КОМП'ЮТЕРНІ СИСТЕМИ ТА КОМПОНЕНТИ

УДК 681.3.05

А. М. Пєтух, В. П. Майданюк, О. О. Ліщук

АНАЛІЗ АЛГОРИТМІВ УЩІЛЬНЕННЯ ДАНИХ ТА ЇХ ПРОГРАМНИХ РЕАЛІЗАЦІЙ

Вінницький національний технічний університет

Анотація. У статті розглянуто статистичні, словникові і арефметичні алгоритми ущільнення даних. З статистичних було виділено три основних класи: неадаптивні, напівадаптивні і адаптивні. Зі словникових виділено два основних алгоритма: LZ77 та LZ78, покращення яких породило багато нових методів. Проведено їх порівняльний аналіз, розглянута технічна сторона, принципи роботи та приведені приклади роботи розглянутих алгоритмів. Наведено список основних прикладних програм та розширень файлів які ефективно застосовують алгоритми ущільнення даних. Проаналізовано які алгоритми вони використовують та у якій сфері. Проведено аналіз та наведені основні методи вбудованих класів для стиснення даних.

Ключові слова: кодування, ущільнення.

Аннотация. В статье рассмотрены статистические, словарные и арефметични алгоритмы сжатия данных. С статистических было выделено три основных класса: неадаптивные, полуадаптивные и адаптивные. Из словарных выделено два основных алгоритма: LZ77 и LZ78, улучшение которых породило много новых методов. Проведен их сравнительный анализ, рассмотрена техническая сторона, принципы работы и приведены примеры работы рассмотренных алгоритмов. Приведен список основных приложений и расширений файлов эффективно применяющих алгоритмы сжатия данных. Проанализированы какие алгоритмы они используют и в какой сфере. Проведен анализ и приведены основные методы встроеных классов для сжатия данных.

Ключевые слова: кодирование, сжатие.

Abstract. In the article analyzed statistics, dictionary and data compaction algorithms arefmetychni. From the statistics were three main classes: neadaptyvni, napivadaptyvni and adaptive. With the dictionary highlighted two main algorithm: LZ77 and LZ78, which has generated much improved new methods. A comparative analysis reviewed the technical side, principles of operation and given examples of the considered algorithms. Are the main applications and file extensions that effectively seals algorithms used data. The analysis algorithms they use and in what area. The analysis and are the main methods vbudovananyh classes for data compression.

Keywords: encoding, compression.

Вступ

Ефективне кодування повідомлень для передачі їх по дискретному каналу без завад ґрунтується на теоремі Шеннона, яку можна сформулювати так [1]:

- повідомлення джерела з ентропією H(z) завжди можна закодувати послідовностями символів з об'ємом алфавіту m так, що середнє число символів на знак повідомлення $l_{ср}$ буде як завгодно близьким до величини $\frac{H(z)}{\log m}$, але не менше за неї. При m=2, $l_{ср} \geq H(z)$.

Теорема не вказує конкретного способу кодування, але з неї видно, що кожний символ кодової комбінації повинен нести максимальну інформацію, тобто кожний символ повинен приймати значення або 0, або 1 по можливості з рівними ймовірностями і кожний вибір повинен бути незалежним від значення попередніх символів.

Ця робота ініціювала багато наукових досліджень по всьому світу, що тривають донині, поклавши початок розвитку методів обробки, передачі та зберігання інформації.

Теорія економного або оптимального кодування (optimal coding) об'єднує в собі декілька різних напрямів. В рамках даної теорії методи прийнято розділяти на методи економного кодування інформації без втрат (lossless) і методи економного кодування інформації з втратами (lossy) [2-4]. Як випливає з назв, обробка інформації методами першої групи не веде до інформаційних втрат, тоді як використання методів другої групи пов'язане з такими втратами і зазвичай ці методи застосовуються для ущільнення зображень і аудіо. Розглянемо більш детально методи першої групи, оскільки вони є більш універсальними і застосовуються як самостійно так і як один з етапів ущільнення з втратами інформації.

Актуальність

Подальший розвиток інструментальних і прикладних програмних засобів тісно пов'язаний з питаннями економії пам'яті для їх зберігання і виконання, що робить вкрай актуальною задачу ущільнення даних без втрат.

Мета

Метою роботи є аналіз сучасного стану застосування методів ущільнення даних без втрат інформації в сучасних інформаційних системах.

