZ78, LZW	середній	вище середньої	середнє	1	рідко	
Методи контекстного моделювання (CM)						
передбачення по частковому співпаданню (PPM)	високий	нижче середньої	достатньо велике	1	рідко	Універсальний, стиснення текстів, зображень
Сортування, групування і перетворення стовпців						
перетворення Барроуза-Уїлера (BWT)	вище середнього	низька	середнє	>1	рідко	Універсальні методи, стиснення текстів
метод переміщення стопки книг (MTF)	нижче середнього	вище середньої	достатньо велике	1	можливо	

Метод диференціального кодування дуже широко використовується в тих випадках, коли природа даних така, що їх сусідні значення незначно відрізняються один від одного, при тому, що самі значення можуть бути як

завгодно великими. Це відноситься до звукових сигналів, особливо до мови, зображень, сусідні пікселі яких мають практично однакові яскравості і колір.

Для застосування в СУБД потрібні специфічні методи і прийоми економного кодування, оскільки звичайні методи:

- не забезпечують необхідної швидкості декодування;
- працюють з блоками досить великого обсягу і не підтримують довільний доступ;
- не враховують структуру даних;
- не враховують статистику за даними для подальшої оптимізації запитів.

Дуже корисною властивістю методу стиснення може бути збереження впорядкованості, що дозволяє оперувати при виконанні запитів безпосередньо закодованими даними. Більшість звичайних методів стиснення не володіють такою особливістю. Результатом застосування стиснення для БД є підвищення продуктивності і швидкості виконання запитів. Підвищення продуктивності за рахунок зменшення розміру даних, що зберігаються, обумовлено наступними ефектами [2]:

- зменшується середній час пошуку та доступу до інформації в пристроях зовнішньої пам'яті;
- підвищується фактична пропускна здатність каналу введення-виведення зовнішньої пам'яті;
- підвищується фактична пропускна здатність обчислювальної мережі;
- збільшується фактичний розмір буфера пристрою зовнішньої пам'яті;
- збільшується фактичний розмір буферного пулу СУБД.

Підвищення швидкості виконання запитів внаслідок використання стиснення викликано:

- збільшенням пропускної спроможності каналів зв'язку;
- зменшенням часу очікування отримання даних з зовнішньої і оперативної пам'яті.

У той же час застосування стиснення для БД пов'язано з рядом негативних ефектів:

- збільшення витрат обчислювальних ресурсів і, можливо, оперативної пам'яті;
- змінна довжина записів вимагає ускладнення програмного забезпечення;
- порушення характеристик даних, що вимагає використання сортуючих методів стиснення;
- зменшення можливості успішного відновлення після помилок.

Забезпечення дійсно високої продуктивності СУБД при використанні стиснення можливо тільки при комплексній зміні механізму виконання запитів. Збільшення різниці в швидкості процесора і елементів пам'яті диктує необхідність використання більш складних методів, що вимагають значного обсягу обчислень, але надають великий ступінь стиснення, які можуть підвищити реальну продуктивність СУБД.

Контрольні питання

1. Які типи надмірності характерні для табличних даних?
2. Назвіть можливість відновлення даних (з втратами або без втрат).
3. Назвіть алгоритм стиснення числових даних у БД, що полягає в кодуванні значень атрибута бітовими послідовностями фіксованої довжини.
4. Назвіть алгоритм стиснення числових даних у БД, що полягає в знаходженні послідовностей даних, що повторюються і заміною їх простою структурою із коду і коефіцієнта повтору.
5. Назвіть алгоритм стиснення даних у БД, що використовується у випадках, коли сусідні значення незначно відрізняються один від одного (при тому, що самі значення можуть бути як завгодно великими).
6. Назвіть алгоритм стиснення строкових даних і даних довільного типу у БД, що визначає процедуру побудови коду змінної довжини, середня надмірність для якого мінімальна для всіх не блокових кодів.
7. Назвіть алгоритм стиснення строкових даних і даних довільного типу у БД, що представляє блок елементів дробом як добуток оцінок ймовірності всіх елементів блоку.
8. Назвіть алгоритм стиснення строкових даних і даних довільного типу у БД, ефективність якого досягається за рахунок економного кодування кінцевих пробілів, що примусово збільшують довжину значення до заданої ширини стовпця.
9. Охарактеризуйте алгоритм стиснення строкових даних і даних довільного типу у БД, що полягає в заміні послідовностей елементів вхідних даних на ідентифікатори фраз деякого словника, які збігаються з послідовністю, що заміщується.
10. Охарактеризуйте алгоритм стиснення строкових даних і даних довільного типу у БД, що кодує різницю між попереднім і поточним символами.
11. Охарактеризуйте алгоритм стиснення строкових даних і даних довільного типу у БД, що дозволяє оцінити ймовірність символу в залежності від попередніх символів.
12. Що є результатом застосування стиснення для БД?

ЛЕКЦІЯ 11

на тему: Особливості зображень та загальний огляд алгоритмів стиснення зображень (частина 1)

В лекції розглядаються характеристики і основні класів зображень, колірні простори, приводиться класифікація додатків, що використовують алгоритми компресії.

План:

1. Особливості зображень та актуальність їх стиснення
2. Колірні простори та класи зображень
3. Класи додатків

1. Особливості зображень та актуальність їх стиснення

Беззаперечним є той факт, що стиснення зображень відіграє все важливішу роль в сучасному представлені цифрових даних. Це спричинено як і класичною проблемою ущільнення даних при розміщенні на фізичних носіях, так і розвитком телекомунікаційних технологій, а саме мобільних мереж, в яких часто існують значні обмеження на об'єм та швидкість передачі даних. Тому постає проблема ефективнішого використання трафіку, значна частина якого складається з цифрових зображень.

Зображення – це особливий вид даних, який має надлишковість в двох вимірах, що дає додаткові можливості для стиснення. Для економії пам'яті та більш ефективного використання ресурсів системи використовують відомі алгоритми стиснення даних без втрат і створюють спеціальні алгоритми кодування зображень, що можуть мати дуже високі коефіцієнти ущільнення.

Алгоритми стиснення зображень - область машинної графіки, що бурхливо розвивається. Основний об'єкт додатку зусиль в ній - зображення - своєрідний тип даних, що характеризується трьома особливостями:

- Зображення (як і відео) звичайно вимагає для зберігання набагато більшого об'єму пам'яті, ніж текст. Так, скромна, не дуже якісна ілюстрація на обкладинці книги розміром 500x800 крапок, займає 1.2 Мб - стільки ж, скільки художня книга з 400 сторінок (60 знаків в рядку, 42 рядки на сторінці). Як приклад можна розглянути також, скільки тисяч сторінок тексту ми зможемо помістити на CD-ROM, і як мало там поміститься нестиснутих фотографій високої якості. Ця особливість зображень визначає актуальність алгоритмів архівації графіки.
- Другою особливістю зображень є те, що людський зір при аналізі зображення оперує контурами, загальним переходом кольорів і порівняно нечутливий до малих змін в зображенні. Таким чином, ми можемо створити

ефективні алгоритми архівації зображень, в яких декомпресоване зображення не співпадатиме з оригіналом, проте людина цього не помітить. Дана особливість людського зору дозволила створити спеціальні алгоритми стиснення, орієнтовані тільки на зображення. Ці алгоритми дозволяють стискати зображення з високим ступенем стиснення і незначними з погляду людини втратами.

- Ми можемо легко помітити, що зображення, у відмінність, наприклад, від тексту, володіє надмірністю в 2-х вимірюваннях. Тобто як правило, сусідні крапки, як по горизонталі, так і по вертикалі, в зображенні близькі за кольором. Крім того, ми можемо використовувати подібність між колірними площинами R, G і B в наших алгоритмах, що дає можливість створити ще більш ефективні алгоритми. Таким чином, при створенні алгоритму компресії графіки ми використовуємо особливості структури зображення.

Всього на даний момент відомо мінімум три сімейства алгоритмів, які розроблені виключно для стиснення зображень, і використані в них методи практично неможливо застосувати до архівації ще яких-небудь видів даних.

Відзначимо, що на розмір файла значно впливає кількість параметрів, таких як варіант реалізації алгоритму, параметри алгоритму (як внутрішні, так і задані користувачем), порядок кольорів у палітрі тощо.

Навіть для доволі простих форматів одне і те ж зображення в однаковому форматі з використанням єдиного алгоритму архівації можливо записати в файл декількома коректними способами. Для складних форматів і алгоритмів архівації часто виникають ситуації, коли існуючі програми зберігають зображення різними способами. Ця ситуація склалась з форматом TIFF (у силу його значної гнучкості). Тривалий час по-різному зберігали зображення в формат JPEG, оскільки Міжнародна організація з стандартизації ISO підготувала тільки стандарт алгоритму, але не стандарт формату, що дало можливість запобігти “війні форматів”. Протилежна ситуація склалась з фрактальною компресією, оскільки на збереження фрактальних коефіцієнтів у файл є стандарт формату, але алгоритм їх знаходження є технологічною таємницею розробників програм-компресорів. У результаті для доволі стандартної програми-декомпресора можуть бути підготовлені файли з коефіцієнтами, які істотно різняться, як за розміром, так і за якістю отримуваного зображення.

Для того, щоб говорити про алгоритми стиснення зображень, ми повинні визначитися з декількома важливими питаннями:

- Які класи зображень існують?
- Які класи додатків, що використовують алгоритми компресії графіки, існують, і які вимоги вони пред'являють до алгоритмів?
- Які критерії ми можемо запропонувати для порівняння різних алгоритмів?

Розглянемо ці питання докладніше.

2. Колірні простори та класи зображень

За видом представлення зображення поділяють на растрові та векторні. Растрові зображення, здатні відображати реальні образи без спотворень. Тому вони значною мірою підходять для фотографій, картин та в деяких інших випадках, коли вимагається максимальна "натуральність". Ці зображення легко вивести на будь-який пристрій виведення інформації електронної обчислювальної машини (наприклад, монітор чи принтер), оскільки в основі їх лежить саме растровий принцип.

Але разом з тим варто зважати на недоліки. Растрове зображення високої якості здатне займати значні об'єми пам'яті та для їх обробки потрібна потужна обчислювальна техніка. Будь-яка зміна розмірів завжди призводить до погіршення якості: при збільшенні пікселі можуть виникнути "з нічого", при зменшенні – частину пікселів буде просто викинуто.

Існує також інший спосіб представлення зображення – векторна (об'ємна) графіка. У цьому випадку застосовуються правила побудови з використанням елементарних графічних об'єктів, що своєю чергою не вимагає значних ресурсів обчислювальної техніки і відповідно відображення рисунку відбувається практично миттєво.

Істотним недоліком векторних зображень є достатня складність за допомогою векторних об'єктів відобразити реалістичне зображення, наприклад, фото. Для цього обов'язково знадобиться велика кількість елементарних об'єктів, які при накладанні утворять вихідне зображення, при цьому розмір отриманого файла інформації може виявитися набагато більшим, ніж відповідне растрове зображення.

Статичні растрові зображення є двовимірним масивом чисел. Елементи цього масиву називають **пікселями** (від англійського "pixel" - "picture element"). Всі зображення можна підрозділити на дві групи - з палітрою і без неї. У зображень з палітрою в пікселі зберігається число - індекс в деякому одновимірному векторі кольорів, званому **палітрою**.

Зображення без палітри бувають в системі кольоропредставлення і в **градаціях сірого** (grayscale). Для останніх значення кожного пікселя інтерпретується як яскравість відповідної точки. Частіше за все зустрічаються зображення з 2, 16 і 256 рівнями сірого. Одна з цікавих практичних задач полягає в приведенні кольорового або чорно-білого зображення до двох градацій яскравості, наприклад, для друку на лазерному принтері.

Людське око може сприймати світлове випромінювання в діапазоні довжин хвиль від 380 до 770 нм – одночасно близько 10 тисяч різних кольорів. Хвилі різної довжини сприймаються оком неоднаково. Найбільш відчутним є зелений колір, потім йде червоний, а за ним – синій. Краще розрізняються кольори ближче

розміщених об’єктів, ніж віддалених. Погано сприймається колір дуже маленьких об’єктів. Можливість розрізняти кольори є індивідуальною. На сприйняття кольору впливає також спосіб його відтворення: одні і ті ж зображення, візуалізовані на різних пристроях, мають різний вигляд. Задача точного відображення кольору є досить складною і до кінця ще не вирішена. Колір є важливим засобом підсилення враження при сприйнятті зображень і підвищення їх інформаційної насиченості. Відчуття кольору формується людським мозком в результаті аналізу світлового потоку, що попадає на сітчатку ока від об’єктів, які випромінюють або відбивають світло.

Фізичні характеристики світлового потоку визначаються параметрами потужності, яскравості та освітленості. Насиченість кольору показує, наскільки даний колір відрізняється від монохроматичного (“чистого”) випромінювання того ж кольорового тону. Ахроматичні кольори (білий, сірий, чорний) характеризуються тільки світлотою, тобто розрізненням ділянок, які в більшій або в меншій мірі відбивають світло. Білий і чорний кольори відповідають граничним значенням діапазону, причому чорний відповідає мінімальній інтенсивності, а білий - максимальній. Хроматичні кольори мають параметри насиченості, світлоти і колірного тону.

Глибина кольору або колірна роздільна здатність в комп’ютерній графіці визначає метод кодування колірної інформації для її відтворення на екрані монітору: 2 біти для білого і чорного кольорів, 8, 16 і 24 біти – відповідно для відображення 256, 65536 і більш ніж 16,5 мільйонів градацій колірного тону.

В відповідності до принципів формування зображення адитивним чи субтрактивним методами розроблені способи розділення відтінку кольору на складові компоненти, які називають моделями кольору. В адитивних моделях нові кольори утворюються шляхом додавання основного кольору з чорним. Змішування всіх основних кольорів дає чистий білий колір, якщо значення їх інтенсивностей максимальні, і чистий чорний, якщо вони дорівнюють нулю. Адитивні моделі використовуються в пристроях, які випромінюють світло. В субстрактивних моделях нові кольори утворюються шляхом віднімання основного кольору від білого. В цьому випадку змішування всіх основних кольорів дає чистий чорний колір, якщо значення їх інтенсивностей максимальні, і чистий білий, якщо вони дорівнюють нулю. Субстрактивні моделі використовуються в пристроях, які відбивають світло. Найбільш поширеними є колірні моделі RGB, CMY, HSB, CIE Lab.

Кольорова палітра – це модель представлення кольору, заснована на використанні колірних координат. Кольорова палітра будується таким чином, щоб будь-який колір був представлений точкою, що має певні координати, причому так, щоб одному набору координат (і точці простору) відповідав один колір. Кількість координат задає розмірність простору.

Найбільш поширений простір - RGB. Це тривимірний колірний простір, де кожен колір описаний набором з трьох кординат - кожна з них відповідає компоненті кольору в розкладанні на червоний (R, Red), зелений (G, Green) і синій (B, Blue) кольори.

Модель RGB (Red-Green-Blue – червоний-зелений-синій) є адитивною. Вона являє собою сполучення в різній пропорції трьох основних кольорів і є основою для електронного відтворення зображень на екрані монітору. При накладанні одного компонента основного кольору на інший яскравість сумарного випромінювання збільшується. Суміщення трьох компонентів дає ахроматичний сірий колір, який при збільшенні яскравості наближується до білого кольору.

Відповідно до моделі HSB колір визначається трьома компонентами: відтінком (Hue), насиченістю (Saturation) і яскравістю (Brightness). При моделюванні кольорів тут не змішують основні кольори, а змінюють їх властивості. Відтінок – це є власне колір в загальноприйнятому розумінні. Насиченість визначається кількістю білого в відтінку: в повністю насиченому відтінку не міститься білого –він вважається чистим; частково насичений відтінок світліший. Яскравість визначає інтенсивність світіння кольору – відтінки з високою інтенсивністю дуже яскраві, а з низькою – темні. Модель HSB прийнято використовувати при створенні зображень на комп’ютері з імітацією прийомів роботи і інструментарію художників. Після створення зображення його рекомендується перетворити в іншу модель, в залежності від способу публікації.

Модель CMY (Cyan-Magenta-Yellow – голубий-пурпурний-жовтий) є субтрактивною і призначена для отримання зображень на білій поверхні. Голубий, пурпурний і жовтий кольори називають доповняльними, тому що вони доповнюють основні кольори до білого. Головною проблемою моделі CMY є те, що накладання один на одного доповняльних кольорів на практиці не дає чистого чорного кольору. Тому в модель включають четвертий компонент чистого чорного кольору (blacK – чорний). Такий різновид моделі має абревіатуру CMYK.

В моделі CIE Lab будь-який колір визначається світлотою (L) і хроматичними компонентами: параметром a, що змінюється в діапазоні від зеленого до червоного, і параметром b, що змінюється в діапазоні від синього до жовтого. Ця модель є апаратно незалежною і часто використовується для перенесення даних між різними пристроями. Сьогодні вона є прийнятим по замовчуванню стандартом для програми Adobe Photoshop.

Значення первинних і ахроматичних кольорів для деяких моделей при 256 градаціях колірного тону показані в таблиці 11.1.

Колірна палітра в комп’ютерній графіці за призначенням подібна до палітри художника, але включає значно більшу кількість кольорів. Електронна палітра складається з певної кількості комірок, кожна з яких містить окремий колірний тон. Конкретна палітра співвідноситься з певною моделлю кольору, так як її кольори створені на основі колірного простору цієї моделі, і містить обмежений

набір кольорів, які називаються стандартними. Графічні програми, як правило, надають на вибір декілька колірних палітр, кожна з яких відповідає певній моделі кольору.

Таблиця 11.1 – Значення первинних і ахроматичних кольорів

	RGB	CMY	HSB
Червоний	255, 0, 0	0, 255, 255	0, 240, 120
Жовтий	255, 255, 0	0, 0, 255	40, 240, 120
Зелений	0, 255, 0	255, 0, 255	80, 240, 120
Синьо-зелений	0, 255, 255	255, 0, 0	120, 240, 120
Синій	0, 0, 255	255, 255, 0	160, 240, 120
Червоно-синій	255, 0, 255	0, 255, 0	200, 240, 120
Чорний	0, 0, 0	255, 255, 255	160, 0, 0
Відтінки сірого	63, 63, 63 127, 127, 127 191, 191, 191	191, 191, 191 127, 127, 127 63, 63, 63	160, 0, 59 160, 0, 120 160, 0, 180
Білий	255, 255, 255	0, 0, 0	160, 0, 240

Склад колірних палітр RGB залежить від вибраної роздільної здатності - 24, 16 або 8 біт. В останньому випадку колірна палітра називається індексною, тому що коди відтінків кольору виражають не колір пікселя, а індекс (номер кольору). До файлів зображень, створених в індексній палітрі, повинна додаватись сама палітра.

Зображення, що готуються до публікації в Internet, прийнято створювати в так званій “безпечній” палітрі, яка містить 216 кольорів. Це викликано обмеженнями, пов’язаними з вимогами до сумісності комп’ютерів різних платформ.

Існують і інші колірні простори різної розмірності - від одновимірних, які можуть описати виключно монохромне зображення (наприклад, PAL D / K, PAL B / G), до шести і десятімірних, таких, наприклад, як простір CMYKLcLm (Cyan - блакитний, Magenta - пурпурний, Yellow – жовтий, blacK - чорний, lightCyan – світло блакитний, lightMagenta – світло пурпурний). Вибір колірного простору визначається областю застосування:

1) друк і поліграфія (CMYK, CMYKLcLm, RGB, PMS, NCS);

2) обробка та редагування (XYZ, HSV, LAB);

3) зберігання і передача (YUV, YCbCr, YFbFr, YPbPr, YCoCgR, RCT).

Простори високої розмірності найчастіше використовують для друку і поліграфії, а простори малої розмірності - для зберігання і передачі зображень.

Колірним перетворенням прийнято називати рівняння переходу з простору RGB (вважається базовим) в будь-який інший колірний простір.

У сучасних алгоритмах стиснення кодування компонент відбувається незалежно (не враховується залежність між сигналами). При цьому наявність взаємної залежності між компонентами (як у випадку RGB) призводить до зменшення можливого ступеня стиснення.

При видаленні взаємозалежності може значно збільшитися ступінь стиснення. Отже, чим більше взаємна ентропія, від якої позбавляються за допомогою нового перетворення, то більший виграш за ступенем стиснення нових компонент у порівнянні зі стандартними компонентами RGB.

Надмірність інформації в компонентах RGB негативно позначається на ефективності стиснення зображень. Внаслідок чого на початковому етапі алгоритми стиснення переводять зображення з простору RGB в інший простір, де компоненти більш незалежні між собою.

Для цієї мети в кодеках використовуються лінійні колірні простори, такі як YUV (YCbCr), YCoCg і т. д., основною властивістю яких є слабка залежність між компонентами і маленькі значення кореляції між компонентами.

YUV перетворення використовується вже досить довго в різних системах, пов'язаних зі статичними і динамічними зображеннями. З 1950 року - в стандартах аналогового телебачення NTSC і PAL / SECAM. Компонента Y ідеально підходила для передачі сигналу для телевізорів, здатних відображати лише чорно-біле зображення. В кольоровому телевізорі додаткові сигнали U і V дозволяли відновлювати всю картинку в кольорі. З приходом цифрової обробки даних перетворення YUV знайшло застосування в алгоритмах стиснення зображень JPEG, MPEG4, H.264/AVC і в широкоформатному телебаченні (HDTV). Але для цифрових систем більш коректно використовувати дискретний аналог YUV, YCbCr.

Сьогодні також широко застосовуються аналогічні YUV колірні перетворення, такі як YFbFr і YCoCg. Ці перетворення мають те ж смислове навантаження, але на відміну від YUV не мають помилок округлення, так як працюють з цілочисельними коефіцієнтами.

У просторі YUV (YCbCr) компоненти мають такі позначення: Y – компонента яскравості (характеристика освітленості точки); U, V (Cr, Cb) - кольорорізнісні компоненти (chrominance).

Для того, щоб коректніше оцінювати ступінь стиснення, потрібно ввести поняття **класу зображень** - сукупність зображень, для яких використання алгоритму архівації дає якісно однакові результати. Наприклад, для одного класу алгоритм дає дуже високий ступінь стиснення, для іншого - майже не стискає, для третього - збільшує файл в розмірі.

Дамо неформальне визначення **найпоширеніших класів зображень**:

Клас 1. Зображення з невеликою кількістю кольорів (4-16) і великими областями, заповненими одним кольором. Плавні переходи кольорів відсутні. Приклади: ділова графіка - гістограми, діаграми, графіки і т.п.

Клас 2. Зображення з плавними переходами кольорів, побудовані на комп'ютері. Приклади: графіка презентацій, ескізні моделі в САПР.

Клас 3. Фотореалістичні зображення. Приклад: відскановані фотографії.

Клас 4. Фотореалістичні зображення з накладенням ділової графіки. Приклад: реклама.

Розвиваючи дану класифікацію, як окремі класи можуть бути запропоновані неякісно відскановані в 256 градацій сірого кольору сторінки книг або растрові зображення топографічних карт. (Відмітимо, що цей клас не тотожний класу 4). Формально будучи 8- або 24-бітовими, вони несуть не растрову, а чисто векторну інформацію. Окремі класи можуть утворювати і зовсім специфічні зображення: рентгенівські знімки або фотографії в профіль і фас з електронного досьє.

Достатньо складною і цікавою задачею є пошук якнайкращого алгоритму для конкретного класу зображень.

Підсумок: Немає сенсу говорити про те, що якийсь алгоритм стиснення краще іншого, якщо ми не позначили класи зображень, на яких порівнюються наші алгоритми.

3. Класи додатків

Розглянемо наступну просту **класифікацію додатків, що використовують алгоритми компресії:**

Клас 1. Характеризуються високими вимогами до часу архівації і розархівування. Нерідко потрібне проглядання зменшеної копії зображення і пошук в базі даних зображень.

Приклади: Видавничі системи в широкому значенні цього слова, причому як друкуючі якісні публікації (журнали) з явно високою якістю зображень і використовуванням алгоритмів архівації без втрат, так і друкуючі газети, і WWW-сервери, де є можливість оперувати зображеннями меншої якості і використовувати алгоритми стиснення з втратами. В подібних системах доводиться мати справу з повнокольоровими зображеннями самого різного розміру (від 640x480 - формат цифрового фотоапарата, до 3000x2000) і з великими двокольоровими зображеннями. Оскільки ілюстрації займають левову частку від загального об'єму матеріалу в документі, проблема зберігання стоїть дуже гостро. Проблеми також створює велика різнорідність ілюстрацій (доводиться використовувати універсальні алгоритми). Єдине, що можна сказати наперед, це те, що переважатимуть фотореалістичні зображення і ділова графіка.

Клас 2. Характеризується високими вимогами до ступеня архівації і часу розархівування. Час архівації ролі не грає.

Іноді подібні додатки також вимагають від алгоритму компресії легкості масштабування зображення під конкретну роздільну здатність монітора у користувача. Приклад: Довідники і енциклопедії на CD-ROM. З появою великої кількості комп'ютерів, оснащених цим приводом, достатньо швидко сформувався ринок програм, що випускаються на лазерних дисках. Місткість одного диска приблизно 650 Мб, ї її, як правило, не вистачає. При створенні енциклопедій і ігор велику частину диска займають статичні зображення і відео. Таким чином, для цього класу додатків актуальності набувають істотно асиметричні за часом алгоритми (симетричність за часом - відношення часу компресії до часу декомпресії).

Клас 3. Характеризується дуже високими вимогами до ступеня архівації.

Додаток клієнта одержує від серверу інформацію по мережі. Приклад: Всесвітня інформаційна павутина - WWW. В цій гіпертекстовій системі достатньо активно використовуються ілюстрації. При оформленні інформаційних або рекламних сторінок хочеться зробити їх більш яскравими і барвистими, що природно позначається на розмірі зображень. Якнайбільше при цьому страждають користувачі, підключені до мережі за допомогою повільних каналів зв'язку. Якщо сторінка WWW перенасичена графікою, то очікування її повної появи на екрані може затягнутися. Оскільки при цьому навантаження на процесор мале, то тут можуть знайти використання ефективно стискаючі складні алгоритми з порівняно великим часом розархівування. Крім того, ми можемо видозмінити алгоритм і формат даних так, щоб проглядати огрублене зображення файлу до його повного отримання.

Можна привести безліч більш вузьких класів додатків. Так, своє використання машинна графіка знаходить і в різних інформаційних системах. Наприклад, вже стає звичним досліджувати ультразвукові і рентгенівські знімки не на папері, а на екрані монітора. Поступово в електронний вигляд переводять і історії хвороб. Зрозуміло, що зберігати ці матеріали логічніше в єдиній картотеці. При цьому без використовування спеціальних алгоритмів велику частину архівів займуть фотографії. Тому при створенні ефективних алгоритмів рішення цієї задачі потрібно врахувати специфіку рентгенівських знімків - переважання розмитих ділянок.

У геоінформаційних системах - при зберіганні аерофотознімків місцевості - специфічними проблемами є великий розмір зображення і необхідність вибірки лише частини зображення на вимогу. Крім того, може бути потрібно масштабування. Це неминуче накладає свої обмеження на алгоритм компресії.

У електронних картотеках і досьє різних служб для зображень характерна подібність між фотографіями в профіль і подібність між фотографіями у фас, яку також необхідна враховувати при створенні алгоритму архівації. Подібність між

фотографіями спостерігається і в будь-яких інших спеціалізованих довідниках. Як приклад можна привести енциклопедії птахів або кольорів.

Підсумок: Немає сенсу говорити про те, що якийсь конкретний алгоритм компресії краще іншого, якщо ми не позначили клас додатків, щодо якого ми ці алгоритми збираємося порівнювати.

Контрольні питання

1. Які параметри треба визначити, перш ніж порівнювати два алгоритми компресії?
2. Чому некоректно порівнювати тимчасові параметри реалізацій алгоритмів компресії, оптимально реалізованих на різних комп'ютерах? Приведіть приклади ситуацій, коли архітектура комп'ютера дає переваги тому або іншому алгоритму.
3. Запропонуйте приклад свого класу зображень.
4. Якими властивостями зображень ми можемо користуватися, створюючи алгоритм компресії? Приведіть приклади.
5. До якого класу відносяться цифрові зображення, що здатні відображати реальні образи без спотворень, але зі значними об’ємами пам`яті?
6. До якого класу відносяться цифрові зображення, зміна параметрів яких завжди призводить до погіршення якості?
7. До якого класу відносяться цифрові зображення, побудовані з використанням елементарних графічних об’єктів?
8. До якого класу відноситься цифрове зображення, в пікселях якого зберігаються індекси деякого одномірного вектору кольорів?
9. До якого класу відноситься цифрове зображення, пікселі якого зберігають певні кольорі координати?
10. До якого класу відноситься цифрове зображення, для якого значення кожного пікселя інтерпретується як яскравість відповідної точки?
11. Які кольорові простори застосовують для друку зображень і поліграфії?
12. Які кольорові простори застосовують для зберігання і передачі зображень?
13. Для яких кольорових просторів є характерним слабка залежність між компонентами і маленькі значення кореляції?

ЛЕКЦІЯ 12

на тему: Особливості зображень та загальний огляд алгоритмів стиснення зображень (частина 2)

В лекції розглядаються основні вимоги додатків до алгоритмів компресії, а також критерії порівняння алгоритмів та приводяться параметри різних алгоритмів стиснення зображень.

План:

1. Вимоги додатків до алгоритмів компресії
2. Критерії порівняння алгоритмів стиснення
3. Короткий огляд алгоритмів стиснення зображень

1. Вимоги додатків до алгоритмів компресії

У попередній лекції ми визначили, які додатки є споживачами алгоритмів архівації зображень. Проте відмітимо, що додаток визначає характер використовування зображень (або велика кількість зображень зберігається і використовується, або зображення викачуються з мережі, або зображення великі за розмірами, і нам необхідна можливість отримання лише частини...). Характер використовування зображень задає ступінь важливості наступних нижче суперечливих вимог до алгоритму:

1. Високий ступінь компресії.

Зауважимо, що далеко не для всіх додатків актуальний високий ступінь компресії. Крім того, деякі алгоритми дають краще співвідношення якості до розміру файлу при високих ступенях компресії, проте програють іншим алгоритмам при низьких ступенях.

2. Висока якість зображень.

Виконання цієї вимоги напряму суперечить виконанню попередньої...

3. Висока швидкість компресії.

Ця вимога для деяких алгоритмів з втратою інформації є взаємовиключною з першими двома. Інтуїтивно зрозуміло, що чим більше часу ми аналізуватимемо зображення, намагаючись одержати щонайвищий ступінь компресії, тим краще буде результат. І, відповідно, чим менше ми часу витратимо на компресію (аналіз), тим нижче буде якість зображення і більше його розмір.

4. Висока швидкість декомпресії.

Достатньо універсальна вимога, актуальна для багатьох додатків. Проте можна привести приклади додатків, де час декомпресії далеко не критичний.

5. Масштабування зображень.

Дана вимога має на увазі легкість зміни розмірів зображення до розмірів вікна активного додатку. Річ у тому, що одні алгоритми дозволяють легко

масштабувати зображення прямо під час декомпресії, тоді як інші не тільки не дозволяють легко масштабувати, але і збільшують ймовірність появи неприємних артефактів після використання стандартних алгоритмів масштабування до декомпресованого зображення. Наприклад, можна привести приклад "поганого" зображення для алгоритму JPEG - це зображення з достатньо дрібним регулярним малюнком (піджак в дрібну клітку). Характер спотворень, що вносяться JPEG алгоритмом, такий, що зменшення або збільшення зображення може дати неприємні ефекти.

6. Можливість показати огрублене зображення (низької роздільної здатності), використавши тільки початок файлу.

Дана можливість актуальна для різного роду мережних додатків, де перекачування зображень може зайняти достатньо великий час, і бажано, одержавши початок файлу, коректно показати preview. Помітимо, що примітивна реалізація вказаної вимоги шляхом записування в початок зображення його зменшеної копії помітно погіршить ступінь компресії.

7. Стійкість до помилок.

Дана вимога означає локальність порушень в зображенні при псуванні або втраті фрагмента файлу, що передається. Дана можливість використовується при широкомовленні (broadcasting - передача за багатьма адресами) зображень по мережі, тобто в тих випадках, коли неможливо використовувати протокол передачі, що повторно запрошує дані у серверу при помилках. Наприклад, якщо передається відеоряд, то було б неправильно використовувати алгоритм, у якого збій приводив би до припинення правильного показу всіх подальших кадрів. Дана вимога суперечить високому ступеню архівації, оскільки інтуїтивно зрозуміло, що ми повинні вводити в потік надмірну інформацію. Проте для різних алгоритмів об'єм цієї надмірної інформації може істотно відрізнятися.

8. Специфіка зображення.

Більш високий ступінь архівації для класу зображень, які статистично частіше застосовуватимуться в нашому додатку. В попередніх питаннях ця вимога вже обговорювалася.

9. Редагованість.

Під редагованістю розуміється мінімальний ступінь погіршення якості зображення при його повторному збереженні після редагування. Багато алгоритмів з втратою інформації можуть істотно зіпсувати зображення за декілька ітерацій редагування.

10. Невелика вартість апаратної реалізації. Ефективність програмної реалізації.

Дані вимоги до алгоритму реально пред'являють не тільки виробники ігрових приставок, але і виробники багатьох інформаційних систем. Так, декомпресор фрактального алгоритму дуже ефективно і коротко реалізується з використанням технології MMX і розпаралелювання обчислень, а стиснення за

стандартом CCITT Group 3 легко реалізується апаратно. Очевидно, що для конкретної задачі нам будуть дуже важливі одні вимоги і менш важливі (і навіть абсолютно байдужі) інші.

Підсумок: На практиці для кожної задачі ми можемо сформулювати набір пріоритетів з вимог, викладених вище, який і визначить самий відповідний в наших умовах алгоритм (або набір алгоритмів) для її вирішення.

2. Критерії порівняння алгоритмів стиснення

Зауважимо, що характеристики алгоритму щодо деяких вимог додатків, сформульованих вище, залежать від конкретних умов, в які буде поставлений алгоритм. Так, **ступінь компресії** залежить від того, на якому класі зображень алгоритм тестується. Аналогічно, **швидкість компресії** нерідко залежить від того, на якій платформі реалізований алгоритм. Перевага одного алгоритму перед іншим може дати, наприклад, можливість використовування в обчисленнях алгоритму технологій нижнього рівня, типу MMX, а це можливо далеко не для всіх алгоритмів. Так, JPEG істотно виграє від використання технології MMX, а LZW ні. Крім того, нам доведеться враховувати, що деякі алгоритми розпаралелюються легко, а деякі ні.

Таким чином, **неможливо скласти універсальний порівняльний опис відомих алгоритмів**. Це можна зробити тільки для типових класів додатків за умови використовування типових алгоритмів на типових платформах. Проте такі дані надзвичайно швидко застарівають.

