

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

**ДОСЛІДЖЕННЯ МЕТОДІВ СТИСНЕННЯ ДАНИХ ДЛЯ ЗМЕНШЕННЯ
РОЗМІРУ ФАЙЛІВ ЗОБРАЖЕНЬ І ТЕКСТОВИХ ДАНИХ**

Здобувача вищої освіти другого (магістерського) рівня,
II курсу, групи КМІ-23-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»



(підпис)

Якубовського Василя Сергійовича

(прізвище, ім'я, по батькові)

Науковий керівник



(підпис)

Омельчук Антон Анатолійович, к.т.н.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Керівник освітньо-
професійної програми



(підпис)

Горбовий Артур Юліанович,
д.т.н, професор

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри протокол від «10» січня 2025 №11



(підпис)

Філоненко Михайло
Миколайович, к.ф.-м.н.,

доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Завідувач кафедри

Ірпінь – 2025

Реєстрація № 15

27.01.2025

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту, на доопрацювання)

27.01.2025

(дата)



(підпис)

Результат захисту

5 (відмінно)
(оцінка)

06.02.2025

(дата)



(підпис)

Голова комісії:

В.М.
(підпис)

В.М. Франчук

Члени комісії:

А.Ю.
(підпис)

А.Ю. Горбовий

М.М.
(підпис)

М.М. Філоненко

І.Д.
(підпис)

І.Д. Погореловська

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	26		06.02.2025 Франчук В.М. <i>В.М.</i> Горбовий А.Ю. <i>А.Ю.</i>
2. Дотримання вимог щодо оформлення	від 0 до 10	9		Філоненко М.М. <i>М.М.</i>
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		Погореловська І.Д. <i>І.Д.</i> Суховерхівка Л.П. <i>Л.П.</i>
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	44		

Оцінювання кваліфікаційної роботи науковим керівником - 46 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
90	A	5 (відмінно)	06.02.2025	Франчук В.М. <i>В.М.</i> Горбовий А.Ю. <i>А.Ю.</i> Філоненко М.М. <i>М.М.</i> Погореловська І.Д. <i>І.Д.</i> Суховерхівка Л.П. <i>Л.П.</i>

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую

Завідувач кафедри  Філоненко М.М.

«23» грудня 2023

**Завдання на кваліфікаційну роботу
здобувачу вищої освіти**

Якубовському Василю Сергійовичу

(прізвище, ім'я, по батькові)

Тема кваліфікаційної роботи

Дослідження методів стиснення даних для зменшення обсягу файлів зображень і текстових даних

1. Затверджена наказом ректора від «20» грудня 2023 № 1701

2. Строк здачі закінченої роботи «07» січня 2025

Цільова установка застосування набутих знань, умінь для виконання порівняльного аналізу інструментів стиснення файлів

3. Мета роботи: пошук ефективних інструментів стиснення графічних та текстових даних.

Об'єкт дослідження: засоби забезпечення компресії файлів.

Предмет дослідження інструменти стиснення обсягів графічних та текстових файлів.

База дослідження Лабораторія цифрових проєктів

4. Вихідні дані до роботи методи, алгоритми, утиліти, які призначені для зменшення обсягу файлів зображень та текстових даних

5. Перелік графічного матеріалу рисунки, лістинги коду, презентація роботи.

6. Зміст випускної кваліфікаційної (дипломної) роботи: 1. Теоретичні аспекти стиснення даних. 1.1. Стиснення даних як стратегічний актив в управлінні даними. 1.2. Систематизація методів стиснення. 2. Формати компресії зображень. 2.1. Особливості графічних зображень та актуальність їх стиснення. 2.2. Експериментальне порівняння форматів компресії зображень. 3. Порівняльний аналіз систем стиснення. 3.1. Аналітичний огляд систем стиснення. 3.2. Порівняння результатів роботи систем стиснення. 3.3. Реалізація алгоритму стиснення JPEG у середовищі Google Colab

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.12.2023	15.12.2023
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	27.05.2024	27.05.2024
3	II розділ (аналіз експериментальної частини про методики досліджень)	02.09.2024	02.09.2024
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проєктні рішення)	18.11.2024	18.11.2024
5	Висновки	10.12.2024	10.12.2024
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	23.12.2024	23.12.2024
7	Отримання відгуку керівника	28.01.2025	28.01.2025
8	Отримання зовнішньої рецензії	31.01.2025	31.01.2025

8. Дата видачі завдання «23» грудня 2023

9. Керівник кваліфікаційної роботи

10. Керівник освітньо-професійної програми

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Омельчук А
(прізвище, ініціал)


(підпис)

Горбовий А
(прізвище, ініціал)


(підпис)

Якубовський

АНОТАЦІЯ

“Дослідження методів стиснення даних для зменшення обсягу файлів зображень і текстових даних”

Мета роботи пошук ефективних інструментів стиснення графічних та текстових даних.

Завдання роботи

- визначити сутність та основні принципи стиснення даних;
- порівняти функціональність безвтратних (LossLess) та втратних (Lossy) алгоритмів стиснення;
- дослідити особливості форматів графічних зображень;
- проаналізувати функціональність утиліт стиснення;
- здійснити програмну реалізацію алгоритму стиснення.

Об’єкт дослідження: засоби забезпечення компресії файлів.

Предмет дослідження: інструменти стиснення обсягів графічних та текстових файлів.

База дослідження: навчальна лабораторія цифрових проєктів.

Інформаційна база дослідження: методи, алгоритми та програмні засоби забезпечення стиснення файлів зображень та текстових файлів.

За результатами дослідження було сформульовано висновки про те, що

- вибір оптимального методу стиснення залежить від типу даних, допустимого ступеня втрат та потужності обчислювальних ресурсів;
- ефективним інструментом стиснення файлів є програмне забезпечення (графічні редактори, утиліти стиснення, архіватори), в якому реалізовано безвтратні та втратні алгоритми стиснення.

Одержані результати можуть бути використані при вирішенні проблем стиснення графічних та текстових файлів.

Кваліфікаційна робота містить: 59 сторінок, 8 таблиць, 8 рисунків, список використаних джерел з 63 найменувань, 8 додатків.

Ключові слова: данні, файли, зображення, алгоритм, стиснення, розпакування, утиліти.

Рік виконання магістерської роботи 2024-2025

Рік захисту роботи 2025

SUMMARY

“Study of data compression methods to reduce the size of image and text files”

The purpose of the work is to search for effective tools for compressing graphic and text data.

The tasks of the work:

- to determine the essence and basic principles of data compression;
- to compare the functionality of lossless (LossLess) and lossy (Lossy) compression algorithms;
- to investigate the features of graphic image formats;
- to analyze the functionality of compression utilities;
- to perform a software implementation of the compression algorithm.

Object of research: means of ensuring file compression.

Subject of research: tools for compressing graphic and text file volumes.

Research base: educational laboratory of digital projects.

Research information base: methods, algorithms and software tools for ensuring compression of image and text files.

According to the results of the research, conclusions were formulated that

- the choice of the optimal compression method depends on the type of data, the permissible degree of loss and the power of computing resources;
- an effective tool for compressing files is software (graphic editors, compression utilities, archivers), in which lossless and lossy compression algorithms are implemented.

The results obtained can be used to solve problems of compressing graphic and text files.

The qualification work contains: 59 pages, 8 tables, 8 figures, a list of used sources from 63 items, 8 appendices.

Keywords: data, files, images, algorithm, compression, decompression, utilities.

Year of completion of the master's thesis 2024-2025

Year of defense of the work 2025

Зміст

ВСТУП.....	3
РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ СТИСНЕННЯ ДАНИХ	6
1.1. Стиснення даних як стратегічний актив в управлінні даними	6
1.2. Систематизація методів стиснення даних.....	11
РОЗДІЛ 2. ФОРМАТИ КОМПРЕСІЇ ЗОБРАЖЕНЬ.	29
2.1. Особливості графічних зображень та актуальність їх стиснення.....	29
2.2. Експериментальне порівняння форматів компресії зображень.....	33
РОЗДІЛ 3. ПОРІВНЯЛЬНИЙ АНАЛІЗ УТИЛІТ СТИСНЕННЯ.	43
3.1. Аналітичний огляд систем стиснення	43
3.2. Порівняння результатів роботи утиліт стиснення.	50
3.3. Реалізація алгоритму стиснення JPEG у середовищі Google Colab.	54
ВИСНОВКИ	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТКИ.	64

ВСТУП

Актуальність теми. Характерною ознакою більшості даних є надлишковість, яка сприяє кращому розумінню їх змісту та сприйняттю. Водночас, надлишковість призводить до зростання обсягів файлів даних, що зумовлює потреби у збільшенні потужностей обчислювальних ресурсів для їх обробки. Одним із способів забезпечення компромісу між розміром файлу та якістю даних цих файлів є їх стиснення за рахунок усунення чи зменшення надлишковості даних. Але щоб забезпечити ефективне стиснення, важливо знайти і застосувати алгоритм стиснення, який найкраще підходить для конкретного типу даних.

Оскільки існує достатньо велика кількість методів, алгоритмів, програмних засобів та підходів до стиснення даних, то їх дослідження з метою подальшого ефективного використання є актуальним для наукового пошуку та практичного застосування в галузі ІТ.

Стан і ступінь дослідження теми в літературі. Вперше про можливість вирішення проблем стиснення даних було зазначено у роботах Абрахама Лемпеля (Abraham Lempel) та Джекоба Зів (Jacob Ziv) [1, 2]. Проблемами розробки алгоритмів стиснення та їх застосуванням у відомих архіваторах займався Девід Саломон (David Salomon) [4, 5]. Відомий фахівець в галузі ІТ, Матт Махоні (Matt Mahoney) досліджував у своїх роботах проблеми стиснення довгих текстів [6, 7]. Питаннями створення алгоритму безвтратного стиснення займався Філіп Катц (Phillip Katz), який розробив алгоритм Deflate, що використовується в утилітах ZIP, PKZIP [8]. Розробкою програмного забезпечення для стиснення файлів займалися Жан Луп Гайлі (Jean-Loup Gailly) та Марк Адлер (Mark Adler), саме вони створили оригінальну утиліту GZIP [52].

Серед українських вчених відомими є Глибовець А. М., Яблонський В. Р. , які опублікували результати порівняння алгоритмів компресії даних [10]. Особливості стиснення даних в мережах IoT досліджують Ю. С. Манжос, Є.В.Соколова [11]. Про

пошук методів ефективного кодування повідомлень свідчать роботи Розенвассера Д.М., Педяша В.В. [13].

Враховуючи актуальність обраної теми, стан і ступінь її дослідження було визначено мету виконання кваліфікаційної роботи: пошук ефективних інструментів стиснення графічних та текстових даних.

Для досягнення визначеної мети були вирішені наступні завдання:

- визначено сутність та основні принципи стиснення даних;
- досліджено безвтратні (LossLess) та втратні (Lossy) алгоритми стиснення;
- виконано порівняння форматів растрових зображень у графічному редакторі GIMP;
- проаналізовано функціональність утиліт стиснення ZSTD, GZIP, BZIP2, XZ;
- реалізовано у хмарному середовищі Google Colab алгоритм стиснення JPEG на мові Python.

Об’єкт дослідження: засоби забезпечення компресії файлів.

Предмет дослідження: інструменти стиснення обсягів графічних та текстових файлів.

Методи дослідження: при виконанні роботи були використані наступні методи: системний аналіз, який застосовувався при роботі з інформаційними ресурсами за темою дослідження; методи оцінювання застосовувалися при аналізі результатів стиснення файлів, при аналізі функціональності утиліт стиснення файлів; порівняльний аналіз виконувався при дослідженні результатів стиснення файлів растрових зображень, результатів роботи утиліт ZSTD, GZIP, BZIP2, XZ; методи групування – при формуванні таблиць результатів систематизації алгоритмів стиснення даних та проведеного експерименту з різними утилітами стиснення файлів; розробка та тестування програми стиснення за алгоритмом JPEG; за допомогою методу узагальнення формувалися висновки.

Інформаційна база дослідження: методи, алгоритми та програмні засоби забезпечення стиснення файлів зображень та текстових файлів.

Структура дипломної роботи містить вступ, основну частину, яка складається з трьох розділів, до якої входять 8 рисунків, 8 таблиць, висновки, перелік використаної літератури, що містить 63 джерел, 8 додатків.

За результатами дослідження було сформульовано висновки про те, що:

- вибір оптимального методу стиснення залежить від типу даних, допустимого ступеня втрати та потужності обчислювальних ресурсів;
- ефективним інструментом стиснення файлів є програмне забезпечення (графічні редактори, утиліти стиснення, архіватори), в якому реалізовано безвтратні та втратні алгоритми стиснення.

Науково-практичне значення одержаних результатів:

- результати тестування утиліт стиснення (ZSTD, GZIP, BZIP2, XZ) можуть бути використані при їх виборі та застосуванні;
- результати дослідження форматів растрових зображень у графічному редакторі GIMP можуть бути використані при вирішенні проблем стиснення графічних зображень;
- реалізований у хмарному середовищі Google Colab алгоритм JPEG може використовуватися для стиснення файлів із заданим рівнем якості.

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ СТИСНЕННЯ ДАНИХ

1.1. Стиснення даних як стратегічний актив в управлінні даними

У сучасному світі, де інформація генерується й накопичується в безпрецедентних масштабах, методи архівування та стиснення даних стають не просто корисною функцією, а й критично важливим інструментом управління даними. Ці методи дозволяють оптимізувати використання простору для зберігання, економити пропускну здатність каналів зв'язку, пришвидшувати передачу даних та покращити їх захист.

У роботі відомого розробника програмного забезпечення Давида Саломона (David Salomon) [5] стиснення даних визначається як “процедура, що перетворює вхідні послідовності даних в іншу послідовність символів та займає менший обсяг пам'яті”. Завдяки цьому необхідно менше технічних ресурсів для їх збереження та передачі по мережі. Це надає змогу зменшити витрати компаній та користувачів на обладнання та час, необхідний для передачі цих даних. Тобто, стиснення даних застосовується з метою оптимізації зберігання інформації та швидшого обміну файлами.

На сайті Techopedia.com стиснення даних трактується як “процес модифікації, кодування або перетворення бітової структури даних для економії місця на диску та ефективної їх передачі. Це досягається шляхом усунення надлишкової інформації або ефективнішого представлення даних у стислому вигляді” [14].

Відомий фахівець в галузі ІТ, Матт Махоні (Matt Mahoney), надає більш технічне визначення, поняття стиснення даних, підкреслюючи, що це метод представлення інформації меншою кількістю бітів. “Основна мета стиснення - усунення надлишковості у даних шляхом застосування математичних алгоритмів” [15].

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, які носять абсолютно випадковий характер, з однаковою ймовірністю появи всіх можливих символів. Отже, процедура стиснення даних спрямована на представлення даних в більш компактній формі за рахунок усунення надлишковості з метою економії ресурсів систем передачі та зберігання даних. Стиснення даних також може підвищувати стійкість систем шифрування, оскільки після зменшення надлишковості дані мають псевдовипадковий характер, а додаткове шифрування лише підвищує складність їх взлому.

Основний принцип алгоритмів стиснення базується на припущенні, що в будь-якому файлі, який містить не випадкові дані, інформація частково повторюється. Використовуючи статистичні математичні моделі, можна визначити ймовірність повтору конкретних комбінацій символів. Після цього можна створити коди, яким відповідатимуть вибрані фрази, і надати фразам, які часто повторюються, найкоротші коди. Для цього використовують різні техніки, наприклад: кодування повторів, стиснення за допомогою словника, ентропійне кодування [16].

У роботі Коваленко А.Є. [17] зазначено, що “стиснення даних, або ефективне кодування, або економне кодування — це кодування інформації для зменшення надлишковості повідомлень”. Поширеними способами побудови економних кодів є кодові таблиці, кодові дерева, метод Шеннона-Фано, метод Хаффмена.

Для різних типів даних використовують різні методи стиснення. Методи стиснення, що ефективні для текстових даних, можуть бути не оптимальними для відео-, аудіо-файлів або зображень, оскільки кожен тип даних має свої унікальні властивості та вимоги. Алгоритми, спеціально розроблені для стиснення аудіо або зображень, враховують специфічні особливості цих форматів, що дозволяє досягти кращих результатів. Найчастіше для музики, відео та аудіо використовують стиснення з втратами (англ. *lossy compression*), або незворотнього кодування.

При виборі методу стиснення даних варто також враховувати сферу його застосування, оскільки є неприпустимими неточності чи помилки, наприклад, у медицині, будівельній справі, космічній сфері, кібербезпеці тощо.

При стисненні з втратами відновлені дані відрізнятимуться від оригіналу, але можуть бути корисними для використання [16].

Для текстової інформації, фрагментів баз даних або програмних файлів, які повинні виконуватися, таке кодування призведе до втрати частини інформації, або навіть до неможливості коректної роботи цих файлів. Тому для цих видів даних використовують методи стиснення без втрат (англ. *lossless compression*), які дозволяють повністю відновити вихідні дані без спотворень, з точністю до біта.

Серед методів стиснення без втрат виокремлюють статичні, адаптивні (динамічні), гібридні [21]. Зокрема, у статичних методах кодові слова фіксуються перед передачею, тобто певний набір інформації буде представлений тим самим кодовим словом, якщо воно з'являється в повідомленні. Статичне кодування вимагає двох проходів: обчислення ймовірностей (або частоти) та другий прохід - для кодування. Прикладом є кодування Хаффмана.