Задачі

1. Класифікація методів ущільнення даних без втрат.
2. Аналіз пакетів прикладних програм та форматів файлів для ущільнення даних.
3. Вбудовані методи ущільнення інформації у мовах програмування.

4. Визначення напрямків розвитку ущільнення даних.

Класифікація методів ущільнення даних без втрат

Серед алгоритмів ущільнення без втрат дві схеми ущільнення - кодування Хаффмана (Huffman) і LZW-кодування (за початковими літерами прізвищ Лемпел (Lempel) і Зів (Ziv) (його авторів) і Уелч (Welch), який його суттєво модифікував), формують основу для багатьох систем ущільнення [2-5]. Ці схеми подають два різних підходи до ущільнення даних. Окремий клас утворюють методи, які відомі як арифметичне кодування. Таким чином алгоритми ущільнення даних без втрат можна розділити на такі групи (рис. 1):

- статистичні методи ущільнення;
- словникові (евристичні) методи ущільнення;
- арифметичне кодування.

Статистичні алгоритми (Шеннона-Фано, Хаффмана, кодування за ступенем новизни, імовірнісне ущільнення та ін.) потребують знання ймовірностей появи символів в повідомленні, оцінкою якої є частота появи символів у вхідних даних [4]. Як правило, ці імовірності невідомі. З урахуванням цього статистичні алгоритми можна поділити на три класи.

1. Неадаптивні - використовують фіксовані, завчасно задані імовірності символів. Таблиця ймовірностей символів не передається разом з файлом, оскільки вона відома завчасно. Недолік: невеликий перелік файлів, для яких досягається прийнятний коефіцієнт ущільнення.
2. Напівадаптивні - для кожного файла будується таблиця частот символів і за її допомогою ущільнюють файл. Разом з ущільненим файлом передається таблиця символів. Такі алгоритми досить непогано ущільнюють більшість файлів, але необхідна додаткова передача таблиці частот символів, а також два проходи початкового файлу для виконання кодування.
3. Адаптивні - починають працювати з фіксованою початковою таблицею частот символів (зазвичай всі символи спочатку рівноймовірні) і в процесі роботи ця таблиця змінюється в залежності від символів, що зустрічаються у файлі. Переваги: однопрохідність алгоритму, не потребує передачі таблиці частот символів, достатньо ефективно стискає широкий клас файлів.

Евристичні (словникові) алгоритми ущільнення (типу LZ77, LZ78), як правило, шукають в файлі рядки, що повторюються, і будують словник фраз, що вже зустрічались. Зазвичай такі алгоритми мають цілий ряд специфічних параметрів (розмір буфера, максимальна довжина фрази і т.п.), підбір яких залежить від досвіду автора роботи, і ці параметри добираються таким чином, щоб досягти оптимального співвідношення часу роботи алгоритму, коефіцієнта ущільнення та переліку файлів, що добре стискаються [2].

Арифметичне кодування, подібно до статистичних методів, використовує як основу технології ущільнення, імовірність появи символу у файлі, однак сам процес арифметичного кодування має принципові відмінності. У результаті арифметичного кодування символьна послідовність (рядок) замінюється дійсним числом більше нуля і менше одиниці [4].

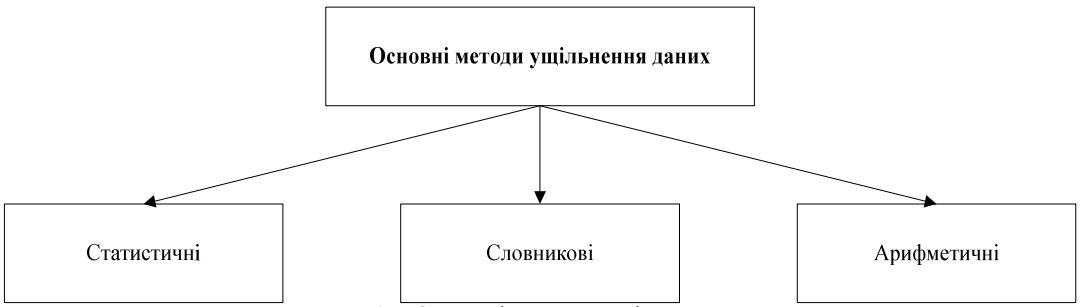

Рисунок 1 – Основні методи ущільнення даних

З появою методу арифметичного кодування проблема генерації коду була фактично вирішена. З тих пір основна увага стала приділятися питанням, пов'язаним з моделюванням. Нові підходи опираються на парадигму ущільнення за допомогою універсального моделювання і кодування *(universal modelling and coding)*, запропоновану Ріссаненом і Ленгдоном (Langdon) в 1981 р. [2,4]. Відповідно до даної концепції процес ущільнення складається з двох самостійних частин:

- моделювання;
- кодування.