Так, наприклад, ще в 1994, інтерес до **показу огрубленого зображення**, використовуючи тільки початок файлу (вимога 6), був чисто абстрактним. Реально ця можливість практично ніде не була потрібна і клас додатків, що використовують дану технологію, був украй невеликий. З вибуховим розповсюдженням Internet, який характеризується передачею зображень по порівняно повільних каналах зв'язку, використовування Interlaced GIF (алгоритм LZW) і Progressive JPEG (варіант алгоритму JPEG), реалізовуючих цю можливість, різко зросло. Те, що новий алгоритм (наприклад, wavelet) підтримує таку можливість, найістотніший плюс для нього сьогодні.

У той же час ми можемо розглянути таку рідкісну на сьогодні вимогу, як **стійкість до помилок**. Можна припустити, що з розповсюдженням широкомовлення в мережі Internet для його забезпечення використовуватимуться саме алгоритми, стійкі до помилок, що тільки починають розглядатися в сьогоднішніх статтях і оглядах.

Зі всіма зробленими вище обмовками, виділимо декілька найважливіших для нас критеріїв порівняння алгоритмів компресії, які і використовуватимемо надалі. Як легко помітити, ми обговорюватимемо менше критеріїв, ніж було

сформульовано вище. Це дозволить уникнути зайвих деталей при короткому викладі даного матеріалу.

1. **Гірший, середній і кращий ступінь стиснення.** Тобто частка, на яку зросте зображення, якщо початкові дані будуть найгіршими; якийсь середньостатистичний ступінь для того класу зображень, на який орієнтований алгоритм; і, нарешті, кращий ступінь. Останній необхідний лише теоретично, оскільки показує ступінь стиснення якнайкращого (як правило, абсолютно чорного) зображення, іноді фіксованого розміру.
2. **Клас зображень,** на який орієнтований алгоритм. Іноді вказано також, чому на інших класах зображень виходять гірші результати.
3. **Симетричність.** Відношення характеристики алгоритму кодування до аналогічної характеристики при декодуванні. Характеризує ресурсоємність процесів кодування і декодування. Для нас найважливішою є симетричність за часом: відношення часу кодування до часу декодування. Іноді нам буде потрібна симетричність по пам'яті.
4. **Чи є втрати якості?** І якщо є, то за рахунок чого змінюється ступінь стиснення? Річ у тому, що у більшості алгоритмів стиснення з втратою інформації існує можливість зміни ступеня стиснення.
5. **Характерні особливості алгоритму** і зображень, до яких його застосовують. Тут можуть вказуватися найважливіші для алгоритму властивості, які можуть стати визначаючими при його виборі.

Використовуючи дані критерії, приступимо до розгляду алгоритмів архівації зображень. Перш, ніж безпосередньо почати розмову про алгоритми, хотілося б сказати наступне. Один і той же алгоритм часто можна реалізувати різними способами. Багато відомих алгоритмів, такі як RLE, LZW або JPEG, мають десятки реалізацій, що розрізняються. Крім того, у алгоритмів буває декілька явних параметрів, варіюючи які, можна змінювати характеристики процесів архівації і розархівування. При конкретній реалізації ці параметри фіксуються, виходячи з найвірогідніших характеристик вхідних зображень, вимог на економію пам'яті, вимог на час архівації і т.д. Тому у алгоритмів одного сімейства кращий і гірший ступені стиснення можуть відрізнятися, але якісно картина не зміниться.

3. Короткий огляд алгоритмів стиснення зображень

Приступимо до розгляду методів стиснення зображень. Вони поділяються на методи стиснення без втрат якості та методи стиснення з втратами якості (рис.12.1).

Методи стиснення зображень без втрат складаються з наступних груп:

1. Методи, орієнтовані на просторові області зображення:
   - ✓ Алгоритми групового кодування (RLE)
   - ✓ Словникові алгоритми. Алгоритм LZW

- ✓ Алгоритм Хаффмана з фіксованою таблицею CCITT GROUP 3
- ✓ Алгоритм JBIG

2. Методи, орієнтовані на частотні області зображення:
   - ✓ Алгоритм Lossless JPEG;
   - ✓ Алгоритм хвильового стиснення WIC (Wavelet Image Compression).

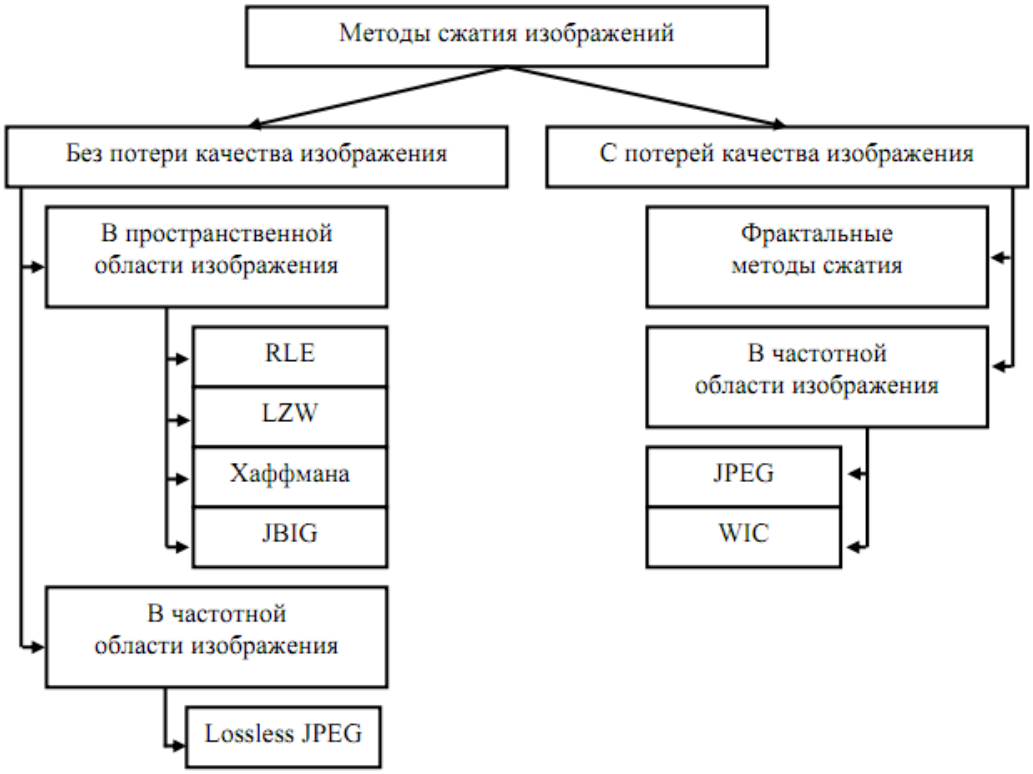

Рисунок 12.1 – Класифікація методів стиснення зображень

3.1 Алгоритми архівації без втрат

Алгоритм RLE *(Run Length Encoding)* — один з найстаріших і найпростіших алгоритмів архівації графіки. Зображення в ньому витягується в ланцюжок байт по рядках растра. Саме стиснення в RLE відбувається за рахунок того, що у зображенні зустрічаються *ланцюжки однакових байт*. Алгоритм орієнтований на зображення з невеликою кількістю кольорів: ділову та наукову графіку. До позитивних сторін алгоритму можна віднести тільки те, що він не вимагає додаткової пам'яті при архівації та розархівації, а також швидко працює.

Алгоритм LZW. Назву **алгоритм LZW** отримав за першими літерами прізвищ його розробників — Lempel, Ziv і Welch. Стиснення в ньому, на відміну від RLE, здійснюється за рахунок *однакових ланцюжків байт*. Процес стиснення виглядає досить просто. Ми зчитуємо послідовно символи вхідного потоку і перевіряємо, чи є у створеній нами таблиці рядків такий рядок. Якщо рядок є, то ми зчитуємо наступний символ, а якщо рядка немає, то ми заносимо в потік код

для попереднього знайденого рядка, заносимо рядок в таблицю і починаємо пошук знову. *Алгоритм LZW* орієнтований на 8-бітові зображення. *LZW* є універсальним алгоритмом - саме його варіанти використовуються у звичайних архіваторах.

Алгоритм Хаффмана — один з класичних алгоритмів, відомих з 60-х років. Використовує тільки частоту появи однакових байт в зображенні. Зіставляє символам вхідного потоку, які зустрічаються більше число раз, ланцюжок біт меншої довжини. І, навпаки, тим, які зустрічається рідко — ланцюжок більшої довжини. Для збору статистики вимагає двох проходів по зображенню. *Алгоритм Хаффмана* практично не застосовується до зображень у чистому вигляді. Зазвичай використовується як один з етапів стиснення в складніших схемах. Цей алгоритм реалізований у форматі **TIFF.** Він є надзвичайно простим у реалізації, швидким і може бути легко реалізований апаратно. Для стиснення чорно-білих зображень використовується алгоритм Хаффмана з фіксованою таблицею CCITT GROUP 3.

Алгоритм JBIG розроблений групою експертів ISO *(Joint Bi-level Experts Group)* спеціально для стиснення однобітних чорно-білих зображень, наприклад, факсів або відсканованих документів. Може також застосовуватися і до 2-х, і до 4-х бітових зображень. При цьому алгоритм розбиває їх на окремі бітові площини. JBIG дозволяє управляти такими параметрами, як порядок розбиття зображення на бітові площини, ширина смуг в зображенні, рівні масштабування.

Алгоритм Lossless JPEG розроблений групою експертів в області фотографії *(Joint Photographic Expert Group)*. На відміну від JBIG, Lossless JPEG орієнтований на повнокольорові 24-бітні або 8-бітові зображення в градаціях сірого без палітри. Він являє собою спеціальну реалізацію JPEG без втрат. Lossless JPEG рекомендується застосовувати в тих додатках, де необхідно побітова відповідність вхідного і декомпресованого зображень.

3.2 Алгоритми архівації з втратами

Алгоритм JPEG — один з найпоширеніших і досить потужних алгоритмів. Практично він є стандартом де-факто для повнокольорових зображень. Оперує алгоритм областями 8x8, на яких яскравість і колір змінюються порівняно плавно. Внаслідок цього, при розкладанні матриці такої області в подвійний ряд косинусів значущими виявляються тільки перші коефіцієнти. Таким чином, стиснення в JPEG здійснюється за рахунок плавності зміни кольорів у зображенні. Алгоритм розроблений групою експертів в області фотографії спеціально для стиснення 24-бітових зображень. JPEG — *Joint Photographic Expert Group* — підрозділ у рамках ISO — Міжнародної організації зі стандартизації. В цілому алгоритм заснований на дискретному косинусному перетворенні (надалі ДКП), що застосовується до матриці зображення для отримання деякої нової матриці коефіцієнтів. Для отримання початкового зображення застосовується зворотне перетворення.

Фрактальний алгоритм. Фрактальна архівація заснована на тому, що зображення представляють в компактнішій формі — з допомогою коефіцієнтів

системи ітерованих функцій (*Iterated Function System — IFS*). IFS являє собою набір тривимірних афінних перетворень, які переводять одне зображення в інше. Перетворенню підлягають точки в тривимірному просторі (x_координата, y_координата, яскравість).

Найвідоміші два зображення, отриманих за допомогою IFS: «трикутник Серпінського» і «папороть Барнслі». «Трикутник Серпінського» задається трьома, а «папороть Барнслі» чотирма афінними перетвореннями. Кожне перетворення кодується кількома байтами, в той час як зображення, побудоване з їх допомогою, може займати і декілька мегабайт.

Алгоритм використовують при стисненні повнокольорових 24 бітних зображень або зображень у градаціях сірого без різких переходів кольорів (фотографії). Бажано, щоб області більшої значимості (для сприйняття) були більш контрастними і різкими, а області меншої значимості — неконтрастними і розмитими.

Рекурсивний (хвильовий) алгоритм. Англійська **назва рекурсивного стиснення — wavelet**. Цей вид архівації відомий досить давно і безпосередньо виходить з ідеї використання когерентності областей. Орієнтований алгоритм на кольорові і чорно-білі зображення з плавними переходами. Ідеальний для картинок типу рентгенівських знімків. Коефіцієнт стиснення задається і варіюється в межах 5-100. До переваг цього алгоритму можна віднести те, що він дуже легко дозволяє реалізувати можливість поступового "проявлення" зображення при передачі зображення по мережі. На відміну від JPEG і фрактального алгоритму даний метод оперує блоками 2x2, 4x4, 8x8 і т.д. За рахунок того, що коефіцієнти для цих блоків зберігаються незалежно, можна досить легко уникнути дроблення зображення на "мозаїчні" квадрати.

В таблиці 12.2 наведені особливості зображення, за рахунок яких відбувається параметри різних алгоритмів стиснення зображень.

Таблиця12.2 – Параметри різних алгоритмів стиснення зображень

Алгоритм	Особливості зображення
RLE	Однакові кольори, що підряд йдуть: 2 2 2 2 2 2 15 15 15
LZW	Однакові підланцюжки: 2 3 15 40 2 3 15 40
Хаффмана	Різна частота появи кольору: 2 2 3 2 2 4 3 2 2 2 4
CCITT-3	Переважання білого кольору в зображенні, великі області, заповнені одним кольором
JPEG	Відсутність різких меж
Рекурсивний	Плавні переходи кольорів і відсутність різких меж
Фрактальний	Подібність між елементами зображення

В таблиці 12.3 наведені параметри різних алгоритмів стиснення зображень.

Таблиця 12.3 – Параметри різних алгоритмів стиснення зображень

Алгоритм	Коефіцієнти стиснення	Симетричність за часом	На що орієнтований	Втрати
RLE	32, 2, 0.5	1:1	3,4-х бітні	Немає
LZW	1000, 4, 5 / 7	1.5:1	1-8 бітні	Немає
Хаффмана	8, 1.5, 1	1-1.5:1	8 бітними	Немає
CCITT-3	213 (3), 5, 0.25	1:1	1-бітні	Немає
JBIG	2-30 разів	1:1	1-бітні	Немає
Lossless JPEG	2 рази	1:1	24-бітові, сірі	Немає
JPEG	2-20 разів	1:1	24-бітові, сірі	Є
Рекурсивний стиск	2-200 разів	1.5:1	24-бітові, сірі	Є
Фрактальний	2-2000 разів	100:1	24-бітові, сірі	Є

У приведеній таблиці виразно видно тенденції розвитку алгоритмів графіки останніх років:

- орієнтація на фотореалістичні зображення з 16 мільйонами кольорів (24 біти);
- використовування стиснення з втратами, можливість за рахунок втрат регулювати якість стислих зображень;
- використовування надмірності зображень в двох вимірюваннях;
- поява істотно несиметричних алгоритмів;
- все більша ступінь стиснення зображень.

Розвиток кодування зображень тісно пов'язаний з двома основними моментами:

1. Ростом швидкодії та ступеня інтеграції електронної елементної бази. Сучасні мікропроцесори з спеціальними апаратними засобами для підвищення швидкодії обробки зображень дозволяють реалізувати навіть програмним способом з прийнятною швидкістю ті методи кодування зображень, які раніше вважалися складними для промислового застосування. Прикладом може бути дискретне косинус не перетворення, яке стало основою стандартів JPEG та MJPEG. Але це не означає, що буде припинено пошук та дослідження більш простих методів кодування, так як нові можливості формують нові більш складні задачі, тому зменшення обчислювальної складності залишиться актуальним.
2. Зниженням собівартості електронної елементної бази. Проведення досліджень методів кодування рухомих і нерухомих зображень завжди вимагало складної і дорогої обчислювальної техніки, доступ до якої мала незначна кількість спеціалістів. Поява недорогих персональних комп’ютерів з розвинутими засобами відображення графічної інформації створила передумови збільшення

інтенсивності досліджень, в яких може брати участь більш широке коло дослідників, що дає надію на нові значні результати в майбутньому.

З урахуванням цього можна зробити такі висновки:

1. Великі зусилля будуть спрямовані на дослідження фрактального методу та його модифікацій, що можливо сприятиме появі нового стандарта ISO, основою якого буде фрактальний метод кодування зображень.

2. Будуть проводитись пошуки та дослідження нових методів кодування, які можуть забезпечити високий коефіцієнт стиснення і високу якість відновленого зображення, незважаючи на їх обчислювальну складність. З іншого боку будуть продовжені пошуки простих методів кодування, будуть досліджуватись комбінації цих методів з урахуванням нових технічних можливостей.

Контрольні питання

1. Назвіть основні вимоги додатків до алгоритмів компресії.
2. Що таке редагованість? Що таке симетричність?
3. Запропонуйте приклад свого класу додатків.
4. Назвіть критерії порівняння алгоритмів стиснення.
5. Приведіть приклади апаратних реалізацій алгоритму стиснення зображень, з якими вам доводилося стикатися (повсякденні і достатньо нові).
6. Чому висока швидкість компресії, висока якість зображень і високий ступінь компресії взаємно суперечливі? Покажіть суперечність кожної пари умов.
7. Як називається можливість зміни розмірів зображення до розмірів вікна активного додатку?
8. Як називається мінімальний ступінь погіршення якості зображення при його повторному збереженні після редагування?
9. Як називається відношення характеристики алгоритму кодування до аналогічної характеристики при декодуванні?
10. Назвіть алгоритм архівації зображень, стиснення в якому відбувається за рахунок ланцюжків однакових байт і орієнтований на зображення з невеликою кількістю кольорів.
11. Назвіть алгоритм архівації зображень, стиснення в якому здійснюється за рахунок однакових ланцюжків байт.
12. Назвіть алгоритм архівації зображень, що використовує тільки частоту появи однакових байт в зображенні і використовується як один з етапів стиснення в складних схемах.
13. Назвіть алгоритм архівації зображень, розроблений спеціально для стиснення однобітових чорно-білих зображень.
14. Назвіть алгоритм архівації зображень, орієнтований на повно кольорові зображення в градаціях сірого без палітри.

ЛЕКЦІЯ 13

на тему: Алгоритми архівації цифрових зображень без втрат

В лекції розглядаються алгоритми стиснення цифрових зображень без втрат, їх переваги і недоліки, сфери використання, а також дані приклади їх використовування в популярних форматах цифрового зображення.

План:

1. Алгоритми групового кодування (RLE)
2. Словникові алгоритми. Алгоритм LZW
3. Алгоритм Хаффмана з фіксованою таблицею CCITT GROUP 3
4. Алгоритм JBIG
5. Алгоритм Lossless JPEG

Стискання растрових і векторних даних здійснюється по-різному. В растрових файлах стискаються тільки дані зображення, а заголовок і решта даних (таблиця кольорів, кінцівка і т.п.) завжди залишаються нестисненими (вони, як правило, займають незначну частину растрового файла). Векторні файли, в яких зберігається математичний опис зображення, а не самі дані, як правило, не мають "рідної" форми стискання. Це викликано тим, що в векторному форматі дані вже представлені в компактній формі і стискання дає дуже незначний ефект. Окрім цього звичайно векторні дані читаються з незначною швидкістю і при додаванні розпаковування цей процес може стати ще більш повільним. Якщо векторний файл все ж стискається, то, як правило, стискаються всі дані, включаючи заголовок.

Методи стиснення без втрат використовуються в основному в наукових і медичних додатках, коли втрата інформації є неприпустимою або самі шуми зображення є головною інформацією, наприклад в системах оцінки якості оптико-електронних систем.

Методи стиснення зображень без втрат складаються з наступних груп:

1. Методи, орієнтовані на просторові області зображення:
   - ✓ Алгоритми групового кодування (RLE)
   - ✓ Словникові алгоритми. Алгоритм LZW
   - ✓ Алгоритм Хаффмана з фіксованою таблицею CCITT GROUP 3
   - ✓ Алгоритм JBIG
2. Методи, орієнтовані на частотні області зображення:
   - ✓ Алгоритм Lossless JPEG;
   - ✓ Алгоритм хвильового стиснення WIC (Wavelet Image Compression).

1. Алгоритми групового кодування (RLE)

Алгоритм RLE *(Run Length Encoding)* — один з найстаріших і найпростіших алгоритмів архівації графіки. Зображення в ньому витягується в ланцюжок байт по рядках растра. Саме стиснення в RLE відбувається за рахунок того, що у вхідному зображенні зустрічаються ланцюжки однакових байт. Заміна їх на пари <лічильник повторень, значення> зменшує надмірність даних.

Проблема всіх аналогічних методів полягає лише у визначенні способу, за допомогою якого алгоритм міг би відрізнити в результуючому потоці байтів кодовану серію від інших - некодованих послідовностей байтів.

Рішення проблеми досягається зазвичай проставлянням міток спочатку кодованих ланцюжків. Такими позначками можуть бути, наприклад, характерні значення бітів в першому байті кодованої серії, значення першого байта кодованої серії і т.п.

Дані методи, як правило, досить ефективні для стиснення растрових графічних зображень (BMP, PCX, TIFF), тому що останні містять досить велику кількість довгих серій повторюваних послідовностей байтів.

Недоліком методу RLE є досить низький ступінь стиснення або вартість кодування файлів з малим числом серій і, що ще гірше – з малим числом повторюваних байтів в серіях. До позитивних сторін алгоритму, мабуть, можна віднести тільки те, що він не вимагає додаткової пам'яті при роботі, і швидко виконується. Цікава особливість групового кодування в форматі PCX полягає в тому, що ступінь архівації для деяких зображень може бути істотно підвищений всього лише за рахунок зміни порядку кольорів в палітрі зображень.

1.1 Перший варіант алгоритму

У даному алгоритмі ознакою лічильника (counter) служать одиниці в двох верхніх бітах файлу (рис. 13.1).

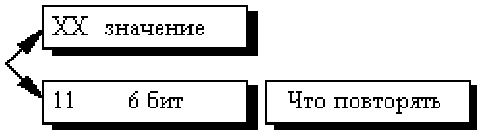

Рисунок 13.1 – Перший варіант алгоритму

6 біт, що відповідно залишилися, витрачаються на лічильник, який може приймати значення від 1 до 64. Рядок з 64 байтів, що повторюються, ми перетворюємо на два байти, тобто стиснемо в 32 рази.

Алгоритм розрахований на ділову графіку - зображення з великими областями кольору, що повторюється. Ситуація, коли файл збільшується, для цього простого алгоритму не така вже і рідкісна. Її можна легко одержати, застосовуючи групове кодування до оброблених кольорових фотографій. Для того,

щоб збільшити зображення в два рази, його треба застосувати до зображення, в якому значення всіх пікселів більше двійкового 11000000 і підряд попарно не повторюються.

Даний алгоритм реалізований у форматі PCX.

1.2 Другий варіант алгоритму

Другий варіант цього алгоритму має великий максимальний ступінь стиснення і менше збільшує в розмірах початковий файл.

Ознакою повтору в даному алгоритмі є одиниця в старшому розряді відповідного байта (рис. 13.2):

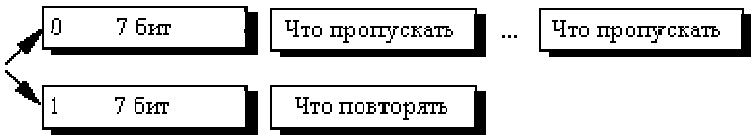

Рисунок 13.2 – Другий варіант алгоритму

Можна легко підрахувати, що в кращому разі цей алгоритм стискає файл в 64 рази (а не в 32 рази, як в попередньому варіанті), в гіршому збільшує на 1/128. Середні показники ступеня компресії даного алгоритму знаходяться на рівні показників першого варіанту.

Схожі схеми компресії використані як один з алгоритмів, підтримуваних форматом TIFF, а також у форматі TGA.

Характеристики алгоритму RLE:

Ступені стиснення: Перший варіант: 32, 2, 0,5. Другий варіант: 64, 3, 128/129. (Кращий, середній, гірший ступінь)

Клас зображень: Орієнтований алгоритм на зображення з невеликою кількістю кольорів: ділову і наукову графіку.

Симетричність: Приблизно одиниця.

Характерні особливості: До позитивних сторін алгоритму, мабуть, можна віднести тільки те, що він не вимагає додаткової пам'яті при архівації і розархівуванні, а також швидко працює. Цікава особливість групового кодування полягає в тому, що ступінь архівації для деяких зображень може бути істотно підвищений всього лише за рахунок зміни порядку кольорів в палітрі зображення.

2. Словникові алгоритми. Алгоритм LZW

Назву алгоритм LZW отримав за першими літерами прізвищ його розробників — Lempel, Ziv і Welch. Стиснення в ньому, на відміну від RLE, здійснюється за рахунок однакових ланцюжків байт. Алгоритм LZW орієнтований на 8-бітові зображення. *LZW* є універсальним алгоритмом - саме його варіанти використовуються у звичайних архіваторах.

2.1 Алгоритм LZ

Існує досить велике сімейство LZ-подібних алгоритмів, що розрізняються, наприклад, методом пошуку ланцюжків, що повторюються. Один з достатньо простих варіантів цього алгоритму, наприклад, припускає, що у вхідному потоці йде або пара <лічильник, зсув відносно поточної позиції>, або просто <лічильник> байт, що "пропускаються" і самі значення байтів (як в другому варіанті алгоритму RLE). При розархівуванні для пари <лічильник, зсув> копіюється <лічильник> байт з вихідного масиву, одержаного в результаті розархівування, на < зсув> байт, а <лічильник> (тобто число рівне лічильнику) значень байт, що "пропускаються", просто копіюється у вихідний масив з вхідного потоку.

Даний алгоритм є несиметричним за часом, оскільки вимагає повного перебору буфера при пошуку однакових підрядків. В результаті нам складно задати великий буфер через різке зростання часу компресії. Проте потенційна побудова алгоритму, в якому на <лічильник> і на <зсув> буде виділено по 2 байти (старший біт старшого байта лічильника - ознака повтору рядка / копіювання потоку), дасть нам можливість стискати всі підрядки, що повторюються, розміром до 32Кб в буфері розміром 64Кб (рис.13.3).

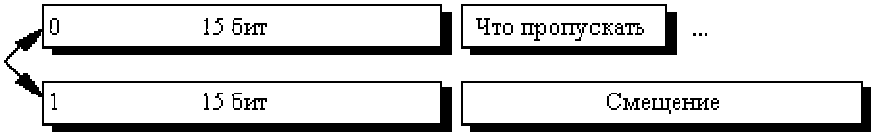

Рисунок 13.3 – Алгоритм LZ

При цьому ми одержимо збільшення розміру файлу у гіршому разі на 32770/32768 (в двох байтах записано, що потрібно переписати у вихідний потік наступні 215 байт), що зовсім непогано. Максимальний ступінь стиснення буде в межах 8192 рази. В межах, оскільки максимальне стиснення ми одержуємо, перетворюючи 32Кб буфера в 4 байти, а буфер такого розміру ми накопичимо не відразу. Проте, мінімальний підрядок, для якого нам вигідно проводити стиснення, повинен складатися в загальному випадку мінімум з 5 байт, що і визначає малу цінність даного алгоритму. До переваг LZ можна віднести надзвичайну простоту алгоритму декомпресії.

2.2 Алгоритм LZW

Цей варіант алгоритму використовує дерево для представлення і зберігання ланцюжків. Очевидно, що це достатньо сильне обмеження на вигляд ланцюжків, і далеко не всі однакові підланцюжки в нашому зображенні будуть використані при стисненні. Проте в цьому алгоритмі вигідно стискати навіть ланцюжки, що складаються з 2 байт.

Процес стиснення виглядає досить просто. Ми зчитуємо послідовно символи вхідного потоку і перевіряємо, чи є у створеній нами таблиці рядків такий рядок.

Якщо рядок є, то ми зчитуємо наступний символ, а якщо рядка немає, то ми заносимо в потік код для попередньо знайденого рядка, заносимо рядок в таблицю і починаємо пошук знову.

Приклад

Хай ми стискаємо послідовність 45, 55, 55, 151, 55, 55, 55. Тоді, згідно викладеному вище алгоритму, ми помістимо у вихідний потік спочатку код очищення <256>, потім додамо до спочатку порожнього рядка "45" і перевіримо, чи є рядок "45" в таблиці. Оскільки ми при ініціалізації занесли в таблицю всі рядки з одного символу, то рядок "45" є в таблиці. Далі ми читаємо наступний символ 55 з вхідного потоку і перевіряємо, чи є рядок "45, 55" в таблиці. Такого рядка в таблиці поки немає. Ми заносимо в таблицю рядок "45, 55" (з першим вільним кодом 258) і записуємо в потік код <45>. Можна коротко представити архівацію так:

"45" - є в таблиці;

"45, 55" - ні. Додаємо в таблицю <258>"45, 55". В потік: <45>;

"55, 55" - ні. В таблицю: <259>"55, 55". В потік: <55>;

"55, 151" - ні. В таблицю: <260>"55, 151". В потік: <55>;

"151, 55" - ні. В таблицю: <261>"151, 55". В потік: <151>;

"55, 55" - є в таблиці;

"55, 55, 55" - ні. В таблицю: "55, 55, 55" <262>. В потік: <259>;

Послідовність кодів для даного прикладу, що потрапляють у вихідний потік: <256> <45> <55> <55> <151> <259>.

Особливість LZW полягає в тому, що для декомпресії нам не треба зберігати таблицю рядків у файл для розпаковування. Алгоритм побудований таким чином, що ми в змозі відновити таблицю рядків, користуючись тільки потоком кодів.

Зауваження 1. Як ви могли помітити, записувані в потік коди поступово зростають. До тих пір, поки в таблиці не з'явиться, наприклад, вперше код 512, всі коди будуть менше 512. Крім того, при компресії і при декомпресії коди в таблиці додаються при обробці одного і того ж символу, тобто це відбувається "синхронно". Ми можемо скористатися цією властивістю алгоритму для того, щоб підвищити ступінь компресії. Поки в таблицю не додано 512 символ, ми писатимемо у вихідний бітовий потік коди з 9 біт, а відразу при додаванні 512 - коди з 10 біт. Відповідно декомпресор також повинен буде сприймати всі коди вхідного потоку 9-бітовими до моменту додавання в таблицю коду 512, після чого сприйматиме всі вхідні коди як 10-бітові. Аналогічно ми поступатимемо при додаванні в таблицю кодів 1024 і 2048. Даний прийом дозволяє приблизно на 15% підняти ступінь компресії (рис. 13.4):

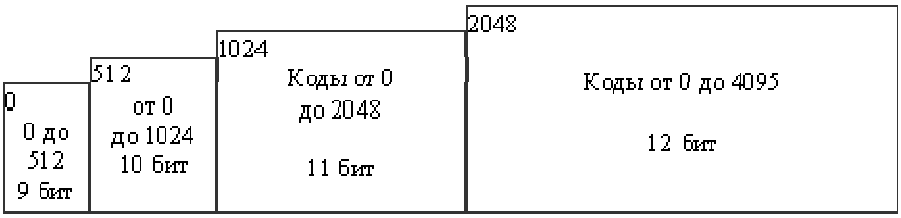

Рисунок 13.4 – Записувані в потік коди

Зауваження 2. При стисненні зображення нам важливо забезпечити швидкість пошуку рядків в таблиці. Ми можемо скористатися тим, що кожний наступний підрядок на один символ довше попередньою, крім того, попередній рядок вже був нами знайдений в таблиці. Отже, достатньо створити список посилань на рядки, що починаються з даного підрядка, як весь процес пошуку в таблиці зведеться до пошуку в рядках, що містяться в списку для попереднього рядка. Зрозуміло, що така операція може бути проведена дуже швидко.

Помітимо також, що реально нам достатньо зберігати в таблиці тільки пару <код попереднього підрядка, доданий символ>. Цій інформації цілком достатньо для роботи алгоритму. Таким чином, масив від 0 до 4095 з елементами <код попереднього підрядка; доданий символ; список посилань на рядки, що починаються з цього рядка> вирішує поставлену задачу пошуку, хоча і дуже повільно.

На практиці для зберігання таблиці використовується таке ж швидке, як у разі списків, але більш компактне по пам'яті рішення - хеш-таблиця. Таблиця складається з 8192 (213) елементів. Кожний елемент містить <код попереднього підрядка; доданий символ; код цього рядка>. Ключ для пошуку завдовжки в 20 біт формується з використанням двох перших елементів, збережених в таблиці як одне число (key). Молодші 12 біт цього числа віддано під код, а наступні 8 біт під значення символу.

Як хеш-функція при цьому використовується:

Index(key)= ((key >> 12) ^ key) & 8191;

Де >> - побітовий зсув управо (key >> 12 - ми отримаємо значення символу) ^ - логічна операція побітового виключаючого АБО, & логічне побітове І.

Таким чином, за лічену кількість порівнянь ми одержуємо шуканий код або повідомлення, що такого коду в таблиці немає.

Підрахуємо кращий і гірший ступінь стиснення для даного алгоритму. Краще стиснення, очевидно, буде одержано для ланцюжка однакових байт великої довжини (тобто для 8-бітового зображення, всі точки якого мають для визначеності колір 0). При цьому в 258 рядок таблиці ми запишемо рядок "0, 0", в 259 - "0, 0, 0" ... в 4095 - рядок з 3839 (=4095-256) нулів. При цьому в потік потрапить 3840 кодів, включаючи код очищення. Отже, порахувавши суму арифметичної прогресії від 2 до 3839 (тобто довжину стислого ланцюжка) і

поділивши її на 3840*12/8 (в потік записуються 12-бітові коди), ми одержимо кращий ступінь стиснення.

Найгірше стиснення буде одержано, якщо ми жодного разу не зустрінемо підрядок, який вже є в таблиці (в ньому не повинно зустрітися жодної однакової пари символів).

У випадку, якщо ми постійно зустрічатимемо новий підрядок, ми запишемо у вихідний потік 3840 кодів, яким відповідатиме рядок з 3838 символів. Без урахування зауваження 1 це складе збільшення файлу майже в 1.5 рази.

LZW реалізований у форматах GIF і TIFF.

Характеристики алгоритму LZW:

Ступені стиснення: Приблизно 1000, 4, 5/7 (Краще, середнє, гірше стиснення). Стиснення в 1000 разів досягається тільки на одноколірних зображеннях розміром кратним приблизно 7 Мб.

Клас зображень: Орієнтований LZW на 8-бітові зображення, побудовані на комп'ютері. Стискає за рахунок однакових підланцюжків в потоці.

Симетричність: Майже симетричний, за умови оптимальної реалізації операції пошуку рядка в таблиці.

Характерні особливості: Ситуація, коли алгоритм збільшує зображення, зустрічається украй рідко. LZW універсальний - саме його варіанти використовуються в звичайних архіваторах.

3. Алгоритм Хаффмана з фіксованою таблицею CCITT GROUP 3

Алгоритм Хаффмана — один з класичних алгоритмів, відомих з 60-х років. Використовує тільки частоту появи однакових байт в зображенні. Зіставляє символам вхідного потоку, які зустрічаються більше число раз, ланцюжок біт меншої довжини. І, навпаки, тим, які зустрічається рідко — ланцюжок більшої довжини. Для збору статистики вимагає двох проходів по зображенню.