Адаптивні або динамічні методи дозволяють зменшити середню частоту дискретизації, засновані на апроксимації вимірювальної величини функцією часу. Адаптивними методами є метод LZ77, LZ78, LZW і адаптивне кодування Хаффмана. У цих методах використовується лише одних прохід. Гібридні методи об'єднують алгоритми статичних та адаптивних методів [3].

Очевидно, що методи стиснення даних із втратами мають значну перевагу перед методами стиснення без втрат за ступенем стиснення початкових даних. Тому саме ці методи зазвичай використовуються для зберігання та передачі зображень. Під час використання методів стиснення з втратами багато уваги приділяють саме сприйняттю вихідного файлу користувачами. Тобто, треба завжди намагатись знайти найкраще співвідношення між ступенем стиснення та якістю результуючого файлу. Звісно, що файл намагаються зробити якомога меншим, але якщо він матиме

дуже погану якість відео або звуку, то, відповідно, його цінність буде набагато меншою, ніж у файлі трохи більшого розміру, але з достатніми характеристиками.

При виборі методів стиснення даних для застосування в конкретній системі враховують наступні характеристики: коефіцієнт стиснення; 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 – при стисненні з втратами [11].

Коефіцієнт стиснення визначається за формулою $k_c = P_{\text{вих}}/P_{\text{вх}}$

де k_c – коефіцієнт стиснення; $P_{\text{вих}}$ – обсяг вихідних даних; $P_{\text{вх}}$ – обсяг вхідних даних. Коефіцієнт стиснення показує скільки в середньому необхідно біт стисненого файлу для представлення одного біта файлу на вході. Значення коефіцієнта стиснення більше 1 свідчить про “негативне стиснення”, тобто отриманий вихідний файл є більшим вхідного.

Одиницями виміру значень коефіцієнта стиснення є bpb (bit per bit, біт на біт), або bpc (bit per character, біт на символ), тобто, скільки в середньому потрібно біт для зберігання однієї літери тексту. При стисненні графічних зображень аналогічно визначається величина bpr (bit per pixel, біт на піксель).

Із коефіцієнтом стиснення пов'язані ще поняття:

- Бітова швидкість (bitrate). Метою компресії є подання даних з найменшою бітовою швидкістю.
- Бітовий бюджет (bit budget), що визначає відсоток у стислому файлі для зберігання таблиць, які будуть використовуватися декодером при декомпресії.

Величина, зворотня коефіцієнту стиснення, називається ступенем або фактором стиснення: $C_c = \Phi_c = P_{\text{вих}}/P_{\text{вх}}$. У цьому випадку значення більше 1 означають стиснення, а менше 1 - розширення.

У залежності від області застосування можуть висуватися різні вимоги до таких характеристик, як швидкодія та необхідний обсяг пам'яті при упаковці та

розпаковці даних. При застосуванні в системах передачі даних параметри швидкодії є критичними насамперед при стисненні, а якщо отримувачем інформації на стороні приймача є людина (наприклад, при телефонній розмові), то і при розпакуванні.

Ступінь спотворення даних при стисненні з втратами оцінюють, як середньоквадратичне відхилення між вихідними даними X та даними, отриманими після розпаковки Y :

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

При виборі алгоритму стиснення обов'язково враховують складність його реалізації, особливо, якщо критичними параметрами є час та вартість розробки системи, та результати порівняння з простішими у реалізації алгоритмами, у яких ступінь стиснення є ненабагато нижчим. Також береться до уваги тип даних, для яких стиснення даним алгоритмом є максимальним [13].

Крім технології стиснення даних, важливим є архівування даних. Архівація даних - стискання одного або багатьох файлів з метою зменшення обсягу даних при їх зберіганні на носіях інформації або при передачі даних по каналах зв'язку, у т. ч. і в мережі Інтернет, та розміщення стислих файлів в одному архівному файлі. [18].

Існує кілька типів архівів, таких як:

- ZIP-архіви: це один з найпоширеніших типів архівів. Вони використовуються для зберігання та стиснення групи файлів у одному архіві. Використовує LZW-стиснення, яке не вносить спотворень і втрат [55].
- TAR-архіви: використовуються в операційних системах UNIX для об'єднання декількох файлів в один архів без стиснення [19].

У контексті управління даними і забезпечення їх ефективного зберігання та передачі, стиснення даних та архівування даних взаємопов'язані концепції. Стиснення даних спрямоване на зменшення обсягу даних за рахунок використання різних методів кодування, що дозволяє економити місце на диску та забезпечувати

швидкодію передачі. У свою чергу, архівування даних передбачає збереження інформації для майбутнього використання або зберігання історії даних, часто інтегруючи в себе процес стиснення. Ці процеси можуть використовуватися окремо або в поєднанні для оптимізації управління даними, економії ресурсів, забезпечення доступності та цілісності даних, і є важливою складовою сучасних систем управління даними.

Загалом сучасні технології стиснення та архівування даних відіграють важливу роль у стратегічному управлінні даними, оскільки дозволяють суттєво оптимізувати зберігання, захист, передачу та обробку інформації, що сприяє досягненню конкурентних переваг та оптимізації бізнес-процесів.

1.2. Систематизація методів стиснення даних

1.2.1. Алгоритми методів стиснення даних без втрат

Проведений аналіз літератури [3, 4, 7, 15] свідчить, що розроблено достатньо велику кількість методів, які дозволяють отримати точну реконструкцію оригінального файлу даних зі стиснутої інформації, при цьому жоден біт не буде втрачений. До таких методів належать найпростіший метод RLE, словникові методи (LZ77, LZ78 і LZW), метод Хаффмана (Huffman), арифметичне кодування (рис. 1.1).

Метод Хаффмана (Huffman), арифметичне кодування - це приклади ентропійного кодування. “Ентропійне кодування - кодування послідовності значень з можливістю однозначного відновлення з метою зменшення обсягу даних (довжини послідовності) за допомогою усереднення ймовірностей появи елементів у закодованій послідовності” [3].

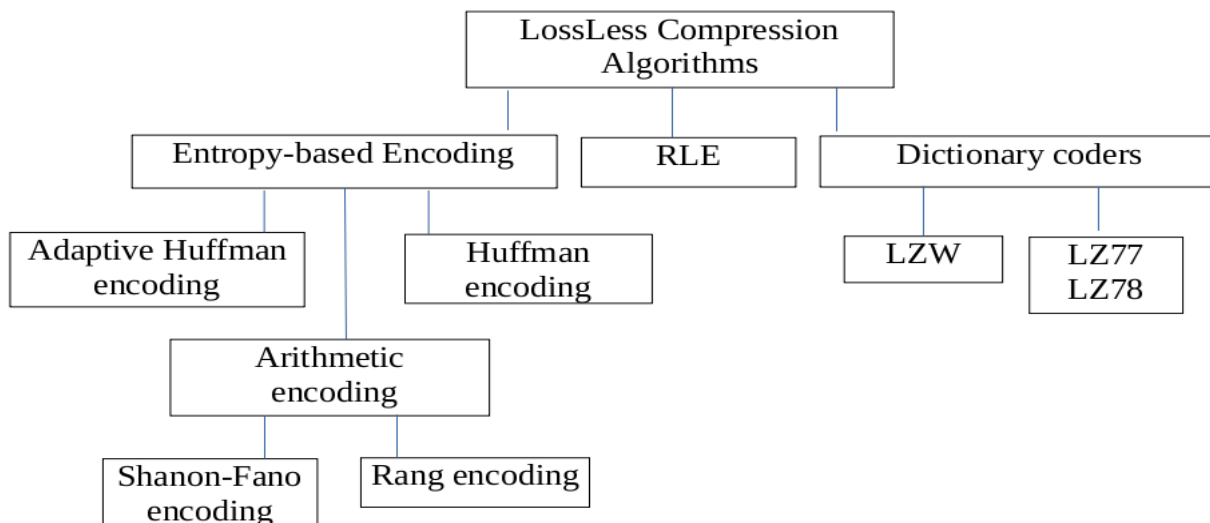


Рис. 1.1. Класифікація методів стиснення без втрат (складено за [3])

Метод стиснення RLE

Одним із найпростіших методів стиснення, який особливо ефективний для даних з великою кількістю повторюваних символів або структур є метод RLE (Run-Length Encoding, або кодування довжини серій). Метод базується на заміні послідовностей повторюваних символів на компактнішу форму, що описує сам символ і кількість його повторень. Згідно алгоритму RLE вхідний потік даних розбивається на серії символів (послідовності одного й того ж символу). Для кожної серії фіксується сам символ та кількість його повторень. Замість довгих серій записується код у форматі: <символ><кількість повторень> [4].

Перевагами методу RLE є простота реалізації та ефективність при роботі з даними, що містять великі блоки однакових символів (наприклад, монохромні зображення або текст із великою кількістю пропусків). Не вимагає додаткової пам'яті при архівації та розархівачії, а також швидко працює. Недоліками методу RLE є низька ефективність для неоднорідних даних, у разі якщо символи часто змінюються, RLE може збільшити розмір даних. Тобто, при незначних відмінностях у зображеннях чи сигналах метод буде неефективним.

Метод RLE використовують для стиснення зображень у форматах BMP, PCX, TGA, TIFF (для певних типів), для стиснення векторних шрифтів для компактного зберігання, для оптимізації передачі даних у телекомунікаціях. RLE може бути частиною складніших методів стиснення у форматах, таких як ZIP або RAR [20].

Словникові методи стиснення

Серед поширених методів стиснення даних без втрат виокремлюють методи стиснення за допомогою словників - алгоритми Лемпеля-Зіва (Lempel-Ziv, LZ77, LZ78), розроблені у 1977 та 1978 роках [1, 2]. Алгоритми сімейства LZ є універсальними алгоритмами стиснення даних, у яких словник формується на підставі вже прочитаної та обробленої частини вхідного повідомлення, тобто адаптивно. Принципова відмінність - лише спосіб формування фраз (послідовності символів).

Основною ідеєю алгоритмів є заміна послідовностей символів на більш короткі послідовності зі словника. Наприклад, для послідовності:

MBCMBCNNNNMBC при MBC = 1 та NNNNN = 2 => 1 1 2 1

Тобто, повторювані послідовності отримують власні коди, які можуть бути набагато коротшими за послідовності символів, які вони замінюють. Також це дозволяє не записувати кожен раз кожне значення символу, оскільки існує лише код, та послідовність байт, яка цьому коду відповідає. Цей алгоритм вирізняється високою швидкістю роботи при кодуванні та декодуванні, невибагливістю до пам'яті та простою апаратною реалізацією. Розпакування стислих файлів здійснюється дуже швидко, тому ці алгоритми використовуються для створення програм, які саморозпаковуються [21].

Словникові алгоритми неодноразово покращувались, наприклад, у 1984 році Террі Велчем була опублікована покращена реалізація алгоритму, яка і отримала назву алгоритм Лемпеля-Зіва-Велча (англ. Lempel-Ziv-Welch algorithm, LZW) [21].

Алгоритм LZW є універсальним алгоритмом - саме його варіанти використовуються у звичайних архіваторах та форматах зображень GIF, TIFF, PDF.

Алгоритм Хаффмана

Це один із класичних алгоритмів стиснення, запропонований Девідом Хаффманом (Huffman) у 1952 році. Алгоритм Хаффмана є оптимальним префіксним кодом для дискретних джерел за критерієм: мінімальна середня довжина кодової комбінації символів (мінімальна надмірність). При цьому, символи з найбільшою ймовірністю появи в початкових даних (кількістю входжень) отримують коротші коди. Чим частіше зустрічається символ, тим коротший код йому присвоюється (менше 8 біт), а для запису рідкісних символів використовують довші коди [21].

Для стиснення чорно-білих зображень використовується алгоритм Хаффмана з фіксованою таблицею CCITT Group 3. Це означає, що даний алгоритм був запропонований третьою групою по стандартизації Міжнародного консультативного комітету з телеграфії і телефонії (Consultative Committee International Telegraph and Telephone). Стандарти Group 3 і Group 4 – це алгоритми стиснення, спеціально розроблені для кодування однобітових даних зображення [5].

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 символ додається у дерево. Для декодера символ ESC є вказівкою на те, що наступні K біт позначають символ первинного алфавіту, при цьому $K = \log_2 N$, де N – кількість символів в первинному алфавіті. При надходженні кожного наступного символу 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 [21].

Алгоритм Шеннона-Фано

У більшості випадків довжина стислої послідовності алгоритмом Шеннона-Фано дорівнює довжині стислої послідовності з використанням кодування Хаффмана. Але є послідовності, в яких методика Шеннона-Фано не дає однозначного кодування, тому метод стиснення Хаффмана вважається більш ефективним.

Перевагою даного методу є його очевидна простота реалізації та, як наслідок цього, висока швидкість кодування/декодування. Основним недоліком є його не оптимальність в загальному випадку. Метод 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 [3].

Коди Хаффмана та 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. У 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 [22].

Алгоритм Deflate

Алгоритм Deflate - це алгоритм стиснення без втрат, який використовує комбінацію алгоритмів LZ77 і алгоритму Хаффмана. Був описаний Філом Кацем для другої версії утиліти створення архівів PKZIP [23]. Deflate вважається вільним від усіх існуючих патентів. Це призвело до його широкого розповсюдження, зокрема у форматі стиснення даних GZIP, форматі зображень PNG і форматі .ZIP, для якого Ф.Кац його і розробив.

Алгоритм Deflate визначає формат стиснутих даних без втрат, який стискає дані з ефективністю, порівнянною з найкращими доступними на даний момент методами стиснення загального призначення.

Алгоритм JBIG

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 [3].

Алгоритм Lossless JPEG

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 [24].

Арифметичне кодування

Арифметичне кодування (англ. Arithmetic coding) - один з алгоритмів ентропійного стиснення. Цей метод вдосконалює кодування Хаффмана, використовуючи двійкові дроби для представлення символів. На відміну від алгоритму Хаффмана, не має жорсткої постійної відповідності вхідних символів - групам біт вихідного потоку. Це дає алгоритму більшу гнучкість у поданні дробових частот появи символів. При арифметичному стисненні повідомлень алфавіту джерела ставиться у відповідність числовий, відкритий справа, інтервал $[0, 10)$, а кожен символ алфавіту зіставляється з різними ділянками цієї числової осі. Ширина інтервалу (діапазон) кожної ділянки залежить від імовірності (частоти) появи символу в повідомленні.

Арифметичне кодування, як правило, перевершує алгоритм Хаффмана за ефективністю стиснення, дозволяє стискати дані з ентропією, меншою 1 біта на кодований символ [3].

Алгоритм PPM

PPM (англ. Prediction by Partial Matching - передбачення щодо часткового збігу) - адаптивний статистичний алгоритм стиснення даних без втрат, що базується на контекстному моделюванні та передбаченні. Модель PPM використовує контекст - множину символів у нестиснутому потоці, що передують даному, щоб передбачати значення символу на основі статистичних даних. Сама модель PPM лише пророкує значення символу, безпосереднє стиснення здійснюється за алгоритмами ентропійного кодування, наприклад, алгоритмом Хаффмана або арифметичного кодування. Опубліковані дослідження алгоритмів сімейства PPM з'явилися в середині 1980-х років. Але до 1990-х років програмні реалізації не були популярні, тому що моделі PPM вимагають значної кількості оперативної пам'яті та обчислювальної потужності. Сучасні реалізації PPM є одними з кращих серед алгоритмів стиснення без втрат для текстів на природній мові [21].

Алгоритм BWT

Алгоритм Burrows-Wheeler Transform (BWT) є потужним методом для перетворення рядків, що використовується при стисненні даних. Принцип його роботи полягає у трансформації вхідного рядка в таку форму, яка полегшує подальше стиснення. Перетворення не стискає дані безпосередньо, але створює рядок, який добре підходить для подальшого стиснення методами Run-Length Encoding (RLE) або Move-to-Front Transform (MTF). Алгоритм BWT змінює структуру вхідного рядка таким чином, що однакові символи групуються разом. Це підвищує ефективність подальшого стиснення. Серед недоліків алгоритму відзначають підвищену складність у порівнянні з іншими методами стиснення, великі вимоги до пам'яті - для великих файлів потрібно більше пам'яті для зберігання зсувів під час виконання алгоритму.

BWT використовується для стиснення текстових даних (наприклад, у форматах bzip2, bz2), а також у спеціалізованих алгоритмах, що обробляють послідовності символів з великими повтореннями [22].

У табл.1.1 наведено порівняння основних характеристики розглянутих алгоритмів стиснення без втрат за кількома критеріями, що дозволяє зрозуміти, який метод буде найкращим залежно від типу даних та вимог до швидкості і ефективності.