Під моделюванням розуміється побудова моделі інформаційного джерела, що породжує дані, які ущільнюються, а під кодуванням – відображення оброблюваних даних в стислу форму подання на підставі результатів моделювання (рис. 2).

"Кодувальник" створює вихідний потік, що є компактною формою подання оброблюваної послідовності, на підставі інформації, що поставляється йому "моделювальником".

Слід відзначити, що поняття "кодування" часто використовують в широкому сенсі для позначення всього процесу ущільнення, тобто включаючи моделювання в даному нами означенні. Таким чином, необхідно розрізняти поняття кодування в широкому сенсі (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 сенсі. Це найменування далеке від досконалості і зустрічає цілком обґрунтовану критику. Далі в цьому розділі процес кодування в широкому сенсі іменуватимемо "кодуванням", а у вузькому сенсі - "статистичним кодуванням" або "власне кодуванням".

Оцінювання ймовірності символів при моделюванні проводиться на підставі відомої статистики і можливо, апріорних припущень, тому часто говорять про завдання статистичного моделювання.

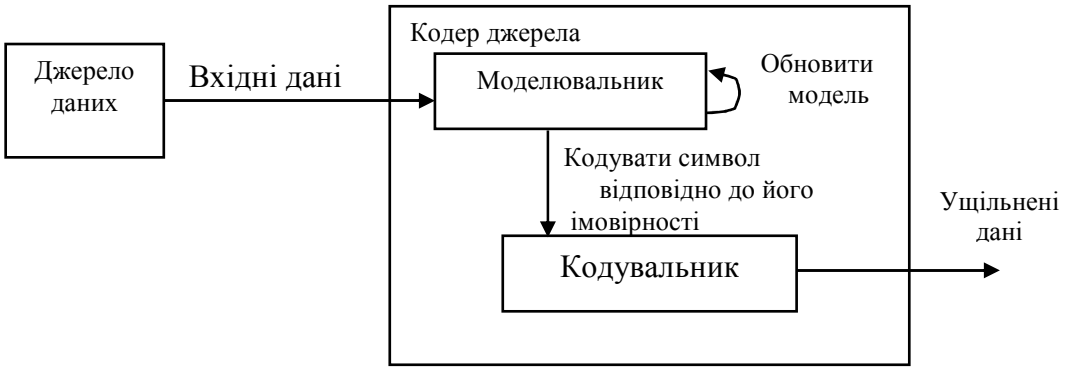

Рисунок 2 – Нова схема ущільнення даних

Можна сказати, що моделювальник передбачає імовірність появи кожного символу в кожній позиції вхідного рядка, звідси ще одне найменування цього компонента - "передбачувач" або "предиктор" (від predictor). Чим точніша оцінка імовірності появи символів, тим більше коди відповідають оптимальним, тим краще ущільнення. На етапі статистичного кодування виконується заміщення символу s_i, з оцінкою вірогідності появи $p(s_i)$ кодом завдовжки – $\log_2 p(s_i)$ біт.

Виділяють 4 варіанти моделювання: 1) статичне; 2) напівадаптивне; 3) адаптивне (динамічне); 4) блоково-адаптивне.

При статичному моделюванні для будь-яких даних, що обробляються, використовується одна і та ж модель. Опис моделі зберігається в структурах кодера і декодера. Недоліком такого підходу є погане ущільнення або навіть збільшення даних, що не відповідають заданій моделі.

Найбільш простий спосіб оцінювання імовірностей реалізується за допомогою напівадаптивного моделювання і полягає в попередньому підрахунку частот появ символів в блоці даних, що ущільнюється.