Класичний алгоритм Хаффмана був розглянутий в попередніх лекціях. Він практично не застосовується до зображень в чистому вигляді, а використовується як один з етапів компресії в складніших схемах.

Близька модифікація алгоритму використовується при стисненні чорно-білих зображень (один біт на піксел). Повна назва даного алгоритму CCITT Group 3. Це означає, що даний алгоритм був запропонований третьою групою по стандартизації Міжнародного Консультаційного Комітету з Телеграфії і Телефонії (Consultative Committee International Telegraph and Telephone).

Це простий алгоритм стиску, запропонований Девідом Хаффманом у 1952 році. Стандарти Group 3 і Group 4 – це алгоритми стиску, спеціально розроблені для кодування однобітових даних зображення. Алгоритми CCITT не є адаптивними, тобто не настоюються для кодування кожного растра з оптимальною ефективністю. У них використовується фіксована таблиця кодових

значень, що були обрані спеціально для представлення документів, які підлягають факсимільній передачі. Перед початком кодування здійснюється частотний аналіз коду зображення і виявляється частота повтору кожного з символів. Символи, які частіше зустрічаються, кодуються меншою кількістю розрядів. При використанні кодування за схемою Хаффмана треба разом із закодованим текстом передати відповідний алфавіт, але для великих фрагментів надлишковість не може бути значною.

Послідовності чорних і білих крапок в ньому замінюються числом, рівним їх кількості. А цей ряд, вже у свою чергу, стискається по Хаффману з фіксованою таблицею.

Набір точок зображення одного кольору, що йдуть підряд, називається серією. Довжина цього набору крапок називається **довжиною серії**.

Приклад

У таблиці задано два види кодів:

- *Коди завершення серій* - задані з 0 до 63 з кроком 1.
- *Складові (додаткові) коди* - задані з 64 до 2560 з кроком 64.

Кожний рядок зображення стискається незалежно. Ми вважаємо, що в нашому зображенні істотно переважає білий колір, і всі рядки зображення починаються з білої крапки. Якщо рядок починається з чорної крапки, то ми вважаємо, що рядок починається білою серією з довжиною 0. Наприклад, послідовність довжин серій 0, 3, 556, 10 ... означає, що в цьому рядку зображення йдуть спочатку 3 чорні крапки, потім 556 білих, потім 10 чорних і т.д.

На практиці в тих випадках, коли в зображенні переважає чорний колір, ми інвертуємо зображення перед компресією і записуємо інформацію про це в заголовок файлу.

Оскільки чорні і білі серії чергують, то реально код для білої і код для чорної серії працюватимуть поперемінно.

У термінах регулярних виразів ми одержимо для кожного рядка нашого зображення (достатньо довгого, що починається з білої крапки) вихідний бітовий потік вигляду:

((<Б-2560>)*[<Б-сст.>]<Б-зв.>(<Ч-2560>)*[<Ч-сст.>]<Ч-зв.>)+

[(<Б-2560>)*[<Б-сст.>]<Б-зв.>]

Де ()* - повтор 0 або більш раз ()+.- повтор 1 або більш раз [] - включення 1 або 0 разів.

Для приведеного раніше прикладу: 0, 3, 556, 10... алгоритм сформує наступний код: <Б-0><Ч-3><Б-512><Б-44><Ч-10>, або, згідно таблиці, 00110101**10**0110010100101101**0000100** (різні коди в потоці виділені для зручності). Цей код володіє властивістю префіксних кодів і легко може бути згорнутий назад в послідовність довжин серій. Легко підрахувати, що для приведеного рядка в 569 біт ми одержали код завдовжки в 33 біти, тобто ступінь стиснення складає приблизно 17 разів.

На рис. 13.5 наведені приклади зображень, для яких вигідно і невигідно використання алгоритму CCITT-3. Зображення, для якого дуже вигідне використання алгоритму CCITT-3 (великі області заповнені одним кольором) - зліва. Зображення, для якого менш вигідне використання алгоритму CCITT-3 (менше областей, заповнених одним кольором. Багато коротких "чорних" і "білих" серій) - справа.

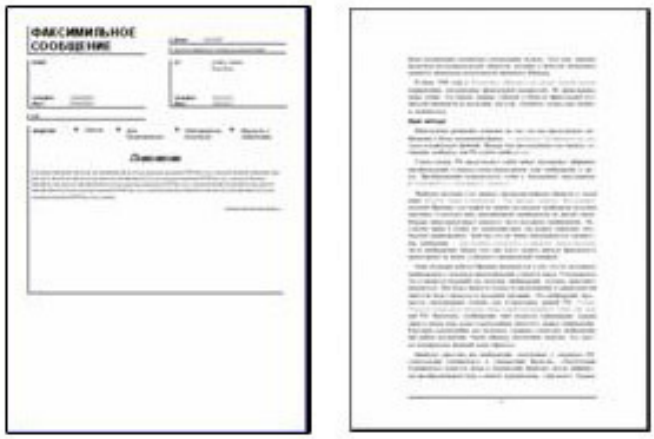

Рисунок 13.5 – Приклади зображень, для яких вигідно і невигідно використання алгоритму CCITT-3

Помітимо, що єдиний "складний" вираз в нашому алгоритмі: L2=МаксимальныйДопКодМеньшеL(L) - на практиці працює дуже просто: L2=(L>>6)*64, де >> - побітовий зсув L вліво на 6 бітів (можна зробити те ж саме за одну побітову операцію & - логічне І).

Цей алгоритм реалізований у форматі TIFF.

Характеристики алгоритму CCITT Group 3

Ступені стиснення: краща прагне до 213.(3), середня 2, у гіршому разі збільшує файл в 5 разів.

Клас зображень: Двокольорові чорно-білі зображення, в яких переважають великі простори, заповнені білим кольором.

Симетричність: Близька до 1.

Характерні особливості: Даний алгоритм надзвичайно простий в реалізації, швидкий і може бути легко реалізований апаратний.

4. Алгоритм JBIG

Алгоритм розроблений групою експертів ISO (Joint Bi-level Experts Group) спеціально для стиснення однобітових чорно-білих зображень. Наприклад, факсів або відсканованих документів. У принципі, може застосовуватися і до 2-х, і до 4-х бітових картинок. При цьому алгоритм розбиває їх на окремі бітові площини.

JBIG дозволяє управляти такими параметрами, як порядок розбиття зображення на бітові площини, ширина смуг в зображенні, рівні масштабування. Остання можливість дозволяє легко орієнтуватися в базі великих за розмірами зображень, проглядаючи спочатку їх зменшені копії. Настроюючи ці параметри, можна використовувати описаний вище ефект "огрубленого зображення" при отриманні зображення по мережі або по будь-якому іншому каналу, пропускна спроможність якого мала в порівнянні з можливостями процесора. Розпаковуватися зображення на екрані буде поступово, як би поволі "виявляючись". При цьому людина починає аналізувати картинку задовго до кінця процесу розархівування.

Алгоритм побудований на базі Q-кодувальника, патентом на який володіє IBM. Q-кодер, так само як і алгоритм Хаффмана, використовує для символів, що частіше з'являються, короткі ланцюжки, а для тих, що рідше з'являються - довгі. Проте, на відміну від нього, в алгоритмі використовуються і послідовності символів.

Характеристики алгоритму JBIG

Ступені стиснення: В 2 - 30 разів.

Клас зображень: Однобітові чорно-білі зображення.

Симетричність: Близька до 1.

Характерні особливості: Характерною особливістю JBIG є різке зниження ступеня стиснення при підвищенні рівня шумів вхідного зображення.

5. Алгоритм Lossless JPEG

Цей алгоритм розроблений групою експертів в області фотографії (Joint Photographic Expert Group). На відміну від JBIG, Lossless JPEG орієнтований на повнокольорові 24-бітові або 8-бітові в градаціях сірого зображення без палітри. Він є спеціальною реалізацією JPEG без втрат.

Lossless JPEG використовує схему передбачення значення пікселя по трьох найближчих сусідніх пікселах, а для стиснення різниці між істинним і передбаченим значенням пікселя використовує ентропійне кодування. Алгоритм стиснення Lossless JPEG не передбачає ні адаптивного передбачення значення кодованого пікселя, ні контекстного моделювання помилки передбачення. Для ентропійного кодування помилки передбачення Lossless JPEG використовує код Хаффмана. В якості альтернативного стандарт допускає використання арифметичного кодування, однак, через патентні обмеження воно не знайшло застосування в практичних реалізаціях Lossless JPEG.

Метод стиснення без втрат, який використовується в стандарті lossless JPEG заснований на методі різницевого (диференціального) кодування. Основна ідея диференціального кодування полягає в наступному. Зазвичай зображення характеризуються сильною кореляцією між точками зображення. Цей факт враховується при різницевому кодуванні, а саме, замість стиснення послідовності

точок зображення x_1, x_2, ... x_N, стиску піддається послідовність різниць $y_i = x_i - x_{i-1}$, i=1,2, ... N, $x_0 = 0$. Числа y_i називають помилками передбачення x_i. У стандарті losslessJPEG передбачено формування помилок передбачення з використанням попередніх закодованих точок в поточному рядку і \ або в попередньому рядку.

Характеристики алгоритму Lossless JPEG

Ступені стиснення: 20, 2, 1.

Клас зображень: Повнокольорові 24-бітові або 8-бітові в градаціях сірого зображення без палітри.

Симетричність: Близька до 1.

Характерні особливості: Lossless JPEG рекомендується застосовувати в тих додатках, де необхідна побітова відповідність початкового і декомпресованого зображень.

З одного боку, приведені вище алгоритми достатньо універсальні і покривають всі типи зображень, з іншою - у них, за сьогоднішніми мірками, дуже маленький ступінь стиснення. Використовуючи один з алгоритмів стиснення без втрат, можна забезпечити архівацію зображення приблизно в два рази. В той же час алгоритми стиснення з втратами оперують з коефіцієнтами 10-200 разів. Крім можливості модифікації зображення, одна з основних причин подібної різниці полягає в тому, що традиційні алгоритми орієнтовані на роботу з ланцюжком. Вони не враховують, так звану, "когерентність областей" в зображеннях. Ідея когерентності областей полягає в малій зміні кольору і структури на невеликій ділянці зображення. Всі алгоритми, про які мова піде в наступних лекціях, були створені пізніше спеціально для стиснення графіки і використовують цю ідею.

Слід зазначити, що і в класичних алгоритмах можна використовувати ідею когерентності. Існують алгоритми обходу зображення по "фрактальній" кривій, при роботі яких воно також витягується в ланцюжок; але за рахунок того, що крива оббігає області зображення по складній траєкторії, ділянки близьких кольорів в вихідному ланцюжку подовжуються.

Контрольні питання

1. На який клас зображень орієнтований алгоритм RLE?
2. Приведіть два приклади "поганих" зображень для першого варіанту алгоритму RLE, для яких файл максимально збільшиться в розмірі.
3. На який клас зображень орієнтований алгоритм CCITT G-3?
4. Приведіть приклад "поганого" зображення для алгоритму CCITT G-3, для якого файл максимально збільшиться в розмірі.
5. Приведіть приклад "поганого" зображення для алгоритму Хаффмана.
6. Порівняйте алгоритми стиснення зображень без втрат.
7. У чому полягає ідея когерентності областей?

8. Вкажіть алгоритм, що не відноситься до методів, орієнтованих на просторові області зображення.

9. Вкажіть алгоритм, що відноситься до методів, орієнтованих на частотні області зображення.

10. Назвіть алгоритм, стиснення в якому відбувається за рахунок заміни ланцюжків однакових байт у зображення на пари <лічильник повторень, значення>.

11. Назвіть алгоритм, орієнтований на 8-бітові зображення, стиснення в якому відбувається за рахунок однакових ланцюжків байт.

12. Назвіть алгоритм стиснення зображень, що використовує тільки частоту появи однакових байт в зображенні називається.

13. Назвіть алгоритм стиснення одно бітових чорно-білих зображень, що використовує не тільки частоту появи однакових байт в зображенні, але й послідовності байт називається.

14. Назвіть алгоритм стиснення зображень, що використовує схему передбачення значення пікселя по трьох найближчих сусідніх пік селях, а для стиснення різниці між істинним і передбаченим значенням пікселя використовує ентропійне кодування називається.

15. Вкажіть зображення, для якого вигідно використовувати алгоритм CCITT GROUP 3?

16. Вкажіть зображення, для якого не вигідно використовувати алгоритм CCITT GROUP 3?

17. Назвіть особливості зображення, за рахунок яких відбувається стиснення для алгоритму групового кодування.

18. Назвіть особливості зображення, за рахунок яких відбувається стиснення для словникового алгоритму LZW.

19. Назвіть особливості зображення, за рахунок яких відбувається стиснення для алгоритму Хаффмана з фіксованою таблицею CCITT GROUP 3.

ЛЕКЦІЯ 14

на тему: Руйнівні алгоритми стиснення цифрових зображень. Алгоритм JPEG

У першій частині лекції розповідається про один з найпопулярніших форматів стислого зображення – JPEG, основні переваги цього формату, а також послідовні кроки, що використовуються в алгоритмі для отримання стислого зображення. Частина лекції, що залишилася, дає вичерпну інформацію про формат JPEG-2000, про його відмітні особливості, сфери використання і алгоритм архівації зображення.

План:

1. Критерії оцінки втрат якості зображень
2. Алгоритм JPEG
3. Алгоритм JPEG 2000

1. Критерії оцінки втрат якості зображень

Першими для архівації зображень стали застосовуватися звичні алгоритми, ті, що використовувалися і використовуються в системах резервного копіювання, при створенні дистрибутивів і т.п. Ці алгоритми архівували інформацію без змін. Проте основною тенденцією останнім часом стало використання нових класів зображень. Старі алгоритми перестали задовольняти вимогам, що пред'являються до архівації. Багато зображень практично не стискалися, хоча "на перший погляд" володіли явною надмірністю. Це привело до створення нового типу алгоритмів - стискаючих з втратою інформації. Як правило, ступінь стиснення і, отже, ступінь втрат якості в них можна задавати. При цьому досягається компроміс між розміром і якістю зображень.

Одна з серйозних проблем машинної графіки полягає в тому, що дотепер не знайдений адекватний критерій оцінки втрат якості зображення. А втрачається вона постійно - при оцифровці, при переведенні в обмежену палітру кольорів, при переведенні в іншу систему кольоропредставлення для друку, і, що для нас особливо важливо, при архівації з втратами. Можна привести приклад простого критерію: середньоквадратичне відхилення значень пікселів (root mean square - RMS):

$$d(x,y) = \sqrt{\frac{\sum_{i=1,j=1}^{n,n} (x_{ij}-y_{ij})^2}{n^2}}$$

Згідно цього критерію зображення буде сильно зіпсовано при пониженні яскравості всього на 5% (око цього не помітить - у різних моніторів налаштування яскравості варіюється набагато сильніше). В той же час зображення з "снігом" -

різкою зміною кольору окремих крапок, слабкими смугами або "муаром" будуть визнані такими, що "майже не змінилися". Свої неприємні сторони є і у інших критеріїв.

Розглянемо, наприклад, максимальне відхилення:

$$d(x,y) = \max_{i,j} |x_{ij} - y_{ij}|$$

Цей показник украй чутливий до розбиття окремих пікселів. Тобто у всьому зображенні може істотно змінитися тільки значення одного пікселя (що практично непомітне для ока), проте згідно цього показника зображення буде сильно зіпсовано.

Показник, який зараз використовують на практиці, називається мірою відношення сигналу до шуму (peak-to-peak signal-to-noise ratio - PSNR).

$$d(x,y) = 10 \cdot \log_{10} \frac{255^2 \cdot n^2}{\sum_{i=1,j=1}^{n,n} (x_{ij} - y_{ij})^2}$$

Даний показник, по суті, аналогічний середньоквадратичному відхиленню, проте користуватися ним дещо зручніше за рахунок логарифмічного масштабу шкали. Їй властиві ті ж недоліки, що і середньоквадратичному відхиленню.

Краще всього втрати якості зображень оцінюють наші очі. Відмінною вважається архівація, при якій неможливо на око розрізнити первинне і розархівоване зображення. Хорошою – коли сказати, яке із зображень піддавалося архівації, можна тільки порівнюючи дві картинки, що знаходяться поряд. При подальшому збільшенні ступеня стиснення, як правило, стають помітні побічні ефекти, характерні для даного алгоритму (задовільна архівація). На практиці, навіть при відмінному збереженні якості, в зображення можуть бути внесені регулярні специфічні зміни. Тому алгоритми архівації з втратами не рекомендується використовувати при стисненні зображень, які надалі збираються або друкувати з високою якістю, або обробляти програмами розпізнавання образів. Неприємні ефекти з такими зображеннями, як ми вже говорили, можуть виникнути навіть при простому масштабуванні зображення.

2. Алгоритм JPEG

JPEG - один з найновіших і достатньо могутніх алгоритмів. Практично він є стандартом де-факто для повнокольорових зображень. Оперує алгоритм областями 8x8, на яких яскравість і колір міняються порівняно плавно. Внаслідок цього, при розкладанні матриці такої області в подвійний ряд по косинусах значущими виявляються тільки перші коефіцієнти. Таким чином, стиснення в JPEG здійснюється за рахунок плавності зміни кольорів в зображенні.

Алгоритм розроблений групою експертів в області фотографії спеціально для стиснення 24-бітових зображень. JPEG - Joint Photographic Expert Group - підрозділ в рамках ISO - Міжнародної організації по стандартизації. Назва алгоритму читається ['jei'peg]. В цілому алгоритм заснований на дискретному косинусоїдальному перетворенні (надалі ДКП), що використовується до матриці зображення для отримання деякої нової матриці коефіцієнтів. Для отримання початкового зображення застосовується зворотне перетворення.

ДКП розкладає зображення по амплітудах деяких частотних компонент. Таким чином, при перетворенні ми одержуємо матрицю, в якій більшість коефіцієнтів або близькі, або рівні нулю. Крім того, завдяки недосконалості людського зору, можна апроксимувати коефіцієнти більш грубо без помітної втрати якості зображення.

Для цього використовується квантування коефіцієнтів (quantization). В найпростішому випадку - це арифметичний побітовий зсув управо. При цьому перетворенні втрачається частина інформації, але може досягатися великий ступінь стиснення.

Базова схема (як працює алгоритм)

Отже, розглянемо алгоритм докладніше. Нехай ми стискаємо 24-бітове зображення.

Крок 1. RGB в YCrCb (YUV)

Переводимо зображення з колірного простору RGB, з компонентами, що відповідають за червону (Red), зелену (Green) і синю (Blue) складові кольору крапки, в колірний простір YCrCb (іноді називають YUV).

У ньому Y - складова яскравості, а Cr, Cb - компоненти, що відповідають за колір (хроматичний червоний і хроматичний синій). За рахунок того, що людське око менш чутливе до кольору, ніж до яскравості, з'являється можливість архівувати масиви для Cr і Cb компонент з великими втратами і, відповідно, великими ступенями стиснення. Подібне перетворення вже давно використовується в телебаченні. На сигнали, що відповідають за колір, там виділяється більш вузька смуга частот.

Спрощений переклад з колірного простору RGB в колірний простір YCrCb можна представити за допомогою матриці переходу:

$$|YCbCr| = |0.2990.5870.1140.5-0.4187-0.08130.1687-0.33130.5| * |RGB|+|0128128|$$

Зворотне перетворення здійснюється множенням вектора YUV на зворотну матрицю.

$$|RGB| = |101.4021-0.34414-0.7141411.7720| * (|YCbCr| - |0128128|)$$

Крок 2. Дискретизація YUV

Розбиваємо початкове зображення на матриці 8x8. Формуємо з кожної три робочі матриці ДКП - по 8 біт окремо для кожної компоненти. При великих ступенях стиснення цей крок може виконуватися трохи складніше. Зображення ділиться по компоненті Y - як і в першому випадку, а для компонент Cr і Cb матриці набираються через рядок і через стовпець. Тобто з початкової матриці розміром 16x16 виходить тільки одна робоча матриця ДКП. При цьому, як неважко помітити, ми втрачаємо 3/4 корисної інформації про колірні складові зображення і одержуємо відразу стиснення в два рази. Ми можемо поступати так завдяки роботі в просторі YCrCb. На результуючому RGB зображенні, як показала практика, це позначається несильно.

Крок 3. ДКП (дискретне косинусоїдальне перетворення)

У спрощеному вигляді ДКП при n=8 можна представити так:

$$Y[u,v] = \frac{1}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} C(i,u) \times C(j,v) \times y[i,j]$$

де

$$C(i,u) = A(u) \times \cos\left(\frac{(2\times i+1)\times u\times \pi}{2\cdot n}\right)$$

$$A(u) = \begin{cases} \frac{1}{\sqrt{2}}, \text{for } u = 0 \\ 1, \text{for } u \neq 0 \end{cases}$$

Застосовуємо ДКП (дискретне косинусоїдальне перетворення) до кожної робочої матриці. При цьому ми одержуємо матрицю, в якій коефіцієнти в лівому верхньому кутку відповідають низькочастотній складовій зображення, а в правому нижньому - високочастотній. Поняття частоти виходить з розгляду зображення як двовимірного сигналу (аналогічно розгляду звуку як сигналу). Плавна зміна кольору відповідає низькочастотній складовій, а різкі скачки - високочастотній.

Крок 4. Квантування

Проводимо квантування. У принципі, це просто розподіл робочої матриці на матрицю квантування поелементно. Для кожної компоненти (Y, U і V), в загальному випадку, задається своя матриця квантування q[u,v] (далі МК).

$$Yq[u,v] = \text{IntegerRound}\left(\frac{Y[u,v]}{q[u,v]}\right)$$

На цьому кроці здійснюється управління ступенем стиснення, і відбуваються найбільші втрати. Зрозуміло, що, задаючи МК з великими коефіцієнтами, ми одержимо більше нулів і, отже, великий ступінь стиснення.

У стандарт JPEG включені рекомендовані МК, побудовані дослідним шляхом. Матриці для більшого або меншого ступеня стиснення одержують шляхом множення початкової матриці на деяке число gamma.

З квантуванням пов'язані і специфічні ефекти алгоритму. При великих значеннях коефіцієнта gamma втрати в низьких частотах можуть бути настільки

великі, що зображення розпадеться на квадрати 8x8. Втрати у високих частотах можуть виявитися в так званому "ефекті Гіббса", коли навкруги контурів з різким переходом кольору утворюється своєрідний "німб".

Крок 5. Зігзаг – сканування

Переводимо матрицю 8x8 в 64-елементний вектор за допомогою зігзаг-сканування, тобто беремо елементи з індексами (0,0), (0,1), (1,0) (2,0)... (рис. 14.1)

Таким чином, на початку вектора ми одержуємо коефіцієнти матриці, що відповідають низьким частотам, а в кінці - високим.

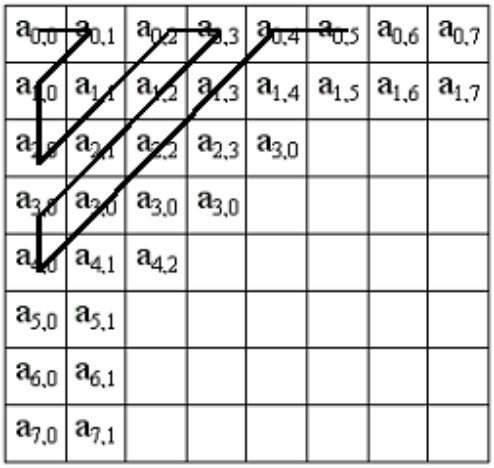

Рисунок 14.1 – Зігзаг-сканування

Крок 6. RLE – групове кодування

Згортаємо вектор за допомогою алгоритму групового кодування. При цьому одержуємо пари типу (пропустити, число), де "пропустити" є лічильником нулів, що пропускаються, а "число" - значення, яке необхідне поставити в наступну чарунку. Так, вектор 42 3 0 0 0 -2 0 0 0 0 1 ... буде згорнутий в пари (0,42) (0,3) (3, -2) (4,1) ... .

Крок 7. Стиснення по Хаффману

Згортаємо пари, що вийшли, кодуванням по Хаффману з фіксованою таблицею.

Процес відновлення зображення в цьому алгоритмі повністю симетричний. Метод дозволяє стискати деякі зображення в 10-15 разів без серйозних втрат (рис. 14.2).

Істотними позитивними сторонами алгоритму є те, що:

1. Задається ступінь стиснення.
2. Вхідне кольорове зображення може мати 24 біти на крапку.

Негативними сторонами алгоритму є те, що:

1. При підвищенні ступеня стиснення зображення розпадається на окремі квадрати (8x8). Це позв'язано з тим, що відбуваються великі втрати в низьких частотах при квантуванні, і відновити початкові дані стає неможливо.
2. Проявляється ефект Гіббса - ореоли по межах різких переходів кольорів

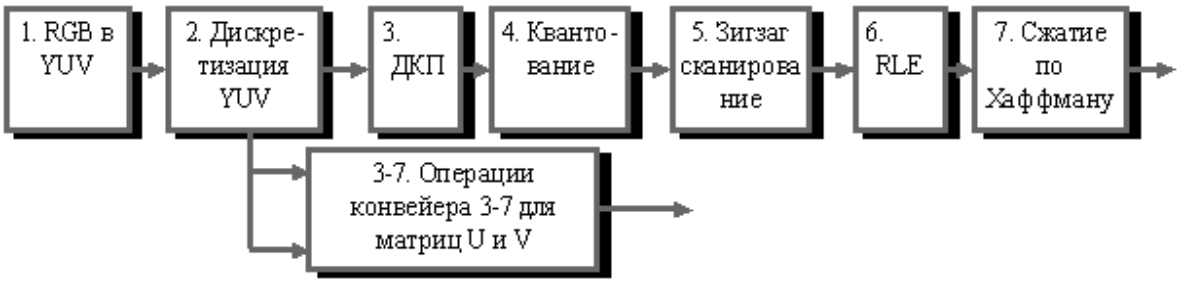

Рисунок 14.2 – Конвейєр операцій, що використовується в алгоритмі JPEG

Отже, стандартизований JPEG відносно недавно - в 1991 році. Але вже тоді існували алгоритми, що стискають сильніше при менших втратах якості. Річ у тому, що дії розробників стандарту були обмежені потужністю існуючої на той момент техніки. Тобто навіть на персональному комп'ютері алгоритм повинен був працювати менше хвилини на середньому зображенні, а його апаратна реалізація повинна бути простою і дешевою. Алгоритм повинен був бути симетричним (час розархівування приблизно рівний часу архівації).

Виконання останньої вимоги зробило можливим появу таких пристроїв, як цифрові фотоапарати, що знімають 24-бітові фотографії на флеш-карту. Потім ця карта вставляється в роз'єм на вашому ноутбуку і відповідна програма дозволяє зчитати зображення. Якби алгоритм був несиметричний, неприємно було б довго чекати, поки апарат "перезарядиться" - стисне зображення.

Не дуже приємною властивістю JPEG є також те, що нерідко горизонтальні і вертикальні смуги на дисплеї абсолютно не видні і можуть виявитися тільки при друці у вигляді муарового узору. Він виникає при накладенні нахиленого растру друку на горизонтальні і вертикальні смуги зображення. Через ці сюрпризи JPEG не рекомендується активно використовувати в поліграфії, задаючи високі коефіцієнти матриці квантування. Проте при архівації зображень, призначених для перегляду людиною, він на даний момент незамінний.

Широке використання JPEG довгий час стримувалося, мабуть, лише тим, що він оперує 24-бітовими зображеннями. Тому для того, щоб з прийнятною якістю подивитися картинку на звичайному моніторі в 256-кольоровій палітрі, було потрібне використання відповідних алгоритмів і, отже, певний час. В додатках, орієнтованих на прискіпливого користувача, таких, наприклад, як ігри, подібні затримки неприйнятні. Крім того, якщо зображення, що є у вас, припустимо, в 8-бітовому форматі GIF перевести в 24-бітовий JPEG, а потім назад в GIF для

перегляду, то втрата якості відбудеться двічі при обох перетвореннях. Проте, виграш в розмірах архівів часто настільки великий (в 3-20 разів), а втрати якості настільки малі, що зберігання зображень в JPEG виявляється дуже ефективним.

Декілька слів необхідно сказати про модифікації цього алгоритму. Хоча JPEG і є стандартом ISO, формат його файлів не був зафіксований. Користуючись цим, виробники створюють свої, несумісні між собою формати, і, отже, можуть змінити алгоритм. Так, внутрішні таблиці алгоритму, рекомендовані ISO, замінюються ними на свої власні. Крім того, легка плутанина присутня при завданні ступеня втрат. Наприклад, при тестуванні з'ясовується, що "відмінна" якість, "100%" і "10 балів" дають картинки, що істотно розрізняються. При цьому, до речі, "100%" якості не означають стиснення без втрат. Зустрічаються також варіанти JPEG для специфічних додатків.

Як стандарт ISO JPEG широко використовується при обміні зображеннями в комп'ютерних мережах. Підтримується алгоритм JPEG у форматах Quick Time, PostScript Level 2, Tiff 6.0 і, на даний момент, займає видне місце в системах мультимедіа.

Характеристики алгоритму JPEG:

Ступінь стиснення: 2-200 (Задається користувачем).

Клас зображень: Повнокольорові 24-бітові зображення або зображення в градаціях сірого без різких переходів кольорів (фотографії).

Симетричність: 1:1

Характерні особливості: В деяких випадках, алгоритм створює "ореол" навкруги різких горизонтальних і вертикальних меж в зображенні (ефект Гіббса). Крім того, при високому ступені стиснення зображення розпадається на блоки 8x8 пікселів.

3. Алгоритм JPEG 2000

Алгоритм JPEG-2000 розроблений тією ж групою експертів в області фотографії, що і JPEG. Формування JPEG як міжнародного стандарту було закінчено в 1992 році. В 1997 стало ясно, що необхідний новий, більш гнучкий і могутній стандарт, який і був допрацьований до зими 2000 року. Основні відмінності алгоритму в JPEG 2000 від алгоритму в JPEG полягають в наступному:

1. **Краща якість зображення при сильному ступені стиснення.** Або, що те ж саме, великий ступінь стиснення при тій же якості для високих ступенів стиснення. Фактично це означає помітне зменшення розмірів графіки "Web-якості", що використовується більшістю сайтів.

2. **Підтримка кодування окремих областей з кращою якістю.** Відомо, що окремі області зображення критичні для сприйняття людиною (наприклад, очі на фотографії), тоді як якістю інших можна пожертвувати (наприклад, задній

план). При "ручній" оптимізації збільшення ступеня стиснення проводиться до тих пір, поки не буде втрачена якість в якійсь важливій частині зображення. Зараз з'являється можливість задати якість в критичних областях, стиснувши решту областей сильніше, тобто ми одержуємо ще більший остаточний ступінь стиснення при суб'єктивно рівній якості зображення.

3. **Основний алгоритм стиснення замінений на вейвлет-стиснення (wavelet – рекурсивний хвильовий алгоритм).** Крім підвищення ступеня стиснення це дозволило позбутися 8-піксельної блоковості, що виникає при підвищенні ступеня стиснення. Крім того, плавний прояв зображення тепер спочатку закладений в стандарт (Progressive JPEG, активно використовується в Інтернет, з'явився набагато пізніше JPEG).

4. **Для підвищення ступеня стиснення в алгоритмі використовується арифметичне стиснення.** Спочатку в стандарті JPEG також було закладено арифметичне стиснення, проте пізніше воно було замінено менш ефективним стисненням по Хаффману, оскільки арифметичне стиснення було захищено патентами. Зараз термін дії основного патенту закінчився, і з'явилася можливість поліпшити алгоритм.

5. **Підтримка стиснення без втрат.** Крім звичного стиснення з втратами новий JPEG тепер підтримуватиме і стиснення без втрат. Таким чином, стає можливим використовування JPEG для стиснення медичних зображень, в поліграфії, при збереженні тексту під розпізнавання оптичними системами і т.д.

6. **Підтримка стиснення однобітових (2-кольорових) зображень.** Для збереження однобітових зображень (малюнки тушшю, відсканований текст і т.п.) раніше повсюдно рекомендувався формат GIF, оскільки стиснення з використанням ДКП вельми неефективне до зображень з різкими переходами кольорів. В JPEG при стисненні 1-бітова картинка приводилася до 8-бітової, тобто збільшувалася в 8 разів, після чого робилася спроба стискати, нерідко менш ніж в 8 разів. Зараз можна рекомендувати JPEG 2000 як універсальний алгоритм.

7. **На рівні формату підтримується прозорість.** Плавно накладати фон при створенні WWW сторінок можна не тільки в GIF, але і в JPEG 2000. Крім того, підтримується не тільки 1 біт прозорості (піксел прозорий / непрозорий), а окремий канал, що дозволить задавати плавний перехід від непрозорого зображення до прозорого фону.

Крім того, на рівні формату підтримуються включення в зображення інформації про копірайт, підтримка стійкості до бітових помилок при передачі і широкомовленні, можна запитувати для декомпресії або обробки зовнішні засоби (plug-ins), можна включати в зображення його опис, інформацію для пошуку і т.д.

Схема роботи (ідея алгоритму)

Базова схема JPEG-2000 (рис. 14.3) дуже схожа на базову схему JPEG. Відмінності полягають в наступному:

1. Замість дискретного косинусного перетворення (DCT) використовується дискретне вейвлєт-перетворення (DWT).

2. Замість кодування по Хаффману використовується арифметичне стиснення.

3. У алгоритм з самого початку закладено управління якістю областей зображення.

4. Не використовується явно дискретизація компонент U і V після перетворення колірних просторів, оскільки при DWT можна досягти того ж результату, але більш акуратно.

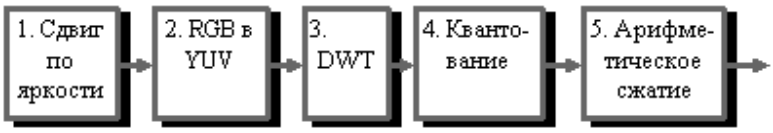

Рисунок 14.3 – Конвейєр операцій, що використовується в алгоритмі JPEG-2000

Розглянемо алгоритм по кроках.