Таблиця 1.1

Порівняння алгоритмів стиснення без втрат					
Алгоритм	Основний принцип	Ефективність стиснення	Швидкість стиснення	Складність реалізації	Приклади використання
Huffman coding	Частотне кодування символів	Низька для великих файлів із різними символами	Висока	Низька	Текстові файли, протоколи передачі даних
Arithmetic coding	Кодування на основі ймовірностей	Висока для текстів із повторюваним и структурами	Середня	Висока	Архіватори, стиснення відео (без втрат)
LZW (Lempel-Ziv-Welch)	Побудова словника повторюваних фрагментів	Середня	Висока	Низька	GIF, TIFF
LZ77	Пошук повторюваних блоків у вікні	Середня	Висока	Середня	ZIP, GZIP
PPM (Prediction by Partial Matching)	Прогнозування на основі контексту	Висока для тексту	Низька	Висока	Архіватори (WinRAR, текстові дані)
Burrows-Wheeler Transform (BWT)	Перетворення послідовностей символів для ефективного кодування	Висока	Середня	Середня	BZIP2, BZ2, стиснення баз даних
DEFLATE	Комбінація LZ77 і Huffman coding	Середня-висока	Висока	Середня	ZIP, PNG, HTTP
Алгоритм Шеннона-	Алгоритм кодування на	Середня	Висока	Низька	Стиснення текстів,

Алгоритм	Основний принцип	Ефективність стиснення	Швидкість стиснення	Складність реалізації	Приклади використання
Фано	основі статистики символів				використовується в деяких форматах
Метод стиснення RLE	Алгоритм, що стискає повторювані елементи послідовності	Низька для даних із високою варіативністю	Висока	Низька	BMP, TIFF, деякі формати зображень
Алгоритм JBIG	Алгоритм стиснення для бінарних зображень	Висока для бінарних зображень	Низька	Середня	Стиснення сканованих документів, факсимільних зображень
Алгоритм Lossless JPEG	Стиснення зображень без втрат на основі трансформації ДКК (Discrete Cosine Transform)	Середня-висока	Середня	Середня	Зображення (PNG, JPEG 2000)

Підсумовуючи, варто зазначити, що алгоритми стиснення без втрат забезпечують точне відновлення оригінальних даних після декомпресії. Вони широко використовуються для текстових, програмних та інших типів даних, де важлива точність.

1.2.2. Алгоритми методів стиснення з втратами

Стиснення з втратами - це один з основних методів стиснення, що значно зменшує розмір файлу за рахунок видалення некритичних деталей, зміни точності представлення кольорів. Цей втратний метод стиснення дозволяє досягнути великого зменшення розміру файлу, але це може призвести до помітної втрати якості, особливо при високому ступені стиснення. Тому при виборі рівня стиснення важливо знайти баланс між розміром файлу та збереженням достатньої якості. При роботі з втратними форматами важливо зберігати оригінальну копію зображення, оскільки втрачена якість не може бути повністю відновлена після стиснення.

Популярними форматами стиснення з втратами є: алгоритми JPEG, JPEG2000, WebP, рекурсивний (хвильовий) алгоритм стиснення зображень (WIC), Fractal Compression.

Алгоритм JPEG

Алгоритм JPEG (Joint Photographic Expert Group - об'єднана експертна група в галузі фотографії) - один з найпоширеніших і досить потужних алгоритмів для стиснення зображень з втратами. Практично він є стандартом де-факто для повнокольорових зображень. Оперує алгоритм областями 8×8 , 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 [26].

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 (ДКП), що застосовується до матриці зображення для отримання деякої нової матриці коефіцієнтів.

Етапи стиснення за алгоритмом JPEG представлені на (рис. 1. 3):

При виконанні першого етапу виконується перетворення кольорового зображення з системи представлення RGB в іншу систему, наприклад YUV, де Y – сигнал яскравості, U і V – різницеві колірні сигнали.



Рис.1.2. Етапи стиснення цифрових зображень за алгоритмом JPEG (складено за [26])

Перетворення виконується від пікселя до пікселя за такими формулами:

$$Y = 0.3R + 0.59G + 0.11B; U = -0.15R - 0.29G + 0.44B; V = 0.62R - 0.52G - 0.1B,$$

де R , G , B – кольорні сигнали зображення (червоний, зелений і синій відповідно).

Представлення кольорового зображення в системі YUV дає можливість використати особливості зорового сприйняття зображень людиною – низьку чутливість до точності представлення кольорів. Наприклад, усереднення значень відліків в межах примикаючих квадратів з розмірами сторін 2×2 , зменшує розмір компонент U і V в чотири рази без помітного зменшення якості зображення. Враховуючи те, що на кожний піксель витрачається три байти (YUV), виграш є значним.

Алгоритм JPEG дозволяє стискати зображення в 10-15 разів без помітних візуальних втрат [26]. Приклад реалізації алгоритму JPEG на мові Python в середовищі Google Colaboratory наведено у п.3.3. цієї роботи (додаток Ж).

Алгоритм JPEG2000.

У 2000 році тією ж групою експертів в області фотографії, що розробляла JPEG було представлено новий, гнучкіший і потужніший стандарт. JPEG 2000 підтримує стиснення з втратами та без втрат, що базується на хвильовому перетворенні. Основні відмінності між алгоритмами JPEG 2000 та JPEG полягають в наступному:

1. Краща якість зображення при значній величині стиснення.
2. Основний алгоритм стиснення замінений на Wavelet. Плавний прояв зображення тепер закладений в алгоритм.
3. Для підвищення міри стиснення використовується арифметичне стиснення, на відміну від менш ефективного стиснення Хаффмана.
4. Підтримка стиснення без втрат - крім звичного стиснення з втратами JPEG2000 підтримує і стиснення без втрат.
5. Підтримка стиснення однобітових (2-кольорових) зображень. Для збереження однобітових зображень (малюнки тушшю, відсканований текст) раніше усюди рекомендувався формат GIF, оскільки стиснення з використанням ДКП дуже неефективне до зображень з різкими переходами кольорів.
6. На рівні формату підтримується прозорість. Плавно накладати фон при створенні веб-сторінок тепер можна не лише в GIF, але і в JPEG 2000 [27].

Рекурсивний (хвильовий) алгоритм стиснення зображень

Алгоритм хвильового стиснення WIC (Wavelet Image Compression) орієнтований на стиснення кольорових і чорно-білих зображень з плавними переходами, ідеальний для рентгенівських фотографій. Коефіцієнт стиснення варіюється в межах 5-100 [28].

Рекурсивне стиснення базується на пірамідальному S-перетворенні, яке може використовуватись для стиснення фотореалістичних зображень як майже без втрат, так і з втратами. Стиснення виконується за два кроки:

- 1) S-перетворення початкового зображення;

2) перетворені дані кодуються одним з статистичних методів.

Обидві операції зворотні, що дозволяє відновити початкове зображення. Однак для отримання великих коефіцієнтів стиснення необхідно зниження точності представлення компонент зображення, отриманих в результаті виконання S-перетворення, але так щоб спотворення не були візуально помітні.

Структура кодування приведена на рис. 1.3. Вхідні дані обробляються фільтрами S-перетворення, які формують компоненти зображення.



Рис. 1.3. Схема хвильового кодування [28]

Адаптивний квантувач призначений для квантування значень відліків компонент з урахуванням особливостей зорового сприйняття. Процес виконання квантування пов'язаний з втратами інформації, однак ці втрати повинні бути непомітними. Тобто візуально якість відновленого зображення не повинна відрізнятися від початкового. Власне стиснення може бути виконано на основі одного з методів статистичного кодування (наприклад, кодування Хаффмана, LZW-кодування, арифметичного кодування).

Перевагами цього алгоритму є можливість поступового відображення картини при передачі її по мережі. Крім того, оскільки на початку зображення ми фактично зберігаємо його зменшену копію, що спрощує його показ по заголовку. На відміну від JPEG і фрактального алгоритмів цей метод оперує блоками, 2x2, 4x4, 8x8 і так далі. Проте за рахунок того, що коефіцієнти для цих блоків зберігаються незалежно, є можливість досить легко уникнути дроблення зображення на “мозаїчні” квадрати [28].

Алгоритм стиснення WebP (Google)

WebP - це сучасний формат стиснення зображень, розроблений Google, який підтримує як стиснення з втратами, так і без втрат. Формат був створений для того, щоб забезпечити високу якість зображень при меншому розмірі файлу, порівняно з іншими форматами, такими як JPEG, PNG або GIF [29].

Ключові характеристики WebP:

1) Типи стиснення:

- Стиснення з втратами: використовується для зменшення розміру файлів шляхом видалення непомітної для людини інформації.
- Стиснення без втрат: забезпечує збереження повної якості зображення, зменшуючи розмір за рахунок оптимізації і видалення надлишкових даних.

2) Алгоритм стиснення:

- Для стиснення з втратами WebP використовує прогнозування руху і кодування на основі блоків (predictive coding).
- Стиснення без втрат в WebP базується на методі, аналогічному до стиснення PNG, але більш ефективному.

3) **Переваги:** може забезпечити розмір файлу на 25-34% менший, ніж у JPEG з подібною якістю; підтримує прозорість, що робить його хорошим вибором для зображень з прозорими ділянками, як PNG; підтримує анімацію, аналогічно GIF, але з більш ефективним стисненням.

4) **Недоліки:** незважаючи на підтримку багатьма сучасними браузерами (Google Chrome, Firefox, Edge, Opera), WebP не підтримується старими браузерами та деякими програмами.

Формат WebP використовується на вебсайтах, де важлива оптимізація швидкості завантаження та економія пропускної здатності. Підходить для зображень з високою якістю при малому розмірі, таких як фотографії та графіка [30].

Алгоритм Vector Quantization (VQ)

Vector Quantization (векторне квантування) - це метод кодування даних, який широко використовується для стиснення зображень, аудіо, відео та інших типів сигналів. Основна ідея VQ полягає в тому, щоб розбити дані на вектори (або блоки) і замінити кожен вектор одним з найкращих представників (центроїдів) з попередньо визначеного набору. Це дозволяє зменшити кількість даних, зберігаючи при цьому основну інформацію [36].

Алгоритм VQ є ефективним для зображень з невеликою кількістю різних кольорів або текстур. Оскільки дані замінюються на індекси з кодової книги, що дозволяє значно зменшити обсяг інформації без значної втрати якості. Заміна великих векторів на малі індекси значно спрощує передачу даних, що важливо для мультимедійних застосувань. Застосовується у форматах JPEG, PGM, VQ-encoded images. Недоліками є великі витрати на побудову кодової книги – цей процес може бути обчислювально складним і потребує багато ресурсів для великих наборів даних. Якість стиснення значною мірою залежить від розміру кодової книги. Надто мала кількість центрів може призвести до значної втрати якості, а надто велика — до меншого стиснення [22].

Фрактальний метод стиснення (Fractal Compression)

Одним із перспективних методів кодування є стиснення зображень з використанням фрактальної графіки. “Фрактальне кодування - це математичний процес для кодування растрів, які містять реальне зображення, в сукупність математичних даних, що описують фрактальні властивості зображення” [31].

Фрактал - це структура, яка складається з подібних форм і малюнків, що зустрічаються в різних розмірах. Цей термін уперше застосував Бенуа Мандельброт (Benoit Mandelbrot) для опису малюнків, що повторювались, які він спостерігав у різних проявах. Мандельброт також виявив, що фрактали можна описувати математично і створювати за допомогою простих алгоритмів та незначної кількості даних [32].

Математична модель, яка використовується при фрактальному стисненні зображень, називається системою ітерованих функцій (Iterated Function System, IFS), які описують самоподібність областей зображення. У роботі Майкла Барнслі [33] зазначено, що “IFS - це набір тривимірних афінних перетворень, що переводять одне зображення в інше. Перетворенню піддаються точки в тривимірному просторі (x координата, y координата, яскравість)”.

Залежно від розподільної здатності і змісту вхідних растрових даних, якості зображення і розміру файлу, процес стиснення одного зображення може зайняти від декількох секунд до декількох годин (і більше) навіть на дуже швидкодіючому комп'ютері. Декодування фрактального зображення є набагато простішим процесом, оскільки уся трудомістка робота була виконана при пошуку усіх фракталів під час кодування. В процесі декодування треба лише інтерпретувати фрактальні коди, перетворивши їх в растрове зображення [31].

Алгоритми фрактального стиснення є найбільш перспективними на сьогоднішній день, оскільки вони забезпечують коефіцієнт стиснення в межах 2-2000. Крім того, при розархівуванні зображення можна виконувати масштабування. Алгоритм орієнтований на повноколірні зображення та зображення в градаціях сірого кольору [34].

Підсумовуючи, варто зазначити, що алгоритми стиснення з втратами жертвують частину точності відновлення оригінальних даних після декомпресії для досягнення більшого коефіцієнта стиснення. Ці алгоритми зазвичай використовуються для стиснення мультимедіа (зображень, аудіо, відео).

У табл.1.2 наведено порівняння основних характеристики розглянутих алгоритмів стиснення з втрат за кількома критеріями, що дозволяє зрозуміти, який алгоритм буде найкращим залежно від типу даних та вимог до швидкості і ефективності. З таблиці видно, що перевагами алгоритмів стиснення з втратами є висока ефективність стиснення, особливо для мультимедійних файлів. Це зумовлює їх застосування при передачі даних через мережу, для зберігання великих обсягів

інформації. Водночас, основним недоліком є невідворотна втрата даних, залежно від рівня стиснення.

Таблиця 1.2

Порівняння алгоритмів стиснення з втратами					
Алгоритм	Основний принцип	Ефективність стиснення	Якість після декомпресії	Складність реалізації	Приклади використання
JPEG	Перетворення ДКП, квантування	Висока	Залежить від налаштувань	Середня	Фотографії, веб-зображення
JPEG 2000	Хвильове перетворення	Дуже висока	Краще за JPEG	Висока	Медичні зображення, архіви
WebP (Google)	Хвильове перетворення	Дуже висока	Висока	Висока	Веб-графіка, зображення
Fractal Compression	Самоподібність у зображеннях	Висока	Висока	Дуже висока	Зображення з повторюваними структурами
Vector Quantization (VQ)	Поділ простору сигналів на вектори для кодування	Висока	Залежить від методу квантування	Середня	Стиснення зображень, аудіо, відео
Wavelet Image Compression (WIC)	Використання хвильового перетворення для багаторівневого стиснення	Дуже висока	Дуже висока	Висока	Стиснення медичних, наукових та цифрових зображень

Отже, проведений огляд технологій та методів стиснення свідчить, що при виборі між методами стиснення з втратами та без втрат необхідно враховувати можливість компромісу між ступенем стиснення та якістю.

Методи з втратами можуть досягти вищих ступенів стиснення, але ціною нижчої якості. Методи без втрат можуть підтримувати високу якість, але ціною менших ступенів стиснення. Ще один компроміс між складністю та продуктивністю. Методи з втратами можуть бути простішими та швидшими для реалізації та виконання, але їх також може бути складніше розпакувати та відновити. Методи без втрат можуть бути складнішими та повільнішими для

реалізації та виконання, але їх також можна легше розпакувати та відновити. Загальне порівняння двох категорій методів стиснення наведені у табл. 1.3.

При виборі методу стиснення необхідно враховувати такі фактори як тип даних, характеристики споживачів цих даних, мета створення та використання даних. Різні типи даних, а це може бути текст, зображення, двійковий файл, можуть мати різні характеристики та вимоги до стиснення. Різні споживачі контенту можуть мати різні очікування та стандарти якості.

Таблиця 1.3

Порівняння методів стиснення

Характеристика методів стиснення	Стиснення без втрат	Стиснення з втратами
Точність відновлення	100%	Часткова (залежить від алгоритму)
Ефективність стиснення	Середня	Дуже висока
Використання ресурсів	Нижча	Вища
Застосування	Текст, програми	Фото, відео, аудіо
Приклади форматів	ZIP, PNG, GIF	JPEG

Тому, щоб забезпечити найефективніше стиснення, важливо правильно вибрати алгоритм стиснення, який найкраще підходить для конкретного типу вхідних даних, з урахуванням всіх вимог та компромісів.

Порівняння методів стиснення

Таблиця 1.3

Характеристика методів стиснення	Стиснення без втрат	Стиснення з втратами
Точність відновлення	100%	Часткова (залежить від алгоритму)
Ефективність стиснення	Середня	Дуже висока
Використання ресурсів	Нижча	Вища
Застосування	Текст, програми	Фото, відео, аудіо
Приклади форматів	ZIP, PNG, GIF	JPEG

Тому, щоб забезпечити найефективніше стиснення, важливо правильно вибрати алгоритм стиснення, який найкраще підходить для конкретного типу вхідних даних, з урахуванням всіх вимог та компромісів.

РОЗДІЛ 2. ФОРМАТИ КОМПРЕСІЇ ЗОБРАЖЕНЬ

Зображення найвищої якості починається з файлу RAW (цифровий негатив), що містить усі нестиснені й необроблені дані зображення, отримані сенсорами цифрової камери чи сканера. Зйомка у форматі RAW забезпечує якісне зображення з високим рівнем деталізації, але файли виходять великого розміру. Тому дані файлу RAW необхідно спочатку імпортувати до відповідного програмного забезпечення (Photoshop, Adobe Lightroom, Capture One), а вже потім редагувати, конвертувати та стискати без жодних втрат, або експортувати в інший формат растрового зображення, наприклад JPEG.

2.1. Особливості графічних зображень та актуальність їх стиснення

Зображення є структурованим відображенням інформації у вигляді візуальних елементів, яке характеризується надлишковістю. Ця надлишковість обумовлена кореляцією між сусідніми елементами (пікселями) або іншими властивостями, такими як повторюваність кольорів, текстур чи форм, що дозволяє застосовувати алгоритми стиснення для зменшення розміру без значної втрати якості [35]. “Зображення — це цифрове представлення інформації у вигляді двовимірного масиву, який можна обробляти, зберігати або передавати за допомогою комп'ютерних технологій” [36].

