























**Описати один з алгоритмів**

**RLE, KWE, Хафмана, …** Лемпеля-Зіва, метод Шенона

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Характерною особливістю більшості типів даних є їх надлишковість. Ступінь надлишковості даних залежить від типу даних. Наприклад, для відеоданих ступінь надлишковості в декілька разів більша ніж для графічних даних, а ступінь надлишковості графічних даних, у свою чергу, більша за ступінь надлишковості текстових даних. Іншим фактором, що впливає на ступінь надлишковості є прийнята система кодування. Прикладом систем кодування можуть бути звичайні мови спілкування, які є ні чим іншим, як системами кодування понять та ідей для висловлення думок. Так, встановлено, що кодування текстових даних за допомогою засобів української мови дає в середньому надлишковість на 20-25% більшу ніж кодування аналогічних даних засобами англійської мови.  Для людини надлишковість даних часто пов'язана з якістю інформації, оскільки надлишковість, як правило, покращує зрозумілість та сприйняття інформації. Однак, коли мова йде про зберігання та передачу інформації засобами комп'ютерної техніки, то надлишковість відіграє негативну роль, оскільки вона приводить до зростання вартості зберігання та передачі інформації. Особливо актуальною є ця проблема у випадку необхідності обробки величезних обсягів інформації при незначних об'ємах носіїв даних. У зв'язку з цим постійно виникає проблема позбавлення надлишковості або стиснення даних. Коли методи стиснення даних застосовуються до готових файлів, то часто замість терміну "стиснення даних" вживають термін "архівування даних", стиснений варіант даних називають архівом, а програмні засоби, що реалізують методи стиснення називаються архіваторами.  В залежності від того, в якому об'єкті розміщені дані, що підлягають стисненню розрізняють:   1. **Стиснення (архівування) файлів**: використовується для зменшення розмірів файлів при підготовці їх до передавання каналами зв'язку або до транспортування на зовнішніх носіях малої ємності; 2. **Стиснення (архівування) папок**: використовується як засіб зменшення обсягу папок перед довготерміновим зберіганням, наприклад, при резервному копіюванні; 3. **Стиснення (ущільнення) дисків**: використовується для підвищення ефективності використання дискового простору шляхом стиснення даних при записі їх на носії інформації (як правило, засобами операційної системи).   Існує багато практичних алгоритмів стиснення даних, але всі вони базуються на трьох теоретичних способах зменшення надлишковості даних. Перший спосіб полягає в зміні вмісту даних, другий - у зміні структури даних, а третій - в одночасній зміні як структури, так і вмісту даних.  Якщо при стисненні даних відбувається зміна їх вмісту, то метод стиснення є незворотнім, тобто при відновленні (розархівуванні) даних з архіву не відбувається повне відновлення інформації. Такі методи часто називаються методами стиснення з регульованими втратами інформації. Зрозуміло, що ці методи можна застосовувати тільки для таких типів даних, для яких втрата частини вмісту не приводить до суттєвого спотворення інформації. До таких типів даних відносяться відео- та аудіодані, а також графічні дані. Методи стиснення з регульованими втратами інформації забезпечують значно більший ступінь стиснення, але їх не можна застосовувати до текстових даних. Прикладами форматів стиснення з втратами інформації можуть бути: JPEG (Joint Photographic Experts Group) для графічних даних; MPG - для для відеоданих; MP3 - для аудіоданих.  