Крок 1. Зсув по яскравості

У JPEG-2000 передбачений зсув по яскравості (DC level shift) кожної компоненти (RGB) зображення перед перетворенням в YUV. Це робиться для вирівнювання динамічного діапазону, що приводить до збільшення ступеня стиснення. Формулу перетворення можна записати як:

$$I'(x,y) = I(x,y) - 2^{ST-1}.$$

Значення ступеня ST для як кожної компоненти R, G і B своє (визначається при стисненні компресором). При відновленні зображення виконується зворотне перетворення:

$$I'(x,y) = I(x,y) + 2^{ST-1}.$$

Крок 2. RGB в YUV (YCrCb)

Переводимо зображення з колірного простору RGB, з компонентами, що відповідають за червону (Red), зелену (Green) і синю (Blue) складові кольору крапки, в колірний простір YUV. Цей крок аналогічний JPEG (див. матриці перетворення в описі JPEG), за тим виключенням, що окрім перетворення з втратами передбачено також і перетворення без втрат. Його матриця виглядає так:

$$\begin{vmatrix} Y \\ U \\ V \end{vmatrix} = \begin{pmatrix} \left\lfloor \frac{R+2G+B}{4} \right\rfloor \\ R-G \\ B-G \end{pmatrix}.$$

Зворотне перетворення здійснюється за допомогою зворотної матриці:

$$\begin{vmatrix} R \\ G \\ B \end{vmatrix} = \begin{pmatrix} U+G \\ Y-\lfloor\frac{U+V}{4}\rfloor \\ V+G \end{pmatrix}.$$

Крок 3. Вейвлєт-перетворення (DWT)

Відповідає за якість. Дискретне wavelet перетворення (DWT) також може бути двох видів - для випадку стиснення з втратами і для стиснення без втрат. Його коефіцієнти задаються таблицями.

Саме перетворення в одновимірному випадку є скалярним добутком коефіцієнтів фільтру на рядок перетворюваних значень (в нашому випадку - на рядок зображення). При цьому парні значення, що виходять, формуються за допомогою низькочастотного перетворення, а непарні за допомогою високочастотного:

$$y_{output}(2n) = \sum_{j=0}^{N-1} x_{input}(j) \cdot h_H(j-2n)$$

$$y_{output}(2n+1) = \sum_{j=0}^{N-1} x_{input}(j) \cdot h_L(j-2n-1)$$

Оскільки більшість $h_L(i)$ рівні 0, то можна переписати приведені формули з меншою кількістю операцій. Для простоти розглянемо випадок стиснення без втрат.

$$y_{out}(2n) = \frac{-x_{in}(2n-1)+2\cdot x_{in}(2n)+6\cdot x_{in}(2n+1)+2\cdot x_{in}(2n+2)-x_{in}(2n+3)}{8}$$

$$y_{out}(2n+1) = -\frac{x_{in}(2n)}{2} + x_{in}(2n+1) - \frac{x_{in}(2n+2)}{2}$$

Легко показати, що даний запис можна еквівалентно переписати, зменшивши ще втричі кількість операцій множення і розподілу (проте тепер необхідно буде підрахувати спочатку всі непарні y). Додамо також операції округлення до найближчого цілого, не перевищуючого задане число a, що позначаються як $\lfloor a \rfloor$:

$$y_{out}(2n+1) = x_{in}(2n+1) - \left\lfloor \frac{x_{in}(2n)+x_{in}(2n+2)}{2} \right\rfloor$$

$$y_{out}(2n) = x_{in}(2n) + \left\lfloor \frac{y_{out}(2n-1)+y_{out}(2n+1)+2}{4} \right\rfloor$$

Приклад.

Розглянемо на прикладі, як працює дане перетворення. Для того, щоб перетворення можна було застосовувати до крайніх пікселів зображення, воно симетрично добудовується в обидві сторони на декілька пікселів, як показано на рис. 14.4. У гіршому разі (стиснення з втратами) нам необхідно добудувати зображення на 4 пікселя.

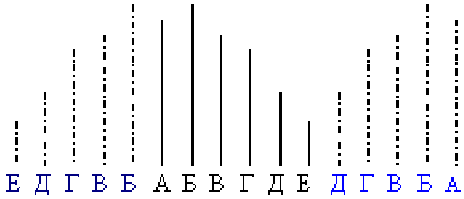

Рисунок 14.4 – Симетричне розширення зображення (яскравості АБ…Е) по рядку вправо і вліво

Хай ми перетворимо рядок з 10 пікселів. Розширимо її значення вправо і вліво і застосуємо DWT перетворення:

n	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11
x_{in}	3	2	1	2	3	7	10	15	12	9	10	5	10	9
y_{out}		0	1	0	3	1	11	4	13	-2	8	-5		

Рядок 1, що вийшов, 0, 3, 1, 11, 4, 13, -2, 8, -5 і є ланцюжком, що однозначно задає початкові дані. Вчинивши аналогічні перетворення з коефіцієнтами для розпаковування, одержимо необхідні формули:

$$x_{out}(2n) = y_{out}(2n) - \left\lfloor \frac{y_{out}(2n-1)+y_{out}(2n+1)+2}{4} \right\rfloor$$

$$x_{out}(2n+1) = y_{out}(2n+1) + \left\lfloor \frac{x_{out}(2n)+x_{out}(2n+2)}{2} \right\rfloor$$

Легко перевірити (використовуючи перетворення упаковки), що значення на кінцях рядків в y_{out} також симетричні щодо n=0 і 9. Скориставшись цією властивістю, розширимо наш рядок вправо і вліво і застосуємо зворотне перетворення:

n	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11
y_{out}		0	1	0	3	1	11	4	13	-2	8	-5	8	-2
x_{out}			1	2	3	7	10	15	12	9	10	5	10	

Як бачимо, ми одержали початковий ланцюжок (($x_{in} = x_{out}$).

Далі до рядка застосовується черезрядкове перетворення, суть якого полягає в тому, що всі парні коефіцієнти переписуються в початок рядка, а всі непарні - в кінець. В результаті цього перетворення на початку рядка формується "зменшена копія" всього рядка (низькочастотна складова), а в кінці рядка - інформація про коливання значень проміжних пікселів (високочастотна складова).

y_{out}	1	0	3	1	11	4	13	-2	8	-5
y'_{out}	1	3	11	13	8	0	1	4	-2	-5

Це перетворення застосовується спочатку до всіх рядків зображення, а потім до всіх стовпців зображення. В результаті зображення ділиться на 4 квадранти. В першому квадранті буде сформована зменшена копія зображення, а в інших трьох - високочастотна інформація. Після чого перетворення повторно застосовується вже тільки до першого квадранта зображення за тими ж правилами (рис. 14.5).

Рисунок 14.5 – Результат вейвлет-перетворення

Для коректного збереження результатів під дані 2 і 3 квадрантів виділяється на один біт більше, а під дані 4-го квадранта - на 2 біти більше. Тобто якщо початкові дані були 8-бітові, то на 2 і 3 квадранти потрібно 9 біт, а на 4-й - 10, незалежно від рівня використання DWT. При записі коефіцієнтів у файл можна використовувати ієрархічну структуру DWT, поміщаючи коефіцієнти перетворень з більшого рівня в початок файлу. Це дозволяє одержати "зображення для попереднього перегляду", прочитавши невелику ділянку даних з початку файлу, а не розпаковуючи весь файл, як це доводилося робити при стисненні зображення цілком. Ієрархічність перетворення може також використовуватися для плавного поліпшення якості зображення при передачі його по мережі.

Крок 4. Квантування

Так само, як і в алгоритмі JPEG, після DWT застосовується квантування. Коефіцієнти квадрантів діляться на наперед задане число. При збільшенні цього числа знижується динамічний діапазон коефіцієнтів, вони стають ближче до 0, і ми одержуємо великий ступінь стиснення. Варіюючи ці числа для різних рівнів перетворення, для різних колірних компонент і для різних квадрантів, ми дуже гнучко управляємо ступенем втрат в зображенні. Розраховані в компресорі оптимальні коефіцієнти квантування передаються в декомпресор для однозначного розпаковування.

Крок 5. Арифметичне стиснення

Для стиснення масивів даних в JPEG 2000 використовується варіант арифметичного стиснення, званий MQ-кодер, прообраз якого (QM-кодер) розглядався ще в стандарті JPEG, але реально не використовувався через патентні обмеження.

Області підвищеної якості

Основна задача, яку ми вирішуємо - підвищення ступеня стиснення зображень. Коли практично досягнута межа стиснення зображення в цілому і різні методи дають дуже невеликий виграш, ми можемо істотно (в рази) збільшити ступінь стиснення за рахунок зміни якості різних ділянок зображення.

Проблемою цього підходу є те, що необхідно якимсь чином одержувати розташування найважливіших для людини ділянок зображення. Наприклад, такою

ділянкою на фотографії людини є особа, а на обличчі - очі. Якщо при стисненні портрета з великими втратами будуть розмиті предмети, що знаходяться на задньому плані - це неістотно. Проте, якщо буде розмита особа або очі - експертна оцінка ступеня втрат буде гіршою (рис. 14.6).

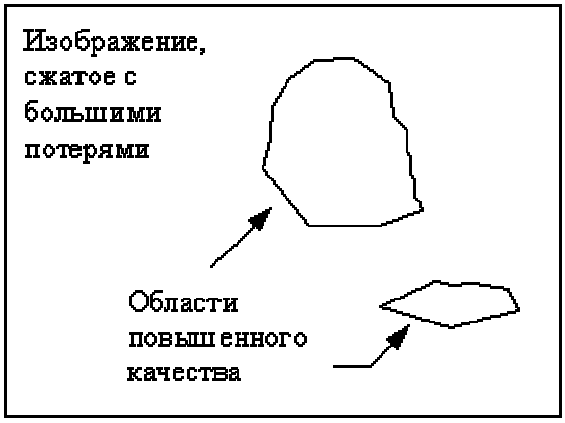

Рисунок 14.6 – Локальне поліпшення якості областей зображення

Роботи по автоматичному виділенню таких областей активно ведуться. Зокрема, створені алгоритми автоматичного виділення облич на зображеннях. Продовжуються дослідження методів виділення найбільш значимих (при аналізі зображення мозком людини) контурів і т.д. Проте очевидно, що універсальний алгоритм найближчим часом створений не буде, оскільки для цього вимагається побудувати повну схему сприйняття зображень мозком людини.

На сьогодні цілком реальне використання напівавтоматичних систем, в яких якість областей зображення задаватиметься інтерактивно. Даний підхід зменшує кількість можливих областей використання модифікованого алгоритму, але **дозволяє досягти більшого ступеня стиснення**.

Такий підхід логічно застосовувати, якщо:

1. Для додатку повинна бути критична (максимальна) ступінь стиснення, причому настільки, що можливий індивідуальний підхід до кожного зображення
2. Зображення стискається один раз, а розтискається багато разів

Як приклади додатків, що задовольняють цим обмеженням, можна привести практично всі мультимедійні продукти на CD-ROM. І для CD-ROM енциклопедій, і для ігор важливо записати на диск якомога більше інформації, а графіка, як правило, займає до 70% всього об'єму диска. При цьому технологія виробництва дисків дозволяє стискати кожне зображення індивідуально, максимально підвищуючи ступінь стиснення.

Цікавим прикладом є WWW-сервери. Для них теж, як правило, виконуються обидві викладених вище умови. При цьому абсолютно не обов'язково

індивідуально підходити до кожного зображення, оскільки за статистикою 10% зображень запрошуватимуться 90% разів. Тобто для великих довідкових або ігрових серверів з'являється можливість зменшувати час завантаження зображень і ступінь завантаженості каналів зв'язку адаптивно.

У JPEG-2000 використовується однобітове зображення-маска, що задає підвищення якості в даній області зображення. Оскільки за якість областей у нас відповідають коефіцієнти DWT перетворення в 2, 3 і 4 квадрантах, то маска перетвориться так, щоб вказувати на всі коефіцієнти, відповідні областям підвищення якості (рис. 14.7):

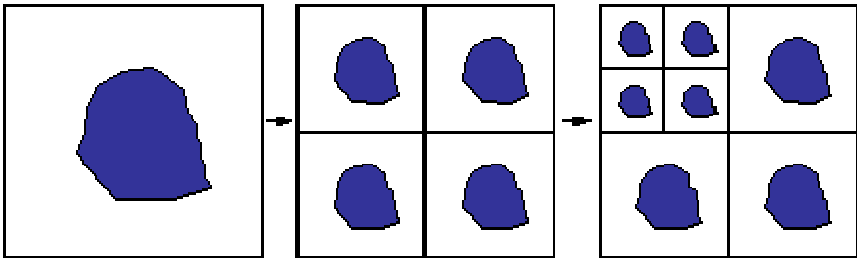

Рисунок 14.7 – Перетворення маски області підвищення якості для обробки DWT коефіцієнтів

Ці області обробляються далі іншими алгоритмами (з меншими втратами), що і дозволяє досягти балансу по загальній якості і ступеню стиснення.

Характеристики алгоритму JPEG-2000:

Ступінь стиснення: 2-200 (Задається користувачем). Можливе стиснення без втрат.

Клас зображень: Повнокольорові 24-бітові зображення. Зображення в градаціях сірого без різких переходів кольорів (фотографії). 1-бітові зображення.

Симетричність: 1-1,5:1

Характерні особливості: Дозволяє видаляти візуально неприємні ефекти, підвищуючи якість в окремих областях. При сильному стисненні з'являється блоковість і великі хвилі у вертикальному і горизонтальному напрямах.

Контрольні питання

1. Вкажіть критерій оцінки втрат якості зображень, який зараз використовується на практиці:
2. Згідно якого критерію зображення буде сильно зіпсовано при пониженні яскравості на 5%?
3. Який критерій визнає прийнятною якістю зображення зі «снігом» і «муаром»?

4. Згідно якого критерію зображення буде сильно зіпсовано при істотній зміні тільки одного пікселя?

5. Який показник оцінки втрат якості зображень використовує логарифмічний масштаб шкали?

6. Архівація зображень, при якій неможливо на око розрізнити первинне і розархівоване зображення, є

7. Якою є архівація зображень, при якій сказати, яке з зображень піддавалося архівації, можна тільки порівнюючи два зображення?

8. Якою є архівація зображень, при якій можна на око відрізнити первинне і розархівоване зображення, навіть не порівнюючи їх?

9. Якою є архівація зображень, при якій первинне і розархівоване зображення суттєво відрізняються і містять побічні візуальні ефекти (артефакти)?

10. Назвіть алгоритм стиснення зображень, що оперує областями 8x8, на яких яскравість і колір змінюються порівняно плавно.

11. Назвіть алгоритм стиснення зображень, заснований на дискретному косинусному перетворенні.

12. Назвіть алгоритм стиснення зображень, що використовує дискретне вейвлєт-перетворення.

13. Назвіть особливості зображення, за рахунок яких відбувається стиснення для алгоритму JPEG?

ЛЕКЦІЯ 15

на тему: Руйнівні алгоритми стиснення цифрових зображень: рекурсивний і фрактальний алгоритми

У першій частині лекції дано опис рекурсивного (хвильового) алгоритму, його особливостей і реалізації. В другій частині мова піде про фрактальну архівацію і про особливості побудови алгоритму компресії та декомпресії зображень.

План:

1. Рекурсивний (хвильовий) алгоритм
2. Фрактальний алгоритм

1. Рекурсивний (хвильовий) алгоритм

Англійська назва рекурсивного стиснення - wavelet. Українською мовою воно перекладається як хвильове стиснення, як стиснення з використанням сплесків, а останнім часом і вейвлет-стиснення. Цей вид архівації відомий досить давно і напряму виходить з ідеї використовування когерентності областей. Орієнтований алгоритм на кольорові і чорно-білі зображення з плавними переходами. Ідеальний для картинок типу рентгенівських знімків. Ступінь стиснення задається і варіюється в межах 5-100. При спробі задати більший коефіцієнт на різких межах, особливо діагональних, виявляється "сходовий ефект" - сходинки різної яскравості розміром в декілька пікселів.

Ідея алгоритму полягає в тому, що ми зберігаємо у файл різницю - число між середніми значеннями сусідніх блоків в зображенні, яка звичайно приймає значення, близькі до 0.

Так два числа a_{2i} і a_{2i+1} завжди можна представити у вигляді $b_i^1 = (a_{2i}+a_{2i+1})/2$ і $b_i^2 = (a_{2i}-a_{2i+1})/2$. Аналогічно послідовність a_i може бути попарно переведена в послідовність $b_i^{1,2}$.

Приклад

Нехай ми стискаємо рядок з 8 значень яскравості пікселів ((a_i): (220, 211, 212, 218, 217, 214, 210, 202). Ми одержимо наступні послідовності b_i^1, і b_i^2: (215.5, 215, 215.5, 206) і (4.5, -3, 1.5, 4). Зауважимо, що значення b_i^2 достатньо близькі до 0. Повторимо операцію, розглядаючи b_i^1 як a_i. Дана дія виконується як би рекурсивно, звідки і назва алгоритму. Ми одержимо з (215.5, 215, 215.5, 206):

(215.25, 210.75) (0.25, 4.75). Одержані коефіцієнти, округляючи до цілих і стиснувши, наприклад, за допомогою алгоритму Хаффмана з фіксованими таблицями, ми можемо помістити у файл.

Помітимо, що ми застосовували наше перетворення до ланцюжка тільки двічі. Реально ми можемо дозволити собі використання wavelet- перетворення 4-6 разів. Більш того, додаткове стиснення можна одержати, використовуючи таблиці алгоритму Хаффмана з нерівномірним кроком (тобто нам доведеться зберігати код Хаффмана для найближчого в таблиці значення). Ці прийоми дозволяють досягти помітних ступенів стиснення.

Алгоритм для двовимірних даних реалізується аналогічно (рис. 15.1). Якщо у нас є квадрат з 4 крапок з яскравостями $a_{2i,2j}, a_{2i+1,2j}, a_{2i,2j+1}, a_{2i+1,2j+1}$, то

$$b^1_{i,j} = (a_{2i,2j} + a_{2i+1,2j} + a_{2i,2j+1} + a_{2i+1,2j+1})/4$$
$$b^2_{i,j} = (a_{2i,2j} + a_{2i+1,2j} - a_{2i,2j+1} - a_{2i+1,2j+1})/4$$
$$b^3_{i,j} = (a_{2i,2j} - a_{2i+1,2j} + a_{2i,2j+1} - a_{2i+1,2j+1})/4$$
$$b^4_{i,j} = (a_{2i,2j} - a_{2i+1,2j} - a_{2i,2j+1} + a_{2i+1,2j+1})/4$$

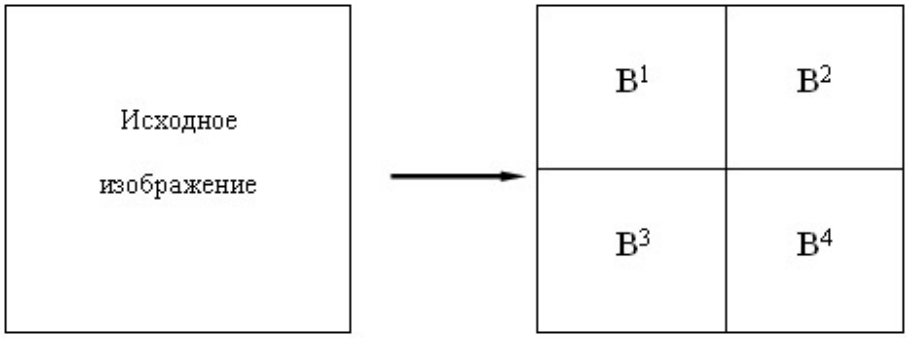

Рисунок 15.1 – Приклад роботи алгоритму для двовимірних даних

Використовуючи ці формули, ми для зображення 512x512 пікселів одержимо після першого перетворення 4 матриці розміром 256x256 елементів (рис. 15.2):

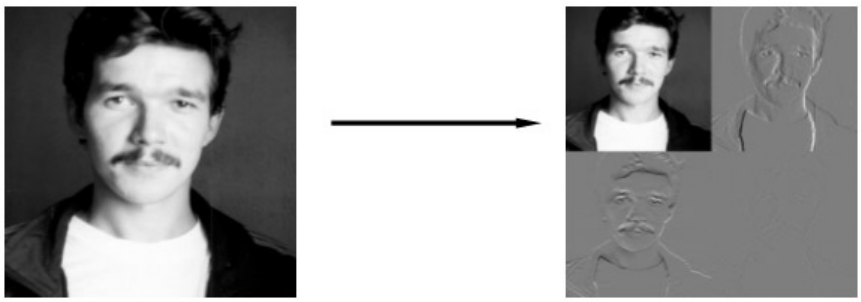

Рисунок 15.2 – Результат після першого перетворення

У першій, як легко здогадатися, зберігатиметься зменшена копія зображення. В другій - усереднені різниці пар значень пікселів по горизонталі. В третій - усереднені різниці пар значень пікселів по вертикалі. В четвертій - усереднені різниці значень пікселів по діагоналі. По аналогії з двовимірним випадком ми можемо повторити наше перетворення і одержати замість першої матриці 4 матриці розміром 128x128. Повторивши наше перетворення утретє, ми одержимо у результаті: 4 матриці 64x64, 3 матриці 128x128 і 3 матриці 256x256. На практиці при записі у файл, значеннями, одержуваними в останньому рядку $b^{\downarrow}_{i,j}$, зазвичай нехтують (відразу одержуючи виграш приблизно на третину розміру файлу - 1- 1/4 - 1/16 - 1/64...).

До переваг цього алгоритму можна віднести те, що він дуже легко дозволяє реалізувати можливість поступового "прояву" зображення при передачі зображення по мережі. Крім того, оскільки на початку зображення ми фактично зберігаємо його зменшену копію, спрощується показ "огрубленого" зображення по заголовку.

На відміну від JPEG даний метод не оперує блоками, наприклад, 8x8 пікселів. Точніше, ми оперуємо блоками 2x2, 4x4, 8x8 і т.д. Проте за рахунок того, що коефіцієнти для цих блоків ми зберігаємо незалежно, ми можемо достатньо легко уникнути дроблення зображення на "мозаїчні" квадрати.

Отже, вейвлетна компресія в сучасних алгоритмах компресії зображень дозволяє значно (до двох разів) підвищити ступінь стиснення чорно-білих і кольорових зображень при порівнянній візуальній якості стосовно алгоритмів попереднього покоління, заснованих на дискретному косинусперетворенні, таких, наприклад, як JPEG.

Для роботи з дискретними зображеннями використовується варіант вейвлет-перетворення, відомий як алгоритм Малла. Вихідне зображення розкладається на дві складові - високочастотні деталі (що складаються в основному з різких перепадів яскравості), і згладжену зменшену версію оригіналу. Це досягається застосуванням пари фільтрів, причому кожна з отриманих складових удвічі менша за вихідне зображення. Як правило, використовуються фільтри з кінцевим імпульсним відгуком, у яких пікселі, що потрапили в невелике "вікно", помножуються на заданий набір коефіцієнтів, отримані значення підсумуються, і вікно зсувається для розрахунку наступного значення на виході. Між вейвлетами й фільтрами є тісний зв'язок. Вейвлети безпосередньо не фігурують в алгоритмах, але якщо ітерувати відповідні фільтри на зображеннях, що складаються з єдиної яскравої точки, то на виході будуть все чіткіше проступати вейвлети.

Оскільки зображення двовимірні, фільтрація здійснюється й по вертикалі, і по горизонталі. Цей процес повторюється багаторазово, причому щораз як вхід використовується згладжена версія з попереднього кроку. Тому що зображення "деталей" складаються зазвичай з набору різких границь, і містять великі ділянки

де інтенсивність близька до нуля. Якщо припустимо зневажити деякою кількістю дрібних деталей, то всі ці значення можна просто занулити. У результаті виходить версія вихідного зображення, що добре піддається стиснення. Для відновлення оригіналу знову застосовується алгоритм Малла, але з парою фільтрів, зворотної дії до вихідного.

Характеристики хвильового алгоритму:

Ступінь: 2-200 (Задається користувачем).

Клас зображень: Повнокольорові 24 бітові зображення або зображення в градаціях сірого без різких переходів кольорів (фотографії).

Симетричність: 1.5:1

Характерні особливості: При високому ступені стиснення виявляється «сходовий ефект». Дозволяє реалізувати можливість поступового «прояву» зображення, спрощується показ «огрубленого» зображення.

2. Фрактальний алгоритм

Ідея методу

Фрактальна архівація заснована на тому, що ми представляємо зображення в більш компактній формі - за допомогою коефіцієнтів системи ітерованих функцій (Iterated Function System - далі по тексту як IFS). Перш, ніж розглядати сам процес архівації, розберемо, як IFS будує зображення, тобто процес декомпресії.

Строго кажучи, IFS є набором тривимірних афінних перетворень, в нашому випадку переводячих одне зображення в інше. Перетворенню піддаються крапки в тривимірному просторі (x_координата, y_координата, яскравість).

Найбільш наочно цей процес продемонстрував Барнслі в своїй книзі "Fractal Image Compression". Там введено поняття Фотокопіюючої Машини, що складається з екрану, на якому зображена початкова картинка, і системи лінз, що проектують зображення на інший екран (рис. 15.3):

- Лінзи можуть проектувати частину зображення довільної форми в будь-яке інше місце нового зображення.
- Області, в які проектуються зображення, не перетинаються.
- Лінза може міняти яскравість і зменшувати контрастність.
- Лінза може дзеркально відображати і повертати свій фрагмент зображення.
- Лінза повинна масштабувати (причому тільки зменшуючи) свій фрагмент зображення.

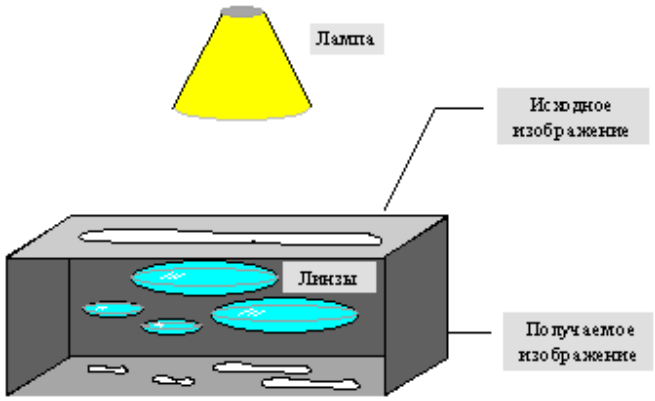

Рисунок 15.3 – Машина Барнслі

Розставляючи лінзи і міняючи їх характеристики, ми можемо управляти одержуваним зображенням. Одна ітерація роботи Машини полягає в тому, що по початковому зображенню за допомогою проектування будується нове, після чого нове береться як початкове. Стверджується, що в процесі ітерацій ми одержимо зображення, яке перестане змінюватися. Воно залежатиме тільки від розташування і характеристик лінз, і не залежатиме від початкової картинки. Це зображення називається "**нерухомою крапкою"** або **атрактором** даної IFS. Відповідна теорія гарантує наявність рівно однієї нерухомої крапки для кожної IFS.

Оскільки відображення лінз є стискаючим, кожна лінза в явному вигляді задає самоподібні області в нашому зображенні. Завдяки самоподібності ми одержуємо складну структуру зображення при будь-якому збільшенні. Таким чином, інтуїтивно зрозуміло, що система итерованих функцій задає **фрактал** (самоподібний математичний об'єкт).

Найбільш відомо два зображення, одержані за допомогою IFS: "трикутник Серпінського" (рис. 15.4) і "папороть Барнслі" рис. 15.5.

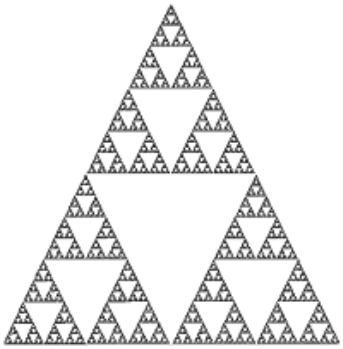
Рисунок 15.4 – Трикутник Серпінського. Задається 3 перетвореннями

"Трикутник Серпінського" задається трьома, а "папороть Барнслі" чотирма афінними перетвореннями (або, в нашій термінології, "лінзами"). Кожне

перетворення кодується буквально ліченими байтами, тоді як зображення, побудоване з їх допомогою, може займати і декілька мегабайт.

Рисунок 15.5 – Папороть Барнслі. Задається 4 перетвореннями

З вищесказаного стає зрозуміло, як працює архіватор, і чому він вимагає так багато часу. Фактично, фрактальна компресія – це пошук самоподібних областей в зображенні і визначення для них параметрів афінних перетворень (рис. 15.6).

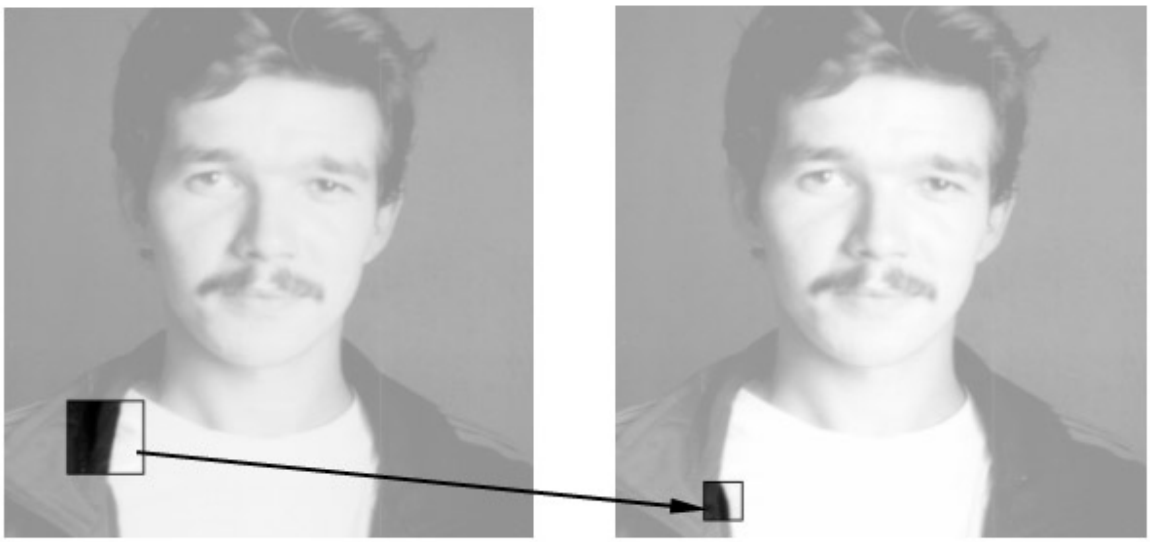

Рисунок 15.6 – Приклад пошуку само подібних областей в зображенні

У гіршому разі, якщо не застосовуватиметься оптимізуючий алгоритм, буде потреба в переборі і порівнянні всіх можливих фрагментів зображення різного розміру. Навіть для невеликих зображень при врахуванні дискретності ми одержимо астрономічне число варіантів, що перебираються. Причому, навіть різке звуження класів перетворень, наприклад, за рахунок масштабування тільки в певну кількість раз, не дає помітного виграшу в часі. Крім того, при цьому втрачається якість зображення. Більшість досліджень в області фрактальної компресії зараз направлені на зменшення часу архівації, необхідного для отримання якісного зображення.

Далі приводяться основні визначення і теореми, на яких базується фрактальна компресія.

Визначення. Перетворення $w : R^2 \to R^2$, представлене у вигляді

$$w(\bar{}) = w\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & b \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

де a, b, c, d, e, f дійсні числа і $(xy) \in R^2$ називається двовимірним афінним перетворенням.

Визначення. Перетворення, представлене у вигляді

$$w(\bar{}) = w\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a & b & t \\ c & d & u \\ r & s & p \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e \\ f \\ q \end{pmatrix}$$

де a, b, c, d, e, f, p, q, r, s, t, u дійсні числа і $(xyz) \in R^3$ називається тривимірним афінним перетворенням.

Визначення. Нехай $f : \mathrm{X} \to \mathrm{X}$ - перетворення в просторі X. Точка $x_f \in \mathrm{X}$ така, що $f(x_f) = x_f$ називається **нерухомою точкою (атрактором) перетворення**.

Визначення. Перетворення $f : \mathrm{X} \to \mathrm{X}$ у метричному просторі (X, d) називається стискаючим, якщо існує число s: $0 \le s < 1$, таке, що

$d(f(x), f(y)) \le s \cdot d(x, y) \forall x, y \in \mathrm{X}$

Зауваження: Формально ми можемо використовувати будь-яке стискаюче відображення при фрактальній компресії, але реально використовуються лише тривимірні афінні перетворення з достатньо сильними обмеженнями на коефіцієнти.

Теорема про стискаюче перетворення

Нехай $f : \mathrm{X} \to \mathrm{X}$ - стискаюче перетворення в повному метричному просторі (X, d). Тоді існує в точності одна нерухома крапка $x_f \in \mathrm{X}$ цього перетворення, і для будь-якої крапки $x \in \mathrm{X}$ послідовність $\{f^n(x) : n = 0, 1, 2...\}$ сходиться до x_f.

Визначення. Зображенням називається функція S, визначена на одиничному квадраті і приймаюча значення від 0 до 1 або $S(x, y) \in [0...1] \forall x, y \in [0...1]$

Нехай тривимірне афінне перетворення $w_i : R^3 \to R^3$, записано у вигляді

$$w_i(\bar{}) = w_i\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & p \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e \\ f \\ q \end{pmatrix}$$

і визначено на компактній підмножині D_i декартова квадрата [0..1]x[0..1](ми користуємося особливим видом матриці перетворення, щоб зменшити розмірність

області визначення з R^3до R^2). Тоді воно переведе частину поверхні S в область R_i, розташовану із зсувом (e,f) і поворотом, заданим матрицею

$$\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

При цьому, якщо інтерпретувати значення функції $S(x,y) \in [0...1]$ як яскравість відповідних крапок, вона зменшиться в p раз (перетворення зобов'язано бути стискаючим) і зміниться на зсув q.

Визначення. Кінцева сукупність W стискаючих тривимірних афінних перетворень w_i, визначених на областях D_i, таких, що $w_i(D_i) = R_i$, називається **системою ітерованих функцій** (IFS).

Системі ітерованих функцій однозначно зіставляється нерухома крапка - зображення. Таким чином, процес компресії полягає в пошуку коефіцієнтів системи, а процес декомпресії - в проведенні ітерацій системи до стабілізації одержаного зображення (нерухомої крапки IFS). На практиці буває достатні 7-16 ітерацій. Області R_i надалі іменуватимуться **ранговими** (після стиснення), а області D_i - **доменними** (до стиснення).

Побудова алгоритму

Основною задачею при компресії фрактальним алгоритмом є знаходження відповідних афінних перетворень. В найзагальнішому випадку ми можемо переводити будь-які за розміром і формою області зображення, проте в цьому випадку виходить астрономічне число перебираних варіантів різних фрагментів, яке неможливе обробити на даний момент навіть на суперкомп'ютері.