Це визначення враховує математичний і технічний аспекти зображень, що важливо в контексті обробки даних та стиснення.

Зображення характеризується трьома особливостями:

- 1) потребує більших обсягів пам'яті, ніж текст, що визначає актуальність алгоритмів архівації даних;
- 2) при аналізі зображення людський зір оперує контурами, загальним переходом кольорів і порівняно нечутливий до незначних змін у зображенні. Ця особливість людського зору дозволила створити спеціальні алгоритми стиснення, які забезпечують високий ступінь стиснення, при цьому декомпресоване зображення не співпадатиме з оригіналом, проте людське око цього не помічає.

З) зображення, на відміну від тексту, володіє надмірністю в 2-х вимірах. Зокрема, сусідні крапки, як по горизонталі, так і по вертикалі, в зображенні близькі за кольором. Крім того, використовується подібність між кольорними площинами R, G і B, що дає можливість створити ще більш ефективні алгоритми [37].

Залежно від способу побудови та кодування, графічне зображення може бути растровим (піксельним), векторним (описаним математичними формулами), **фрактальним та тривимірним**.

Растрові графічні зображення складаються з пікселів (англ. «picture element» - елемент малюнку), які мають свій колір, яскравість та є елементами загальної композиції. Растрові зображення можуть передати складні кольорові градієнти та деталі, що робить їх ідеальними для фотореалістичних форматів зображень та фотографій. Ці зображення легко виводяться на монітор чи принтер, оскільки в основі їх лежить саме растровий принцип.

Усі растрові файли можуть бути збережені в одній із кольірних моделей (палітрі): RGB, CMY, HSB, CIE Lab, або 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 [5].

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.

Модель CMY (Cyan-Magenta-Yellow – блакитний-пурпурний-жовтий) призначена для отримання зображень на білій поверхні. Блакитний, пурпурний і жовтий кольори називають доповнюючими, тому що вони доповнюють основні кольори до білого. Накладання один на одного доповнюючих кольорів на практиці не дає чистого чорного кольору, тому в модель включають четвертий компонент чистого чорного кольору (black – чорний). Така модель має аббревіатуру CMYK.

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: чистим вважається відтінок, який не містить білого; частково насичений відтінок світліший. Яскравість визначає інтенсивність світіння кольору – відтінки з високою інтенсивністю дуже яскраві, а з низькою – темні.

У моделі CIE Lab будь-який колір визначається світлотою (L) і хроматичними компонентами: параметром а, 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.

Водночас при збільшенні розміру растрового зображення його якість погіршується, з'являються пікселі та розмиття, тобто втрачається якість при масштабуванні. Для зображень з високою роздільною здатністю характерні великі розміри файлів, робота з якими потребує значних обсягів пам'яті та потужних обчислювальних ресурсів, у тому числі для їх зберігання та передачі.

Тому актуальним є питання стиснення графічних файлів. Але стиск растрових зображень є особливим і залежить від типу формату графічного файлу та алгоритму, який в ньому реалізовано.

Векторна графіка – це вид комп'ютерної графіки, в якому зображення подається у вигляді сукупності окремих об'єктів, описаних математично. Принципи векторної графіки базуються на, відмінному від піксельної графіки, математичному апараті і забезпечують побудову лінійних контурів, за допомогою сплайнових кривих: NURBS-кривих та кривих Без'є [38].

Векторні зображення формуються за допомогою математичних рівнянь, що не вимагає значних обчислювальних ресурсів та забезпечує практично миттєве відображення рисунків. Векторні зображення можна збільшувати та зменшувати

без втрати якості, оскільки вони визначаються математично. Можна переносити елементи, змінювати їх розмір, форму і колір, але це не відіб'ється на якості їх віртуального представлення. Векторна графіка не залежить від роздільної здатності, тобто може бути представлена на різноманітних вихідних пристосуваннях без втрати якості. Це робить їх ідеальними для логотипів та графіків. Векторні формати зображень зазвичай займають менше місця на диску в порівнянні з растровими, що сприяє використанню саме цього формату для зберігання зображень [37].

Водночас, для зображень з великою кількістю деталей і колірних градієнтів векторний формат може бути не найкращим — якість може бути нижчою, ніж у растрових, розмір отриманого файла інформації також може виявитися набагато більшим, ніж відповідне растрове зображення. Крім того робота з векторними картинками може бути складнішою, ніж з растровими і потребувати спеціальних навичок та програмного забезпечення.

Щодо фрактальних зображень, то вони не належать безпосередньо до растрових чи векторних форматів у традиційному сенсі. Вони є окремим видом графічних зображень, заснованих на математичних формулах. Фрактальний підхід унікальний тим, що дозволяє створювати дуже деталізовані та масштабовані зображення. Фрактальні зображення при візуалізації можуть бути растровими, якщо зображення представлено у вигляді пікселів, або векторними, якщо описуються геометричними примітивами.

2.2. Експериментальне порівняння форматів компресії зображень

2.2.1. Формати растрових зображень

Один із найбільш поширених форматів для зберігання растрових зображень є формат **JPEG** (Joint Photographic Expert Group), який *реалізує метод стиснення з втратами*, що дозволяє суттєво зменшити розмір графічних файлів без помітної втрати якості зображення для ока людини.

Область застосування формату досить вузька – розповсюдження високоякісної напівтонової графіки в Інтернеті. Формат підтримується практично всіма сучасними графічними програмами та веб-браузерами.

Файли стиснених растрових зображень мають розширення JPEG або JPG. Позитивним є те, що цей формат підтримується всіма браузерами, редакторами (Adobe Photoshop, GIMP, Microsoft Paint, Lightroom) та пристроями. Як недолік цього формату, виділяють неможливість зберігати зображення з прозорим фоном, не підтримується робота з шарами - JPEG - це плоский формат.

JPEG підтримує 16 мільйонів кольорів, що робить його ідеальним для фотографій, він також використовується для збереження графіки для електронної пошти та великих зображень. Зображення, збережене у форматі JPG, можна зменшити до дуже маленьких розмірів та надіслати електронною поштою для узгодження з клієнтом. Файли JPG з високою роздільною здатністю та мінімальним стисненням підходять для редагування та подальшого друку [39].

Растрові графічні зображення зберігаються також у форматі **GIF** (Graphics Interchange Format), який використовується у веб-дизайні та анімації. У цьому форматі реалізовано алгоритм стиснення без втрат LZW (Lempel-Ziv-Welch), який працює за принципом словникового кодування, де замість повторюваних шаблонів даних записуються їх короткі коди [22]. GIF підтримує прозорість та 256 кольорів, яких може не вистачати для складних зображень та повнокольорових фотографій. Не підтримується також робота з шарами. Підходить для створення простої графіки, логотипів. GIF-файли містять усі кадри та часові дані в одному документі [38].

Альтернативою цього формату є **PNG** (для статичних зображень) і **WebP** (для анімацій), які підтримують набагато більшу палітру кольорів і забезпечують високу якість.

З форматом **GIF** працюють Adobe Photoshop, GIMP, онлайн-інструменти для створення анімацій (наприклад, EZGIF).

Формат **PNG** (Portable Network Graphics), розроблений як більш просунута альтернатива GIF. Зображення PNG підтримує прозорість (альфа-канал), і може відображати більш високу глибину кольору, передаючи мільйон відтінків. Цей формат давно став веб-стандартом і одним із найпоширеніших в Інтернеті. Добре підходить для зображень із текстом, логотипами та графікою, зберігає оригінальну якість зображення. Але не підтримує роботу з шарами. Також PNG найкраще оптимізовано для екранів, він не підійде для друкованих проектів та фотографій [40].

У форматі **PNG** використовується алгоритм стиснення без втрат **Deflate**, який поєднує словникове кодування **LZ77** для пошуку повторюваних фрагментів даних та кодування **Хаффмана** для кодування символів змінною кількістю бітів на основі їхньої частоти [3]. З цим форматом працюють програми Adobe Photoshop, GIMP, CorelDRAW, Inkscape.

Растровий формат **BMP** (Bitmap Picture) зазвичай не використовує алгоритмів стиснення і зберігає зображення у необробленому виді, у вигляді сітки пікселів. Це забезпечує не лише високу якість зображення, але й великий розмір файлів. У цьому форматі можлива підтримка алгоритму стиснення без втрат RLE (Run-Length Encoding) для зображень з глибиною кольору 8 біт. Формат BMP підтримує прозорість, але не підтримує роботу з шарами. Із зображеннями у форматі BMP просто працювати у графічних редакторах Paint, Adobe Photoshop, GIMP та виводити їх на друк [41].

Растровий формат **TIFF** (Tagged Image File Format) є дуже гнучким форматом, використовується у професійній фотографії та поліграфії. Підтримує багато кольірних просторів і може зберігати зображення в різних режимах стиснення (RLE, LZW, Deflate, JPEG). Розмір файлу TIFF залежить від вибраного методу стиснення. Формат TIFF підтримує прозорість та можливість працювати з шарами. Деякі програми, особливо хмарні, можуть не підтримувати цей формат. З цим форматом працюють програми Adobe Photoshop, GIMP, Adobe Lightroom, CorelDRAW [41].

Растровий формат **TGA** (Truevision Graphics Adapter), також відомий як **TARGA**, призначений для використання в графічних додатках і підтримує високоякісні зображення, включаючи зображення з альфа-каналом для прозорості. TGA залишається популярним у середовищах комп'ютерної графіки, особливо в ігровій індустрії. Підтримує палітри кольорів RGB і RGBA (з альфа-каналом) та глибину кольору 8, 16, 24 або 32 біт на піксель, що забезпечує високоякісні зображення. Для базового стиснення використовується метод RLE (Run-Length Encoding), можливе збереження зображення без стиснення (raw image), що формує файли великих розмірів. Використовується при обміні графічними файлами в середовищі художників та дизайнерів, при створенні ігрової графіки (текстури, карти нормалей) [42].

Отже, растрові зображення можуть зберігатися у форматах BMP, JPEG, PNG, GIF, TIFF, TGA, WEBP, RAW. Ці формати відрізняються за такими властивостями, як ступінь стиснення, прозорість, підтримка кольорів тощо.

Є також растрові формати PSD (Photoshop Document), ICO (Icon Windows), PCX (один із перших растрових форматів), HEIC (High Efficiency Image Format), EXR (OpenEXR), які не представлені у даному розділі роботи, у зв'язку з обмеженнями по кількості сторінок.

На рис. 2.1. представлено результати опрацювання графічних растрових зображень за допомогою графічного редактора **GIMP** (The GNU Image Manipulation Program) на прикладі файлів ukraine.jpg, розміром 30.2 Kb та 2025.jpeg, розміром 398.2Kb.

Name	Size	Type
10 raw ukraine.ppm	253.2 KiB	PPM image
10 ascii ukraine.ppm	924.3 KiB	PPM image
9 2025.webp	117.7 KiB	WebP image
9 2025.png	846.7 KiB	PNG image
9 2025.jpeg	398.2 KiB	JPEG image
9 2025.gif	292.8 KiB	GIF image
9 2025.bmp	1.6 MiB	Windows BMP image
8 без стиснен ukraine.tga	253.2 KiB	TGA image
8 RLA ukraine.tga	182.4 KiB	TGA image
7 ukraine.gif	40.2 KiB	GIF image
7 interlace ukraine.gif	42.8 KiB	GIF image
6.2 ukraine.eps	237.7 KiB	EPS image
6.1 ukraine.eps	239.1 KiB	EPS image
5 без стисн ukraine.tif	257.7 KiB	TIFF image
5 LZW ukraine.tif	127.9 KiB	TIFF image
4 по замовчув ukraine.png	80.1 KiB	PNG image
4 16bpc graya ukraine.png	94.8 KiB	PNG image
4 8bpc graya ukraine.png	37.7 KiB	PNG image
3 ukraine.jpg	29.6 KiB	JPEG image
3 jpeg ukraine.jpg	30.2 KiB	JPEG image
2 thumbnail ukraine.webp	21.6 KiB	WebP image
2 lossy ukraine.webp	13.9 KiB	WebP image
2 lossless ukraine.webp	64.2 KiB	WebP image
1 compatib ukraine.bmp	253.3 KiB	Windows BMP image
1 adv 32 x8r8g8b8 ukraine.bmp	337.6 KiB	Windows BMP image
1 adv 24 r8g8b8 ukraine.bmp	253.3 KiB	Windows BMP image
1 16 r5g5b5 ukraine.bmp	168.9 KiB	Windows BMP image

"1 16 r5g5b5 ukraine.bmp" | 168.9 KiB (172,938 bytes) | windows BMP i...

Рис. 2.1. Результати експорту файлів ukraine.jpg та 2025.jpeg у формати BMP, GIF, PNG, TIFF, EPS, TGA та WEBP

Файл ukraine.jpg був експортований у формати BMP, GIF, PNG, TIFF, EPS, TGA та WEBP з різною кольоровою палітрою (16bit R5G5B5, 24bit R8G8B8, 32bit X8R8G8B8), без стиснення, з стисненням без втрат (Lossless) та з втратами (Lossy), у формі мініатюри (табл. 2.1).

Результати експорту файлу ukraine.jpg

Формат	Розмір, Kb	Примітка
JPG	30.2	
BMP	168.9	16bit R5G5B5 Без стиснення
	253.3	24bit R8G8B8 Без стиснення
	337.6	32bit X8R8G8B8 Без стиснення
WEBP	13.9	lossly
	64.2	Lossless
	21.6	Thumbnail мініатюра
TGA	182.4	Стиснення RLA
	253.2	Без стиснення
PNG	37.7	8BPC GRAYA
	94.8	16BPC GRAYA
	80.1	По замовчуванню
TIFF	127.9	LZW-кодування (Lossless)
	257.7	Без стиснення
EPS	239.1	
	237.7	
GIF	40.2	
	42.8	

У додатку В представлено вміст цих файлів, який свідчить про майже непомітні для людського ока відмінності, крім файлу у форматі PNG з палітрою 16bpc graya — чорно-біле зображення.

Але вміст табл. 2.2 свідчить про ефективність використання форматів WEBP, JPG, PNG. Найменший розмір мають файли у сучасному форматі WEBP (13.9Kb, зі стисненням), який також забезпечує якісне зображення.

Файли TIFF та TGA є великими за розміром, навіть при застосуванні алгоритму стиснення LZW (127.9 Kb) та при стисненні алгоритмом RLA (182.4Kb)

При цьому найбільший розмір мають файли у форматі BMP, TGA та EPS.

При стисненні файлу `ukraine.bmp` було задано декілька варіантів у колірній моделі RGB в бітності 16, 24 і 32 та враховувалася можливість підтримки прозорості (альфа-канал). За результатами роботи з BMP форматом були зроблені наступні висновки: якщо не потрібно зберігати прозорість і достатньо кольорів з обмеженою палітрою (16 біт), то колірної моделі **R5G5B5** буде достатньо. У разі, якщо необхідна обробка прозорості (при обмеженій палітрі 16 біт), то колірна модель **A1R5G5B5** може бути кращим варіантом. Якщо потрібно використовувати додатковий біт для маркування або спеціальних ефектів, то вибір колірної моделі **X1R5G5B5** може бути достатнім.

Залежно від вимог до якості кольору та прозорості, вибір між цими форматами може змінюватися.

Отже, отримані результати роботи з файлом `ukraine.jpg` свідчать, що найменшим є файл у форматі WEBP (13.9Kb).

Найбільшими є BMP-файли (168.9Kb, 253.3Kb, 337.6Kb), вони займають значно більше місця порівняно з WEBP та іншими сучасними форматами.

Це показує, як ефективність алгоритмів стиснення впливає на розмір файлів, які представлені у різних форматах.

У формати BMP, PNG, WEBP, GIF також був експортований файл 2025.JPEG (рис.2.1). Результати наведені у табл. 2.2.

Один із найменших файлів за розміром є файл 20 25.WEBP — 117.7Kb.

Файл 2025.PNG (846.7Kb) є значно більшим за WEBP через відсутність ефективного стиснення.

Таблиця 2.2

Результати експорту файлу 2025.JPEG

Формат	Розмір	Примітка
JPEG	398.2 Kb	
BMP	1.6 Mb	Найбільший файл через відсутність стиснення
WEBP	117.7 Kb	lossy
PNG	846.7 Kb	Без стиснення
GIF	292.8 Kb	

Файл 2025.GIF (292.8Kb) є меншим за PNG, але більший за WEBP. Крім того, якість зображення у форматі GIF виявилася найгіршою серед всіх інших форматів (додаток А).

Найбільшим за розміром є файл 2025.BMP (1.6Mb) через відсутність стиснення.

Узагальнюючи всю інформацію про растрові формати, варто зазначити, що найбільш ефективними є формати WEBP, JPEG та PNG.

2.2.2. Формати векторних зображень

Проведений аналіз показав, що для векторних зображень найпоширенішими форматами є: **AI** (Adobe Illustrator), **SVG** (Scalable Vector Graphics), **EPS** (Encapsulated PostScript), **CDR** (CorelDRAW), **DXF** (Drawing Exchange Format), **WMF** (Windows Metafile), **PDF** (Portable Document Format).