Якщо при стисненні даних відбувається тільки зміна структури даних, то метод стиснення є зворотнім. У цьому випадкові з архіву можна відновити інформацію повністю. Зворотні методи стиснення можна застосовувати до будь-яких типів даних, але вони дають менший ступінь стиснення у порівнянні з незворотними методами стиснення. Приклади форматів стиснення без втрати інформації: GIF (Graphics Interchange Format), TIFF (Tagged Image File Format) - для графічних даних; AVI - для відеоданих; ZIP, ARJ, RAR, CAB, LH - для довільних типів даних. Існує багато різних практичних методів стиснення без втрати інформації, які, як правило, мають різну ефективність для різних типів даних та різних обсягів. Однак, в основі цих методів лежать три теоретичних алгоритми:   * алгоритм RLE (Run Length Encoding); * алгоритми групи KWE(KeyWord Encoding); * алгоритм Хафмана.   **Алгоритм RLE**  В основі алгоритму RLE лежить ідея виявлення послідовностей даних, що повторюються, та заміни цих послідовностей більш простою структурою, в якій вказується код даних та коефіцієнт повторення. Наприклад, нехай задана така послідовність даних, що підлягає стисненню:  1 1 1 1 2 2 3 4 4 4  В алгоритмі RLE пропонується замінити її наступною структурою: 1 4 2 2 3 1 4 3, де перше число кожної пари чисел -це код даних, а друге - коефіцієнт повторення. Якщо для зберігання кожного елементу даних вхідної послідовності відводиться 1 байт, то вся послідовність займатиме 10 байт пам'яті, тоді як вихідна послідовність (стиснений варіант) займатиме 8 байт пам'яті. Коефіцієнт стиснення, що характеризує ступінь стиснення, можна обчислити за такою формулою:  де Vx- обсяг пам'яті, необхідної для зберігання вихідної (результуючої) послідовності даних, Vn- вхідної послідовності даних.  Чим менше значення коефіцієнта стиснення, тим ефективніший метод стиснення. Зрозуміло, що алгоритм RLE буде давати кращий ефект стиснення при більшій довжині послідовності даних, що повторюється. У випадкові розглянутого вище прикладу, якщо вхідна послідовність матиме такий вигляд: 1 1 1 1 1 1 3 4 4 4, то коефіцієнт стиснення буде рівний 60%. У зв'язку з цим найбільша ефективність алгоритму RLE досягається при стисненні графічних даних (особливо для однотонових фонових зображень).  **Алгоритми групи KWE**  В основі алгоритму стиснення за ключовими словами покладено принцип кодування лексичних одиниць групами байт фіксованої довжини. Прикладом лексичної одиниці може бути звичайне слово. На практиці, в ролі лексичних одиниць вибираються послідовності символів, що повторюються, які кодуються ланцюжком символів (кодом) меншої довжини. Результат кодування зводиться в таблицю, утворюючи так званий словник.  Існує досить багато реалізацій цього алгоритму, серед яких найбільш поширеними є алгоритм Лемпеля-Зіва (алгоритм LZ) та його модифікація алгоритм Лемпеля-Зіва-Велча (алгоритм LZW). Словником в даному алгоритмі є потенційно нескінченний список фраз. Алгоритм починає роботу з майже пустого словника, що містить тільки один закодований рядок, так званий NULL-рядок. Коли зчитується черговий символ вхідної послідовності даних, він додається до поточного рядка. Процес продовжується доти, поки поточний рядок відповідає якій-небудь фразі з словника. Але рано або пізно поточний рядок перестає відповідати якій-небудь фразі словника. У цей момент, коли поточний рядок являє собою останній збіг зі словником плюс щойно прочитаний символ повідомлення, кодер видає код, що складається з індексу збігу і наступного за ним символа, що порушив збіг рядків. Крім того, нова фраза, що складається з індексу збігу і наступного за ним снмвола, додається в словник. У наступний раз, коли ця фраза з'явиться в повідомленні, вона може бути використана для побудови більш довгої фрази, що підвищує міру стиснення інформації.  Алгоритм LZW побудований навколо таблиці фраз (словника), яка відображає рядки символів стиснуваного повідомлення в коди фіксованої довжини. Таблиця володіє так званою властивістю передування, тобто для кожної фрази словника, що складається з деякої фрази w і символа К фраза w також міститься в словнику. Якщо всі частинки словника повністю заповнені кодування перестає бути адаптивним (кодування відбувається виходячи з вже існуючих в словнику фраз).  