У спрощеному варіанті алгоритму зроблені наступні обмеження на області:

1. Всі області є квадратами із сторонами, паралельними сторонам зображення. Це обмеження достатньо жорстке. Фактично ми збираємося апроксимувати все різноманіття геометричних фігур лише квадратами.
2. При переведенні доменної області в рангову зменшення розмірів проводиться рівно в два рази. Це істотно спрощує як компресор, так і декомпресор, оскільки задача масштабування невеликих областей є нетривіальною.
3. Всі доменні блоки - квадрати і мають фіксований розмір. Зображення рівномірною сіткою розбивається на набір доменних блоків.
4. Доменні області беруться "через крапку" і по X, і по У, що відразу зменшує перебір в 4 рази.
5. При переведенні доменної області в рангову поворот куба можливий тільки на 0°, 90°, 180° або 270°. Також допускається дзеркальне віддзеркалення. Загальне число можливих перетворень (рахуючи порожнє) - 8.

6. Масштабування (стиснення) по вертикалі (яскравості) здійснюється у фіксоване число раз - в 0,75.

Ці обмеження дозволяють:

1. Побудувати алгоритм, для якого потрібне порівняно мале число операцій навіть на достатньо великих зображеннях.

2. Дуже компактно представити дані для запису у файл. Нам потрібно на кожне афінне перетворення в IFS:

- два числа для того, щоб задати зсув доменного блоку. Якщо ми обмежимо вхідні зображення розміром 512x512, то достатньо буде по 8 біт на кожне число;
- три біти для того, щоб задати перетворення симетрії при переведенні доменного блоку в ранговий;
- 7-9 біт для того, щоб задати зсув по яскравості при переведенні.

Інформацію про розмір блоків можна зберігати в заголовку файлу. Таким чином, ми затратили менше 4 байт на одне афінне перетворення. Залежно від того, який розмір блоку, можна вирахувати, скільки блоків буде в зображенні. Таким чином, ми можемо одержати оцінку ступеня компресії.

Наприклад, для файлу в градаціях сірого 256 кольорів 512x512 пікселів при розмірі блоку 8 пікселів афінних перетворень буде 4096 (512/8 512/8). На кожне буде потрібно 3.5 байти. Отже, якщо початковий файл займав 262144 (512 512) байт (без урахування заголовка), то файл з коефіцієнтами займатиме 14336 байт. Ступінь стиснення - 18 разів. При цьому ми не враховуємо, що файл з коефіцієнтами теж може володіти надмірністю і архівуватися методом архівації без втрат, наприклад LZW.

Негативні сторони запропонованих обмежень:

1. Оскільки всі області є квадратами, неможливо скористатися подібністю об'єктів, за формою далеких від квадратів (які зустрічаються в реальних зображеннях достатньо часто).
2. Аналогічно ми не зможемо скористатися подібністю об'єктів в зображенні, коефіцієнт подібності між якими сильно відрізняється від 2.
3. Алгоритм не зможе скористатися подібністю об'єктів в зображенні, кут між якими не кратний 90°.

Така плата за **швидкість компресії** і за простоту упаковки коефіцієнтів у файл.

Сам алгоритм упаковки зводиться до перебору всіх доменних блоків і підбору для кожного відповідного йому рангового блоку.

Для кожного рангового блоку робимо його перевірку зі всіма можливими доменними блоками (у тому числі з тими, що пройшли перетворення симетрії), знаходимо варіант з найменшою мірою (найменшим середньоквадратичним відхиленням) і зберігаємо коефіцієнти цього перетворення у файл. Коефіцієнти - це (1) координати знайденого блоку, (2) число від 0 до 7, характеризуюче

перетворення симетрії (поворот, віддзеркалення блоку), і (3) зсув по яскравості для цієї пари блоків. Зсув по яскравості обчислюється як:

$$q = \left[\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} - \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij} \right] / n^2$$

де r_{ij} - значення пікселів рангового блоку (R), а d_{ij} - значення пікселів доменного блоку (D). При цьому міра обчислюється як:

$$d(R,D) = \sum_{i=1}^{n} \sum_{j=1}^{n} (0.75 r_{ij} + q - d_{ij})^2$$

Ми не обчислюємо квадратного кореня з міри і не ділимо її на n, оскільки дані перетворення монотонні і не перешкодять нам знайти екстремум, проте ми зможемо виконувати на дві операції менше для кожного блоку.

Одна з можливих схем кодування зображень фрактальним методом, запропонована Джеквіном (Jacquin), містить такі етапи:

- зображення розділяється на примикаючі одна до одної області розміром nxn (рангові області);
- задається набор доменних областей. Доменні області можуть перекриватися, вони не повинні обов'язково закривати всю поверхню зображення. Розміри доменних областей звичайно вибирають 2nx2n;
- для кожної рангової області підбирається доменна область, яка після афінних перетворень найбільш точно апроксимує рангову область. На практиці застосовується вісім варіантів відображення одного квадрата в інший з використанням афінних перетворень. Це повороти зображення на кути 90, 180, 270, (-90) градусів відносно його центра і перетворення симетрії відносно ортогональних осей, які проходять через центр фрагмента перпендикулярно його сторонам;
- точність апроксимації визначається за допомогою середньоквадратичного критерія.

Схема алгоритму декомпресії зображень

Декомпресія алгоритму фрактального стиснення надзвичайно проста. Необхідно провести декілька ітерацій тривимірних афінних перетворень, коефіцієнти яких були одержані на етапі компресії.

Як початкове може бути узято абсолютно будь-яке зображення (наприклад, абсолютне чорне), оскільки відповідний математичний апарат гарантує нам збіжність послідовності зображень, одержуваних в ході ітерацій IFS (IFS – система ітерованих функцій), до нерухомого зображення (близькому до початкового). Зазвичай для цього достатньо 16 ітерацій.

Оскільки ми записували коефіцієнти для блоків r_{ij} (які в нашому окремому випадку є квадратами однакового розміру) послідовно, то виходить, що ми

послідовно заповнюємо зображення по квадратах сітки розбиття використовуванням афінного перетворення.

Як можна підрахувати, кількість операцій на один піксел зображення в градаціях сірого при відновленні надзвичайно мала (N операцій складання "+" і N операцій множення "*", де N - кількість ітерацій, тобто 7-16). Завдяки цьому, декомпресія зображень для фрактального алгоритму проходить швидше декомпресії, наприклад, для алгоритму JPEG. В простій реалізації JPEG на крапку приходиться 64 операції складання "+" і 64 операції множення "*". При реалізації швидкого ДКП можна одержати, 7 додавань і 5 множень на крапку, але це без урахування кроків RLE, квантування і кодування по Хаффману. При цьому для фрактального алгоритму множення відбувається на раціональне число, одне для кожного блоку. Це означає, що ми можемо, по-перше, використовувати цілочисельну раціональну арифметику, яка швидше за арифметику з плаваючою крапкою. По-друге, можна використовувати множення вектора на число - більш просту і швидку операцію, що часто закладається в архітектуру процесора (процесори SGI, Intel MMX, векторні операції Athlon і т.д.). Для повнокольорового зображення ситуація якісно не змінюється, оскільки переведення в інший колірний простір використовують обидва алгоритми.

Оцінка втрат і способи їх регулювання

При короткому викладі спрощеного варіанту алгоритму були пропущено багато важливих питань. Наприклад, що робити, якщо алгоритм не може підібрати для якого-небудь фрагмента зображення подібний йому? Достатньо очевидне рішення - розбити цей фрагмент на більш дрібні, і спробувати пошукати для них. В той же час зрозуміло, що цю процедуру не можна повторювати до безкінечності, інакше кількість необхідних перетворень стане така велика, що алгоритм перестане бути алгоритмом компресії. Отже, ми допускаємо втрати в якійсь частині зображення.

Для фрактального алгоритму компресії, як і для інших алгоритмів стиснення з втратами, дуже важливі механізми, за допомогою яких можна буде регулювати ступінь стиснення і ступінь втрат. До теперішнього часу розроблений достатньо великий набір таких методів. По-перше, можна обмежити кількість афінних перетворень, явно забезпечивши ступінь стиснення не нижче фіксованої величини. По-друге, можна зажадати, щоб за ситуації, коли різниця між оброблюваним фрагментом і якнайкращим його наближенням буде вище за певне порогове значення, цей фрагмент дробився обов'язково (для нього обов'язково заводиться декілька "лінз"). По-третє, можна заборонити дробити фрагменти розміром менше, припустимо, чотирьох крапок. Змінюючи порогові значення і пріоритет цих умов, ми дуже гнучко управлятимемо коефіцієнтом компресії зображення в діапазоні від побітової відповідності до будь-якого ступеня стиснення. Помітимо, що ця гнучкість набагато вище, ніж у найближчого "конкурента" - алгоритму JPEG.

Отже фрактальний алгоритм - це алгоритм стиску зображень з втратами, заснований на застосуванні систем ітерованих функцій до зображень. Даний алгоритм відомий тим, що в деяких випадках дозволяє одержати дуже високі коефіцієнти стиснення (кращі приклади - до 1000 разів при прийнятній візуальній якості) для реальних фотографій природних об'єктів, що недоступно для інших алгоритмів стиску зображень у принципі. Через складну ситуацію з патентуванням широкого поширення алгоритм не одержав.

Основа методу фрактального кодування – це виявлення самоподібних ділянок у зображенні. Метод використовує системи доменних і рангових блоків зображення, блоків квадратної форми, що покривають все зображення. Цей підхід став основою для більшості методів фрактального кодування, застосовуваних сьогодні. Відповідно до даного методу зображення розбивається на безліч неперекритих рангових підзображень і визначається безліч неперекритих доменних підзображень. Для кожного рангового блоку алгоритм кодування знаходить найбільш підходящий доменний блок і афінне перетворення, що переводить цей доменний блок у даний ранговий блок. Структура зображення відображається в систему рангових блоків, доменних блоків і перетворень.

Основна складність фрактального стиснення полягає в тому, що для знаходження відповідних доменних блоків потрібен повний перебір. Оскільки при цьому переборі щораз повинні порівнятися два масиви, дана операція виходить досить тривалою. Порівняно простим перетворенням її можна звести до операції скалярного добутку двох масивів, однак ця операція також тривала.

Серед відомих способів підвищення швидкодії кодування зображень фрактальним методом можно виділити такі:

- пошук доменних блоків, для яких F не перевищує заданного значення;
- локальний та сублокальний пошук;
- ізометричне передбачення;
- класифікація доменних і рангових блоків. Ранговий порівнюється з доменними блоками того ж самого класу.

Характеристики фрактального алгоритму:

Ступінь стиснення: 2-2000 (Задається користувачем).

Клас зображень: Повнокольорові 24 бітові зображення або зображення в градаціях сірого без різких переходів кольорів (фотографії). Бажано, щоб області більшої значимості (для сприйняття) були більш контрастними і різкими, а області меншої значимості - неконтрастними і розмитими.

Симетричність: 100:1

Характерні особливості: Дозволяє одержати дуже високі коефіцієнти стиснення. Може вільно масштабувати зображення при розархівуванні, збільшуючи його в 2-4 рази без появи "сходового ефекту". При збільшенні ступеня компресії з'являється "блоковий" ефект на межах блоків в зображенні.

Контрольні питання

1. У чому різниця між алгоритмами з втратою інформації і без втрати інформації?
2. Приведіть приклади заходів втрати інформації і опишіть їх недоліки.
3. За рахунок чого стискає зображення алгоритм JPEG?
4. У чому полягає ідея фрактального алгоритму компресії?
5. У чому полягає ідея рекурсивного (хвильового) стиснення?
6. Назвіть алгоритм стиснення зображень, що використовує ідею когерентності областей в зображенні.
7. Назвіть алгоритм стиснення зображень, що виконує пошук само подібних областей в зображенні і визначення для них параметрів афінних перетворень.
8. Назвіть особливості зображення, за рахунок яких відбувається стиснення для фрактального алгоритму.
9. Назвіть алгоритм стиснення зображень, що легко дозволяє реалізувати можливість поступового «прояву» зображення.
10. Де зберігатиметься зменшена копія зображення після його вейвлєт-перетворення?
11. Де, після вейвлєт-перетворення зображення, зберігатимуться усереднені різниці пар значень пікселів по горизонталі?
12. Де, після вейвлєт-перетворення зображення, зберігатимуться усереднені різниці пар значень пікселів по вертикалі?
13. Де, після вейвлєт-перетворення зображення, зберігатимуться усереднені різниці пар значень пікселів по діагоналі?
14. Використовуючи фрактальний алгоритм, скільки афінних перетворень потрібно для «трикутника Серпінського»?
15. Використовуючи фрактальний алгоритм, скільки афінних перетворень потрібно для «папоротті Барнслі»?
16. Охарактеризуйте кінцеву сукупність стискаючих тривимірних афінних перетворень, що для стиснення використовує коефіцієнти, а для декомпресії – ітерації системи до стабілізації одержаного зображення.
17. Назвіть області, на яких визначені тривимірні афінні перетворення, до стиснення фрактальним алгоритмом.
18. Назвіть області, на яких визначені тривимірні афінні перетворення, після стиснення фрактальним алгоритмом.
19. Назвіть особливості зображення, за рахунок яких відбувається стиснення для рекурсивного алгоритму:
20. Який алгоритм краще: JPEG чи фрактальний?

ЛЕКЦІЯ 16

на тему: Загальний огляд алгоритмів стиснення відеоданих

У цій лекції мова піде про алгоритми стиснення відео. Буде даний докладний перелік специфічних вимог, що пред'являються до цих алгоритмів. В лекції детально розглянутий алгоритм стиснення MPEG, від історії його створення до послідовності кроків, що виконуються при стисненні зображення.

План:

1. Основні поняття та принципи стиснення відеоданих
2. Вимоги додатків до алгоритмів стиснення відео
3. Визначення класів програмного і апаратного забезпечення для алгоритмів стиснення відео
4. Огляд стандартів стиснення відеоданих
5. Базові технології стиснення відео

1. Основні поняття та принципи стиснення відеоданих

В середині сорокових років минулого століття, відразу після закінчення Другої світової війни, група молодих інженерів, серед яких були Ален Тьюринг, Джон Атанасов, Джон Мошлі і Преспер Еккман, почала розробляти перші електронні комп'ютери. Піонерам-винахідникам комп'ютери представлялися швидкими і надійними пристроями для здійснення обчислень над числами. Однак дуже скоро більшість розробників комп'ютерів усвідомили, що їх можна застосовувати не тільки в вузько обчислювальних цілях. Перші нечислові додатки, розроблені в п'ятдесяті роки, обробляли тексти, потім прийшла черга зображень (шістдесяті роки), комп'ютерної анімації (сімдесяті роки) і оцифрованого звуку (вісімдесяті роки). В реальний час комп'ютери в основному використовуються для комунікацій і розваг, тому вони виконують різноманітні мультимедійні додатки, в яких доводиться обробляти тексти, зображення, відео та звук. Всі ці оцифровані масиви доводиться відображати, редагувати і передавати по лініях зв'язку іншим користувачам.

Будь-які типи комп'ютерних даних можуть тільки суттєво виграти від застосування ефективного стиснення. Однак, особливо корисною компресія стає при роботі з файлами, що містять відео. Уже файл одиничного зображення має досить великий обсяг. Що ж говорити про відеофайл, який складається з тисяч зображень? При обробці зображень, як ми знаємо, часто застосовується стиснення з втратами. При роботі ж з відео це стає просто необхідним. Стиснення зображень ґрунтується на кореляції пікселів, а компресія відео може використовувати не

тільки кореляцію близьких пікселів кожного кадру, але і кореляцію між послідовними кадрами.

Стиснення даних є надзвичайно важливим і актуальним практичним завданням у зв'язку з інтенсивним розвитком комп'ютерних засобів комунікації. На сьогодні це одне з надзвичайно важливих завдань для науковців усього світу, оскільки функціонування супутникових систем цифрового телебачення, цифрових фото- і відеокамер, відеотелефонів, інтернет-систем відеоконференцій неможливе без стиснення відеозображень.

Відео – це по суті тривимірний масив кольорових пікселів. Два виміри означають вертикальний і горизонтальний розміри кадру, а третій вимір – момент часу. **Кадр** – це масив усіх пікселів, які розміщені перед камерою в певний момент часу, або просто зображення.

Стиснення відео було б неможливе, якби кожен кадр був унікальним, а розміщення пікселів – повністю випадковим, але це не так. Тому можна стискати, по-перше, саме зображення – наприклад, фотографія блакитного неба без сонця фактично зводиться до опису граничних точок і градієнта заливки. По-друге, можна стискати схожі сусідні кадри. Зрештою алгоритми стиснення картинок і відео схожі, якщо розглядати відео як тривимірне зображення з часом як третю координату.

Основною складністю при роботі з відео є великі об'єми дискового простору, необхідного для зберігання навіть невеликих фрагментів. Причому навіть використання сучасних алгоритмів стиснення не змінює ситуацію кардинально. При записі на один компакт-диск "в побутовій якості", на нього можна помістити декілька тисяч фотографій, приблизно 10 годин музики і всього півгодини відео. Відео "телевізійного" формату 720x576 пікселов 25 кадрів в секунду в системі RGB вимагає потоку даних приблизно в 240 Мбіт/сек (тобто 1.8 Гб в хвилину). При цьому традиційні алгоритми стиснення зображень, орієнтовані на окремі кадри, не рятують ситуації, оскільки навіть при зменшенні потоку в 10 разів він складає достатньо великі величини.

Стиснення відео призводить до зменшення кількості даних, які використовуються для подання відеопотоку. Стиснення відео дозволяє ефективно зменшувати потік даних, необхідний для передачі відео по каналах радіомовлення, зменшувати простір, необхідний для зберігання даних на CD, DVD або жорстких дисках. Недоліки: якщо стиснення відбувається із втратами, виникають характерні і добре помітні артефакти – наприклад, блочність (розбиття зображення на блоки 8×8 пікселів), замилювання (втрата дрібних деталей зображення) і т. ін. Існують і способи стиснення відео без втрат, але натепер вони зменшують дані недостатньо.

Стиснення або компресія відеоінформації являє собою процес її ущільнення, при якому цифрові дані приводяться до форми, що потребує меншого об'єму пам'яті для їх зберігання. Стиснення даних досягається за рахунок усунення

надлишковості компонентів даних, без яких можна обійтися для правильного відображення вихідної інформації.

Стиснення відео ґрунтується на двох важливих принципах. Перший – це просторова надлишковість, притаманна кожному кадру відеоряду. Другий принцип заснований на тому факті, що більшу частину часу кожен кадр подібний до свого попередника. Це називають часовою збитковістю. Таким чином, типовий метод стиснення відео розпочинається з кодування першого кадру з допомогою декількох алгоритмів стиснення зображення. Потім кодується кожний наступний кадр, знаходяться розбіжності між цим кадром і його попередником і кодується ця розбіжність. Якщо новий кадр сильно відрізняється від попереднього, то його можна кодувати незалежним чином.

Багато типів даних мають в собі статичну надлишковість. Такі дані можна ефективно стискувати використовуючи компресію без втрат. У разі використання стиснення без втрат результат декомпресії точно (біт до біта) відповідатиме оригіналу. Проте стисненням без втрат неможливо досягти високих коефіцієнтів стиснення на реальному (не штучному) відео. Найкращий результат, який можна досягнути використанням стандартного алгоритму JPEG-LS, дає коефіцієнт стиснення приблизно в 3-4 рази по відношенню до вихідного об‘єму даних.

Для досягнення більшої ефективності стиснення відео даних приходиться застосовувати стиснення з втратами. Методи стиснення відео з втратами ґрунтуються на усуненні суб‘єктивної надлишковості тих елементів зображення, які можна усунути без помітного впливу на зорове сприйняття відео. При стисненні використовується декілька типів надмірності:

- **Когерентність областей зображення** - мала зміна кольору зображення в сусідніх пікселях (властивість, яку експлуатують всі алгоритми стиснення зображень з втратами).
- **Надмірність в колірних площинах** - використовується велика важливість яскравості зображення для сприйняття.
- **Подібність між кадрами** - використовування того факту, що на швидкості 25 кадрів в секунду, як правило, сусідні кадри майже не змінюються.

Перші два пункти використовуються також в алгоритмах стиснення графіки. Використання подібності між кадрами в найпростішому і найбільш часто використовуваному випадку означає кодування не кожного нового кадру, а його різниці з попереднім кадром. Для відео типу "голова, що говорить" (передача новин, відеотелефони) велика частина кадру залишається незмінною й навіть такий простий метод дозволяє значно зменшити потік даних. Більш складний метод полягає в знаходженні для кожного блоку в кадрі, який стискується, найменш відмінного від нього блоку в кадрі, який використовується в якості базового. Далі кодується різниця між цими блоками. Цей метод істотно більш ресурсномісткий.

Починаючи з появи першого стандарту компресії відео MPEG-1 (MPEG - Moving Picture Experts Group) відбувається неперервний розвиток і удосконалення методів компресії і декомпресії цифрового відео. На даний час, це питання представляє собою надзвичайно актуальну сферу наукових досліджень.

Визначимося з основними поняттями, які використовуються при стисненні відео. Відеопотік характеризується **роздільною здатністю, частотою кадрів і системою представлення кольорів**. З телевізійних стандартів прийшла роздільна здатність в 720x576 і 640x480, і частоти в 25 (стандарти PAL або SECAM) і 30 (стандарт NTSC) кадрів в секунду. Для низької роздільної здатності існують спеціальні назви CIF - Common Interchange Format, складає 352x288 і QCIF - Quartered Common Interchange Format, складає 176x144. Оскільки CIF і QCIF орієнтовані на невеликі потоки, то з ними працюють на частотах від 5 до 30 кадрів в секунду.

2. Вимоги додатків до алгоритмів стиснення відео

Для алгоритмів стиснення відео характерна більшість тих же вимог додатків, які пред'являються до алгоритмів стиснення графіки, проте є і певна специфіка:

1. Довільний доступ - мається на увазі можливість знайти і показати будь-який кадр за обмежений час. Забезпечується наявністю в потоці даних так званих точок входу - кадрів, стиснутих незалежно (тобто як звичайне статичне зображення). Прийнятним часом пошуку довільного кадру вважається 1/2 секунди.
2. Швидкий пошук вперед/назад - мається на увазі швидкий показ кадрів, не наступних один за одним в початковому потоці. Вимагає наявності додаткової інформації в потоці. Ця можливість активно використовується різними програвачами.
3. Показ кадрів у зворотному напрямі. Рідко потрібен в додатках. При жорстких обмеженнях на час показу чергового кадру виконання цієї вимоги може різко зменшити ступінь стиснення.
4. Аудіо-візуальна синхронізація - найсерйозніша вимога. Дані, необхідні для того, щоб добитися синхронності аудіо і відео доріжок, істотно збільшують розмір фільму. Для відеосистеми це означає, що, якщо ми не встигаємо дістати і показати в потрібний момент часу якийсь кадр, то ми повинні уміти коректно показати, наприклад, кадр, наступний за ним. Якщо ми показуємо фільм без звуку, то можна дозволити собі трохи більш повільний або більш швидкий показ. За часів порівняно недосконалого німого кіно кадри йшли настільки нерівномірно, наскільки нерівномірно крутив ручку камери оператор. Показ без звуку фільму, знятого такими недосконалими методами, сприймається нормально навіть за умови, що частота кадрів, що показуються, постійна (і герої фільму то

пересуваються карикатурно швидко, то повільно). Проте дивитися фільм, в якому відеосистема не встигає за звуком, - стає мукою.

5. Стійкість до помилок - вимога, обумовлена тим, що більшість каналів зв'язку ненадійна. Зіпсоване перешкодою зображення повинне швидко відновлюватися. Вимога достатньо легко задовольняється необхідним числом незалежних кадрів в потоці. При цьому також зменшується ступінь стиснення, оскільки на екрані 2-3 секунди (50-75 кадрів) може бути одне і те ж зображення, але ми будемо вимушені навантажувати потік незалежними кадрами.

6. Час кодування/декодування. В багатьох системах (наприклад, відеотелефонах) загальна затримка на кодування – передачу – декодування повинна складати не більше 150 мс. Крім того, в додатках, де необхідне редагування, нормальна інтерактивна робота неможлива, якщо час реакції системи складає більше 1 секунди.

7. Редагованість. Під редагованістю розуміється можливість змінювати всі кадри так само легко, неначе вони були записані незалежно.

8. Масштабованість - простота реалізації концепції "відео у вікні". Ми повинні уміти швидко змінювати висоту і ширину зображення в пікселях. Масштабування здатне породити неприємні ефекти в алгоритмах заснованих на ДКП (дискретному косинусному перетворенні). Коректно реалізувати цю можливість для MPEG на даний момент можна, мабуть, лише при достатньо складних апаратних реалізаціях, тільки тоді алгоритми масштабування істотно не збільшуватимуть час декодування. Цікаво, що масштабування достатньо легко здійснюється в фрактальних алгоритмах. В них, навіть при збільшенні зображення у декілька разів, воно не розпадається на квадрати, тобто відсутній ефект "зернистості". Якщо необхідно зменшувати зображення (що, хоча і рідко, але потрібно), то з такою задачею добре справляються алгоритми, засновані на wavelet перетворенні.

9. Невелика вартість апаратної реалізації. При розробці хоча б приблизно повинна оцінюватися і враховуватися кінцева вартість. Якщо ця вартість велика, то навіть при використовуванні алгоритму в міжнародних стандартах, виробники пропонуватимуть свої, більш конкурентоздатні, алгоритми і рішення. На практиці ця вимога означає, що алгоритм повинен реалізовуватися невеликим набором мікросхем.

Описані вимоги до алгоритмів суперечливі. Очевидно, що високий ступінь стиснення має на увазі архівацію кожного подальшого кадру з використанням попереднього. в той же час вимоги на аудіо-візуальну синхронізацію і довільний доступ до будь-якого кадру за обмежений час не дають можливості витягнути всі кадри в ланцюжок. І, проте, можна спробувати дійти деякого компромісу. Збалансована реалізація, що враховує систему суперечливих вимог, може досягатися на практиці за рахунок налаштувань компресора при стисненні конкретного відео.

3. Визначення класів програмного і апаратного забезпечення для алгоритмів стиснення відео

Під процедурою визначення вимог, що пред'являються до алгоритму, розуміється з'ясування класів програмного і апаратного забезпечення, на які він орієнтований і, відповідно, вироблення вимог до нього.

Носії інформації, на які орієнтований алгоритм:

1. **DVD-ROM** - порівняно низька вартість, дуже висока щільність запису інформації роблять його перспективним пристроєм для зберігання цифрового відео.
2. **CD-ROM** - низька вартість при високій щільності запису інформації роблять його досить привабливим пристроєм для зберігання цифрового відео. До недоліків відноситься порівняно малий об'єм, проте диск володіє рекордно низьким відношенням вартості диска до об'єму.
3. **Жорсткий диск** - найшвидший і гнучкий пристрій, що володіє дуже малим часом пошуку, що необхідне для деяких додатків. Проте він має високе співвідношення вартості диска до об'єму.
4. **Перезаписувані оптичні диски** - один з найперспективніших пристроїв, здатних поєднувати в собі переваги CD-ROM (низька собівартість зберігання інформації, великий об'єм, довільний доступ) і жорсткого диска (можливість перезапису).
5. **Комп'ютерні мережі** (як глобальні, так і локальні). Характеризуються можливістю швидко одержувати практично необмежені об'єми інформації. До недоліків мереж, з якими борються так звані технології QoS (Quality Service - гарантована якість сервісу), відносяться можливі затримки пакетів, і довільна зміна пропускної спроможності каналу.

Програмне забезпечення, що використовує відео-компресію, можна підрозділити на дві групи - **симетричне** і **асиметричне**.

Асиметричні додатки пред'являють серйозні вимоги до декодера (як правило, за часом і пам'яті), але для них байдужі витрати ресурсів при кодуванні. Прикладом є різні мультимедіа енциклопедії, путівники, довідники, ігри і просто фільми. При такій постановці задачі з'являється можливість застосувати складні алгоритми компресії, що дозволяють одержати великий ступінь стиснення даних.

Симетричні додатки пред'являють однаково жорсткі вимоги до часу, пам'яті і інших ресурсів, як при кодуванні, так і при декодуванні. Прикладами такого роду додатків можуть служити відеопошта, відеотелефон, відеоконференції, редагування і підготовка відеоматеріалів.

4. Огляд стандартів стиснення відеоданих

У 1988 році в рамках Міжнародної Організації по Стандартизації (ISO) почала роботу група MPEG (Moving Pictures Experts Group) - група експертів в області цифрового відео (ISO-IEC/JTC1/SC2/WG11/MPEG). Група працювала в напрямах, які можна умовно назвати MPEG-Video - стиснення відеосигналу в потік з швидкістю до 1,5 Мбіт/сек, MPEG-Audio - стиснення звуку до 64, 128 або 192 Кбіт/сек на канал і MPEG-System - синхронізація відео і аудіо потоків. Нас в основному цікавитимуть досягнення MPEG-Video, хоча очевидно, що багато рішень в цьому напрямі приймалися з урахуванням вимог синхронізації.

Як алгоритм, MPEG має декількох попередників. Це, перш за все, універсальний алгоритм JPEG. Його універсальність означає, що JPEG показує непогані результати на широкому класі зображень.

Якщо бути більш точним, то стандарт MPEG, як і інші стандарти на стиснення, описує лише вихідний бітовий потік, неявно задаючи алгоритми кодування і декодування. При цьому їх реалізація перекладається на програмістів-розробників. Такий підхід відкриває широкі горизонти для тих, хто бажає оптимально реалізувати алгоритм для конкретного обчислювального пристрою (контроллера, ПК, розподіленої обчислювальної системи, операційної системи, відеокарти і т.п.). При спеціалізованих реалізаціях можуть бути враховані вельми специфічні вимоги на час роботи, витрати пам'яті і якість одержуваних зображень. Алгоритми стиснення відео вельми гнучкі і часто для різних підходів до реалізації, можна одержати істотну різницю за якістю відео, при одному і тому ж ступені стиснення. Більш того - для одного і того ж стислого файлу за допомогою різних алгоритмів декодування можна одержати фільми, що істотно розрізняються по візуальній якості. Часто "проста" реалізація стандарту дає смикаючий відеоряд з добре помітними блоками, тоді як програми відомих виробників програють цей же файл цілком плавно і без впадаючої в очі блоковості. Ці нюанси необхідно добре собі представляти, коли мова заходить про порівняння різних стандартів.

У вересні 1990 р. був представлений попередній стандарт кодування MPEG-1. В січні 1992 роботу над MPEG-1 була завершено, і почата робота над MPEG-2, в задачу якого входив опис потоку даних з швидкістю від 3 до 10 Мбіт/сек. Практично в той же час була почата робота над MPEG-3, який був призначений для опису потоків 20-40 Мбіт/сек. Проте незабаром з'ясувалося, що алгоритмічні рішення для MPEG-2 і MPEG-3 принципово близькі і можна безболісно розширити рамки MPEG-2 до потоків в 40 Мбіт/сек. В результаті робота над MPEG-3 була припинена. MPEG-2 був остаточно допрацьований до 1995 року.

У 1991 групою експертів по відеотелефонах (EGVT) при Міжнародному консультативному комітеті з телефонії і телеграфії (CCITT) запропонований стандарт відеотелефонів px64 Kbits. Запис px64 означає, що алгоритм орієнтований на паралельну передачу оцифрованого відеозображення по p каналах

з пропускною спроможністю 64 Кбіта/сек. Таким чином, охоплюючи декілька телефонних ліній, можна одержувати зображення цілком прийнятної якості. Одним з головних обмежень при створенні алгоритму був час затримки, який повинен був складати не більше 150 мс. Крім того, рівень перешкод в телефонних каналах достатньо високий, і це, природно, знайшло віддзеркалення в алгоритмі. Можна вважати, що px64 Kbits - попередник MPEG-а для потоків даних менше 1,5 Мбіт/сек і специфічного класу відео.

У групі при CMTT (сумісний комітет при CCITT і CCIR - International Consultative Committee on bRoadcasting) роботи були направлені на передачу оцифрованого відео по виділених каналах з високою пропускною спроможністю і радіолініях. Відповідні стандарти H21 і H22 орієнтовані на 34 і 45 Мбіт/сек, і сигнал передається з дуже високою якістю.

MPEG-4 спочатку був задуманий як стандарт для роботи з наднизькими потоками. Проте в процесі досить довгої підготовки стандарт зазнав абсолютно революційних змін і зараз власне стиснення з низьким потоком входить в нього як одна складова частина, причому достатньо невелика за розміром. Наприклад, сам формат сьогодні включає такі речі, як синтез мови, рендеринг зображень і опису параметрів візуалізації особи на стороні програми перегляду.

Ре́ндеринг, комп'ютерна візуалізація *(англ. rendering — візуалізація, вимальовування, подання) — в комп'ютерній графіці — це процес отримання зображення за моделлю з допомогою комп'ютерної програми. Тут модель — це опис тривимірних об'єктів (3D, 3Д) на визначеній мові програмування і у вигляді структури даних. Такий опис може містити геометричні дані, положення точки спостерігача, інформацію про освітлення. А зображення — це цифрове растрове зображення.*

Слово рендеринг в Україні вживають для вказування процесу візуалізації, що виконується за допомогою програмного забезпечення, а рендер — для позначення готового зображення, тобто як синоніми до словосполучень: комп'ютерна візуалізація — комп'ютерний рендеринг, візуалізований об'єкт — рендер.

На додаток до вищезазначених існують стандарти, які є не удосконаленням попередніх стандартів стиснення, а визначають різні мови опису:

MPEG-7: Стандарт індексації мультимедіа-вмісту.

MPEG-21: MPEG описує стандарт як мультимедійне середовище розробки.

Розробка MPEG-7 була почата в 1996. Власне до алгоритмів стиснення відео цей стандарт має ще менше відношення, ніж MPEG-4, оскільки його основна задача полягає в описі контента і управлінні їм.

Паралельно весь цей час існували формати Motion-JPEG і Motion-JPEG2000, призначені в основному для зручності обробки стислого відео. Розглянемо основні стандарти і алгоритми, що є їх основою, детальніше.

Отже, успіхи цифрової відеоіндустрії (перш за все широкоформатного цифрового відео і DVD-відео) базувалися на міжнародному стандарті ISO/IEC

13818, широко відомому під абревіатурою MPEG-2. Необхідність кращого стиснення відеоданих спонукало розробку подальших стандартів відеостиснення, відомих під назвами ISO/IEC 14496 Part 2 (MPEG - 4 Visual) і рекомендації організації ITU, - E H.264/ISO/IEC 144496 Part 10 (скорочено H.264). Стандарти MPEG - 4 Visual і H.264 мають загальне походження і багато загальних рис. Вони були розроблені на основі більш ранніх стандартів стиснення. Проте вони розвивають старі стандарти в істотно різних напрямках. Стандарт MPEG-4 Visual віддаляється від прямокутного відеокадру, пропонує гнучкий і відкритий погляд на візуальні комунікації і використовує високоефективне відеостискування і об'єктно-орієнтовану обробку даних. Стандарт H.264 має прагматичніший погляд. Він прагне виконувати ті ж дії, що і попередні стандарти (забезпечуючи механізм стискування для прямокутних кадрів), але з більшою ефективністю і стійкістю, забезпечуючи сумісність зі всіма широко поширеними типами додатків, такими як широкоекранне телебачення, зберігання візуальної інформації і передача потокового відео.