AI (Adobe Illustrator) – це власний формат Adobe Illustrator, однієї з найпопулярніших програм для створення векторної графіки (Логотипи, інфографіка, брендові матеріали). AI є спеціалізованим форматом для векторної графіки, який зберігає всі шари, шрифти, кольори та інші налаштування дизайну.

AI-файли можуть містити складні ілюстрації та детальні дизайни, які можуть редагуватися у подальшій роботі над зображеннями [43].

SVG (Scalable Vector Graphics) - є одним з найпопулярніших форматів векторної графіки. Використовується при розробці веб-дизайну, логотипів, іконок. Цей формат підтримується більшістю веб-браузерів та дозволяє інтегрувати векторні зображення безпосередньо у HTML-код. SVG використовує XML-формат для опису графіки, що дозволяє зображенням бути масштабованими без втрати якості.

Перевагами SVG формату є масштабування зображення без втрати якості; можливість редагування за допомогою текстових редакторів або програм для векторної графіки; підтримка анімації та взаємодії через JavaScript і CSS; легко інтегрується в веб-сайти [44].

EPS (Encapsulated PostScript) - це формат, який часто використовується для передачі векторних зображень між різними програмами, може містити векторну графіку, а також растровані зображення. Часто використовується в професійному друці та графічному дизайні при роботі з логотипами, графіками для публікацій, друкованими матеріалами. Широко підтримується в графічних редакторах та програмних засобах для друку. Може бути вбудований в інші документи, наприклад, у PDF або PostScript. Підтримує кольорові моделі CMYK, що важливо для друкарської продукції [45].

CDR (CorelDRAW Bitmap) – це основний формат векторного графічного редактора CorelDRAW. Формат CDR став універсальним для інших програм завдяки використанню окремої компресії для векторних і растрових зображень, можливості вбудовувати шрифти, величезному робочому полю 45х45 метрів, підтримці багатосторінковості. Широко використовується в поліграфії та дизайні [46].

DXF (Drawing Exchange Format) - формат, розроблений Autodesk для обміну векторною графікою між різними CAD (Computer-Aided Design) програмами.

Використовується в CAD-системах для створення технічних креслень, архітектурних планів [47].

WMF (Windows Metafile) – універсальний формат векторних графічних файлів для Windows-додатків. Використовується для зберігання колекції графічних зображень Microsoft Clip Gallery, які інтегруються в документи MS Office, швидко відображаються на екрані завдяки простій структурі [48].

PDF (Portable Document Format) – цей формат підтримує векторну графіку, але здебільшого використовується для збереження документів. Підтримує комбінацію растрових і векторних елементів. Часто використовується для обміну файлами дизайну, оскільки багато програм можуть читати PDF [49].

При роботі з векторною графікою можуть бути використані наступні програмні засоби:

Adobe Illustrator - одна з найпопулярніших програм для створення векторної графіки. Вона пропонує широкий набір інструментів для професійного дизайну.

CorelDRAW – потужний інструмент для роботи з векторною графікою. Він часто використовується для створення логотипів, ілюстрацій та рекламних матеріалів.

Inkscape - безкоштовна програма з відкритим вихідним кодом для роботи з векторною графікою. Вона пропонує багато функцій, подібних до Adobe Illustrator, та є гарною альтернативою для тих, хто шукає безкоштовні рішення.

Підсумовуючи, варто зазначити, що векторна графіка призначена для створення високоякісних та масштабованих зображень, чіткість та професійний вигляд яких зберігається в будь-яких умовах, від веб-дизайну до друкованих матеріалів.

Вибір між растровим та векторним форматами графічних зображень залежить від конкретних цілей та завдань проєкту.

У табл. 2.3 представлені переваги та недоліки растрових і векторних зображень.

Таблиця 2.3.

Основні характеристики растрових і векторних зображень

Вид графічного зображення	Растровий	Векторний
Переваги	1. Реалістичність зображень. 2. Природність кольорів. 3. Можливість отримання зображень з використанням спеціальних пристроїв	• Невеликі за розміром файли зображень. • Збереження якості після масштабування. • Простота редагування окремих елементів зображення
Недоліки	• Великі за розміром файли зображень. • Пікселізація зображення після збільшення (ефект блочності). • Складність редагування окремих елементів зображення	• Складність реалістичного відтворення об'єктів навколишнього середовища. • Відсутність пристроїв для автоматизованого створення зображення

Якщо потрібні високоякісні фотографії чи деталізовані зображення, краще використовувати растровий формат. Але, якщо потрібен логотип, який повинен залишатися чітким за будь-якого масштабу, то векторна графіка буде оптимальним рішенням. Стиснення зображень балансує між зменшенням розміру файлу та прийнятним рівнем візуальної якості, оскільки значне стиснення може призвести до погіршення якості зображення, а великі файли створюють проблеми ефективного використання трафіку.

РОЗДІЛ 3. ПОРІВНЯЛЬНИЙ АНАЛІЗ УТИЛІТ СТИСНЕННЯ

3.1. Аналітичний огляд систем стиснення

Проведений аналіз свідчить про достатньо велику кількість розроблених утиліт для виконання процедури стиснення та відновлення файлів — brotli [50], gzip [51], bzip2 [52], pbzip2 (паралельний компресор файлів bzip2, v1.1.9) [54], lbzip2 [55], XZ [56] та інші. У роботі [7] надано опис понад 200 утиліт, перелік яких представлено у Додатку Б. Кожна з цих утиліт побудована на основі певних алгоритмів, має свої опції стиснення, якими задаються швидкість та ступінь стиснення (екстремальний, максимальний, мінімальний, типовий) файлів.

Для виконання порівняльного аналізу функціональних можливостей було обрано чотири утиліти, до використання яких є вільний доступ: **bzip2**, **gzip**, **Zstd** (**Zstandard**), **XZ** (**LZMA**).

3.1.1. Програма стиснення файлів bzip2

Програма стиснення файлів **bzip2** - це безкоштовна вільна утиліта командного рядка з відкритим початковим кодом (BSD-подібна ліцензія), що розроблена і підтримується Джуліаном Сьюардом [52]. Утиліта працює в усіх основних операційних системах та сумісна з Windows 2000 і XP і з усіма пізнішими версіями. Для її встановлення використовується файл bzip2-105-x86-win32.exe. Також ця програма працює з даними у розмірах від 100 до 900 Кбайт .

Програма bzip2 - це високоякісний компресор даних, який заснований на алгоритмі стиснення даних без втрат, з використанням перетворення Берроуза-Вілера (Burrows Wheeler Transform, BWT). Програма здійснює перетворення послідовностей символів, які багато разів чергуються, на рядки однакових символів, потім застосовує перетворення MTF (англ. move-to-front), і в кінці - кодування Хаффмана. Ця утиліта працює з даними у розмірах від 100 до 900 Кбайт. Стиснення, як зазначено у посібнику по bzip2 [53], значно краще, ніж досягається

звичайними компресорами на основі LZ77/LZ78, і наближається до продуктивності сімейства статистичних компресорів PPM.

Програма стиснення bzip2 забезпечує значно швидшу декомпресію порівняно зі швидкістю стиснення.

Стиснутий файл, який формується командою **bzip2 original_name file**, має ті ж тимчасові мітки, права доступу, власника, як і оригінальний файл, що дає можливість зберегти ці атрибути при розпакуванні файлів. Програма надає стиснутим файлам розширення .bz2 (original_name.bz2).

Розпаковування стисненого файлу виконується командою: **bzip2 -d file.bz2**. Для розпаковування також можна використовувати альтернативу: **bunzip2**. За замовчуванням **bzip2** і **bunzip2** не перезапишуть існуючі файли. Якщо не потрібно зберігати початковий файл, використовується прапорець **-f**.

При стисканні однією командою кількох файлів спільний файл не буде створюватися - кожен файл буде стиснений окремо. У разі необхідності створення єдиного стиснутого файлу, попередньо за допомогою **tar** створюється загальний файл, який потім стискається **bzip2**.

Утиліта **tar** (Tape Archive) використовується для об'єднання кількох файлів в один архів, але **без стиснення**. Це дозволяє об'єднувати структуру каталогів та файли в один архівний файл для зручного збереження, передачі та відновлення структури директорій та файлів у Unix-подібних системах. Сам по собі **tar** не стискає файли, але він використовується разом з іншими алгоритмами стиснення, такими як **gzip** або **bzip2**, для створення стиснених архівів (наприклад, файли .tar.gz або .tar.bz2) [19].

Для створення стисненого **bzip2** архіву **tar** використовується команда:

tar -cvf - файли | bzip2 -9 > назва_архіва.tar.bz2

Для розпаковування архіву **tar**, стисненого **bzip2** призначена команда:

bzip2 -cd назва_архіву.tar.bz2 | tar -xvf -

Корисними також є команди:

bzcat - розпаковує файли в stdout;

bzip2recover - відновлює дані з пошкоджених файлів **bzip2**.

Більш детальний опис опцій утиліти bzip2 (програма має п'ятнадцять параметрів) може бути отриманий через параметр -h: **bzip2 -h**.

3.1.2. Утиліта стиснення gzip

Ще однією із програм стиснення та відновлення файлів, є утиліта gzip (скорочення від GNU zip), яка використовується в основному в UNIX-системах. Утиліта gzip (version 1.13) зменшує розмір заданого файлу за допомогою алгоритму Deflate, який базується на алгоритмі кодування Лемпеля-Зіва (LZ77) та алгоритмі Хаффмана, і забезпечує стиснення без втрат. Результати стискання залежить від розміру вхідних даних та розподілу у них типових підрядків [51]. Зазвичай, текст у формі початкового коду програм або літературної англійської мови gzip стискає на 60-70%. Загалом, рівень стиснення є набагато кращим за рівень LZW (використано у compress), кодування Хаффмана (використано у pack) та адаптивне кодування Хаффмана (використано у compact).

Для упакування кількох файлів зазвичай їх спочатку архівують в один файл утилітою tar, а потім цей файл стискають gzip. Стиснуті у такий спосіб файли мають подвійне розширення .tar.gz.

Кожен стиснутий файл отримує розширення .gz зі збереженням режиму доступу, власника та часу внесення змін. Для операційних систем MSDOS, OS/2 FAT, Windows NT FAT та Atari типовим розширенням стиснутих файлів є .z. Програма є невибагливою до ресурсів операційної системи і, як варіант, підходить для машин з невеликими потужностями [57].

Утиліта gzip стискає лише звичайні файли. Зокрема, вона ігнорує символічні посилання. Якщо назва стисненого файла задовга для поточної файлової системи, gzip її вкорочує. Типово, gzip зберігає початкову назву файла і час модифікації у зміненому файлі. Ця інформація використовується при розпакуванні файла із

параметром -N. Це є корисним, якщо назву файлу було вкорочено, або коли час модифікації не було збережено під час перенесення файла.

Завдяки налаштуванням користувача, можна спочатку визначити ступінь компресії даних: пакувати швидше з меншою компресією даних або повільніше з можливістю сильного стиснення. Для цього в програмі передбачені відповідні ключі від «1» до «-9», де «1» - найшвидше стиснення, а «-9» - найкраще стиснення. Для роботи з даними за замовчуванням встановлений ключ «-6».

Стиснені файли можна відновити до їх початкової форми за допомогою команди `gzip -d` або `gunzip` або `zcat`.

Команда `gunzip` приймає список файлів з рядка команди і замінює кожен з файлів, що має розширення: `.gz`, `-gz`, `.z`, `-z` або `_z` (без врахування регістру символів), і вміст яких починається з належного контрольного числа, на розпакований файл, назва якого не містить початкового суфікса.

`Gunzip` також розпізнає спеціальні суфікси `.tgz` і `.taz`, як скорочення від `.tar.gz` і `.tar.Z`, відповідно.

Поточна версія `gunzip` може розпаковувати файли, які створено `gzip`, `zip`, `compress` або `rpack`. Визначення формату вхідних даних відбувається автоматично. При використанні перших двох форматів `gunzip` перевіряє 32-бітну контрольну суму CRC (cyclic redundancy check – перевірка циклічної надлишковості). Для `rpack` утиліта `gunzip` перевіряє обсяг нестиснених даних. Стандартним форматом `compress` не передбачено перевірок цілісності.

Команда `zcat` є ідентичним `gunzip -c` та розпаковує або список файлів з рядка команди або дані із стандартного джерела вхідних даних і записує розпаковані дані до стандартного виведення. `zcat` розпаковуватиме файли, які мають належну контрольну суму, незалежно від того, мають вони суфікс назви `.gz` чи ні.

Більш детальний опис опцій утиліти `gzip` можна отримати за допомогою параметра `h`, задавши команду: `gzip -h`.

3.1.3. Утиліта стиснення Zstandard

Утиліта Zstandard (Zstd) - це швидкий алгоритм стиснення без втрат, орієнтований на сценарії стиснення даних у реальному часі. Розроблена Yann Collet (компанія Facebook) для забезпечення швидкого стиснення та декодування даних із високим ступенем ефективності. Остання версія Zstandard v1.5.6 випущена у 2024 році містить покращення продуктивності стиснення на різних рівнях (особливо на рівнях 1 - 4 та 5 -12), оптимізацію стиснення та виправлення помилок [58].

Основними особливостями утиліти Zstandard є:

- висока швидкість роботи: Zstd забезпечує швидке стиснення і декомпресію, що робить його в рази швидшим за інші утиліти, такі як gzip чи bzip2;
- гнучкість: у Zstd є 22 рівні стиснення (0 – 22), що дозволяє налаштовувати компроміс між швидкістю та ступенем стиснення (при низьких рівнях - максимальна швидкість стиснення; при високих рівнях - максимальний ступінь стиснення);
- підтримка багатопотоковості: Zstd може використовувати кілька ядер процесора для прискорення процесу стиснення;
- сумісність: Zstd підтримує контейнери з іншими форматами, такими як gzip, lz4, а також легко інтегрується в різні системи [59].

Для створення стисненої версії файлу *original_name file.txt* виконується команда: `zstd original_name file.txt`, результатом є стиснений файл: *original_name file.txt.zst*.

Для відновлення (декомпресії) оригінального файлу виконується команда:

zstd -d *original_name file.zst*

Для стиснення файлу, наприклад, за 19 рівнем стиснення виконується команда: **zstd** -19 *original_name file.txt*

Для стиснення всього каталогу: **tar** cf - *каталог* | **zstd** > *каталог.tar.zst*

Спочатку створюється архів **tar**, а потім здійснюється стисненням за допомогою **Zstd**.

Утиліта **Zstd** знаходить широке використання для:

1. Швидкого стиснення файлів із мінімальними втратами продуктивності. Перевагою є компроміс між швидкістю та ефективністю стиснення.
2. Архівації великих обсягів даних (логи, бази даних, резервні копії).
3. Стиснення в реальному часі - утиліта оптимізована для роботи на високопродуктивних серверах і дозволяє стискати дані на льоту. Це зручно для обробки потокових даних у режимі реального часу.
4. Оптимізації сховищ - допомагає економити дисковий простір за рахунок зменшення розмірів файлів.
5. Передачі даних: використовується для оптимізації трафіку при передачі стиснених даних по мережі, що зменшує затримки та підвищує швидкість передачі.
6. Стиснення в сучасних системах - утиліта інтегрована у різні файлові системи та програмне забезпечення, наприклад: Linux Kernel, Docker (для стиснення образів контейнерів), Compression в Git (для прискорення роботи Git через стиск файлів).

Zstandard використовується в мережі Facebook для стиснення даних у своїй інфраструктурі. В ОС Linux zstd використовується як вбудований алгоритм стиснення в ядро Linux (Btrfs, SquashFS). У системах контролю версій Git ця утиліта застосовується для оптимізації швидкості зберігання та передачі репозиторіїв [59].

3.1.4. Утиліта для стиснення даних XZ

Утиліта стиснення даних **XZ** (5.6.3 /2024-10-01) базується на алгоритмі LZMA (Lempel-Ziv-Markov chain algorithm), який забезпечує високий ступінь стиснення, високу швидкість стиснення та розпакування даних. При цьому є вибагливим до споживання пам'яті [60].

Основне призначення утиліти **XZ**:

1. Стиснення великих файлів: **XZ** використовується для максимального зменшення розміру файлів, що є корисним для великих архівів, образів систем або резервних копій.
2. Архівування та резервне копіювання: завдяки високому ступеню стиснення **XZ** підходить для архівування великих обсягів даних, коли час стиснення менш важливий за кінцевий розмір файлу.
3. Поширення програмного забезпечення: у багатьох дистрибутивах Linux пакунки стиснуті за допомогою **XZ** (наприклад, .tar.xz).
4. Стиснення текстових та лог-файлів: **XZ** особливо ефективна для текстових даних, де алгоритм LZMA показує відмінні результати.
5. Оптимізація місця на диску: **XZ** застосовується там, де важливо мінімізувати використання дискового простору, наприклад, у вбудованих системах або при зберіганні архівів [60].

Утиліта **XZ**, що базується на алгоритмі LZMA, використовує такі принципи:

Словникове кодування (LZ77) при пошуку та заміні повторюваних послідовностей даних.

Кодування Маркова - для побудови моделей на основі імовірностей наступних символів.

Арифметичне кодування - для ефективного кодування символів у невеликі фракції чисел, що зменшує обсяг стиснутих даних.