Алгоритми стиснення цієї групи найефективніші для текстових даних великих обсягів і малоефективні для файлів малих розмірів (за рахунок необхідності зберігання словника).  **Алгоритм Хафмана**  В основі алгоритму Хафмана лежить ідея кодування бітовими групами. Спочатку проводиться частотний аналіз вхідної послідовності даних, тобто встановлюється частота входження кожного символу, що зустрічається у ній. Після цього символи сортуються по спаданню частоти входження.  Основна ідея полягає в наступному: чим частіше зустрічається символ, тим меншою кількістю біт він кодується. Результат кодування зводиться в словник, що необхідний для декодування.  Розглянемо простий приклад, що ілюструє роботу алгоритму Хафмана. Нехай задано текст, в якому літера 'А' входить 10 разів, літера 'B' - 8 раз, 'C'- 6 разів , 'D' - 5 разів, 'E' і 'F' - по 4 рази. Тоді один з можливих варіантів кодування за алгоритмом Хафмана наведений у таблиці 1.  Таблиця 1.   |  |  |  | | --- | --- | --- | | Символ | Частота входження | Бітовий код | | A | 10 | 00 | | B | 8 | 01 | | C | 6 | 100 | | D | 5 | 101 | | E | 4 | 110 | | F | 4 | 111 |   Як видно з таблиці 1, розмір вхідного тексту до стиснення рівний 37 байт, тоді як після стиснення - 93 біт, тобто майже 12 байт (без врахування довжини словника). Коефіцієнт стиснення рівний 32%. Алгоритм Хафмана універсальний, тобто його можна застосовувати для стиснення даних будь-яких типів, але він малоефективний для файлів малих розмірів (за рахунок необхідності зберігання словника).  На практиці програмні засоби стиснення даних синтезують ці три "чистих" алгоритми, оскільки їх ефективність залежить від типу та обсягу даних. У таблиці 2 наведені найпоширеніші формати стиснення та відповідні їм програми-архіватори, що використовуються на практиці.  Таблиця 2.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Формат стиснення | Операційна система MS DOS | | Операційна система Windows | | | Програма архівування | Програма розархівування | Програма архівування | Програма розархівування | | ARJ | Arj.exe | Arj.exe | WinArj.exe | WinArj.exe | | RAR | Rar.exe | Unrar.exe | WinRar.exe | WinRar.exe | | ZIP | Pkzip.exe | Pkunzip.exe | WinZip.exe | WinZip.exe |   Крім того, сучасні архіватори надають користувачеві повний спектр послуг для роботи з архівами, основними з яких є:   1. створення нового архіву; 2. додавання файлів в існуючий архів; 3. розпакування файлів з архіву; 4. створення архівів, що саморозпаковуються (self-extractor archive); 5. створення розподілених архівів фіксованих розмірів для носіїв малої ємності; 6. захист архівів паролями від несанкціонованого доступу; 7. перегляд вмісту файлів різних форматів без попереднього розархівування; 8. пошук файлів і даних всередині архіву; 9. перевірка на віруси в архіві до розпакування; 10. вибір та налаштування коефіцієнта стиснення.   **Контрольні запитання**   1. Які фактори впливають на ступінь надлишковості даних? 2. Що таке архів? Які програмні засоби називаються архіваторами? 3. Чому методи стиснення при яких відбувається зміна вмісту даних називаються незворотніми? 4. Наведіть приклади форматів стиснення з втратами інформації. 5. В чому полягає перевага зворотніх методів стиснення над незворотніми? А недолік? 6. Яка існує залежність між коефіцієнтом стиснення та ефективністю методу стиснення? 7. В чому полягає основна ідея алгоритму RLE? 8. В чому полягає основна ідея алгоритмів групи KWE? 9. В чому полягає основна ідея алгоритму Хафмана? 10. Які ви знаєте програми-архіватори? Коротко охарактеризуйте їх. |

**Необхідність і доцільність стиснення:**

• зменшення обсягу файлів;

• економія місця на носіях;

• швидкість обміну інформацією;

• резервне копіювання;

• архівація під час шифрування даних.

**Ступінь стискання залежить від:**

• методу кодування;

• програми-архіватора;

• типу файла, що архівується.