Для різних сфер використання цифрового відео висувалися різні вимоги до стиснення, які привели до формування ряду стандартів з різними областями застосування:

- необхідність стиснення для відеоконференцій привело до виникнення стандартів ITU H.261 для ISDN-відеоконференцій, для відеоконференцій в телефонних мережах і H.263 для відеоконференцій в мережах ATM і по широкополосним каналам;
- необхідність стиснення відеопослідовностей для зберігання на CD-ROM (з умовою забезпечення 1.2 Мбіт/с для відеопотоку і 256 Кбіт/с ISO MPEG-1;
- для мовлення і зберігання на DVD, з бітрейтом від 2 до 15 Мбіт/с для відео і аудіо, був розроблений стандарт ISO MPEG-2;
- необхідність кодування окремих аудіо-візуальних об’єктів як природного походження, так і синтезованих, привело до створення ISO MPEG-4. Цей стандарт включає в себе декілька частин, в яких розглядається, окрім кодування відео, аудіокодування, кодування об’єктів і т.д. До відео відносяться частини 2 (ISO 14496-2 або MPEG-4 Part 2) і 10 (ISO 14496-10 або MPEG-4 Part 10).

5. Базові технології стиснення відео

5.1 Опис алгоритму компресії

Для стиснення цифрового відео необхідно мати дві системи, які доповнюють одна іншу: компресор (кодер) та декомпресор (декодер). Кодер перетворює відеодані в стиснену форму для подальшої передачі або зберігання відео, а декодер робить зворотне перетворення, повертаючи стисненні відеодані до

вигляду придатного для показу глядачу. Пару кодер/декодер прийнято називати "кодеком."

Більшість методів кодування відео використовують часову та просторову надлишковість. В часовій області існує значна подібність між відеокадрами які знаходяться в близькі моменти часу. Суміжні по часу кадри (які ідуть один за одним) мають високий ступінь кореляції, особливо при великій частоті кадрів. В просторовій області також спостерігається висока залежність величин пікселів (семплів), які знаходяться поруч.

Кодек перетворює початковий відеоряд за допомогою певної моделі. Модель кодування - це ефективне кодоване представлення відеоданих, за допомогою якого можна реконструювати ці дані з певним ступенем точності. У ідеалі модель повинна представляти послідовність з найменшим числом бітів і найбільшою можливою точністю. Ці дві мети (висока якість і ефективність стиснення) зазвичай суперечать одна одній, оскільки високий ступінь стиснення відеоданих припускає істотне зниження якості на виході декодера.

Відеокодек складається з наступних основних функціональних блоків:

• блоку перетворення колірного простору;
• блоку усунення часової надлишковості;
• блоку усунення просторової надлишковості;
• блоку ентропійного кодера.

Більшість алгоритмів використовують перетворення з кольорового простору RGB в кольоровий простір YUV (YCbCr), де Y - компоненту, що відповідає за яскравість, а U(Cb) і V(Cr) – компоненти, що відповідають за колір, для відділення яскравості від кольору. Людський зір чутливіший до яскравості, чим до кольору. Користуючись цим, можна ще до апроксимації зменшити кількість інформації про колір в 2 або 4 рази. Спрощено переклад з колірного простору RGB в колірний простір YUV можна представити за допомогою матриці переходу:

$$\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} = \begin{vmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0{,}5 & -0{,}4187 & -0{,}0813 \\ 0{,}1687 & -0{,}3313 & 0{,}5 \end{vmatrix} * \begin{vmatrix} R \\ G \\ B \end{vmatrix} + \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix}$$

Перетворення [Y Cb Cr] у [R G B] (зворотно до попереднього перетворення). RGB-колір може бути обчислений безпосередньо з Y, Cb, Cr у такий спосіб:

$$\begin{vmatrix} R \\ G \\ B \end{vmatrix} = \begin{vmatrix} 1 & 0 & 1{,}402 \\ 1 & -0{,}34414 & -0{,}71414 \\ 1 & 1{,}772 & 0 \end{vmatrix} * \left(\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} - \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix} \right)$$

Очевидно, що при зменшенні інформації про колір знижується якість зображення.

Технологія стиснення відео в MPEG розділяється на дві частини: зменшення надмірності відеоінформації в тимчасовому вимірюванні, засноване на тому, що

сусідні кадри, як правило, відрізняються не сильно, і стиснення окремих зображень.

Для того, щоб задовольнити суперечливим вимогам і збільшити гнучкість алгоритму, розглядається чотири типи кадрів:

- I-кадри – кадри, стислі незалежно від інших кадрів (I-Intra pictures)
- P-кадри - кадри, стислі з використанням посилання на одне зображення (P-Predicted)
- B-кадри - кадри, стислі з використанням посилання на два зображення (B-Bidirection)
- DC-кадри - незалежно стислі кадри з великою втратою якості (використовуються тільки при швидкому пошуку).

I-кадри забезпечують можливість довільного доступу до будь-якого кадру, будучи своєрідними вхідними крапками в потік даних для декодера. P-кадри використовують при архівації посилання на один I- або P-кадр, підвищуючи тим самим ступінь стиснення фільму в цілому. B-кадри, використовуючи посилання на два кадри, що знаходяться попереду і позаду, забезпечують найвищий ступінь стиснення. Самі як посилання використовуватися не можуть. Послідовність кадрів у фільмі може бути, наприклад, така: IBBPBBPBBPBBIBBPBB... Або, якщо ми не економимо на ступені стиснення, така (рис. 16.1):

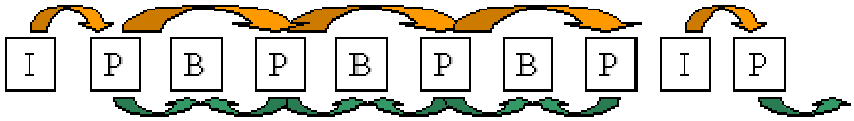

Рисунок 16.1 – I-кадри - незалежно стислі (I-Intrapictures), P-кадри - стислі з використанням посилання на одне зображення (P-Predicted), B-кадри - стислі з використанням посилання на два зображення (B-Bidirection)

Частота I-кадрів вибирається залежно від вимог на час довільного доступу і надійності потоку при передачі через канал з помилками. Співвідношення P- і B-кадрів підбирається, виходячи з вимог до величини компресії і обмежень декодера. Як правило, декодування B-кадрів вимагає більше обчислювальних потужностей, проте дозволяє підвищити ступінь стиснення. Саме варіювання частоти кадрів різних типів забезпечує алгоритму необхідну гнучкість і можливість розширення. Зрозуміло, що для того, щоб розпакувати B-кадр, ми повинні вже розпакувати ті кадри, на які він посилається. Тому для послідовності IBBPBBPBBPBBIBBPBB кадри у фільмі будуть записані так: 0**312645..., де цифри - номери кадрів, а зірочкам відповідають або B-кадри з номерами -1 і -2, якщо ми знаходимося в середині потоку, або порожні кадри (нічого), якщо ми на початку фільму. Подібний формат володіє достатньо великою гнучкістю і здатний задовольняти самим різним наборам вимог.

Загальна схема усунення просторової надлишковості і ентропійного кодування для стандартів MPEG і H.263/264 визначає процес стискування окремого кадру відеопотоку і збігається з процедурою стискування статичного повнокольорового зображення за стандартом JPEG (рис.16.2).

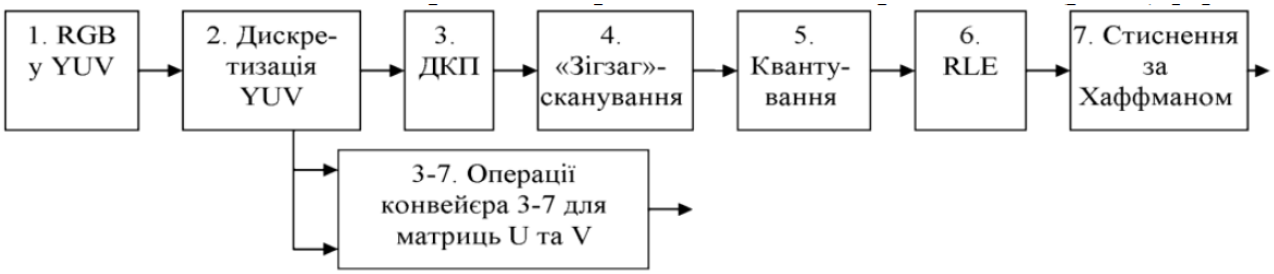

Рисунок 16.2 – Основні етапи процедури стиснення по стандарту JPEG

Одним з основних понять при стисненні декількох зображень є поняття макроблоку. При стисненні кадр з колірного простору RGB переводиться в колірний простір YUV. Кожна з площин зображення (У, U, V), що стискається, розділяється на блоки 8x8, з якими працює ДКП. Причому площини U і V, відповідні компоненті кольоровості беруться з дозволом в два рази меншим (по вертикалі і горизонталі), ніж початкове зображення. Таким чином, ми відразу одержуємо стиснення в два рази, користуючись тим, що око людини гірше розрізняє колір окремої точки зображення, ніж її яскравість. Блоки 8x8 групуються в макроблоки.

Макроблок - це група з чотирьох сусідніх блоків в площині яскравості компоненти У (матриця пікселів 16x16 елементів) і два відповідних їм по розташуванню блоки з площин кольоровості U і V. Таким чином, кадр розбивається на незалежні одиниці, що мають повну інформацію про частину зображення. При цьому розмір зображення повинен бути кратний 16.

Окремі макроблоки стискаються незалежно, тобто в B-кадрах ми можемо стиснути макроблок конкретний як I-блок, P-блок з посиланням на попередній кадр, P-блок з посиланням на подальший кадр і, нарешті, як B-блок.

5.2 Загальна схема алгоритму

У цілому весь конвейєр перетворень можна представити так:

1. Підготовка макроблоків. Для кожного макроблоку визначається, яким чином він буде стислий. В I-кадрах всі макроблоки стискаються незалежно. В P-кадрах блок або стискається незалежно, або є різницею з одним з макроблоків в попередньому опорному кадрі, на який посилається P-кадр.
2. Переведення макроблоку в колірний простір YUV. Отримання потрібної кількості матриць 8x8.

3. Для Р-блоків і В-блоків проводиться обчислення різниці з відповідним макроблоком в опорному кадрі.

4. ДКП (дискретне косинусне перетворення)

5. Зігзаг-сканування.

6. Квантування.

7. Групове кодування.

8. Кодування по Хаффману.

При декодуванні весь конвейєр повторюється для зворотних перетворень, починаючи з кінця.

Контрольні питання

1. Назвіть основні вимоги до алгоритмів стиснення відео.
2. Що таке І-кадри, Р-кадри? Як вибирається частота І – кадрів? Як підбирається співвідношення Р – кадрів і В – кадрів?
3. Як називаються кадри, що стискаються незалежно від інших кадрів, згідно стандарту MPEG та забезпечують можливість довільного доступу до будь-якого кадру?
4. Як називаються кадри, що стискаються з використанням посилання на одне зображення, підвищуючи ступінь стиснення, згідно стандарту MPEG?
5. Як називаються кадри, що стискаються з використанням посилання на два зображення, забезпечуючи найвищий ступінь стиснення (стандарт MPEG)?
6. Назвіть принцип стиснення відео, який ґрунтується на твердженні, що більшу частину часу кожний кадр подібний до свого попередника.
7. Назвіть тип надмірності відео, що використовує властивість малої зміни кольору зображення в сусідніх пікселях.
8. Назвіть тип надмірності відео, що використовує важливість яскравості зображення для сприйняття.
9. Назвіть тип надмірності відео, який використовує той факт, що на швидкості 25 кадрів в секунду, як правило, сусідні кадри майже не змінюються.
10. Назвіть вимогу, щодо можливості знайти і показати будь-який кадр за обмежений час.
11. Назвіть вимогу, щодо можливості змінювати всі кадри так само легко, неначе вони були записані незалежно.
12. Назвіть вимогу, щодо можливості швидко змінювати висоту і ширину зображення в пікселях.
13. Яке програмне забезпечення використовує відео-компресію і пред`являє жорсткі вимоги до часу, пам’яті і інших ресурсів, як при кодуванні, так і при декодуванні?
14. Яке програмне забезпечення використовує відео-компресію і пред`являє серйозні вимоги тільки до декодера (як правило, за часом і пам’яттю)?

ЛЕКЦІЯ 17

на тему: Технології і алгоритми стиснення відеоданих (частина 1)

У лекції розглядаються стандарти, технології і алгоритми стиснення цифрового відео, розглянуті поширені алгоритми стиснення відео без втрат.

План:

1. Технології стиснення цифрового відео
2. Огляд технологій та алгоритмів стиснення відео
3. Алгоритми стиснення відео без втрат

1. Технології стиснення цифрового відео

Існує безліч технологій стиснення цифрового відео, однак зупинимося на тих з них, які лягли в основу найбільш популярних компресорів відео. Деякі з розглянутих компресорів використовують не одну технологію стиснення, а деяку їх сукупність. Наприклад, і Indeo 3.2, і Cinepak використовують векторне квантування. Міжнародні стандарти MPEG-1, MPEG-2, MPEG-4, H.261 і H.263 використовують комбіновану технологію ДКП і компенсацію руху. Деякі сучасні алгоритми використовують технологію ДВП (Discrete Wavelet Transform, або DWT). Інші технології включають Фрактальное стиснення зображень (Fractal Image Compression).

Для початку давайте визначимо поняття «якість стиснення відео». Незважаючи на всю суб'єктивність такого поняття, все ж спробуємо умовно розділити якість стиснення на наступні рівні (ступені стиснення).

1. Стиснення без втрат якості

Стиснення зображень може здійснюватися без втрат якості лише в тому випадку, якщо в процесі стиснення не було втрат даних. В результаті отримане після декомпресії зображення буде в точності (побітно) збігатися з оригіналом. Прикладом такого стиснення може служити формат GIF для статичної графіки і GIF89a для відео.

2. Стиснення з втратами якості

Стиснення може відбуватися з втратами якості, якщо в процесі стиснення інформація була втрачена. Однак з точки зору людського сприйняття стисненням з втратами слід вважати лише таке стиснення, при якому можливе на око відрізнити результат стиснення від оригіналу. Таким чином, незважаючи на те що два зображення - оригінал і результат стиснення з використанням того чи іншого компресора – побітно можуть не збігатися, проте різниця між ними може бути зовсім непомітною. Прикладом може служити алгоритм JPEG для стиснення статичної графіки та алгоритм M-JPEG для стиснення відео.

3. Стиснення без втрат з точки зору сприйняття

Формально будучи стисненням з втратами якості, схема стиснення може в той же час здаватися стисненням без втрат з точки зору сприйняття її людиною. Більшість технологій стиснення з формальної втратою якості мають так званий Фактор Якості Стиснення (ФЯС), що характеризує саме сприйняту якість і варіюється в межах від 0 до 100. При факторі якості стиснення, що дорівнює 100, сприймаються характеристики якості стисненого відео не відрізняються від оригіналу, з точки зору сприйняття.

4. Стиснення з природною втратою якості

JPEG і MPEG і інші технології стиснення з втратою якості іноді стискають, без втрат переступаючи за грань стиснення з точки зору сприйняття відеоінформації. Проте стислі відео і статичні зображення цілком прийнятні для адекватного сприйняття їх людиною. Іншими словами, в даному випадку спостерігається так звана природна деградація зображення, при якій губляться деякі дрібні деталі сцени. Схоже може відбуватися і в природних умовах, наприклад під час дощу або туману. Зображення в таких умовах, як правило, помітно, проте деталізація його зменшується.

5. Стиснення з неприродною втратою якості

Низька якість стиснення, в значній мірі спотворює зображення і вносить в нього штучні (неіснуючі в оригіналі) деталі сцени, називається неприродним стисненням з втратою якості. Прикладом тому може служити деяка «блочність» в сильно стислому MPEG-і і в інших компресорах, що використовують технологію ДКП. Неприродність полягає в першу чергу в порушенні найважливіших з точки зору сприйняття людиною характеристик зображення - контурів. Досвід показує, що саме контури дозволяють сприймаючому апарату людини правильно ідентифікувати той чи інший візуальний об'єкт.

Отже, алгоритми стиснення відеоданих можна поділити на дві групи:

– алгоритми стиснення без втрат (GIF 89a, CorePNG, HuffYUV, Lagarith);

– алгоритми стиснення із втратами (MPEG-X, H.XXX).

Стиснення цифрових відеоданих без втрат виконується шляхом заміни оригінальної послідовності бітів іншою послідовністю – що містить опис оригінальної послідовності. При цьому скорочення відбувається за рахунок фрагментів, що повторюються, і фрагментів, що містять закономірний ряд змін. У разі потреби отримати початковий потік даних його відновлюють за потоком-описом.

Стиснення даних із втратами виконують за двома схемами:

1. У трансформувальних кодеках фрейми зображень трансформуються в новий базисний простір і виконується квантування. Трансформація може здійснюватися або для всього фрейму цілком (як, наприклад, у схемах на основі wavelet-перетворення), або поблоково (характерний приклад – JPEG). Результат потім стискується методами ентропії.

2. У кодеках, які використовують метод передбачення, попередні й наступні дані використовуються для того, щоб передбачити поточний семпл зображення. Помилка між передбаченими даними і реальними разом з додатковою інформацією потім квантується і кодується.

У деяких системах обидві наведені техніки комбінуються через використання трансформуючих кодеків для стискання помилкових сигналів, згенерованих на стадії передбачення.

Відзначимо, що всі широко відомі відеокомпресори використовують технології стиснення з втратами якості. При досить високих коефіцієнтах стиснення все вони будуть стискати з неприродною втратою якості.

Таким чином, вибираючи той чи інший компресор для стиснення цифрового відео, необхідно досягти стиснення, принаймні з природними втратами якості.

2. Огляд технологій та алгоритмів стиснення відео

2.1 Групове кодування (Run Length Encoding, RLE)

Компресорами, що використовують технологію RLE, є Microsoft RLE (MRLE). RLE використовується також для кодування коефіцієнтів в ДКП, застосовується в MPEG-1234, H.261, H.263 і JPEG.

RLE кодує послідовність повторюваних елементів зображення або одноколірних елементів одним кодовим словом. RLE добре стискає зображення, в яких спостерігається повторення контурів або кольорів окремих елементів. У повнокольорових зображеннях повторень кольору значно менше, тому стиснення повнокольорового відео з використанням технології RLE позбавлене всякого сенсу.

Переваги і недоліки:

1. Працює виключно з 8-бітовими зображеннями.
2. Не підходить для стиснення повнокольорового відео.

2.2 Векторна квантизация (Vector Quantization, VQ)

Компресорами, що використовують технологію VQ, є Indeo 3.2 і Cinepak. Обидва вони застосовують колірну схему YUV (а не RGB).

Основна ідея векторного квантування полягає в розбитті зображення на блоки (розміром 4x4 пікселя в колірній схемі YUV для компресорів Indeo і Cinepak). Як правило, деякі блоки виявляються схожими один на одного. В цьому випадку компресор ідентифікує клас схожих блоків і замінює їх одним загальним блоком. Крім того, генерується двійкова таблиця (карта) таких загальних блоків з найкоротших кодових слів. VQ-декодер потім, використовуючи таблицю, збирає зображення по блоках із загальних блоків.

Процес кодування тривалий і трудомісткий, так як кодеру необхідно виявляти приналежність кожного блоку зображення до якогось спільного блоку.

Однак завдання декодування у цьому випадку зводиться до задачі побудови зображення по заданій карті із загальних блоків і не займає багато апаратних і часових ресурсів. Таблицю або карту також називають ще і кодовою книгою, а двійкові коди, що входять до неї, - кодовими словами, відповідно. Найбільше стиснення з використанням алгоритму VQ досягається шляхом зменшення числа класів загальних блоків, тобто припущенням про схожість щодо більшого числа блоків зображення, і, як наслідок, зменшенням кодової книги. У міру зменшення розмірів кодової книги якість відтвореного відео погіршується. В результаті на зображенні з'являється штучна «блочність».

Важливою особливістю технології VQ є те, що при стисненні відео одна і та ж кодова книга може використовуватися для декількох кадрів зображення.

Переваги і недоліки:

1. Процес кодування дуже трудомісткий і практично неможливий без спеціального додаткового обладнання.

2. Процес декодування дуже швидкий.

3. Блокові спотворення при високих коефіцієнтах стиснення.

4. Технології, що використовують алгоритми ДКП, ДВП можуть досягати більш високих рівнів стиснення.

2.3 Дискретне косинусне перетворення (ДКП)

Компресори, що використовують ДКП: Motion JPEG; Editable MPEG; MPEG-1; MPEG-2; MPEG-4.

ДКП є широко використовуваним при стисненні зображень перетворенням. Стандарт стиснення статичної графіки JPEG, використовуваний у відеоконференціях стандарт H.263, цифрові відеостандарти MPEG (MPEG-1, MPEG-2 і MPEG-4) - всі вони використовують ДКП. У цих стандартах використовується, зокрема, 2-мірне ДКП, що застосовується послідовно до блоків зображення розмірністю 8 x 8 пікселів. ДКП обчислює 64 (8x8 = 64) коефіцієнта, які потім проходять процес квантування, забезпечуючи тим самим реально стиснення. У більшості зображень більшість ДКП-коефіцієнтів в силу своєї малості після квантування обнуляються. Ця властивість ДКП і лежить в основі безлічі алгоритмів стиснення, які використовують ДКП.

До того ж відомо, що людське око набагато менш чутливе до високочастотних компонентів зображення, які представляють великими коефіцієнтами ДКП. До цих більших значень коефіцієнтів може бути застосований (і, як правило, застосовується) більший фактор квантування. Після квантування коефіцієнти піддаються обробці алгоритмом RLE. Далі для частих комбінацій використовуються короткі кодові слова, для більш рідких - відносно довгі. Здійснюється ентропійне кодування.

Застосування ДКП на блоці з N вибірок потребує N * N операцій множення і підсумовування. Однак завдяки рекурсивної структурі матриці ДКП реально

потрібна набагато менша кількість математичних операцій. Ця властивість робить ДКП реально застосовним на сучасних математичних процесорах персональних ЕОМ.

Переваги і недоліки:

1. Блоковість при високій компресії.

2. Закруглення гострих кутів зображення. Випадкове «розмивання» гострих країв зображень.

3. Кодування є дуже затратним. Тільки останнім часом вдалося здійснити процес кодування програмно, а не апаратно.

2.4 Дискретне Wavelet-перетворення (DWT)

Компресори, що використовують DWT (Discrete Wavelet Transform): Intel Indeo 5.x; Intel Indeo 4.x

DWT-алгоритм заснований на передачі сигналу, наприклад зображення, через пару фільтрів: низькочастотний і високочастотний. Фільтр низьких частот видає грубу форму вихідного сигналу. Високочастотний фільтр видає сигнал різниці або додаткової деталізації.

У свою чергу, результат на виході високочастотного фільтра (додатковий сигнал деталізації) може бути підданий тій самій процедурі і так далі.

Переваги і недоліки:

1. Більшість як статичних, так і динамічних зображень, стиснутих за допомогою алгоритму DWT, не має характерної для алгоритму ДКП блокової структури.

2. Відносна якість зображень, стиснутих з використанням DWT, перевершує якість зображень, стиснутих за допомогою ДКП, при тих же коефіцієнтах стиснення.

3. DWT трохи розмазує, заокруглює гострі контури зображення. Так званий контурний шум або ефект Гіббса.

2.5 Різниця кадрів

Компресорами, що використовують технологію різниці кадрів, є: Cinepak

Алгоритм різниці кадрів використовує ту властивість, що в багатьох відео зображення від кадру до кадру мало чим відрізняється. У міру застосування алгоритму різниці кадрів для кодування кожного наступного кадру і отримання при цьому малих коефіцієнтів, які важко кодуються, в кадри поступово вкрадається помилка. Це вимагає включення в відеоряд так званих ключових кадрів, які кодуються без врахування попередніх і є так званими «опорними точками» в відео.

Переваги і недоліки:

1. В цілому може забезпечувати стиснення, краще, ніж незалежне стиснення окремих кадрів.

2. Помилки, що виникають в ході кодування, накопичуючись, потребуватимуть додаткового ключового кадру.

2.6 Компенсація руху

Компресорами, що використовують технологію компенсації руху, є: MPEG-1,2 і 4.

Компенсація руху заснована на використанні ряду складних алгоритмів. Сфера, де дана технологія стиснення ефективна, як правило, зводиться до відеоряду, в якому об'єкт змінює своє місце розташування відносно нерухомого фону. Об'єкти, що змінюються за формою, що наближаються або віддаляються (рухома камера), не підлягають ефективному стисненню за допомогою алгоритму компенсації руху. Стиснення можливо завданням вектора зміщення елементів зображення замість зберігання великих значень нових координат даних елементів зображення. Основним блоком (щодо якого задається вектор зміщення інших блоків) може бути будь-який блок зображення розміром 16x16 пікселів, максимально схожий на кодуємий блок. Ясно, що кадр, на який посилаються таким чином інші кадри, повинен бути декодований раніше. MPEG дозволяє виконувати передбачення в обох напрямках шляхом введення так званих B-кадрів (bi-directionally predicted).

Переваги і недоліки:

1. У порівнянні з механізмом різниці кадрів механізм компенсації руху дозволяє досягати більшого ступеня стиснення.

2. Кодування вимагає спеціальної апаратури і занадто багато роботи.

3. Технологія компенсації руху використовується в таких міжнародних стандартах стиснення цифрового відео, як: MPEG, H.261 і H.263.

4. Найбільше стиснення досягається в сценах зі зниженим рухом.

3. Алгоритми стиснення відео без втрат

3.1 GIF 89a

GIF (від англ. Graphics Interchange Format — «формат обміну зображеннями») — 8-бітний растровий графічний формат, що використовує до 256 чітких кольорів із 24-бітного діапазону RGB. Формат було розроблено компанією CompuServe у 1987 році, і з того часу набув широкої популярності у всесвітній павутині завдяки своїй відносній простоті та мобільності.

Для стискання файлів використовує LZW-компресію.

Існує дві специфікації формату GIF — GIF 87a і GIF 89a.

Перша специфікація була створена в 1987 компанією CompuServe для заміни застарілого формату RLE. GIF став популярним під час розвитку інтернету, оскільки, на відміну від інших форматів, дозволяв використовувати більш компактні за розміром файлу зображення на веб-сторінках.

Одними із головних особливостей другої специфікації формату є підтримка анімації та прозорості.

Хоча до теперішнього часу формат багато в чому застарів, і для його заміни створені формат PNG для статичних зображень та APNG для анімованих зображень, він як і раніше широко використовується. GIF-формат залишається затребуваним при створенні анімації.

3.2 CorePNG

CorePNG - це метод стиснення без втрат RGB-відеокодека, заснований на використанні методу стиснення зображень PNG. Кожен кадр відео стискається з використанням PNG-стиснення, дозволяючи використовувати всі можливості формату PNG, але також наслідуючи всі його недоліки.

PNG (англ. Portable network graphics) - растровий формат зберігання графічної інформації, що використовує стиснення без втрат за алгоритмом Deflate.

DEFLATE – це алгоритм, що використовує комбінацію алгоритму LZ і алгоритму Хаффмана. Цей метод кодування використовує принцип ковзаю чого вікна і враховує інформацію, що раніше зустрічалася, тобто інформацію, що вже відома для кодувальника й декодувальника (друге й наступне входження деякого рядка символів у повідомленні заміняються посиланнями на її перше входження).

DEFLATE достатньо простий у реалізації, швидкий, широко використовується у програмному забезпеченні. Недоліком алгоритму є мала ефективність при кодуванні незначного обсягу даних.

PNG був створений як вільний формат для заміни GIF, тому в Інтернеті з'явився рекурсивний акронім «PNG is Not GIF» (PNG - HE GIF).

CorePNG підтримує кодування відео в колірному просторі RGB з глибиною кольору 24 або 32 біт. Підтримка інших колірних просторів, таких як YUY2 або YV12, також надається.

CorePNG підтримує запис P-кадрів, званих delta кадрами, в відеопотоці вони кодують тільки відмінності між попереднім і поточним кадрами. Однак, ця функція позначена як нестабільна.

3.3 HuffYUV

Huffyuv (або HuffYUV) - кодек, призначений для стиснення відео без втрат. Незважаючи на «YUV» в назві, він використовує колірний простір не YUV, а YCbCr. Алгоритм Huffyuv передбачає кожний піксель кадру і потім кодує помилку за алгоритмом Хаффмана.

3.4 Lagarith

Lagarith - відкритий кодек, створений для кодування відеоданих без втрат інформації. Розроблено Беном Грінвудом (Ben Greenwood). Базується на відомому кодеку Huffyuv, приблизно можна порівняти з ним по швидкодії, перевершує за

ступенем стиснення. Кодек добре стискає відео з переважанням статичних зображень. Це досягається за рахунок підтримки недійсних фреймів, тобто якщо попередній фрейм ідентичний поточному, то він використовується знову, а поточний відкидається.

Особливості

• Lagarith працює в колірних просторах RGB і YUY.
• Останні версії підтримують багатопроцесорність.
• Забезпечується стиснення на 10-30% краще в порівнянні з Huffyuv.
• Від Huffyuv успадковане переведення YUY відео в RGB.

Контрольні питання

1. Які параметри треба визначити, перш ніж порівнювати два алгоритми стиснення відео?
2. Приведіть приклади ситуацій, коли архітектура комп'ютера дає переваги тому або іншому алгоритму стиснення відео.
3. Якими властивостями відеопотоку ми можемо користуватися, створюючи алгоритм стиснення? Приведіть приклади.
4. Що таке аудіо-візуальна синхронізація? Чому виконання її вимог значно знижує ступінь стиснення?
5. Назвіть алгоритм стиснення відео, який кодує послідовність елементів зображення, що повторюються, або одноколірних елементів одним кодовим словом та не підходить для стиснення повнокольорового відео.
6. Охарактеризуйте технологію стиснення відео, що полягає в розбитті зображення на блоки, ідентифікації схожих блоків і заміні їх одним загальним, а також генерації двійкової таблиці загальних блоків з найкоротших кодових слів.
7. Назвіть алгоритм стиснення відео, що використовує властивість подібності (схожості) зображення від кадру до кадру.
8. Охарактеризуйте технологію стиснення відео, що є ефективною для відеоряду, де об’єкт змінює своє місце розташування відносно нерухомого фону і задається вектором зміщення елементів зображення.
9. Вкажіть алгоритми, що відносяться до стиснення відео без втрат.
10. Назвіть формат обміну зображеннями, що використовує LZW-компресію.
11. Назвіть растровий формат зберігання графічної інформації, що використовує стиснення за алгоритмом Deflate.
12. Назвіть алгоритм стиснення, що використовує комбінацію алгоритмів LZ і Хаффмана.

ЛЕКЦІЯ 18

на тему: Технології і алгоритми стиснення відеоданих (частина 2)

У лекції розглядаються стандарти, технології і алгоритми стиснення цифрового відео з втратами і шляхи підвищення ступеня стиснення відеоданих.

План:

1. Алгоритми стиснення відео з втратами
2. Шляхи підвищення ступеня стиснення відеоданих

1. Алгоритми стиснення відео із втратами

Відомо, що сумісна робота декількох алгоритмів називається відеокодеком. Відеокодеки, які використовують різні стандарти, як правило не сумісні. Далі представлено основні стандарти, які використовують методи та алгоритми компресії-декомпресії відеоданих. Слід зазначити, що стандарти є гарантією коректної компресії-декомпресії відеоданих.

1.1 Стандарти ISO - алгоритми групи MPEG-Video

Motion-JPEG

Motion-JPEG (або M-JPEG) є найпростішим алгоритмом стиснення відео. В ньому кожний кадр стискається незалежно алгоритмом JPEG. Цей прийом дає високу швидкість доступу до довільних кадрів, як в прямому, так і в зворотному порядку проходження. Відповідно легко реалізуються плавні "перемотування" в обох напрямах, аудіо-візуальна синхронізація і, що саме головне - редагування. Типові операції JPEG зараз підтримуються на апаратному рівні більшістю відеокарт і даний формат дозволяє легко оперувати великими об'ємами даних при монтажі фільмів. Незалежне стиснення окремих кадрів дозволяє накладати різні ефекти, не боячись, що взаємний вплив сусідніх кадрів внесе додаткові спотворення у фільм.

Характеристики Motion-JPEG

Потік, роздільна здатність (стиснення): Потік і роздільна здатність довільні, стиснення в 5-10 разів

Плюси: Забезпечує швидкий довільний доступ. Легко редагувати потік. Низька вартість апаратної реалізації.

Мінуси: Порівняно низький ступінь стиснення.

MPEG-1

Алгоритм MPEG-1 в цілому відповідає описаній в попередній лекції загальній схемі побудови алгоритмів стиснення.

Характеристики MPEG-1

Потік, роздільна здатність: 1.5 Мбіт/с, 352x240x30, 352x288x25

Плюси: Порівняно простий в апаратній реалізації, містить перетворення, підтримувані на апаратному рівні великою кількістю відеокарт.

Мінуси: Невисокий ступінь стиснення. Мала гнучкість формату.

MPEG-2

MPEG-2 займається стисненням оцифрованого відео при потоці даних від 3 до 10 Мбіт/сек. Багато що в ньому запозичене з формату CCIR-601. CCIR-601 є стандартом цифрового відео з розміром передаваного зображення 720x486 при 60 напівкадрах в секунду. Рядки зображення передаються з чергуванням, і два напівкадри складають кадр. Цей прийом нерідко застосовують для зменшення мерехтіння. Хроматичні канали (U і V в YUV) передаються розміром 360x243 60 разів в секунду і також чергують вже між собою. Подібний розподіл називається 4:2:2. Переведення з CCIR-601 в MPEG-2 простий: треба поділити в 2 рази компоненту яскравості по горизонталі, поділити потік в 2 рази в тимчасовому вимірюванні (прибравши чергування), додати другу хроматичну компоненту і викинути "зайві" рядки, щоб розмір по вертикалі ділився на 16. Ми одержимо потік YUV кадрів розміром 352x240 з частотою 30 кадрів в секунду.