Це дозволяє LZMA досягти високого ступеня стиснення, особливо для даних із великою кількістю повторюваних символів [60].

Стиснення файлу виконується команда:

```
xz original_name file.txt.
```

Результатом є стиснутий файл з розширенням .xz:

```
original_name file.txt.xz.
```

Розпакування файлу: `xz -d original_name file.txt.xz,`

де параметр `-d` означає `decompress`.

Для архівування всього каталогу з подальшим стисненням використовується `tar` у зв'язці з XZ: `tar -cJf архів.tar.xz каталог`.

Параметри: `-c` — створити архів; `-J` — використовувати XZ для стиснення; `-f` — вказати ім'я файлу архіву.

Приклад розпакування архіву: `tar -xJf архів.tar.xz`,
де `-x` — означає витягування даних з архіву.

Параметр `-9` визначає максимальне стиснення та пам'ять. Типовим є використання параметру стиснення `-6`. Параметр `-e` (екстремальний) визначає краще стиснення за рахунок часу стиснення.

Перевагами XZ (LZMA) є:

- Високий ступінь стиснення, краще, ніж у `gzip` і `bzip2`.
- Ефективність для текстових даних — добре підходить для логів, коду та текстових файлів.
- Сумісність - підтримується в більшості дистрибутивів Linux і є стандартним додатком у багатьох системах.
- Низьке споживання пам'яті під час декомпресії [56].

Варто додати, що сайт `Kernel.org` нові версії патчів і ядра Linux випускає лише в форматах `tar.xz` і `tar.gz` [61].

До недоліки XZ (LZMA) відносять:

- Повільне стиснення, особливо на високих рівнях.
- Великі вимоги до пам'яті під час стиснення - особливо на максимальних рівнях.
- Повільна обробка бінарних файлів - для них `Zstandard` або `gzip` можуть бути ефективнішими [58].

3.2. Порівняння результатів роботи утиліт стиснення

Для тестування утиліт був використаний архівний файл **linux 5.1-rc5** (ядро Linux — `kernel.org`, реліз 5-го кандидата), попередньо заархівований за допомогою

tar (<https://git.kernel.org/torvalds/t/linux-5.1-rc5.tar.gz>). Розмір файлу **tar** archive становить 832 MB, що потребувало певного часу на його копіювання та перенесення на локальний Робочий стіл персонального комп'ютера.

Для порівняння функціональних можливостей утиліт стиснення **zstd** (**Zstandard**), **gzip**, **bzip2**, **xz** (**LZMA**) було обрано час, швидкість, ступінь, розмір стиснення та час і швидкість відновлення. Для виконання тестування використовувалися скрипти, результати виконання яких наведено у додатках Б, В, Г, Д. **Зведені результати тестування представлені у додатку Є.**

Конфігурація системи, в якій виконувалося тестування алгоритмів стиснення:

- CPU AMD Ryzen 7 PRO 5850U
- RAM 16GB
- SSD 512GB M.2 NVMe
- OS Debian GNU/Linux 12

Аналіз результатів тестування утиліт свідчить, що швидкість декомпресії є значно вищою порівняно зі швидкістю стиснення, крім **zstd**. При цьому **gzip** здійснює відновлення набагато швидше, ніж **bzip2**. Близькими є швидкості відновлення файлів утиліт **zstd** та **xz**.

Найвищий ступінь стиснення забезпечує утиліта **xz** (level 9) - 8.2704, але при цьому і час стиснення був найбільшим — 337 сек. Мінімальний час стиснення файлу (832 MB) - 2.57 сек. забезпечує **zstd** (level 1), але при цьому ступінь стиснення має невисоке значення - 4.9.

Але за показниками ступінь стиснення і розмір стиснення утиліта **bzip2** має переваги над **gzip**. Водночас програма **gzip** є менш вибагливою до ресурсів операційної системи, тому переваги можна надати самій цій утиліті.

Порівняльний аналіз результатів тестування свідчить, що утиліта **Zstandard** має кращі показники за параметрами швидкість стиснення, час стиснення. Але програє по показнику ступінь та швидкість відновлення.

Отже, найбільшу швидкість стиснення демонструє **zstd (level 1)** — **323.41 MB/s**, а найменшу - **xz (level 9)** – 2.46 MB/s (рис. 3.1).

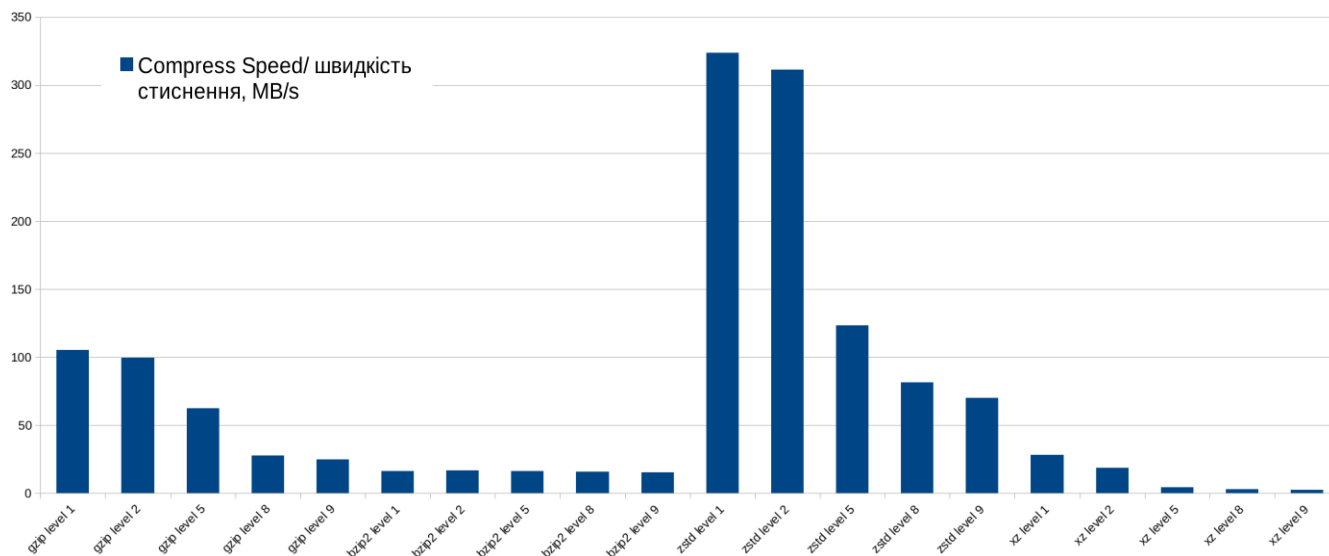


Рис. 3.1. Порівняння утиліт за параметром “Швидкість стиснення”

Найменший час стиснення - 2.57 сек. демонструє **zstd (level 1)**, а найбільший - утиліта **xz (level 9)** — 337.07 сек (рис.3.2).

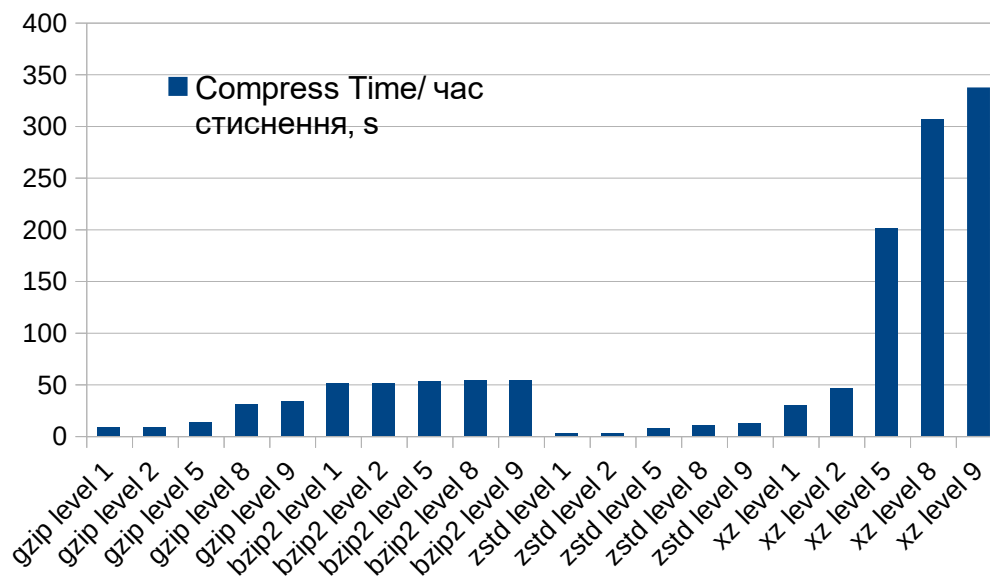


Рис. 3.2. Порівняння утиліт за параметром “Час стиснення”

Найвищий ступінь стиснення — 8.27 забезпечує xz (level 9), а найменший серед розглянутих утиліт - **gzip** (level 1) — 4.1799 (рис.3.3).

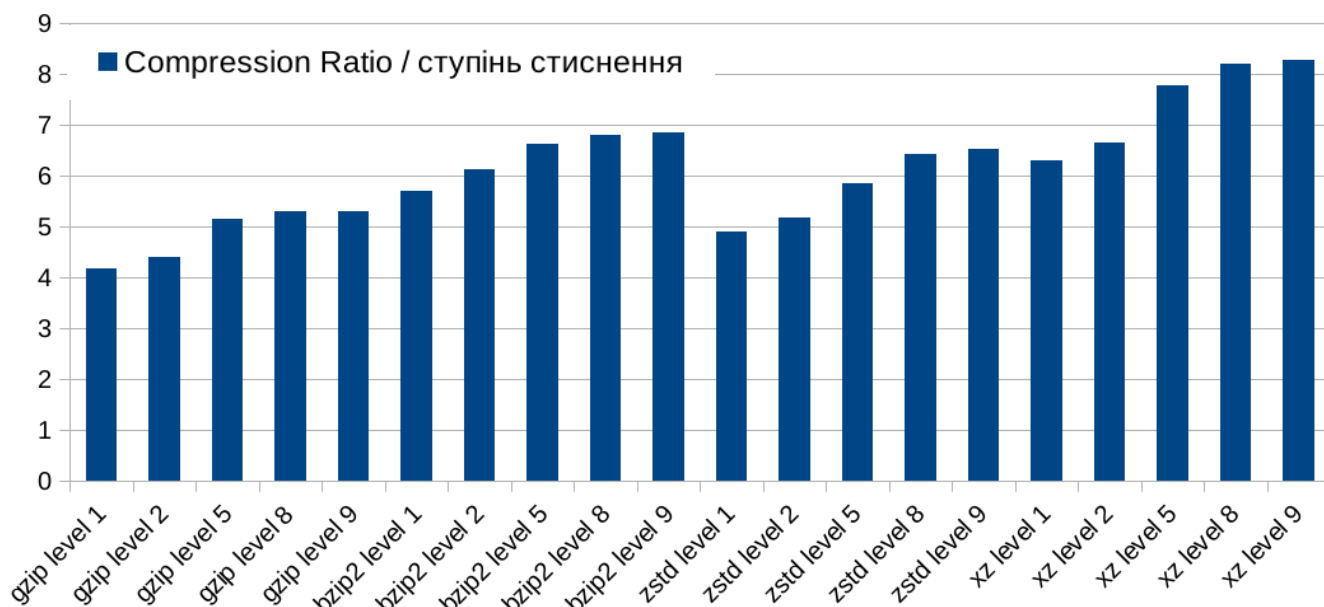


Рис. 3.3. Порівняння утиліт за параметром “Ступінь стиснення”

Найшвидше відновлює файли **gzip** (level 8) — 221.64 МВ/сек., а найменша швидкість відновлення у **bzip2** (level 9) — 56.04 МВ/сек.

Відповідно, найменший час відновлення демонструє gzip (level 8) - 3.75 сек., а найдовший - bzip2 (level 9) — 14.83 сек.

За параметром розмір стиснення переваги має xz (level 9) — 105381748 В, а найбільший розмір після стиснення забезпечує gzip (level 1) — 208511455 В.

Результати порівняльно аналізу узагальнено у табл. 3.1.

Таблиця 3.1

Порівняння результатів роботи утиліт gzip, bzip2, zstd, xz (LZMA)

Параметр	zstd	gzip	bzip2	xz (LZMA)
Швидкість стиснення	Дуже висока	Висока	Низька	Низька
Швидкість декомпресії	Середня	Висока	Низька	Низька
Ступінь стиснення	Високий	Середній	Вищий за gzip	Максимальний

Отже, вибір утиліти для забезпечення стиснення файлів залежить від цілей та задач, які необхідно вирішувати. Якщо потрібно швидко і з великим ступенем стиснути файл варто застосовувати утиліту `zstd`.

У разі якщо потрібно максимально стиснути текстові чи лог-файл і час при цьому не є ключовим показником, то з цією задачею добре справиться утиліта `xz`. Саме її варто використовувати для виконання резервного копіювання.

Середній ступінь стиснення, з високою швидкістю виконання, забезпечує утиліта `gzip`.

3.3. Реалізація алгоритму стиснення JPEG у середовищі Google Colab

Google Colaboratory (або Google Colab) - це хмарне середовище, яке дозволяє виконувати Python-код у веб-браузері без необхідності встановлення будь-якого програмного забезпечення на комп'ютері.

Colab заснований на популярному форматі Jupyter Notebook, що поєднує код, текст, графіки та результати в одному інтерактивному документі. Colab дозволяє легко підключатися до зовнішніх баз даних, веб-API або завантажувати файли з локального комп'ютера. **Google Colab** підтримує повний стек Python-бібліотек, таких як NumPy, Pandas, TensorFlow, PyTorch, Matplotlib тощо.

Робота в **Google Colab** розпочинається зі створення ноутбука або завантаження вже існуючого з Google Drive. Код створюється в редакторі **Jupyter Notebook**, а виконується на серверах Google, що дозволяє використовувати потужні обчислювальні ресурси, які відсутні на звичайних ПК [62].

У середовищі **Google Colaboratory** була реалізована на мові Python програма для виконання стиснення файлів з використанням алгоритму JPEG.

У програмі реалізовано функції:

1) Завантаження та відображення зображення:

- Користувач завантажує зображення за допомогою інтерактивного завантажувача.

- Завантажене зображення відображається на екрані.

2) Стиснення зображення та візуалізація результатів:

- Стиснення зображення виконується за алгоритмом JPEG з різними рівнями якості: 70, 25, 10.

- Відображається кожне стиснене зображення, розмір стисненого файлу (Kb) та рівень якості вказується розмір стисненого файлу.

Структура коду містить:

- **Бібліотеки:**

```
from PIL import Image - для обробки зображень;
import matplotlib.pyplot as plt - для візуалізації зображень;
import io - для роботи з потоками даних;
import os - для перевірки розміру файлів;
from google.colab import files - для завантаження файлів у середовищі Colab.
```

- **Функції:**

```
upload_and_display_image - для завантаження та відображення
оригінального зображення у середовищі Colab.
```

```
compress_and_visualize - виконує стиснення завантаженого зображення за
алгоритмом JPEG для кількох рівнів якості та відображає результати стиснення.
```

- **Основний блок, який виконує** завантаження зображення, потім - стиснення та візуалізацію.

Опис змінних, що використовуються у програмі, наведено у табл. 3.5

Опис змінних, що використовуються у програмі

Змінна	Тип	Джерело	Примітка
uploaded	dict	Результат виконання <code>files.upload()</code> .	Зберігає завантажений файл у вигляді словника, де ключ — це ім'я файлу, а значення — вміст файлу у байтах.
filename	str	Отримується із ключів словника <code>uploaded:filename = list(uploaded.keys())[0]</code>	Назва першого файлу, завантаженого користувачем
image	PIL.Image. Image	Створюється за допомогою бібліотеки PIL: <code>image = Image.open(io.BytesIO(uploaded[filename]))</code>	Об'єкт зображення, завантаженого користувачем. Використовується для обробки та відображення зображення.
qualities	list	<code>qualities = [70, 25, 10]</code>	Список рівнів якості стиснення для алгоритму JPEG: 70 - висока, 25 - середня, 10 - низька
buffer	io.BytesIO	Створюється під час кожної ітерації стиснення: <code>buffer = io.BytesIO()</code>	Буфер у пам'яті для тимчасового зберігання стисненого зображення у форматі JPEG.
compressed_size	int	Визначається через метод <code>tell()</code> об'єкта <code>buffer</code> : <code>compressed_size = buffer.tell()</code>	Розмір стисненого зображення, байт
compressed_image	PIL.Image. Image	Завантажується з буфера: <code>compressed_image = Image.open(buffer)</code>	Стиснене зображення, відновлене з буфера <code>buffer</code>
original_size	int	Отримується через функцію <code>os.path.getsize</code> : <code>original_size = os.path.getsize(filename)</code>	Розмір оригінального завантаженого файлу, байт
quality	int	Елемент списку <code>qualities</code> в кожній ітерації циклу.	Поточний рівень якості стиснення, який береться із списку <code>qualities</code> .
i	int	Генерується циклом: <code>for i, quality in enumerate(qualities):</code>	Індекс ітерації у циклі <code>for</code> , використовується для доступу до елементів списку <code>qualities</code> та визначення позиції на графіку

Додаткові змінні, що використовуються у програмі:

`plt.subplot(1, 3, i + 1)` - для визначення позиції графіка на екрані для візуалізації стисненого зображення;

`plt.figure(figsize=(6, 6))` та `plt.figure(figsize=(12, 6))` - розміри графіка для відображення оригінального та стиснених зображень.