При адаптивному моделюванні у міру кодування модель змінюється за заданим алгоритмом після ущільнення кожного символу. Однозначність декодування досягається тим, що спочатку кодер і декодер мають ідентичну і звичайно дуже просту модель, модифікація якої при кодуванні і декодуванні виконується однаковим чином. Коефіцієнти ущільнення при адаптивному моделюванні можуть навіть досягати значень близьких до напівадаптивного.

Блоково-адаптивне моделювання є окремим випадком адаптивної стратегії. В цьому випадку модель обновлюється не після обробки кожного символу, а після обробки блока символів. Довжина блока може змінюватись в процесі кодування.

Аналіз розповсюджених типів даних виявляє сильну залежність ймовірності появи символів від попередніх символів у повідомленні, тобто більшість джерел повідомлень є джерелами з пам'яттю. Наприклад, для більшості європейських мов врахування безумовної імовірності символів дозволяє зменшити середню довжину коду до 4,5 біта на символ, а при використанні інформації про один попередній символ до 3,6 біта на символ. Врахування інформації про два попередні символи зменшує середню довжину коду до 3,2 біта на символ. Покращення ущільнення при врахуванні попередніх символів відзначається і для інших типів даних: об'єктних файлів, зображень, аудіоданих.

Моделювання, яке дає оцінку імовірності появи символу в залежності від попередніх, або контексту, називається контекстним моделюванням. Контекстне моделювання практично завжди застосовується як адаптивне.

Контекст в широкому смислі – це сукупність символів, що оточують поточний символ. Розрізняють також лівосторонні і правосторонні контексти, тобто послідовності символів, що примикають до поточного символу зліва і справа, відповідно. При контекстному моделюванні під контекстом розуміють саме лівосторонній контекст, оскільки контекстне моделювання практично завжди застосовується як адаптивне.

Якщо довжина контексту обмежена, то такий підхід називається контекстним моделюванням обмеженого порядку (finite context modeling), при цьому під порядком розуміється максимальна довжина N контекстів, що використовуються.

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. Серед інших методів контекстного моделювання в широкому сенсі можна відзначити такі:

- моделі станів (використовується при динамічному марковському ущільненні – Dinamic Markov Compression або DMC);
- граматичні моделі (використовуються в алгоритмі SEQUITUR);

- моделі з використанням штучних нейронних мереж для побудови передбачувача [2].

В світлі концепції універсального моделювання і кодування заслуговують на увагу методи ущільнення без втрат на основі перетворень. Мета використання перетворень в ущільненні даних - перетворення потоку вхідних подій до вигляду, що дозволяє використовувати простіші і ефективніші моделі. Фактично вони перетворюють одні види надмірності в інші, простіше модельовані. Тобто, перетворення дозволяє подавати оброблювану інформацію в особливій формі, ідеально відповідній для подальшого ефективного кодування. Незвичність підходу полягає в наявності фактично двох етапів моделювання: перший етап - це робота перетворення, направлена на отримання «зручного» інформаційного подання, а другий - побудова допоміжної моделі, на основі якої буде закодовано дане подання.

До таких перетворень відносять перетворення MTF та перетворення BWT (Burrows-Wheeler Transform) [6-7]. Однак якщо MTF давно використовується при ущільненні як перетворення, так і самостійний метод ущільнення, то по-перше перетворення BWT може використовуватись тільки як перетворення, а по-друге за рахунок використання перетворення BWT сумісно з MTF можна досягнути значних коефіцієнтів ущільнення. Перетворення BWT застосовується для перетворення ланцюжкової надмірності в надмірність повторення подій [6]. Спочатку вхідний потік подій циклічно зсувається на одну позицію і записується під початковим вхідним потоком стільки раз, скільки подій у вхідному потоці. Отримана квадратна матриця сортується по рядках зліва направо. Доведено, що для відновлення початкового потоку подій достатньо останнього стовпця матриці (так званого префіксного стовпця) і номера рядка початкового потоку подій після сортування. Префіксний стовпець має велику надмірністю повторення подій і локальну надмірність розподілу ймовірності.

Існують швидкі і ефективні реалізації даного перетворення, що не вимагають повної побудови квадратної матриці в пам'яті (що привело б до високих вимог до використовуваної пам'яті).

Вихідний потік перетворення BWT, як правило, або кодується за методом RLE, або додатково перетворюється за методом MTF і далі кодується або префіксним, або арифметичним кодером. Перетворення BWT використовується для ущільнення текстової і графічної інформації.