Проблеми починаються, коли з'являється можливість збільшити потік даних і підвищити якість зображення. Це не така проста задача, як здається. Проблема полягає в чергуванні напівкадрів у вхідному форматі. Тривіальне рішення - працювати з кадрами 720x486 при 30 кадрах в секунду, як із звичайним відео. Цей шлях приводить до неприємних ефектів при швидкому русі об'єктів на екрані. Між двома початковими напівкадрами 720x243 зсув стає помітним, а оскільки наш кадр формується з початкових напівкадрів через рядок, то при стисненні відбувається розмивання об'єкту, що рухається. Винне в цьому ефекті ДКП, і якось виправити ситуацію, не зменшивши ступеня стиснення відео, або не втративши у візуальній якості не можна. Достатньо поширеним є використання "деінтерлейсинга" (від англійського deinterlacing - видалення чергування рядків). Ця операція дозволяє видалити чергування рядків, зміщуючи парні рядки в одному напрямі, а непарні в іншому, пропорційно відносному руху об'єкту в даній області екрану. В результаті ми одержуємо візуально більш якісне зображення, але дещо більш тривалу передобробку перед стисненням.

Іншим рішенням є архівація парних і непарних кадрів в потоці незалежно. При цьому ми, звичайно, позбудемося артефактів, що виникають при швидкому русі об'єктів, але істотно зменшимо ступінь стиснення, оскільки не використовуватимемо найважливішої речі - надмірності між сусідніми кадрами, яка дуже велика.

Характеристики MPEG-2

Потік, роздільна здатність: 3-15 Мбіт/с, універсальний

Плюси: Підтримка достатньо серйозних звукових стандартів Dolby Digital, висока універсальність, порівняна простота апаратної реалізації.

Мінуси: Недостатній на сьогодні ступінь стиснення, недостатня гнучкість.

MPEG-4

MPEG-4 кардинально відрізняється від стандартів, що приймалися раніше. Розглянемо найцікавіші і корисні нововведення.

- **Розрахунок тривимірних сцен і робота з синтетичними об'єктами.** До складу декодера MPEG-4 як складова частина входить блок візуалізації тривимірних об'єктів (Animation Framework eXtension - AFX - те, що в просторіччі називають даними для тривимірного движка). Ті, хто кодував відео, знають, скільки проблем доставляють титри і взагалі будь-які об'єкти, що накладаються поверх фільму (логотипи, заставки і т.п.). Якщо добре виглядає основний план - будуть зіпсовані об'єкти, що накладаються, якщо добре виглядають вони - буде низьким загальний ступінь стиснення. В MPEG-4 пропонується розв'язати проблему кардинально. Об'єкти, що накладаються, розраховуються окремо і накладаються потім. Крім того, можна використовувати відеопотік навіть як текстуру, що накладається на поверхні об'єктів, що розраховуються. Така гнучка робота з тривимірними об'єктами дозволяє істотно підняти ступінь стиснення при помітно кращій якості зображення. Більш того - ніхто не заважає робити відеоролики взагалі без живого відео, а тільки з розрахованих (синтетичних) об'єктів. Розмір їх опису буде в рази менше ніж розмір аналогічних фільмів, стислих просто як потік кадрів. До речі, окремо в стандарті передбачена робота з "спрайтами" - статичними зображеннями, що накладаються на кадр. При цьому розмір спрайту може бути як зовсім маленький (логотип каналу в куточку екрану), так і перевищувати розмір кадру і "прокручуватися" (тобто як "спрайт" може бути заданий фон, а невеликі відео-об'єкти, наприклад, голову диктора, будуть на нього накладати). Це дає значну гнучкість при створенні MPEG-4 фільмів і дозволяє помітно зменшити об'єм кодованої інформації.
- **Об'єктно-орієнтована робота з потоком даних.** Тепер робота з потоком даних стає об'єктно-орієнтованою. При цьому дані можуть бути живим відео, звуковими даними, синтетичними об'єктами і т.д. З них створюються сцени, цими сценами можна управляти. Для простих смертних при цьому мало що зміниться, проте для програмістів об'єктне середовище означає кардинальне спрощення роботи з виникаючими складними структурами.
- **Додаванняя в потік двійкового коду "C++ подібної" мови BIFS.** За допомогою BIFS в потік додаються описи об'єктів, класів об'єктів і сцен. Також на ньому можна міняти координати, розміри, властивості, поведінку і реакцію об'єктів на дії користувача. Свого часу Flash був названий революцією 2D графіки в Інтернеті. Аналогічний прорив в області відео творив MPEG-4.

- **Активна глядацька позиція.** Як було помічено вище, BIFS дозволяє задавати реакцію об'єктів сцени на дії користувача. Потенційно можливе видалення, додавання або переміщення об'єктів, введення команд з клавіатури. Подієва модель запозичена з мови моделювання віртуальної реальності VRML, що розвивалася вже довгий час. Для тих, хто грав в написані на VRML ігри, очевидно, що в MPEG-4 абсолютно реально створювати "квест"-подібні (і не тільки) ігри. Широкий простір відкривається для створення навчальних і розважальних програм. Представляєте, викачуєте з Інтернету один файл, який відразу в собі містить все, що необхідне для невеликого курсу лекцій, причому ви можете прослуховувати його, бачучи голову викладача, що говорить, або відключивши його, можете збільшити фрагменти ("спрайт") з матеріалами. А в кінці - пройти короткий тест на розуміння предмету. До речі - в стандарті передбачена обробка команд на стороні серверу, тобто програма-переглядач може відіслати дані на сервер і одержати звідти оцінку. Відмінність від попередніх стандартів революційна.
- **Синтезатор обличь і фігур.** В стандарт закладений інтерфейс до модуля синтезу обличь і фігур. Наприклад, у файлі зберігаються ключові дані про профіль особи і текстуру особи, а при записі фільму зберігаються тільки коефіцієнти зміни форми. Для передач типу новин, цей прийом дозволяє в десятки разів скоротити розмір файлу при чудовій якості.
- **Синтезатор звуків і мови.** Крім синтезу обличь в стандарт MPEG-4 також закладені алгоритми синтезу звуків, і навіть мови(!).
- **Поліпшені алгоритми стиснення відео.** В стандарті передбачені блоки, що відповідають за потоки 4.8-65Кбіт/с з прогресивною розгорткою і великі потоки з підтримкою черезстрокової розгортки. Для передачі по ненадійних каналах можливе використовування перешкодостійких методів кодування (за рахунок незначного збільшення об'єму передаваних даних різко знижується ймовірність спотворення зображення). При передачі відео з одночасним переглядом закладена можливість огрубити зображення, якщо декодер через обмеження каналу зв'язку не встигає одержати всю інформацію. Всього в стандарт закладено 3 рівні деталізації. Ця можливість дозволить легко адаптувати алгоритм для трансляцій відео по мережі.
- **Підтримка профілів на рівні стандарту.** Зрозуміло, що реалізація всіх можливостей стандарту перетворює декодер на вельми складну і велику конструкцію. При цьому далеко не для всіх додатків необхідні якісь складні специфічні функції (наприклад, синтез мови). Творці стандарту поступили просто: вони визначили наборів профілів, кожний з яких включає набір обов'язкових функцій. Якщо у фільмі записано, що йому для програвання необхідний такий профіль і декодер цей профіль підтримує, то стандарт гарантує, що фільм буде програний правильно.

Вище стисло перераховані деякі відмінності MPEG-4 від попередніх стандартів. Треба відзначити, що на момент створення стандарту гострої потреби в описаних вище речах ще не було. Інакше кажучи, ми маємо справу з добре продуманою роботою по формуванню стандарту, яка була закінчена на той час, як в ньому виникла перша необхідність.

Творцями MPEG-4 врахований досвід попередників (зокрема VRML), коли дуже рання поява стандарту і відсутність в ньому механізму профілів серйозно підірвала його масове використання. Сподіватимемося, що масовому використанню MPEG-4 такі проблеми не загрожують.

Характеристики MPEG-4

Потік, роздільна здатність: 4,8 Кбіт/с - 20 Мбіт/с, підтримуються всі основні стандарти відеопотоків.

Плюси: Підтримка достатньо прогресивних звукових стандартів, високий ступінь універсальності, підтримка нових технологій (різні види синтезу звуку і зображення).

Мінуси: Висока складність реалізації.

2.2 Рекомендації ITU (Міжнародного союзу електрозв`язку) алгоритми групи H.XXX

H.261

Стандарт H.261 специфікує кодування і декодування відеопотоку для передачі по каналу p*64 Кбіт, де p=1..30. Як канал може виступати, наприклад, декілька телефонних ліній.

Вхідний формат зображення - дозволи CIF або QCIF у форматі YUV (CCIR 601) частота кадрів від 30 fps і нижче. Використовується зменшення дозволу в 2 рази для компонент кольоровості.

У вихідний потік записуються два типи кадрів: INTRA - стислі незалежно (відповідають I-кадрам) і INTER - стислі з посиланням на попередній кадр (відповідають P-кадрам). В кадрі, що передається не обов'язково присутні всі макроблоки зображення, якщо блок змінився трохи передавати його немає значення. Стиснення в INTRA кадрах здійснюється по схемі стиснення окремого зображення. В INTER кадрах проводиться аналогічне стиснення різниці кожного передаваного макроблоку з самим "схожим" макроблоком з попереднього кадру (компенсація руху). Для згладжування артефактів ДКП передбачена можливість вживання розмиття усередині кожного блоку 8x8 пікселів. Стандарт вимагає, щоб INTRA кадри зустрічалися в потоці не рідше ніж через кожні 132 INTER кадри (щоб не нагромаджувалася погрішність кодування і була можливість відновитися у разі помилки в потоці).

Ступінь стиснення залежить в основному від методу знаходження "схожих" макроблоків в попередньому кадрі, чи алгоритму рішення передавати конкретний макроблок, вибору способу кодування кожного макроблоку (INTER/INTRA) і

вибору коефіцієнтів квантування результатів ДКП. Жоден з перерахованих питання стандартом не регламентуються, залишаючи свободу для побудови власних оптимальних алгоритмів.

Характеристики H.261

Потік, роздільна здатність: p*64 Кбіт, p=1..30

Плюси: Простий в апаратній реалізації.

Мінуси: Невисокий ступінь стиснення. Обмеження на формат.

H.263

Даний стандарт є розширенням, доповненням і значним ускладненням H.261. Він містить "базовий" стандарт кодування, практично не відмінний по алгоритмах стиснення від H.261, плюс безліч опциональных його розширень. Стисло перерахуємо найважливіші відмінності:

1. **Використання арифметичного кодування** замість кодів Хаффмана. Дає можливість на 5-10% підвищити ступінь стиснення.
2. **Можливість задання векторів зсуву, вказуючих за межі зображення.** При цьому граничні пікселі використовуються для прогнозу пікселів зовні зображення. Даний прийом ускладнює алгоритм декодування, але дозволяє значно поліпшити зображення при різкій зміні плану сцени.
3. **Можливість задання вектора зсуву для кожного блоку 8x8** в макроблоку, що у ряді випадків істотно збільшує стиснення і знижує блоковість зображення.
4. **Поява** B-кадров, яка дозволяє збільшити ступінь стиснення, за рахунок ускладнення і збільшення часу роботи декодера.
5. **Підтримка великого числа форматів вхідних відеоданих:** CIF, sub-QCIF, QCIF, 4CIF, 16CIF і окремо що настроюються. Основна відмінність від більш універсальних форматів полягає в адаптації для декількох фіксованих дозволів, що дозволяє робити менш універсальні, але більш швидкі процедури обробки кадрів. Побудований таким чином декодер працює дещо швидше.
6. **Компенсація руху з субпіксельною точністю.** Можливість зсунути блок на полпиксела також збільшує ступінь стиснення, але збільшує час роботи декодера.
7. **Особливий режим стиснення INTRA макроблоків з посиланням на сусідні макроблоки** в оброблюваному кадрі, особливий режим квантування і спеціальна таблиця Хаффмана для поліпшення стиснення I-кадров у ряді випадків.
8. **Згладжування меж блоків декодованого зображення** для зменшення ефекту "блоковості". Часто при різкому русі в кадрі при стисненні алгоритм виявляється вимушений підвищити ступінь квантування блоків після ДКП щоб укластися у відведений на передачу бітовий потік. При цьому в кадрі виникають добре вам знайомі по JPEG блоки розміром 8x8. Як показала практика, "зрощення"

меж, коли крайні пикселы блоків зсовують по яскравості так, щоб зменшити різницю, дозволяє часто помітно підвищити візуальну якість фільму.

9. **Зміна дозволу і деформація базового кадру, що** використовується як базове при стисненні.

10. **Різні режими квантування і кодування по Хаффману.**

Характеристики H.263

Потік, роздільна здатність: 0.04-20 Мбіт/с, sub-QCIF, QCIF, CIF, 4CIF, 16CIF і дозволи, що окремо настроюються.

Плюси: Алгоритм H.263 також як H.261 допускає швидку апаратну реалізацію, проте при цьому дозволяє добитися більшого ступеня стиснення при тій же якості. Підтримує стиснення звуку.

Мінуси: По кількості закладених ідей знаходиться між MPEG-2 і MPEG-4.

H.264

H.264/MPEG-4 AVC — міжнародний стандарт відеокомпресії. Був розроблений групою фахівців організації ITU під назвою H.26L. У 2001 році група ITU об'єдналась з MPEG-Visual та розробку стандарту було продовжено в команді Joint Video Team (JVT). Метою проекту була розробка методів стиснення відео-даних, що гарантували б, у порівнянні з існуючими, щонайменше вдвічі менший бітрейт зі збереженням рівня якості як для мобільних приладів, так і для телебачення. У 2003 році обидві організації затвердили стандарт. Назва стандарту ITU: H.264. Стандарт ISO/IEC має назву MPEG-4/AVC (*Advanced Video Coding*) і є десятою частиною стандарту MPEG-4 (*MPEG-4/Part 10, ISO/IEC 14496-10*).

У порівнянні з MPEG-2 (DVD-Video), стиснення H.264 істотно ефективніше, що забезпечує кращу якість зображення і менший обсяг файлу. H.264 прийнятий як стандарт для стиснення відео високої чіткості (HD, HDTV), розповсюджуваного на оптичних носіях нового покоління Blu-ray HD DVD, використовується в мобільних пристроях, підтримується в Apple QuickTime, поширюється в ситемах цифрового телемовлення, відеоконференцзв`язку та відеоспостереження.

H.264 є гібридним стандартом блокового кодування відеоданих з використанням компенсації руху. Компенсація ґрунтується на використанні векторів переміщення областей кадру для прогнозування змін у зображенні. Оскільки для відео зображень характерний високий ступінь кореляції між двома послідовними кадрами, то використаємо це для кодування не зображення в цілому, а лише векторів переміщення різних частин зображення. Кодується при цьому прогнозована різниця між поточним кадром і його областями, які присутні на інших кадрах у зміщеному щодо оригінального положенні. Ця техніка називається «проміжне передбачення».

Станадартом H.264 передбачається розбиття зображення на макроблоки розміром до 16x16 пікселів кожний. Макроблоки поєднуються в групи (одну чи декілька). Отже окреме зображення може бути закодоване як одна або кілька груп.

Використання групування макроблоків дозволяє застосовувати різні методи корекції помилок, різні типи кодування макроблоків, а також такі інструменти як роздільне кодування напівкадрів. У кольорових відео зображеннях кодування яркісної складової відбувається окремо від колірної. З огляду на особливості людського зору, при цьому, як правило, використовується дискретизація колірного сигналу відносно яскравого.

Кодер отримує дані для прогнозування макроблоків на основі попередньо-закодованих даних, або на основі поточного кадру (*інтрапрогнозування*) або на основі інших кадрів, які вже було закодовано і передано (*інтерпрогнозування*). Кодер виділяє прогнозовану інформацію із поточного макроблоку і утворює залишок (корисну різницю). Пошук підхожого інтерпрогнозування зазвичай описують як оцінку руху, а виділення інтерпрогнозування із поточного макроблоку як компенсацію руху.

Характеристики H.264

Потік, роздільна здатність: 4,8 Кбіт/с - 20 Мбіт/с, підтримуються всі основні стандарти відеопотоків.

Плюси: Алгоритм H.264 був цілком успішним проектом. Це дуже гнучкий кодек, який набув широкого затосування в мережах розповсюдження потокового відео, на супутникових платформах, а також під час запису Blu-ray дисків. Він дуже хороший для масштабування, завдяки чому він був запропонований як стандарт для 3D з частотою кадрів 48-60 в секунду.

Мінуси: Проблема кодека H.264 полягає в тому, що будучи в принципі здатним кодувати відео в цих форматах, він не може забезпечити ступінь стиснення, який би зміг суттєво зменшити розміри одержуваних після стиснення файлів і тим самим заслужив би міжнародне визнання як засіб просування нових форматів відео. Це врахував наступний алгоритм H.265, який був розроблений таким чином, щоб використовувати нові технології стиснення і більш розумну модель кодування / декодування, найбільш економно використовувати пропускні ресурси каналу.

H.265

H.265 або **HEVC** (англ. *High Efficiency Video Coding — Високоефективне кодування відео*) — сучасний стандарт стиснення відео, який розглядається як наступник нині популярного H.264, затверджений Міжнародним союзом електрозв'язку (*MCE*) в кінці січня 2013 року. H.265 розробляється спільно двома групами експертів — ISO/IEC Moving Picture Experts Group (*MPEG*) та ITU-T Video Coding Experts Group (*VCEG*) як стандарти ISO/IEC 23008-2 *MPEG-H Part 2* та ITU-T *H.HEVC*. MPEG та VCEG створили спільну групу, *Joint Collaborative Team on Video Coding* (JCT-VC; укр. *Об'єднана група з відеокодування*), для розробки HEVC.[2][3] Як стверджується, новий стандарт дозволить покращити якість відео та зменшити вдвічі бітрейт при тій же якості для H.264/MPEG-4

AVC. Затверджений MCE кодек підтримує роздільну здатність 7680x4320 (8K), як зазначається, для перегляду відео, закодованого в HEVC, мережеве з'єднання має бути як мінімум 20-30 Мбіт/сек.

Чорновий варіант H.265 був запропонований експертами в кінці 2012 року. Ряд компаній, що виробляють побутову електроніку на початок 2013 року вже мали продукти, які підтримують HEVC.

- Як вимоги до стандарту запропоновано багато нових можливостей:
- Компенсація руху з точністю до 1/8-пікселя
- Адаптивне передбачення помилок кодування в просторової і частотної областях
- Адаптивний вибір матриці квантування
- Заснована на порівнянні схема вибору і кодування вектора руху
- Режим-залежна зміна налаштування внутрішньокадрового кодування

Технологія HEVC дозволяє постачальникам відео передавати високоякісні відеоматеріали з меншим навантаженням на мережу. Відзначимо основні функціональні нововведення, застосовані в H.265:

- **Особливі можливості для довільного доступу і зрощування цифрових потоків.** У H.264/MPEG-4 AVC цифровий потік повинен завжди починатися з блоку адресації IDR, а в HEVC підтримується довільний доступ.
- **Зображення розбивається на одиниці дерева кодування** (CTU), кожна з яких містить блоки дерева кодування (CTB) яскравості і кольоровості. У всіх колишніх стандартах кодування відео використовувався фіксований розмір масиву для вибору яскравості – 16 x 16. HEVC підтримує блоки CTB різного розміру, який вибирається в залежності від потреб кодувальника з точки зору пам`яті і обчислювальної потужності.
- **Кожен блок кодування** (CB) **може бути рекурсивно розділений на блоки перетворення** (TB). На відміну від колишніх стандартів в HEVC один блок TB може охоплювати кілька блоків передбачення (PB) для перехресних одиниць кодування (CU).
- **Направлене передбачення з 33 різними напрямками орієнтації для блоків перетворення** (TB) розміром від 4 x 4 до 32 x 32. Можливий напрямок передбачення – все 360 градусів. HEVC підтримує різні методики кодування передбачення інтракадрів.

У довгострокові перспективі H.265, швидше за все, замінить H.264 в якості головного рішення для розширеної обробки відео. Втім все буде залежати ще і від того, наскільки сильніше буде розряджати батареї процес обробки відео H.265 в порівнянні з H.264. Паралельна модель H.265 кодування, безсумнівно, повинна добре показати себе на фоні багатоядерних пристроїв майбутнього.

Характеристики H.265

Потік, роздільна здатність: підтримка відеопотоку до 300 кадрів в секунду, висока роздільна здатність – 7680 x 4320.

Плюси: Кодек H.265 дуже підходить для IP-відеонагляду, оскільки економно використовує пропускну здатність каналів зв'язку. Алгоритми стиснення задіюють більше ресурсів, але вони є більш ефективними. Коли використовується кодек H.265, є можливість підтримки відеопотоку до 300 кадрів в секунду. Вдалося зменшити розмір переданих і збережених файлів на 40-60% у порівнянні з кодеком H.264. Навантаження на мережу знижується, а якість відео – висока. H.265 передбачає роботу з відео, яке має високу роздільну здатність.

Більш висока чіткість зображення, менші вимоги до пропускної здатності каналів, економія місця при зберіганні відео, стимулювання розвитку більш високих стандартів у відео нагляді, не потрібно кардинально переробляти існуючу інфраструктуру.

Мінуси: H.265/HEVC накладає виключно високі вимоги до обчислювальних потужностей і клієнтських пристроїв, а також до внутрішніх серверів транскодування. Вимоги до обчислювальної потужності процесора зростають однозначно. Декодування відео H.265 вимагає в 10 разів більше ресурсів процесора, ніж декодування H.264.

Фактична продуктивність, як і раніше, далека від необхідної в реальному середовищі. Відсутність паралельної схеми. Неефективне налаштування векторизації.

2. Шляхи підвищення ступеня стиснення відеоданих

2.1 Використання векторів зсувів блоків

Найпростіший спосіб враховувати подібність сусідніх кадрів - це віднімати кожний блок кадру, що стискається, з відповідного блоку попереднього. Проте більш гнучким є алгоритм пошуку векторів, на які зсунулися блоки поточного кадру по відношенню до попереднього. Для кожного блоку в зображенні ми знаходимо блок близький по деякій метриці (наприклад, по сумі квадратів різниці пікселів) в попередньому кадрі в деякій області поточного положення блоку. Якщо мінімальна відстань по вибраній метриці з блоками в попередньому кадрі більше вибраного порогу - блок стискається незалежно (рис. 18.1).

Таким чином, разом з кожним блоком в потік тепер зберігаються координати зсуву максимально схожого блоку в попередньому I- або P-кадрі, або ознака того, що дані стислі незалежно. Ці координати задають вектор зсуву блоку (motion vector). В ситуаціях, коли камера наїжджає на об'єкт або дає панораму, використовування векторів зсувів блоків дозволяє значно зменшити амплітуду різниці кадрів, і як наслідок - значно підняти ступінь стиснення.

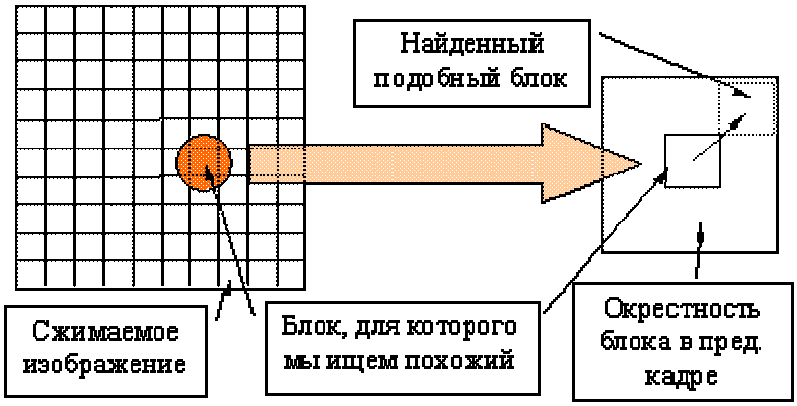

Рисунок 18.1 – Алгоритм пошуку векторів

Якщо ми проаналізуємо реальні фільми, то виявиться, що часто блок зсовується не на кратне число пікселів, а, наприклад, на 10.4 пікселя (камера швидко рухається управо, план зйомки зсовується рівномірно і проходить повний кадр розміром 352x240 за 1.35 секунди). При цьому виявляється, що для підвищення ступеня стиснення вигідно будувати 4 області пошуку векторів зсувів: початкову, зсунуту на півпікселя по горизонталі, зсунуту на півпікселя по вертикалі і зсунуту на півпікселя по горизонталі і по вертикалі (по діагоналі), які будуються за допомогою достатньо швидких алгоритмів білінійної або шматково-лінійної апроксимації. Цей прийом також дозволяє зменшити різницю між блоками і підвищити ступінь стиснення при мінімальній додатковій інформації, яку треба зберігати у файл (плюс 2 біти на кожний блок). Правда будувати апроксимовані блоки доведеться і при декомпресії, проте це порівняно дешева за часом операція, яка вельми трохи збільшує загальний час декомпресії.

Також треба розуміти, що алгоритм пошуку оптимальних векторів зсуву полягає, взагалі кажучи, в переборі. Існують різні методи зменшення цього перебору, і настройки відео-кодеків, регулюючі швидкість стиснення нерідко варіюють саме параметри методу перебору.

2.2 Можливості по розпаралелюванню

Навіть один погляд на цей узагальнений алгоритм дозволяє помітити, що він порівняно легко розпаралелюється. Зображення 320x288 містить 330 макроблоків, які можна кодувати і декодувати незалежно. Кожний макроблок, у свою чергу, містить шість блоків даних для ДКП. Розпаралелювати ДКП дуже важливо, оскільки, не рахуючи пошуку векторів зсуву, це найповільніша операція. Помітимо також, що решта перетворень легко конвейєризується. В результаті ми одержуємо **паралельно-конвейєрну** схему обробки потоку відеоданих.

Достатньо заманливо виглядає можливість розпаралелювати обробку різних кадрів, але тут ми стикаємося з складнощами. Як правило, компресор будується так, щоб після стиснення, зображення піддавалося зворотним перетворенням.

Таким чином, ми одержуємо кадр з втратами і архівуємо решту кадрів, відштовхуючись від нього. Це дозволяє не накопичувати помилки, отримані ще при квантуванні. Таким чином, якщо на екрані між кадрами спостерігалися великі зміни, і якість зображення довелося знизити, то при стабілізації зображення якість швидко підвищується практично до якості початкового відеоряду. Неприємний ефект, породжуваний цим прийомом, полягає в тому, що з'являється мерехтіння окремих точок (або областей) зображення, значення кольору в яких округляється то у більшу, то у меншу сторону.

При розпаковуванні наші можливості по паралельній обробці різних кадрів достатньо обмежені, оскільки велика залежність між кадрами в потоці (великий відсоток P- і B-кадров).

2.3 Інші шляхи підвищення ступеня стиснення

Описаний вище алгоритм в цілому близький більшості алгоритмам стиснення відео, що використовуються зараз на практиці. Проте нові (або добре забуті старі) ідеї з'являються щорічно. Якщо для алгоритмів стиснення без втрат можна говорити про зростання ступеня стиснення на 1% в рік (щодо попереднього року) для достатньо великого тестового масиву даних, то для алгоритмів стиснення відео звичайно йдеться про 3-5% надбавки ступеня стиснення для достатньо великого відеофрагменту при тій же візуальній якості.

Проте старе добре "правило важеля", винайдене ще Архімедом, дотримується і тут. І якщо з одного боку підвищується ступінь стиснення, то з другого боку доводиться пройти, прикладаючи ту ж силу, набагато більший шлях. В даному випадку росте складність програми і зменьшується швидкість роботи, як при компресії, так і при декомпресії.

Перерахуємо основні шляхи підвищення ступеня стиснення:

- **Зміна алгоритму стиснення I-кадров.** Вище приведений алгоритм, заснований на ДКП. Сьогодні все частіше використовуються алгоритми, засновані на вейвлєтах.
- **Зміна алгоритму стиснення без втрат.** Вище приведений алгоритм, що використовує стиснення по алгоритму Хаффмана. Проте недавно закінчився основний патент на арифметичне стиснення (що дає перевагу 2-15%) і, відповідно, все частіше використовується саме воно.
- **Зміна алгоритму роботи з векторами зсуву блоків.** Підбір векторів зсувів блоків по найменшому середньоквадратичному зсуву не є оптимальним. Існують алгоритми, що дають кращий результат при деяких додаткових витратах часу при стисненні.
- **Використання обробки коефіцієнтів.** Можна намагатися одержати більше інформації про зображення із збережених коефіцієнтів. Наприклад, можливе швидке порівняння коефіцієнтів після ДКП в сусідніх блоках і їх усереднювання по достатньо складним алгоритмам. Цей прийом помітно знижує

кількість артефактів, які вносяться в зображення ДКП, при цьому допускаючи реалізацію, що працює в реальному часі.

- **Використання обробки отриманих кадрів.** Відома біда алгоритмів стиснення зображень - неминуча "блоковість" (добре помітні межі макроблоків). У принципі існують алгоритми, що працюють з кадрами зовсім без використання блоків (навіть без векторів зсуву блоків), але такий підхід поки не виправдовує себе ні по ступеню стиснення, ні по швидкості роботи декодера. Проте можна побудувати достатньо швидкі алгоритми наступної обробки, які достатньо акуратно приберуть видимі межі між блоками, не вносячи істотних перешкод в саме зображення. Існують також алгоритми, що прибирають на льоту ефект Гіббса (*поява шуму (ореола) навколо кордонів з різким переходом кольору*) і т.п. Таким чином, істотно поліпшується візуальна якість зображення. Це також означає, що можна підвищити ступінь стиснення при тій же візуальній якості.
- **Поліпшення алгоритмів масштабування зображень.** Як правило, відео на комп'ютері проглядають у весь екран. При цьому навіть використанні дуже простого і швидкого кусково-лінійного масштабування здатне кардинально понизити швидкість програвання ролика. Тобто примітивна операція масштабування зображення працюватиме помітно довше, ніж складний алгоритм декодера. Проте швидкості сучасних комп'ютерів швидко ростуть і алгоритми використовують розширений набір команд. Тобто з'являється можливість використовувати складніші і якісні алгоритми масштабування на весь екран, одержуючи більш високу якість, ніж для білінійної інтерполяції, що використалася раніше (в більшості відеокарт реалізованої апаратний).
- **Використання попередньої обробки відео.** Якщо ми хочемо одержати достатньо високий ступінь стиснення, то можна наперед передбачити, що в нашому зображенні постраждають високі частоти. Воно стане згладженим, пропадуть багато дрібних деталей. При цьому, як правило, з'являються додаткові артефакти у вигляді смужок, хвиль, ореолів у різких границях. Значно підвищити якість зображення після кодування дозволяє попередня обробка з видаленням високих частот. При цьому існують алгоритми, які оброблюють потік таким чином, що візуально якість зображення не змінюється, проте після декодера ми одержуємо істотно більш якісне зображення.

Вище перераховані лише окремі напрями робіт. Фактично за 90-і роки зміни і поліпшення торкнулися всіх модулів алгоритму, побудованого за класичною схемою. Свій внесок в цей процес вносять також виробники мікропроцесорів і особливо Intel. Процесори, починаючи з P4, спеціально призначені для обробки потокових даних. Особливості архітектури процесора (зокрема невеликий в порівнянні з процесорами AMD кеш першого рівня), дають значну перевагу одним алгоритмам і роблять неефективними інші. Проте загальне вдосконалення алгоритмів стиснення відео йде дуже швидко і найближчим часом можна чекати тільки збільшення швидкості появи нових розробок.

Контрольні питання

1. Приведіть приклади програмно-апаратної реалізації алгоритмів стиснення відео (повсякденні і достатньо нові).
2. Приведіть приклади областей використовування відео не критичних до вимоги "стійкості до помилок".
3. Вкажіть алгоритми, що відносяться до стиснення відео з втратами.
4. Для якого стандарту є характерним розрахунок тривимірних сцен і робота з синтетичними об`єктами?
5. Для якого стандарту є характерною можливість завдання векторів зсуву, вказуючи за межі зображення?
6. Для якого стандарту є характерною можливість компенсації руху з субпіксельною точністю?
7. Для якого стандарту є характерною компенсація руху, що ґрунтується на використанні векторів переміщення областей кадру для прогнозування змін у зображенні?
8. Для якого стандарту є характерною можливість компенсації руху з точністю до 1/8 пікселя та ефективне використання пропускної здатності каналів зв’язку?

гармонік доводиться відкидати. При дуже низьких бітрейтах (16-24 кілобита / с) музику складно сприймати, а голос, хоча і залишається розбірливим, набуває досить "психоделічного" забарвлення.

Змінний бітрейт дозволяє зменшити розмір файлу при такій же якості за рахунок усунення надмірності. Інакше кажучи, немає необхідності кодувати тишу з бітрейтом 256 Кбіт/с, якщо її з точно таким же якістю можна відтворити з бітрейтом 64 Кбіт/с.

Контрольні питання

1. Назвіть параметри звукового сигналу.
2. Як називається середня кількість звукової енергії, що проходить в одиницю часу через одиницю поверхні?
3. Як називається мінімальний звуковий тиск, при якому ще існує слухове відчуття?
4. Як називається одиниця суб’єктивної висоти тону?
5. Як називається підвищення порогу чутності одного тону (або сигналу) при одночасному впливі іншого тону (шуму або сигналу) ?
6. Як називається періодичний вимір аналогового сигналу і використання отриманих миттєвих значень замість хвилі?
7. Як називається отримання гранично точних миттєвих значень аналового сигналу і подальшого їх округлення?
8. Охарактеризуйте основні види алгоритмів стиснення аудіосигналів.
9. Як називається обсяг інформації в одиницю часу (величина потоку даних)?

ЛЕКЦІЯ 20

на тему: Загальний огляд алгоритмів стиснення аудіосигналів (частина 2)

У лекції дана характеристика алгоритмів стиснення аудіосигналів, розглянуті загальна схема аудіокодера MPEG, *проведено аналіз форматів та можливостей алгоритмів стиснення аудіосигналів.*

План:

1. Характеристика алгоритмів стиснення аудіосигналів
2. Аналіз форматів стиснення звукових даних
3. Аналіз можливостей алгоритмів стиснення аудіосигналів

1. Характеристика алгоритмів стиснення аудіо сигналів

1.1. Диференціальна імпульсно-кодова модуляція

У звичайній імпульсно-кодової модуляції кожен відлік кодується незалежно від інших. Зміна амплітуди між послідовними відліками в середньому дещо мала. Отже, схема кодування, яка враховує надмірність відліків, вимагатиме більш низької бітової швидкості.

При диференціальній імпульсно-кодовій модуляції (ДІКМ) по каналу зв'язку передаються не значення відліків повідомлення, а різниця між відліками. Один з поширених способів ефективної передачі безперервних повідомлень - спосіб передачі з прогнозом.

Послідовність корельованих відліків вихідного сигналу подають на один з входів віднімає пристрою, на інший вхід надходить сигнал передбачення, сформований з попередніх відліків. Отриманий сигнал помилки передбачення надходить в тракт передачі. Оскільки в сигналі помилки якраз і містяться нові відомості, що становлять різницю між істинним і передбаченим значенням, то такий спосіб передачі називається передачею з прогнозом.

Відома велика кількість варіантів реалізації кодування з передбаченням. Основна відмінність зводиться до різниці операцій формування сигналу помилки: в одних системах сигнал помилки формується в аналоговій формі, а потім кодується, в інших - спочатку кодується вихідний аналоговий сигнал, а потім формується сигнал помилки.