Результати виконання програми представлені на рис. 3.4.

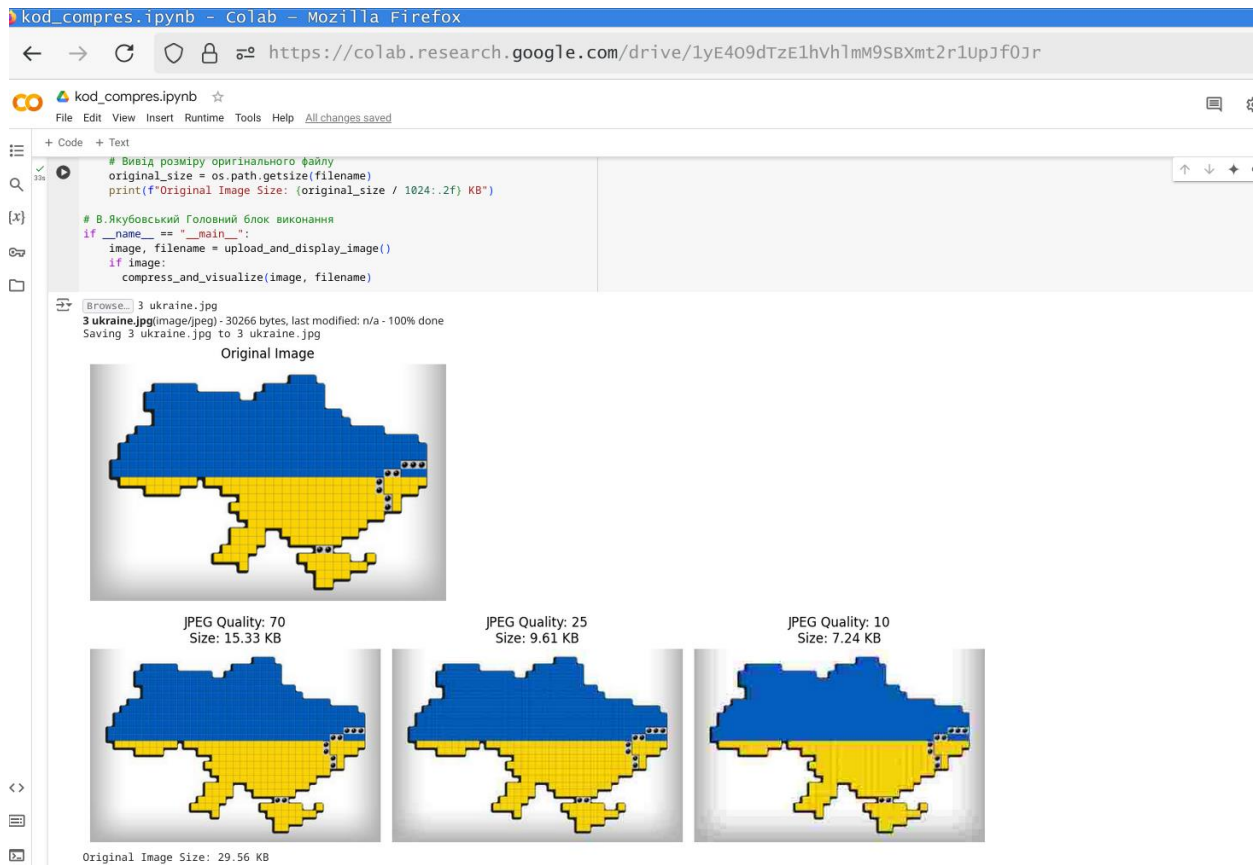


Рис.3.4. Результати виконання: стиснений файл за алгоритмом JPEG із значеннями рівнів якості 70, 25, 10

З рисунку видно, що найкраще стиснення файлу - 7.24Kb, але якість зображення при цьому є низькою.

Зображення файлу розміром 9.61Kb теж має низьку якість.

Зображення файлу розміром 15.33Kb має найкращу якість у порівнянні з попередніми файлами.

Код програми наведено у додатку Ж.

ВИСНОВКИ

Результати виконання кваліфікаційної роботи свідчать, що

- вибір оптимального методу стиснення залежить від типу даних, допустимого ступеня втрати та потужності обчислювальних ресурсів;
- алгоритми стиснення без втрат, що базуються на словникових методах, кодуванні Хаффмана та арифметичному кодуванні, є оптимальними для текстів, баз даних і ситуацій, де потрібне повне відновлення даних;
- втратні алгоритми стиснення варто застосовувати для медійних файлів, де важливим є компроміс між розміром і якістю;
- для растрових зображень ефективними є формати WEBP, JPEG, PNG; для векторних зображень найпоширенішими форматами є: AI, SVG, CDR;
- застосування утиліти з відкритим програмним кодом (zstd, gzip), у разі необхідності, забезпечує високу швидкість стиснення та розпакування даних;
- у хмарному середовищі Google Colab можлива реалізація алгоритмів стиснення на мові Python з використанням формату Jupyter Notebook,

У підсумку, використання інструментів стиснення даних може забезпечити:

- 1) Ефективне зберігання даних, оскільки їх стиснення дозволяє суттєво зменшити потреби у дисковому просторі, сприяє оптимізації ресурсів на рівні серверів чи дата-центрі;
- 2) Прискорення передачі даних, наприклад, при використанні утиліт ZSTD, GZIP, які забезпечують високу швидкість стиснення та розпакування даних. Завдяки цьому зменшується час передачі даних через мережі, що є критично важливим для обміну в реальному часі та при віддаленому доступі.
- 3) Економію фінансових ресурсів, що особливо важливо при використанні послуг хостингу, виконанні хмарних обчислень, зберіганні великих обсягів даних Big Data.

Перспективними є дослідження щодо використання штучного інтелекту у процесах виконання швидкого стиснення та відновлення даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ziv J., Lempel A. A Universal Algorithm for Sequential Data Compression, IEEE Transactions on Information Theory 23 (3), 1977, pp. 337 - 342.
2. Ziv J., Lempel A. Compression of individual sequences via variable-rate coding. IEEE transactions on Information Theory, 1978, 24(5), pp. 530 - 536.
3. Mark Nelson. The Data Compression Book 2nd edition. URL: <https://www.hlevkin.com/hlevkin/02imageprocC/The%20Data%20Compression%20Book%202nd%20edition.pdf>
4. David Salomon. Data Compression: The Complete Reference. URL: <https://link.springer.com/book/10.1007/978-1-84628-603-2>
5. David Salomon. A Concise Introduction to Data Compression. URL: <https://www.davidsalomon.name/DCugAdvertis/DCug.html>
6. Mahoney M. Fast Text Compression with Neural Networks, Proc. AAAI FLAIRS, Orlando, 2000, AAAI. URL: <http://mattmahoney.net/dc/mmahoney00.pdf>
7. Mahoney M. Large Text Compression Benchmark. 2024. URL: <http://mattmahoney.net/dc/text.html>
8. PK Protect/PKZIP/Smartcrypt/SecureZIP. URL: <https://support.pkware.com/pkzip/>
9. Klein S. T. Dana Shapira On the Randomness of Compressed Data. URL: <https://www.semanticscholar.org/paper/On-the-Randomness-of-Compressed-Data-Klein-Shapira/0bf75aa478f7a21a8b06ba1f5d133270f5f85066>]
10. Глибовець А. М., Яблонський В. Р. Експериментальне порівняння алгоритмів компресії даних. URL: <http://nrpcomp.ukma.edu.ua/article/view/185530/185139>
11. Манжос Ю. С. Соколова Є. В. Метод стиснення даних у мережі Інтернет речей //Радіoeлектронні і комп'ютерні системи, 2020, №4(96). С. 57 - 67.

12. Лужецький В. А. Савицька Л. А. Розробка та дослідженням методів адаптивного ущільнення даних на основі лінійної форми Фібоначчі //Східно-Європейський журнал передових технологій. 2015. №1/9 (73). С. 16-22.
13. Розенвассер Д. М. Порівняння методів ефективного кодування повідомлень / В. В. Педяш, Д. М. Розенвассер // Sciences of Europe. 2020. №58. С. 28-35. URL: <https://hdl.handle.net/11300/26567>
14. Data Compression. URL: <https://www.techopedia.com/definition/884/data-compression>
15. Matt Mahoney Data Compression Programs. URL: <http://mattmahoney.net/dc/>
16. Глибовець А. М., Яблонський В. Р. Експериментальне порівняння алгоритмів компресії даних. URL: <http://nrpcmp.ukma.edu.ua/article/view/185530/185139>
17. Коваленко А.Є. Теорія інформації і кодування: навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2020. 248 с.
18. Загальні поняття про архівацію та стиснення файлів. URL: http://alextexnok.blogspot.com/p/v-behaviorurldefaultvmlo_73.html
19. GNU TAR. URL: <https://www.gnu.org/software/tar/>
20. Ismoil Shifoev. Run-Length Encoding (RLE) Algorithm: Step-by-Step Guide. URL: <https://medium.com/@ishifoev/run-length-encoding-rle-algorithm-step-by-step-guide-b0b89f3a4a9f>
21. Беркман Л.Н., Бондарчук А.П., Гайдур Г.І., Чумак Н.С. Кодування джерел інформації та каналів зв'язку: навч.посібник. Київ: ННІТІ ДУТ, 2018.
22. Khalid Sayood Ph.D. Introduction to Data Compression (The Morgan Kaufmann Series in Multimedia Information and Systems) 4th Edition. 2012. 768 pages
23. PKZIP. URL: <https://www.pkware.com/products/pkzip>

24. Корпань Я. В. Методи та алгоритми компактного представлення графічної інформації в комп'ютерних системах / Технологічний аудит та резерви виробництва. – 2015. – № 3/2 (23). – С. 32–36.
25. Зубко Р.А. Алгоритми стиснення зображень /Східно-Європейський журнал передових технологій. 2013. №1/2(61). С. 40-44
26. John W. O'Brien (2005) The JPEG Image Compression Algorithm. APPM-331. FINAL PROJECT, (December 2). URL: <https://www.scribd.com/doc/192407766/Jpeg-compression>
27. JPEG і JPEG2000. URL: <https://www.adobe.com/ua/creativecloud/file-types/image/comparison/jpeg-vs-jpeg-2000.html>
28. Капшій О. В., Коваль О. І., Русин Б. П. Вейвлет-перетворення у компресії та попередній обробці зображень /НАН України, Фіз.-мех. ін-т ім. Г. В. Карпенка. Львів: Сполом, 2008. 206 с.
29. An image format for the Web. URL: https://developers.google.com/speed/webp#webp_support.
30. Формат зображень WEBP. URL: <https://web24.com.ua/uk/about-webp-format/>
31. Зубко Р.А. Стиснення зображень фрактальним методом /Східно-Європейський журнал передових технологій. 2014. №6/2(71). С.23-28.
32. Mandelbrot B. The Fractal Geometry of Nature. W.H. Free-4. man, San Francisco, 1982. 468 p.
33. Barnsley M. F. Fractals Everywhere. Boston: Academic Press3. Professional, 1993. 531 p.
34. Зубко Р.А. Покращення ефективності стиснення зображень фрактальним методом /Вісник Університету «Україна», 2017, № 1 (19), С.190-196.
35. Rafael C. Gonzalez, Richard Eugene Woods. Digital Image Processing. Prentice Hall, 2008. 954 p.

36. Computer graphics: principles and practice / John F. Hughes, Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley. Third edition. 2014. 1263p.
37. Бондаренко М.Ф., Помазанов С.В., Шубін І.Ю. Програмні засоби створення мультимедіа. Навч. посібник, Харків: СМІТ, 2005. 185 с.
38. Ніл Леонард, Гевін Емброуз. Основи. Графічний дизайн 02: Дизайнерське дослідження. ArtHuss. 2019. 192 с.
39. John W. O'Brien (2005) The JPEG Image Compression Algorithm. APPM-331. FINAL PROJECT, (December 2). URL: <https://www.scribd.com/doc/192407766/Jpeg-compression>
40. Greg Roelofs. PNG: The Definitive Guide. Publisher: O'Reilly. 1999. 321 p.
41. James D. Murray and William vanRyper. Encyclopedia of Graphics File Formats, Second Edition. 1996, O'Reilly & Associates. 11158 p.
42. Bourke, P. (1999). "Truevision TGA File Format Specification and Implementation." URL: <http://paulbourke.net>
43. Документація IA. URL: <https://docs.fileformat.com/uk/image/ai/>
44. Документація SVG. URL: <https://docs.fileformat.com/uk/image/svgz/>
45. Документація EPS. URL: <https://docs.fileformat.com/uk/page-description-language/eps/>
46. Документація CDR. URL: <https://docs.fileformat.com/uk/image/cdr/>
47. Документація DXF. URL: <https://docs.fileformat.com/uk/image/dxf/>
48. WMF. URL: <https://ua5.org/graphic/328-windows-metafile-wmf.html>
49. PDF file format: Basic structure [updated 2020]. URL: <https://www.infosecinstitute.com/resources/hacking/pdf-file-format-basic-structure/>
50. Gzip vs Brotli. URL: <https://evrodim.org.ua/tech/brotli-vs-gzip/>
51. Gzip. URL: <http://www.gzip.org/>
52. bzip2. URL: <http://www.bzip.org/>
53. bzip2. URL: http://www.bzip2.org/bzip2/docs/manual_toc.html

54. pbzip2. URL:
<https://www.onworks.net/uk/osdistributions/programs/pbzip2-online>
55. lbzip2. URL: <https://github.com/kjn/lbzip2>
56. Пакунок: xz-utils. URL: <https://packages.debian.org/uk/sid/xz-utils>
57. GNU Gzip: General file (de)compression. URL:
<https://www.gnu.org/software/gzip/manual/gzip.html>
58. Zstandard Compression and the 'application/zstd' Media Type. URL:
<https://www.rfc-editor.org/rfc/rfc8878.html>
59. Zstandard. URL: <https://github.com/facebook/zstd>
60. LZMA. URL: <https://www.7-zip.org/>
61. The Linux Kernel. URL: <https://www.kernel.org/happy-new-year-and-good-bye-bzip2.html>
62. Офіційна документація Google Colab. URL:
<https://colab.research.google.com/notebooks/>
63. Курс Python для наукових обчислень. URL:
<https://realpython.com/python-data-science/>

ДОДАТКИ

Додаток А

1 compatib ukraine.bmp (253.3Kb)



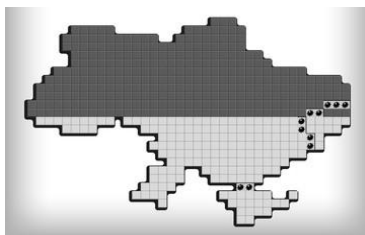
2 lossy ukraine.webp (13.9Kb)



2 lossless ukraine.webp (64.2Kb)



4 16bpc graya ukraine.png (94.8Kb)



7 ukraine.gif (40.2Kb)



Продовження додатку А

9 2025.webp (117.7Kb)



9 2025.gif (292.8Kb)