Пакети прикладних програм та формати файлів для ущільнення даних

7-Zip — популярний, безкоштовний файловий архіватор з високим ступенем ущільнення.

Основні переваги формату 7z: 1) відкрита архітектура; 2) високий коефіцієнт ущільнення; 3) шифрування AES-256; 4) можливість вибору будь-якого методу ущільнення, конверсії і шифрування; 5) підтримка файлів з розміром до 16000000000 GB; 6) підтримка файлів з іменами у форматі юні код; 7) сильна компресія; 8) ущільнення заголовків архіву.

7z має відкриту архітектуру, підтримуючи таким чином будь-який інший метод компресії. На даному етапі 7z підтримує методи наведені в табл. 1[8].

Таблиця 1 – Методи які використовує формат 7z

Метод	Опис
LZMA	Покращена і оптимізована версія алгоритму LZ77
LZMA2	Покращена версія LZMA
PPMD	PPMdH Дмитра Шкаріна з невеликими змінами
BCJ	Конвертер для виконуваних файлів 32-bit x86
BCJ2	Конвертер для виконуваних файлів 32-bit x86
BZip2	Стандартний алгоритм BWT
Deflate	Стандартний алгоритм базований на LZ77

ZIP — формат ущільнення та архівації даних. Файл цього формату зазвичай має розширення .zip і зберігає в ущільненому або не ущільненому вигляді один або декілька файлів. Використовує *LZW*-кодування, яке не вносить спотворень і втрат. Використовується і підтримується усіма сучасними архіваторами.

gzip (скорочення від GNU zip) — утиліта ущільнення і відновлення (декомпресії) файлів, що використовує алгоритм Deflate, який ґрунтується на словниковому алгоритмі ущільненні LZ77. Використовується в основному в UNIX-системах і є стандартом де-факто для ущільнення даних. gzip виконує тільки одну функцію: кодування і декодування одного файлу, він не вміє упаковувати декілька файлів в один архів. При ущільненні до оригінального розширення файлу додається суфікс .gz. Для упаковки кількох файлів зазвичай їх спочатку архівують в один файл утилітою tar, а потім цей файл ущільнюють gzip. Таким чином, ущільнені архіви зазвичай мають подвійне розширення .tar.gz.

Останнім часом gzip активно застосовується для ущільнення веб-контенту, gzip підтримує більшість сучасних браузерів та веб-серверів.

bzip2 — безкоштовна вільна утиліта командного рядка (а також алгоритм) з відкритим початковим кодом для ущільнення даних. bzip2 ущільнює більшість файлів ефективніше, але повільніше, ніж gzip або zip. bzip2 використовує перетворення Барроуза-Вілера (англ. Burrows-Wheeler transform) для

перетворення послідовностей символів, що багато разів чергуються, на рядки однакових символів, потім застосовує перетворення MTF (англ. move-to-front), і в кінці кодування Хаффмана.

Zlib — вільна крос-платформова бібліотека для ущільнення даних. Є узагальненням алгоритму ущільнення Deflate, який використовується компресорі gzip. Перша публічна версія бібліотеки 0.9 була випущена 1 травня 1995 року для використання разом з бібліотекою libpng. Поширюється за ліцензією zlib[9].

zlib є важливим компонентом багатьох програмних платформ, включаючи Linux, Mac OS X та iOS, також використовується в гральних консолях PlayStation 3/4, Wii, Xbox 360. zlib є стандартом де-факто, сотні додатків для *nix операційних систем (наприклад, GNU/Linux) використовують zlib. На інших платформам вона також знаходить застосування. Можна виділити такі області застосування zlib:

- Ядро Linux – реалізація мережевих протоколів із ущільнення, прозоре ущільнення, яке інтегроване у файлові системи, ущільнення завантажувального образу ядра для збереження на дисках (з розпаковкою під час завантаження).
- libpng, реалізація формату зображень PNG, використовує Deflate для потокового ущільнення даних.
- Сучасні HTTP-сервери (nGINX, Apache) використовують zlib для реалізації ущільнення для протоколу HTTP.
- Клієнт і сервер OpenSSH, для опціонального ущільнення, яке підтримується протоколом SSH.
- Бібліотека GnuTLS також може використовувати zlib для ущільнення з'єднань Transport Layer Security.
- Пакет програм 1С: Предприятие версій 7.7 і 8 використовує zlib для ущільнення файлів своєї бази даних (при файловому режимі організації даних) і даних у таблицях SQL (при серверній організації даних).