У системах з ДІКМ зазвичай застосовують нерівномірне квантування сигналу помилки, так як найбільш вірогідні малі помилки. Оскільки потужність шуму квантування становить певну частку потужності квантуємого процесу, а потужність помилки передбачення істотно менше потужності повідомлення, шум квантування при ДІКМ менше, ніж при звичайній ІКМ при тому ж числі рівнів. ДІКМ забезпечує однакове з ІКМ якість передачі при меншій кількості символів в кодової комбінації. Кореляція між відліками зростає в міру скорочення інтервалу між ними. Тому в межі при великій частоті дискретизації

число рівнів квантування сигналу помилки можна зменшити до двох і перейти до одноразрядних систем.

Такий спосіб кодування називають дельта-модуляцією (ДМ). Дельта модуляція (ДМ), відрізняється від ДІКМ тим, що передається тільки знак сигналу помилки.

Поряд з рівномірним (лінійним) квантуванням, при якому крок квантування постійний в межах всього динамічного діапазону, існує можливість узгодження величини кроку квантування з амплітудою сигналу.

Для малих сигналів величина кроку квантування мала, а для великих - зростає. При цьому з ростом амплітуди сигналу зростає і потужність шумів квантування. Якщо використовувати логарифмічну залежність між вихідним і вхідним сигналами, то можна підтримувати співвідношення сигнал / шум квантування нижче заданої межі, як для малих, так і для великих за рівнем сигналів (рис. 20.1).

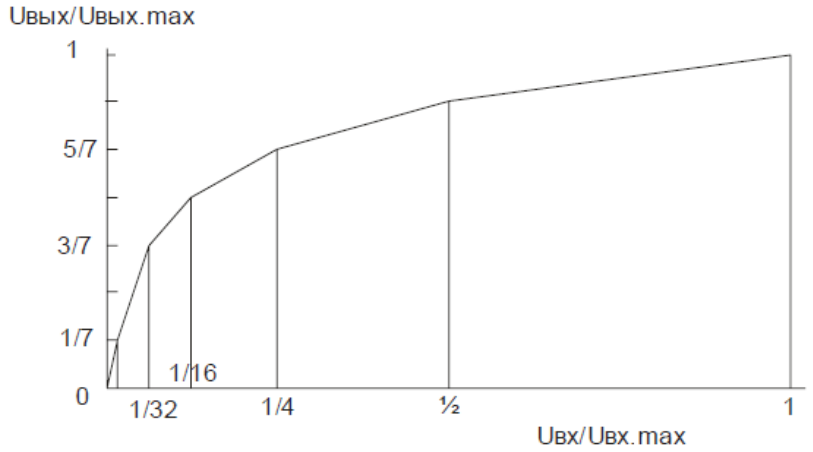

Рисунок 20.1 – Кусково-лінійна апроксимація при адаптивному квантуванні

При використанні такого методу використовується більше біт для слабких сигналів, де шум квантування найбільш помітний. Широко поширений формат, який використовує мю-функцію (μ - Law), часто характеризують як формат, що стискає 12-бітові відліки в 8-бітові.

Адаптивна диференціальна імпульсно-кодова модуляція

Багато реальних джерел, як уже було сказано вище, є квазістаціонарними за своєю природою. Одна з властивостей квазістаціонарності характеристик випадкового виходу джерела полягає в тому, що його дисперсія і автокореляційна функція повільно змінюються з часом. Кодери ІКМ і ДІКМ, проте, проектуються в припущенні, що вихід джерела стаціонарний. Ефективність і робочі характеристики таких кодерів можуть бути поліпшені, якщо вони будуть адаптуватися до повільного змінювання в часі зі статистикою джерела.

Одне поліпшення, яке зменшує динамічний діапазон шуму квантування, - це використання адаптивного квантувача. Інше - зробити адаптивним провісник в ДІКМ. При цьому коефіцієнти провидця можуть час від часу змінюватися, щоб відобразити мінливу статистику джерела сигналу. Визначені таким чином

коефіцієнти провісника можуть бути разом з помилкою квантування e ~ n передані приймачу, який використовує такий же провісник. На жаль, передача коефіцієнтів провидця призводить до збільшення необхідної бітової швидкості, частково компенсуючи зниження швидкості, досягнуте за допомогою квантувача з небагатьма бітами (небагатьма рівнями квантування) для зменшення динамічного діапазону помилки e_n, одержуваної при адаптивному прогнозі.

1.2 Кодування в частотних піддіапазонах

Одним з факторів, що знижують ефективність диференціального кодування, є частота. При кодуванні звуків низької частоти зазвичай виходить множина невеликих збільшень, в той час як при кодуванні звуків високої частоти - множина збільшень великої величини. Один із способів підвищення ефективності диференціального кодування полягає в розподілі частотного діапазону звуку на кілька частин або виділення частотних піддіапазонів і подальшої компресії кожного з них окремо.

Знання про те, як влаштований слух людини, допоможуть ще більш ефективно використовувати цей метод компресії. Так як людина в одних діапазонах частот чує краще, ніж в інших, можна використовувати різні параметри стиснення для різних піддіапазонів. Піддіапазони, розташовані поблизу області чутності, залишаються практично без змін, в той час як менш помітні піддіапазони піддаються більшому стиску, або взагалі не враховуються. Методи кодування піддіапазонами засновані на математичному апараті, що використовується для виділення піддіапазонів, і на ретельному вивченні слуху людини, необхідному для розробки рекомендацій по обробці кожного з діапазонів. Таке кодування дозволяє стиснути ІКМ звукові дані в 10-20 разів.

1.3 Аудіо MPEG-1

Стандарт MPEG-1 стиснення відеофайлів складається з двох основних частин: стиснення відео і стиснення звуку.

Специфікація для визначення MPEG-1 має нормативний і описовий розділи. Нормативний розділ містить специфікації стандарту: таблиці з різними параметрами і кодами Хаффмана, які використовуються в стандарті MPEG. Описовий розділ ілюструє вибрані концепції, пояснює причини вибору того чи іншого підходу, містить необхідні базові відомості, наприклад, алгоритм задає психоакустичну модель.

Основний принцип стиснення аудіо заснований на стисненні з втратами. Загальний принцип такий: у вхідному WAV файлі, як відомо, зберігається повна інформація про вхідний звук, оцифрованийу і проквантований з частотою 44 кГц. Саме ця інформація і зберігається на звичайних аудіо-CD. Відповідно до теореми Котельникова, цієї інформації абсолютно достатньо для відтворення всіх частот вихідного сигналу, менших половини частоти квантування. Тобто всі частоти до 22кГц включно відтворюються так само, як вони звучали при оцифруванні.

Для стиснення аудіоданих найвдаліше рішення було знайдено, розроблено та запатентовано вченими з німецького університету імені Фраунгофера. Формат файлів, (і стандарт) який вони розробили, носив назву MPEG Layer-3 (скорочено MP3). Завдяки тому, що ліцензія дозволяла необмежене і безкоштовне некомерційне використання формату, він набув широкого поширення і популярності і є домінуючим форматом стиснення.

При кодуванні в MP3 вхідний звуковий файл ріжеться на фрагменти, тривалістю по 50 мілісекунд, кожен з яких аналізується окремо. При аналізі фрагмент розкладається на гармоніки за методом Фур'є, з яких відповідно до теорії сприйняття звуку людським вухом викидаються ті гармоніки, які людина не сприймає або сприймає гірше на фоні інших. Крім того, викидаються звуки, замасковані внаслідок інертності слуху. Інформація про ті гармоніки, що залишилися після фільтрації і записується в MP3 файл, який в результаті виходить набагато менше за розміром, ніж вихідний WAV.

При відтворенні виконується зворотне перетворення, при якому гармоніки, що залишилися, знову перетворюються в звукову хвилю. Добутий звук не збігається з вхідним, але оскільки відкидалися малозначущі і не чутні звуки, відрізнити сигнал від вхідного для людського вуха досить складно.

Алгоритм стиснення MPEG, як і будь-який інший алгоритм стиснення, можна розділити на три етапи:

• Попередня обробка;

• основне перетворення;

• кодування і упаковка компонент перетворення.

На етапі попередньої обробки проводиться, в загальному випадку, підготовка вихідного потоку аудіоданих до виконання процедури основного перетворення. Зокрема, можна виділити два види такої підготовки: розбиття на блоки і фільтрація шумів.

На другому етапі за допомогою дискретного перетворення Фур'є (ДПФ) вхідні відліки ІКМ перетворюються в 512 спектральних складових. Таким чином здійснюється перехід від тимчасового уявлення сигналу до частотного.

На етапі кодування і упаковки компонент проводитися аналіз частотної області психоакустичною моделлю, яка відкидає нечутні компоненти спектра і обчислює крок квантування, при якому шум квантування буде не чути. Також на цьому етапі здійснюється саме квантування залишилися спектральних відліків і, далі, вони піддаються кодування за методом Хаффмана.

Алгоритм відновлення сигналу набагато простіше і складається з двох етапів:

• відновлення відліків;

• зворотне перетворення.

На етапі відновлення відліків відбувається декодування спектральних компонент. На другому етапі за допомогою зворотного ДПФ проводитися перехід до тимчасового поданням сигналу.

На рис. 20.2 приведені структурні схеми кодера і декодера MPEG.

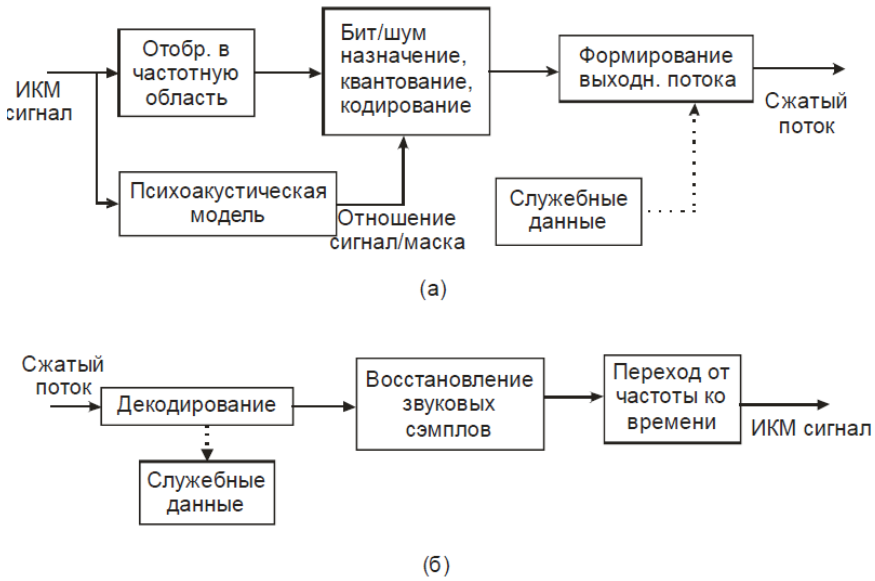

Рисунок 20.2 – Кодер (а) і декодер (б) MPEG.

Відображення в частотну область. Перший крок кодування аудіосигналу полягає в перетворенні його в частотну область. Для цього використовують дискретне перетворення Фур'є (ДПФ).

Так як аудіо дані можуть бути досить великі, то для обчислення спектру використовується "вікно" сканування. Після обчислення спектра "вікно" зсувається вправо.

Зрушення може проводитися як на один відлік сигналу, так і відразу на кілька. У кодеку MPEG зрушення "вікна" проводитися на 32 відліку, при розмірі "вікна" 512 або 1024. Весь спектр ділитися на 32 частотні смуги рівної ширини (частотні піддіапазони). На наступному етапі відбувається усунення надмірності за допомогою аналізу спектру сигналу в психоакустичній моделі для кожної з частотних смуг. Зауважимо, що подібні ідеї використовуються в сучасних стандартах при кодуванні частотних діапазонів.

Дискретне косинусное перетворення. У більшості сучасних стандартах стиснення звуку (в тому числі, в стандарті MPEG) застосовується дискретне косинусное перетворення (ДКП).

Психоакустична модель. Психоакустична модель дозволяє скоротити спектр сигналу без зміни сприйняття стисненого аудіо сигналу, так як вона враховує особливості людського слуху і відкидає нечутні частоти, які маскуються більш гучними звуками.

Психоакустична модель дає можливість кодеру визначити поріг допустимого шуму квантування для кожного поддіпазона. Ця інформація буде використана алгоритмом призначення бітів, що в поєднанні з кількістю наявних бітів задасть число рівнів квантування для кожної підполоси.

Стандарт стиснення звуку MPEG дозволяє значну свободу при реалізації моделей. Витонченість цієї реалізації в конкретному кодері залежить від необхідного ступеня стиснення. У додатках широкого споживання, в яких не вимагається високий фактор стиснення, психоакустична модель може бути

зовсім відсутньою. У цьому випадку алгоритм призначення бітів не використовує співвідношення сигнал / маскування (SMR - signal to mask ratio).

Основні кроки моделі полягають у наступному:

1. Спектральні значення частотних смуг поділяються на тональні (подібні синусоїді) і нетональні (шумоподібні) компоненти.

Отримані компоненти виключаються з вихідного спектра, що залишилися компоненти є нетональними.

2. Проріджувати спектр тональних компонент. Виключаються компоненти, що лежать нижче абсолютного порога чутності. Решта компоненти проріджуються за допомогою "вікна" шириною 0,5 Барк.

3. У кожному піддіапазоні обчислюється поріг маскування для тональних і не тональних компонент. Масковані компоненти відкидаються.

4. Обчислюється загальне ставлення сигнал / маскування (SMR) для кожної субполоси кодування.

Стиснення коефіцієнтів. На наступному етапі відбувається квантування і стиснення спектральних компонент. Кількість рівнів квантування вибираються на підставі співвідношення сигнал / маскування, отриманого на попередньому етапі.

Стиснення коефіцієнтів здійснюється методами стиснення без втрат, наприклад алгоритмом Хаффмана.

Декодер. Декодер MPEG простіше кодера і складається всього з двох етапів:

• відновлення відліків спектральної щільності;

• виконання зворотного перетворення Фур'є.

Відновлення відліків спектральної щільності відбувається з даних стислих алгоритмом без втрат, в основному використовується алгоритм Хаффмана.

2. Аналіз форматів стиснення звукових даних

Якість аудіотреків в MPEG-1 може варіюватися в дуже великих межах - від високоякісного до вкрай низького. Остаточно всі формати стиснення аудіо були стандартизовані в 1992 році європейської комісією по стандартам ISO. Залежно від використовуваного кодера і ступеня стиснення аудіоінформація відеоролика може бути представлена в наступному вигляді: моно, dual mono, стерео, інтенсивне стерео (стереосігнали, чиї частоти перевищують 2 кГц об'єднуються в моно), m / s стерео (один канал - сума сигналів, інший - різниця) і по частоті дискретизації можуть бути: 48, 44.1 і 32 кГц.

Серед існуючих форматів стиснення звукових даних з втратами можна відзначити четвірку найбільш популярних форматів - MP3, AAC, OGG Vorbis і WMA.

MP3 (MPEG Layer 3) – ліцензований формат файлу для зберігання звукової інформації. Це найпопулярніший формат стиснення звукових даних на сьогоднішній день. Формат MP3 заснований на маскуючому ефекті і призначений для значного зменшення обсягу даних, який необхідний для

прослуховування запису і збереження якості звуку максимально близького до вихідного. Такий файл із середнім бітрейтом (128 Кбіт/с) призводить до отримання файлу приблизно в 1/11 від початкового файлу, який має середній бітрейт близько 1411 Кбіт/с. Якість отриманого файлу залежить від бітрейту, з яким можуть створюватися mp3-файли. Найбільш прийнятним є бітрейт від 128-320 Кбіт/с, але для нормального якісного звучання достатньо лише 256 Кбіт/с.

Слід враховувати, що у форматі MP3 програми, що стискають звук, не є стандартизованими, тобто кожен грамотний програміст може реалізувати свою схему стиснення. А стандартам підкорюються тільки декодери, це призводить до того, що якість відтворення формату MP3 далеко не завжди залежить від плеєра – програвача файлу.

Особливістю MP3 є Variable Bit Rate – кодер змінює ступінь стиснення залежно від характеру звуку. Такий підхід призводить до зменшення підсумкового розміру файлу або, при збільшенні вимог до якості, при тому ж розмірі файлу дозволяє отримати краще звучання.

AAC (Advanced Audio Coding) - формат звукового файлу з меншою втратою якості при кодуванні, ніж MP3 при однакових розмірах. Формат також дозволяє стискати без втрати якості. Спочатку створювався як наступник MP3 з поліпшеною якістю кодування. Формат AAC, офіційно відомий як ISO / IEC 13818-7, вийшов у світ в 1997 як нова, сьома, частина родини MPEG-2. Існує також формат AAC, відомий як MPEG-4 Частина 3.

OGG Vorbis - вільний формат стиснення звуку, що вийшов влітку 2002 року. OGG Vorbis абсолютно безкоштовний, відкритий і вільний від патентів. OGG Vorbis володіє найбільш сучасною і якісною психоакустичною моделлю, через що співвідношення бітрейт/якість значно нижче, ніж у конкурентів. Як результат - якість звуку краще, а розмір файлу менше. При цьому OGG Vorbis підтримує до 255 окремих каналів з частотою дискретизації до 192 кГц і розрядністю до 32 біт (чого не дозволяє жоден інший формат стиснення звукових даних з втратами).

OGG Vorbis за замовчуванням використовує змінний бітрейт, при цьому його параметри не обмежені якимись жорсткими значеннями, і він може варіюватися навіть на 1 кбіт/с. Найвищий бітрейт чітко не обмежений, тому при максимальній якості кодування він може варіювати від 400 до 700 кбіт/с. Такою ж гнучкістю володіє частота дискретизації - користувачам надається можливість зробити будь-який вибір у межах від 2 до 192 кГц.

WMA (Windows Media Audio) - ліцензований формат файлу, розроблений компанією Microsoft для зберігання і трансляції звукової інформації. Номінально формат WMA характеризується хорошою здатністю стиснення, що дозволяє йому конкурувати за параметрами з форматами MP3, OGG Vorbis і AAC. Якість формату WMA не є однозначно еквівалетною, а перевага навіть перед MP3 однозначною, як це стверджується компанією Microsoft.

Характеристика, основні переваги і недоліки розглянутих популярних форматів стиснення звукових даних наведені в таблиці 20.1 та таблиці 20.2 відповідно.

Таблиця 20.1 – Характеристика основних форматів стиснення звукових даних

Формат	Квантування, біт	Частота дискретизації, кГц	Число каналів	Бітрейт, Кбіт/с	Ступінь стиснення
MP3	плаваючий	до 48	2	до 320	~11:1 з втратами
AAC	плаваючий	до 96	до 48	до 529	з втратами
OGG Vorbis	до 32	до 192	до 255	до 1000	з втратами
WMA	до 24	до 96	до 8	до 768	2:1, є версія без втрат

Таблиця 20.2 – Аналіз властивостей основних форматів стиснення звукових даних

Формат	Переваги	Недоліки
MP3	1. Заклав основу ідеї спрощення сигналу шляхом використання психоакустики, має кращі позиції за співвідношенням розмір/якість. 2. Величезна поширеність і популярність. 3. Доступність MP3 практично на всіх існуючих комп'ютерних платформах. 4. Використання змінного бітрейта, що дозволяє при меншому розмірі файлу отримати краще звучання. 5.Велика кількість кодерів.	1. Обмежений бітрейт (320 Кбіт/с), різке падіння якості при зниженні бітрейта. 2. Неможливість професійного застосування (кожне нове збереження розкодованого MP3-запису призводить до погіршення якості). 3. Відсутність у стандарті DRM-можливостей (управління авторськими правами). 4. Помітне падіння якості для сильно стиснутих музичних файлів. 5. Комерціалізованість (кожен виробник, який створює новий MP3-кодер, платить відрахування).
AAC	1. Має безліч доповнень, спрямованих на поліпшення якості вихідного звукового сигналу. 2. Дозволяє зберігати в закодованому звуковому сигналі інформацію про авторські права (watermarks). 3. Передбачає три профілю кодування (Main, LC і SSR), що впливають на час кодування і якість одержуваного цифрового потоку. 4. У порівнянні з MP3 помітно збільшена ефективність компресії, а якість звучання файлу при бітрейті 128 кбіт/с порівняна з якістю 192 кбіт/с MP3. 5. Дозволяє створювати багатоканальні файли. 6. При кодуванні на низьких бітрейтах є можливість створення файлів AAC HE (високої ефективності).	1. Технологія watermarks є перешкодою на шляху поширення файлів, створених за допомогою AAC. 2. Не є зворотньо сумісним (NBC - non backwards compatible) з рівнями MPEG-1 не дивлячись на те, що являє собою продовження MPEG-1 Layer I, II, III. 3. Велика кількість кодерів AAC несумісні один з одним на тлі тривалого процесу стандартизації. 4. Формат AAC підтримує порівняно невелика кількість програми для кодування звукових файлів.

Продовження таблиці 20.2

OGG Vorbis	1. Використовує оригінальний математичний алгоритм і власну психоакустичної модель. 2. Безкоштовний, відкритий і вільний від патентів. 2. Розрахований на стиснення даних на всіх бітрейтах без обмежень в режимі змінного бітрейта. Передбачена можливість зміни бітрейта потоку без його декодування. 3. Є можливість кодування декількох каналів аудіо (до 255), можливість редагування вмісту файлів. 4. Підтримується потокове відтворення (streaming) в Інтернеті. 5. Дає в середньому помітно кращі результати кодування на середніх і високих бітрейтах в порівнянні з MP3 (160 Кбіт/с і вище).	1. Для зберігання даних використовується власний універсальний формат. 2.Використання на низьких бітрейтах майже не виправдано. 3. Існує всього декілька кодерів OGG. 4. Після конвертації MP3 в OGG Vorbis до частин аудіосигналу, відкинутих MP3 кодеком, додаються частини, відкинуті кодеком OGG Vorbis, що призводить до погіршення якості. 5. Відмова від патентованих технологій не дозволяє кодеку OGG Vorbis домогтися надвисоких результатів.
WMA	1. Здатний давати більш високу якість звучання, ніж MP3 на порівняних бітрейтах, особливо, на низьких (нижче 128 Кбіт/с). 2. WMA-файли займають менше місця, ніж MP3, при порівнянній якості. 3. Наявність вбудованої DRM - цифрової системи управління авторськими правами. 4. Передбачено кодування без втрати якості, багатоканальне кодування об'ємного звуку і голосу. 5. Висока популярність серед користувачів Windows.	1. Помітно відстає реалізація апаратними програвачами, в порівнянні з MP3. 2. Вбудована DRM-система обмежує свободу користувача. 3. Кодер WMA всього один. 4. Формат WMA має велику кількість помилок різного роду.

3. Аналіз можливостей алгоритмів стиснення аудіо сигналів

Для того щоб грамотно вибрати мовний кодек, досить уявлення про використаний в ньому метод (на якому базується алгоритм кодування) і про процес узгодження сигналу, отриманого після цифрової обробки, з цифровим каналом зв'язку.

Оскільки аналізовані методи кодування є методами стиснення звуку з втратами, то при відновленні (декодуванні) звукового сигналу спостерігаються спотворення сигналу.

За ефективністю (тобто співвідношенням бітрейта до заданого якості) методи кодування можна розділити на три групи. До першої групи - низькоефективних кодеров - відносяться ІКМ і АДМ (адаптивна дельта-модуляція), які забезпечують задовільну якість звуку тільки при швидкостях передачі вище 24 кбіт / с. Кодери другої групи - ОПА, МПК, ЛПКВ, АДИКМ - дозволяють реалізувати задовільний і відмінну якість звучання при швидкостях 8-32 кбіт / с. У третю групу входять ЛПКВ- і ЛПМВ-кодери, що забезпечують відмінну якість при низьких швидкостях.

У більшості алгоритмів ефективного кодування мови ймовірність одноразової помилки становить 10.4-10.3 на символ. При залежності

завадостійкості від швидкості передачі кодери можна розділити на дві групи: в першу входять алгоритми з сильною залежністю, в другу - зі слабкою.

Під завадостійкістю мається на увазі максимальна ймовірність помилки, при якій якість звучання сигналу (відношення сигнал / шум) знижується не більше ніж на 10% (у випробуваннях за методом парних порівнянь відмінності між сигналами складають не більше 20%).

Необхідно відзначити, що більшість кодерів призначені для високоякісних цифрових каналів, але, незважаючи на це, допустима і максимально отримувана ймовірності помилки є стандартними характеристиками всіх ефективних кодерів.

Висновки. Розгляд існуючих кодеків дозволяє сформулювати ряд основних тенденцій:

1. Домінуюче становище при побудові низькошвидкісних кодеків мови методу кодування на основі лінійного передбачення.

2. Зростання частки адаптивних процедур обробки сигналів в сучасних системах кодування мови.

3. Однозначний зв'язок якості синтезованої мови на низьких швидкостях кодування зі ступенем адаптації відповідних кодеків мовних сигналів.

4. Найбільш перспективними є алгоритми стиснення типу лінійного передбачування і його варіацій, а також векторного кодування HVXC, покладені в основу методів MPEG, AAC і більшості сучасних стандартів.

5. Використання більш складних алгоритмів ентропійного кодування (контекстно-адаптивного арифметичного кодування).

Таким чином, аналіз ступеня адаптації сучасних кодеків вказує на перспективність переходу систем кодування мови до багатопараметричної адаптації в умовах апріорної і поточної невизначеності в описі моделей мовного сигналу і зовнішнього середовища функціонування кодека.

Найбільша компресія досягається в методах, які враховують особливості людського слуху, що використовують розбиття на піддіапазони і подальшого проведення в них аналізу. Деякі приклади таких методів наведені в табл. 20.3.

Контрольні питання

1. Охарактеризуйте алгоритми стиснення аудіосигналів.
2. Охарактеризуйте диференціальну імпульсно-кодова модуляцію.
3. Охарактеризуйте кодування в частотних під діапазонах.
4. Охарактеризуйте стандарт аудіо MPEG-1.
5. Проведіть аналіз форматів стиснення звукових даних.
6. Проведіть аналіз можливостей алгоритмів стиснення аудіосигналів.
7. Назвіть модель стиснення аудіо, що заснована на видаленні свідомо нечутних частот з більш ретельним збереженням звуків, які добре розрізняються людським вухом.
8. Назвіть модель стиснення аудіо, що використовує властивість слухового апарату людини визначати напрямок лише середніх частот.

Таблиця 20.3 – Властивості алгоритмів компресії аудіоданих

Наименование алгоритма компрессии аудиоданных	Метод компрессии	Скорость передачи, кбит/с на канал	Величина компрессии	Области применения
ASPEC	Кодирование с преобразованием	64-192	1:6	ISDN
ATRAC	Субполосное кодирование с преобразованием	256	1:5	MiniDisk
MUSCAM	Субполосное кодирование	128-256	1:4	DAB (Digital Audio Broadcasting)
MUSICAM	Субполосное кодирование	128-256	1:4	DAB
MPEG-1, Layer 1 и Layer 2	Субполосное кодирование (MUSICAM)	32-448 (Layer 1) 32-384 (Layer 2)	1:4 (Layer 1) 1:6 (Layer 2)	DAB (Layer 2, 128-256 кбит/с), DBS (Direct Broadcast Satellite, Layer 2, 224 кбит/с), DCC (Digital Compact Cassete, Layer 1, 384 кбит/с)
MPEG-1, Layer 3	Субполосное кодирование с преобразованием	32-320	1:9	Internet-вещание
MPEG-2	Субполосное кодирование/ субполосное кодирование с преобразованием	32-384	> 1:9	Многоканальное стереофоническое вещание
MPEG-2 AAC	Субполосное кодирование с преобразованием	16-384	1:15	Многоканальное стереофоническое вещание
MPEG-4	Субполосное кодирование с преобразованием/ параметрическое кодирование	2-64	-	Мультимедиа приложения
Dolby AC-3	Кодирование с преобразованием	32-384	1:13	Кинематограф, HDTV, спутниковое вещание
Гибридное	Субполосное кодирование с преобразованием/	32-64	1:(15-20)	Радиовещание, хранение информации

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Основна література

1. Бурачок Р. Телекомунікаційні системи передавання інформації. Методи кодування [Текст]: навч. посіб. / Р. Бурачок, М. Климаш, Б. Коваль; МОН України, НУ «Львівська політехніка». – Львів: Вид-во Львівської політехніки, 2015. – 476 с.

2. Ватолин Д., Ратушняк А., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. - М.: ДИАЛОГ-МИФИ, 2003. – 384 с.

Додаткова література

3. Фисенко В.Т., Фисенко Т.Ю., Компьютерная обработка и распознавание изображений: учеб. пособие. - СПб: СПбГУ ИТМО, 2008. – 192 с.

4. Арюшенко В., Шелухин О., Афонин М. Цифровое сжатие видео-информации и звука. – М.: Дашков и Ко, 2004. – 423 с.

5. Вернер М. Основы кодирования. Учебник для ВУЗов. – М.: Техносфера. –2004. – 288 с.

6. Вычислительные системы, сети, телекоммуникации /В.Л.Бройдо. – СПб.: Питер, 2002. – 688 с.

7. Ватолин Д. Media Data Compression Сжатие без потерь – 2003-120 слайдов.

8. Ватолин Д., Смирнов М., Ратушняк А., Юкин В. Методы сжатия данных // Диалог-МИФИ, 2002.

9. Дмитриев В.И. Прикладная теория информации. М.Высшая школа, 1989

10. Жураковський Ю. П., Полторак В. П. Теорія інформації та кодування. - К.: Вища школа, 2001. – 255 с.

11. Квєтний Р.Н. Основи техніки передавання інформації / М.М. Компанець, С.Г. Кривогубченко, А.Я. Кулик / Підручник. - Вінниця: Універсам, 2002. – 304 с.

12. Крапивенко А.В. Методы и средства обработки аудио- и видеоданных. - М.: Вузовская книга, 2010

13. Лидовский В. В. Теория информации: Учебное пособие. – М.: Компания Спутник+, 2004. – 111 с.

14. Миано Дж. Форматы и алгоритмы сжатия изображений в действии. – М.: Триумф, 2003. – 336 с.

15. Сорока Л.С. Основи теорії інформації / Л.С. Сорока / Навчальний посібник. - Х.: ХНУ ім. В.Н. Каразіна, 2007. – 264 с.

16. Сэломон Д. Сжатие данных, изображений и звука. – М.: Техносфера, 2004. – 368 с.

17. Смирнов М. А. Обзор применения методов безущербного сжатия в СУБД [Электронный ресурс] – Режим доступа: http://compression.ru/download/articles/db/smirnov_2003_database_compression_re

view/index.html.

18. Сжатие данных [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org/wiki/Сжатие данных.

19. Дударь З.В., Егоров С.В. Исследование и оптимизация методов сжатия текстовой информации / З. Дударь, С. Егоров. – Харьков : Вестник ХНТУ, 2012. – № 1(44).

20. Нечипоренко О.В., Миценко С.А. Классификационная схема моделей баз данных для лазерных технологических комплексов / О. Нечипоренко, С. Миценко. – Черкаси: Вісник ЧДТУ, 2013. – № 2.

21. Алгоритмы сжатия [Электронный ресурс] – Режим доступа: http://mf.grsu.by/UchProc/livak/en/po/comprsite/theory_contents.html.

22. Корпань Я. В. Методи та алгоритми компактного представлення графічної інформації в комп'ютерних системах / Я. В. Корпань // Технологічний аудит та резерви виробництва. – 2015. – № 3/2 (23). – С. 32–36.

23. Корпань Я. В. Аналіз методів та алгоритмів компресії-декомпресії цифрових відеоданих / Я. В. Корпань // Вісник Хмельницького національного університету. – 2015. – № 3. – С. 175–179.

24. Нечипоренко О. В. Дослідження ефективності методів стиснення звукових даних / О. В. Нечипоренко // Вісник Хмельницького національного університету. – 2015. – № 4. – С. 127–131.

25. Nechyporenko O. Analysis of methods and technologies of human face recognition / O. Nechyporenko, Y. Korpan // Technology audit and production reserves. 2017 - №5/2(37). s. 4-10.

26. Web parsing: задачи, проблемы, инструменты [Електронний ресурс]: «Inostudio». – Режим доступу: https://inostudio.com/ru/article/web-parsing.html – Дата доступу: березень 2018.

27. Data Mining [Електронний ресурс]: «UserGroup». – Режим доступу: http://msugvnua000.web710.discountasp.net/Posts/Details/3316 – Дата доступу: березень 2018.

28. An introduction to big data [Електронний ресурс]: «Opensource». – Режим доступу: https://opensource.com/resources/big-data – Дата доступу: березень 2018.

29. Big Data and Data Mining [Електронний ресурс]: «Habrahabr». – Режим доступу: https://habrahabr.ru/post/267827/– Дата доступу: березень 2018.

30. Data Mining [Електронний ресурс]: «Habrahabr». – Режим доступу: https://habrahabr.ru/post/95209/– Дата доступу: березень 2018.

31. Document Object Model [Електронний ресурс]: «W3C». – Режим доступу: https://www.w3.org/DOM/ – Дата доступу: березень 2018.

32. Подходы к извлечению данных с веб-ресурсов [Електронний ресурс]: «Habrahabr». – Режим доступу: https://habrahabr.ru/post/99918/ – Дата доступу: березень 2018.

33. Подходы к извлечению данных с веб-ресурсов [Електронний ресурс]: «Habrahabr». – Режим доступу: https://habrahabr.ru/company/dataart/blog/262817/ – Дата доступу: березень 2017. – Загол. з екрану.

34. Web Mining: интелектуальный анализ данных в сети Internet [Електронний ресурс]: «Google». – Режим доступу: https://sites.google.com/site/upravlenieznaniami/tehnologii-upravlenia-znaniami/text-mining-web-mining/web-mining – Дата доступу: березень 2018.

35. Datacol – универсальный парсер с визуализацией сбора данных [Електронний ресурс]: «Profithunter». – Режим доступу: https://www.profithunter.ru/obzory/datacol-universalnyiy-parser-s-vizualizatsiey-sbora-dannyih/– Дата доступу: березень 2018.

36. Datacol – парсер для сбора информации с сайтов [Електронний ресурс]: «Vlada-rykova». – Режим доступу: https://vlada-rykova.com/parser-sajtov/– Дата доступу: березень 2018.

37. Универсальный парсер контента – программа Content Downloader [Електронний ресурс]: «Seobook». – Режим доступу: http://seobook.info/universalnyy-parser-kontenta/ – Дата доступу: березень 2018.

38. Перелинковка сайта [Електронний ресурс]: «Pr-cy». – Режим доступу: http://pr-cy.ru/lib/saytostroenie/Perelinkovka-sayta-Kak-eto-sdelat – Дата доступу: березень 2017.