PAQ - серія вільних архіваторів з текстовим інтерфейсом, які спільними зусиллями розробників піднялися на перші місця рейтингів багатьох тестів ущільнення даних, хоча і ціною процесорного часу і обсягу пам'яті. Ущільнення даних опирається на парадигму ущільнення за допомогою універсального моделювання і кодування *(universal modelling and coding)*, запропоновану Ріссаненом і Ленгдоном (Langdon), алгоритм ґрунтується на ідеї контекстного моделювання та арифметичному кодуванні.

GIF (Graphics Interchange Format) — 8-бітний растровий графічний формат, що використовує до 256 кольорів із 24-бітного діапазону RGB. Набув широкої популярності у всесвітній павутині завдяки своїй відносній простоті та мобільності. Особливостями формату є підтримка анімації та прозорості. Для ущільнення файлів використовує LZW-компресію.

PNG (Portable Network Graphics) — растровий формат збереження графічної інформації, що використовує ущільнення без втрат. PNG використовує відкритий, не патентований алгоритм ущільнення Deflate, вільні реалізації якого доступні в Інтернет. Цей же алгоритм використовують і багато інших програм компресії даних.

JPEG – стандарт ущільнення та формат збереження фотографічних зображень з втратами інформації, на етапі власне ущільнення використовує кодування довжин серій та Хаффмана, а **JPEG-2000** використовує арифметичне кодування.

MPEG – стандарт ущільнення та формат збереження відео з втратами інформації, MPEG-1 та MPEG -2 на етапі власне ущільнення використовує кодування Хаффмана, а MPEG-4 використовує арифметичне кодування.

Вбудовані методи ущільнення інформації у мовах програмування

Мова програмування С++

С++ не має вбудованих методів ущільнення, але в мережі Інтернет можна знайти багато бібліотек написаних іншими програмістами. Наприклад, **LZMA SDK** від 7-Zip. **LZMA** є основним методом компресії формату 7z, що використовується за замовчуванням. **LZMA** забезпечує високий коефіцієнт ущільнення і добре підходить для ущільнення виконуваних файлів додатків.

Мова програмування Java

Для роботи з архівами в специфікації Java існують два пакети - java.util.zip і java.util.jar відповідно для архівів zip і jar. Різниця форматів jar і zip полягає тільки в розширенні архіву zip. Jar являє собою ZIP-архів, в якому міститься частина програми на мові Java.

Для читання Zip-архівів застосовується клас ZipInputStream. Для запису Zip-файлу застосовується клас ZipOutputStream.

Для кожного запису, який потрібно додати до Zip-файлу, створюється об'єкт ZipEntry. Далі викликається метод putNextEntry класу ZipOutputStream для початку процесу запису нового файлу. Після цього дані самого файлу відправляються у потік ZIP. По завершенні викликається метод closeEntry. Потім всі ці дії виконуються повторно для всіх інших файлів, які потрібно зберегти в ZIP-архіві.

Мова програмування С#

Для ущільнення даних існує клас ZipFile простору імен System.IO.Compression, який надає статичні методи для створення, вилучення і відкриття ZIP-архівів[10].

Клас ZipFile містить методи наведені у табл.2.

Таблиця 2 – Методи класу ZipFile

CreateFromDirectory ()	Створює ZIP архів, що містить файли і каталоги з зазначеного каталогу.
CreateFromDirectory ()	Створює ZIP-архів, що містить файли і каталоги з зазначеного каталогу, використовує зазначений рівень ущільнення і необов'язково включає базовий каталог.
CreateFromDirectory ()	Створює ZIP-архів, що містить файли і каталоги з зазначеного каталогу, використовує зазначений рівень ущільнення і кодування символів для імен записів і необов'язково включає базовий каталог.
ExtractToDirectory ()	Витягує всі файли з вказаного ZIP-архіву в каталог файлової системи.
ExtractToDirectory ()	Витягує всі файли з вказаного ZIP-архіву в каталог файлової системи і використовує зазначене кодування для імен записів.
Open ()	Відкриває ZIP-архів за вказаним шляхом і в заданому режимі.
Open ()	Відкриває ZIP-архів за вказаним шляху в зазначеному режимі і з використанням зазначеного кодування символів для імен записів.
OpenRead ()	Відкриває для читання ZIP-архів за вказаним шляхом.

Мова програмування Python

Для роботи з zip архівами у python існує стандартний модуль zipfile. Він містить наступні методи:

- ZipFile() – відкриває ZIP-файл для запису зчитування або додавання до існуючого файлу.
- Close() – закриває файл архіву, Close() викликається перед виходом з програми.
- Getinfo() – повертає об'єкт ZipInfo з інформацією про архів.
- Infolist() – повертає список, що містить об'єкт ZipInfo для кожного члена архіву.
- Namelist() – повертає список імен членів архіву.
- Read () – повертає байти файлу в архіві. Архів повинен бути відкритий для читання або запису.
- Write() – запис файлу в архів з використанням заданого типу ущільнення.
- Extract() – розархівовує з відкритого архіву файл з заданим іменем.
- Extractall() – розархівовує всі файли з відкритого архіву.

Висновки

Аналіз методів ущільнення та їх практичної реалізації показав, що більшість відомих архіваторів комбінують декілька методів для досягнення прийнятного коефіцієнту ущільнення, хоча і ціною процесорного часу і обсягу пам'яті. Але завдяки збільшенню швидкодії комп'ютерів швидкість роботи стає менш критичною.

Найбільш перспективними є підходи, які ґрунтуються на словникових методах ущільнення (LZW, LZMA та інші) та контекстному моделюванні і арифметичному кодуванні (PAQ).

Не слід очікувати появи революційних рішень в області ущільнення даних, зусилля дослідників будуть направлені в основному на більш ретельне дослідження існуючих підходів та їх комбінацій, підвищення коефіцієнту ущільнення буде досягатись за рахунок збільшення обчислювальних витрат та обсягів пам'яті.

Щодо до форматів файлів для збереження ущільнених даних, то тут спостерігається велике різноманіття, тому назріла необхідність розробки міжнародного стандарту для формату такого файлу, хоча дефакто таким є формат файлів ZIP. Наявність такого стандарту є важливою для розробників бібліотек ущільнення даних для різних мов програмування, що в перспективі дозволить різним додаткам безпосередньо маніпулювати ущільненими даними.

Література

1. Claude E. Shannon. The Mathematical Theory of Communication / Claude E. Shannon, Weaver Warren. – University of Illinois Press, Urbana, 1963. – 63с.
2. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. / Ватолин Д., Ратушняк А., Смирнов М., Юкин В. – М.: ДИАЛОГ-МИФИ, 2002. – 384 с.
3. Сэломон Д. Сжатие данных, изображений и звука. / Д. Сэломон – М.: Техносфера, 2004. – 368 с.
4. Кожем'яко В.П. Аналіз та перспективи розвитку кодування зображень / В.П. Кожем'яко, В.П. Майданюк, К.М. Жуков - Вісник ВПІ, 1999, № 3. – 42-48с.
5. Майданюк В. П. Кодування та захист інформації. / В. П. Майданюк - Вінниця: ВНТУ, 2009. - 164 с.
6. Майданюк В. П. Ущільнення даних без втрат на основі перетворень / В. П. Майданюк, Кириченко О. В. – Оптико-електронні інформаційно-енергетичні технології. – 2008. - № 2(16) – С. 71-76.
7. Mark Nelson. Data Compression with the Burrows-Wheeler Transform / Mark Nelson – Dr. Dobb's Journal, 1996 – 103с.
8. 7-Zip - потужний архіватор, що підтримує безліч форматів архівів - Режим доступу: http://7-zip.org.ua/7z.html – Назва з домашньої сторінки Інтернету.
9. zlib Матеріал з Вікіпедії — вільної енциклопедії - Режим доступу: https://uk.wikipedia.org/wiki/Zlib – Назва з екрана.
10. MSDN – сеть разработчиков Microsoft – Режим доступу: https://msdn.microsoft.com/ru-ru/library/system.io.compression(v=vs.110).aspx – Назва з домашньої сторінки Інтернету.

Відомості про авторів

Пєтух Анатолій Михайлович – д.т.н. , професор кафедри програмного забезпечення ВНТУ.

Майданюк Володимир Павлович — к.т.н. , доцент кафедри програмного забезпечення ВНТУ.

Ліщук Олександр Олександрович — магістр з програмного забезпечення систем ВНТУ.