9 2025.bmp (1.6Mb)



gzip test

```
[gzip 1.5]
[compress lvl: 1 ] gzip -1 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 7.92s
[compress speed:] 104.94 MB/s
compression ratio: 4.1799 871557120 / 208511455 KB
```

```
[decompress] gzip -d /home/gziptest/linux-5.1-rc5.tar.gz
[decompress stats:] real: 4.17s
[decompress speed:] 199.32 MB/s
```

```
[gzip 1.5]
[compress lvl: 2 ] gzip -2 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 8.34s
[compress speed:] 99.66 MB/s
compression ratio: 4.3933 871557120 / 198382570 KB
```

```
[decompress] gzip -d /home/gziptest/linux-5.1-rc5.tar.gz
[decompress stats:] real: 4.04s
[decompress speed:] 205.73 MB/s
```

```
[gzip 1.5]
[compress lvl: 5 ] gzip -5 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 13.36s
[compress speed:] 62.21 MB/s
compression ratio: 5.1437 871557120 / 169440817 KB
```

```
[decompress] gzip -d /home/gziptest/linux-5.1-rc5.tar.gz
[decompress stats:] real: 3.86s
[decompress speed:] 215.33 MB/s
```

```
[gzip 1.5]
[compress lvl: 8 ] gzip -8 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 30.41s
[compress speed:] 27.33 MB/s
compression ratio: 5.3068 871557120 / 164233117 KB
```

```
[decompress] gzip -d /home/gziptest/linux-5.1-rc5.tar.gz
[decompress stats:] real: 3.75s
[decompress speed:] 221.64 MB/s
```

```
[gzip 1.5]
[compress lvl: 9 ] gzip -9 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 33.42s
[compress speed:] 24.87 MB/s
compression ratio: 5.3094 871557120 / 164150784 KB
```

```
[decompress] gzip -d /home/gziptest/linux-5.1-rc5.tar.gz
[decompress stats:] real: 3.76s
[decompress speed:] 221.05 MB/s
```

bzip2 test

```
[bzip2 1.0.6]
[compress lvl: 1 ] bzip2 -1 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 51.09s
[compress speed:] 16.26 MB/s
compression ratio: 5.6991 871557120 / 152926645 KB
```

```
[decompress] bzip2 -d /home/gziptest/linux-5.1-rc5.tar.bz2
[decompress stats:] real: 14.03s
[decompress speed:] 59.24 MB/s
```

```
[bzip2 1.0.6]
[compress lvl: 2 ] bzip2 -2 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 50.27s
[compress speed:] 16.53 MB/s
compression ratio: 6.1351 871557120 / 142058911 KB
```

```
[decompress] bzip2 -d /home/gziptest/linux-5.1-rc5.tar.bz2
[decompress stats:] real: 14.45s
[decompress speed:] 57.52 MB/s
```

```
[bzip2 1.0.6]
[compress lvl: 5 ] bzip2 -5 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 52.21s
[compress speed:] 15.91 MB/s
compression ratio: 6.6153 871557120 / 131747470 KB
```

```
[decompress] bzip2 -d /home/gziptest/linux-5.1-rc5.tar.bz2
[decompress stats:] real: 14.59s
[decompress speed:] 56.96 MB/s
```

```
[bzip2 1.0.6]
[compress lvl: 8 ] bzip2 -8 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 53.67s
[compress speed:] 15.48 MB/s
compression ratio: 6.8077 871557120 / 128023871 KB
```

```
[decompress] bzip2 -d /home/gziptest/linux-5.1-rc5.tar.bz2
[decompress stats:] real: 14.83s
[decompress speed:] 56.04 MB/s
```

```
[bzip2 1.0.6]
[compress lvl: 9 ] bzip2 -9 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 54.03s
[compress speed:] 15.38 MB/s
compression ratio: 6.8570 871557120 / 127103698 KB
```

```
[decompress] bzip2 -d /home/gziptest/linux-5.1-rc5.tar.bz2
[decompress stats:] real: 14.83s
[decompress speed:] 56.04 MB/s
```

zstd test

```
[zstd v1.4.0]
[compress lvl: 1 ] zstd -T8 -q -f -1 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 2.57s
[compress speed:] 323.41 MB/s
compression ratio: 4.9065 871557120 / 177631505 KB
```

```
[decompress] zstd -T8 -q -f -d /home/gziptest/linux-5.1-rc5.tar.zst
[decompress stats:] real: 6.54s
[decompress speed:] 127.09 MB/s
```

```
[zstd v1.4.0]
[compress lvl: 2 ] zstd -T8 -q -f -2 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 2.67s
[compress speed:] 311.30 MB/s
compression ratio: 5.1729 871557120 / 168483936 KB
```

```
[decompress] zstd -T8 -q -f -d /home/gziptest/linux-5.1-rc5.tar.zst
[decompress stats:] real: 7.20s
[decompress speed:] 115.44 MB/s
```

```
[zstd v1.4.0]
[compress lvl: 5 ] zstd -T8 -q -f -5 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 6.74s
[compress speed:] 123.32 MB/s
compression ratio: 5.8511 871557120 / 148953662 KB
```

```
[decompress] zstd -T8 -q -f -d /home/gziptest/linux-5.1-rc5.tar.zst
[decompress stats:] real: 7.03s
[decompress speed:] 118.23 MB/s
```

```
[zstd v1.4.0]
[compress lvl: 8 ] zstd -T8 -q -f -8 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 10.22s
[compress speed:] 81.32 MB/s
compression ratio: 6.4341 871557120 / 135457716 KB
```

```
[decompress] zstd -T8 -q -f -d /home/gziptest/linux-5.1-rc5.tar.zst
[decompress stats:] real: 6.17s
[decompress speed:] 134.71 MB/s
```

```
[zstd v1.4.0]
[compress lvl: 9 ] zstd -T8 -q -f -9 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 11.90s
[compress speed:] 69.84 MB/s
compression ratio: 6.5200 871557120 / 133673590 KB
```

```
[decompress] zstd -T8 -q -f -d /home/gziptest/linux-5.1-rc5.tar.zst
[decompress stats:] real: 6.11s
[decompress speed:] 136.03 MB/s
```

xz test

```
[xz 5.2.2]
[compress lvl: 1 ] xz -1 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 29.62s
[compress speed:] 28.06 MB/s
compression ratio: 6.3047 871557120 / 138237140 KB
```

```
[decompress] xz -f -d /home/gziptest/linux-5.1-rc5.tar.xz
[decompress stats:] real: 7.85s
[decompress speed:] 105.88 MB/s
```

```
[xz 5.2.2]
[compress lvl: 2 ] xz -2 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 45.46s
[compress speed:] 18.28 MB/s
compression ratio: 6.6408 871557120 / 131242224 KB
```

```
[decompress] xz -f -d /home/gziptest/linux-5.1-rc5.tar.xz
[decompress stats:] real: 7.33s
[decompress speed:] 113.39 MB/s
```

```
[xz 5.2.2]
[compress lvl: 5 ] xz -5 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 200.59s 7
[compress speed:] 4.14 MB/s
compression ratio: 7.7627 871557120 / 112273700 KB
```

```
[decompress] xz -f -d /home/gziptest/linux-5.1-rc5.tar.xz
[decompress stats:] real: 6.50s
[decompress speed:] 127.87 MB/s
```

```
[xz 5.2.2]
[compress lvl: 8 ] xz -8 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 305.67s
[compress speed:] 2.71 MB/s
compression ratio: 8.1944 871557120 / 106359392 KB
```

```
[decompress] xz -f -d /home/gziptest/linux-5.1-rc5.tar.xz
[decompress stats:] real: 6.35s
[decompress speed:] 130.89 MB/s
```

```
[xz 5.2.2]
[compress lvl: 9 ] xz -9 /home/gziptest/linux-5.1-rc5.tar
[compress stats:] real: 337.07s
[compress speed:] 2.46 MB/s
compression ratio: 8.2704 871557120 / 105381748 KB
```

```
[decompress] xz -f -d /home/gziptest/linux-5.1-rc5.tar.xz
[decompress stats:] real: 6.30s
[decompress speed:] 131.93 MB/s
```

Таблиця результатів тестування утиліт gzip, bzip2, zstd, xz

Utility/ утиліта	Compress Speed/ швидкість стиснення, MB/s	Compress Time/ час стиснення, s	Compression Ratio / ступінь стиснення	Decompress Speed/ швидкість відновлення, MB/s	Decompress Time/час відновлення, s	Compress Size / розмір стиснення, B
gzip level 1	104.94	7.92	4.1799	199.32	4.17	208511455
gzip level 2	99.66	8.34	4.3933	205.73	4.04	198382570
gzip level 5	62.21	13.36	5.1437	215.33	3.86	169440817
gzip level 8	27.33	30.41	5.3068	221.64	3.75	164233117
gzip level 9	24.87	33.42	5.3094	221.05	3.76	164150784
bzip2 level 1	16.26	51.09	5.6991	59.24	14.03	152926645
bzip2 level 2	16.53	50.27	6.1351	57.52	14.45	142058911
bzip2 level 5	15.91	52.21	6.6153	56.96	14.59	131747470
bzip2 level 8	15.48	53.67	6.8077	56.04	14.83	128023871
bzip2 level 9	15.38	54.03	6.8570	56.04	14.83	127103698
zstd level 1	323.41	2.57	4.9065	127.09	6.54	177631505
zstd level 2	311.30	2.67	5.1729	115.44	7.20	168483936
zstd level 5	123.32	6.74	5.8511	118.23	7.03	148953662
zstd level 8	81.32	10.22	6.4341	134.71	6.17	135457716
zstd level 9	69.84	11.90	6.5200	136.03	6.11	133673590
xz level 1	28.06	29.62	6.3047	105.88	7.85	138237140
xz level 2	18.28	45.46	6.6408	113.39	7.33	131242224
xz level 5	4.14	200.59	7.7627	127.87	6.50	112273700
xz level 8	2.71	305.67	8.1944	130.89	6.35	106359392
xz level 9	2.46	337.07	8.2704	131.93	6.30	105381748

Програмна реалізація алгоритму JPEG

```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "source": [
        "from PIL import Image\n",
        "import matplotlib.pyplot as plt\n",
        "import io\n",
        "import os\n",
        "from google.colab import files\n",
        "\n",
        "# В.Якубовський Функція для завантаження та відображення зображення\n",
        "def upload_and_display_image():\n",
        "    uploaded = files.upload()\n",
        "    if not uploaded:\n",
        "        return None\n",
        "\n",
        "    filename = list(uploaded.keys())[0]\n",
        "    image = Image.open(io.BytesIO(uploaded[filename]))\n",
        "\n",
        "    plt.figure(figsize=(6, 6))\n",
        "    plt.imshow(image)\n",
        "    plt.title('Original Image')\n",
        "    plt.axis('off')\n",
        "    plt.show()\n",
        "\n",
        "    return image, filename\n",
        "\n",
        "# В.Якубовський Функція для стиснення та візуалізації\n",
        "def compress_and_visualize(image, filename):\n",
        "    # Відображення оригінального зображення вже є у upload_and_display_image\n",
        "\n",
        "    plt.figure(figsize=(12, 6))\n",
        "    # Стиснення з різною якістю\n",
        "    qualities = [70, 25, 10] # Рівні якості JPEG\n",
        "\n",
        "    for i, quality in enumerate(qualities):\n",
        "        # Стиснення зображення\n",
        "        buffer = io.BytesIO()\n",
        "        image.save(buffer, format='JPEG', quality=quality)\n",
        "        compressed_size = buffer.tell()\n",
        "\n",
        "        # Відновлення зображення з буфера\n",
        "        buffer.seek(0)
```

Продовження додатку Ж

```

"        compressed_image = Image.open(buffer)\n",
"\n",
"        # Відображення стисненого зображення\n",
"        plt.subplot(1, 3, i + 1) # Змінено з 1,3,i+2 на 1,3,i+1\n",
"        plt.imshow(compressed_image)\n",
"        plt.title(f'JPEG Quality: {quality}\\nSize: {compressed_size /
1024:.2f} KB')\n",
"        plt.axis('off')\n",
"\n",
"        # Збереження (опціонально)\n",
"        # compressed_image.save(f'compressed_{filename}_{quality}.jpg')\n",
"\n",
"        plt.tight_layout()\n",
"        plt.show()\n",
"\n",
"        # Вивід розміру оригінального файлу\n",
"        original_size = os.path.getsize(filename)\n",
"        print(f\"Original Image Size: {original_size / 1024:.2f} KB\")\n",
"\n",
"# В.Якубовський Головний блок виконання\n",
"if __name__ == \"__main__\":\n",
"    image, filename = upload_and_display_image()\n",
"    if image:\n",
"        compress_and_visualize(image, filename)"
],
"metadata": {
    "colab": {
        "base_uri": "https://localhost:8080/",
        "height": 722
    },
    "id": "Lmk5hu7Ly-pu",
    "outputId": "a66b6bf0-b934-4e10-f800-06bd06c16784"
},
"execution_count": 2,
"outputs": [
    {
        "output_type": "display_data",
        "data": {
            "text/plain": [
                "<IPython.core.display.HTML object>"
            ],
            "text/html": [
                "\n",
                "        <input type=\"file\" id=\"files-748092cc-0669-4e2f-8fa8-
960ef3ed5f83\" name=\"files[]\" multiple disabled\n",
                "            style=\"border:none\" />\n",
                "        <output id=\"result-748092cc-0669-4e2f-8fa8-960ef3ed5f83\">\n",
                "            Upload widget is only available when the cell has been executed in
the\n",
                "            current browser session. Please rerun this cell to enable.\n",
                "        </output>\n",
                "        <script>// Copyright 2017 Google LLC\n",
                "        /\n",
                "        // Licensed under the Apache License, Version 2.0 (the \"License\");\n",
                "        // you may not use this file except in compliance with the License.\n",
                "        // You may obtain a copy of the License at\n",
                "        /\n",
                "        //      http://www.apache.org/licenses/LICENSE-2.0\n",
                "        /\n"
            ]
        }
    ]
}

```

Продовження додатку Ж

```

software\n",
BASIS,\n",
implied.\n",
and\n",
    "// Unless required by applicable law or agreed to in writing,
    "// distributed under the License is distributed on an \"AS IS\"
    "// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
    "// See the License for the specific language governing permissions
    "// limitations under the License.\n",
    "\n",
    "/*\n",
    " * @fileoverview Helpers for google.colab Python module.\n",
    " */\n",
    "(function(scope) {\n",
    "function span(text, styleAttributes = {}) {\n",
    "  const element = document.createElement('span');\n",
    "  element.textContent = text;\n",
    "  for (const key of Object.keys(styleAttributes)) {\n",
    "    element.style[key] = styleAttributes[key];\n",
    "  }\n",
    "  return element;\n",
    "}\n",
    "\n",
    "// Max number of bytes which will be uploaded at a time.\n",
    "const MAX_PAYLOAD_SIZE = 100 * 1024;\n",
    "\n",
    "function _uploadFiles(inputId, outputId) {\n",
    "  const steps = uploadFilesStep(inputId, outputId);\n",
    "  const outputElement = document.getElementById(outputId);\n",
    "  // Cache steps on the outputElement to make it available for the next
    "  // to uploadFilesContinue from Python.\n",
    "  outputElement.steps = steps;\n",
    "\n",
    "  return _uploadFilesContinue(outputId);\n",
    "}\n",
    "\n",
    "// This is roughly an async generator (not supported in the browser
    yet),\n",
    "// where there are multiple asynchronous steps and the Python side is
    going\n",
    "// to poll for completion of each step.\n",
    "// This uses a Promise to block the python side on completion of each
    step,\n",
    "// then passes the result of the previous step as the input to the next
    step.\n",
    "function _uploadFilesContinue(outputId) {\n",
    "  const outputElement = document.getElementById(outputId);\n",
    "  const steps = outputElement.steps;\n",
    "\n",
    "  const next = steps.next(outputElement.lastPromiseValue);\n",
    "  return Promise.resolve(next.value.promise).then((value) => {\n",
    "    // Cache the last promise value to make it available to the next\n",
    "    // step of the generator.\n",
    "    outputElement.lastPromiseValue = value;\n",
    "    return next.value.response;\n",
    "  });\n",
    "}\n",

```

```

"\n",
"/**\n",
" * Generator function which is called between each async step of the
upload\n",
" * process.\n",
" * @param {string} inputId Element ID of the input file picker
element.\n",
" * @param {string} outputId Element ID of the output display.\n",
" * @return {!Iterable<!Object>} Iterable of next steps.\n",
" */\n",
"function* uploadFilesStep(inputId, outputId) {\n",
"  const inputElement = document.getElementById(inputId);\n",
"  inputElement.disabled = false;\n",
"\n",
"  const outputElement = document.getElementById(outputId);\n",
"  outputElement.innerHTML = '';\n",
"\n",
"  const pickedPromise = new Promise((resolve) => {\n",
"    inputElement.addEventListener('change', (e) => {\n",
"      resolve(e.target.files);\n",
"    });\n",
"  });\n",
"\n",
"  const cancel = document.createElement('button');\n",
"  inputElement.parentElement.appendChild(cancel);\n",
"  cancel.textContent = 'Cancel upload';\n",
"  const cancelPromise = new Promise((resolve) => {\n",
"    cancel.onclick = () => {\n",
"      resolve(null);\n",
"    };\n",
"  });\n",
"\n",
"  // Wait for the user to pick the files.\n",
"  const files = yield {\n",
"    promise: Promise.race([pickedPromise, cancelPromise]),\n",
"    response: {\n",
"      action: 'starting',\n",
"    },\n",
"  };\n",
"\n",
"  cancel.remove();\n",
"\n",
"  // Disable the input element since further picks are not allowed.\n",
"  inputElement.disabled = true;\n",
"\n",
"  if (!files) {\n",
"    return {\n",
"      response: {\n",
"        action: 'complete',\n",
"      },\n",
"    };\n",
"  }\n",
"\n",
"  for (const file of files) {\n",
"    const li = document.createElement('li');\n",
"    li.append(span(file.name, {fontWeight: 'bold'}));\n",
"    li.append(span(\n",
"      `(${file.type} | 'n/a') - ${file.size} bytes,` +\n",
"      `last modified:`,

```

```

"                file.lastModifiedDate ?
file.lastModifiedDate.toLocaleDateString() :\n",
"                'n/a' } - `));\n",
"    const percent = span('0% done');\n",
"    li.appendChild(percent);\n",
"\n",
"    outputElement.appendChild(li);\n",
"\n",
"    const fileDataPromise = new Promise((resolve) => {\n",
"        const reader = new FileReader();\n",
"        reader.onload = (e) => {\n",
"            resolve(e.target.result);\n",
"        };\n",
"        reader.readAsArrayBuffer(file);\n",
"    });\n",
"    // Wait for the data to be ready.\n",
"    let fileData = yield {\n",
"        promise: fileDataPromise,\n",
"        response: {\n",
"            action: 'continue',\n",
"        },\n",
"    };\n",
"\n",
"    // Use a chunked sending to avoid message size limits. See
b/62115660.\n",
"    let position = 0;\n",
"    do {\n",
"        const length = Math.min(fileData.byteLength - position,
MAX_PAYLOAD_SIZE);\n",
"        const chunk = new Uint8Array(fileData, position, length);\n",
"        position += length;\n",
"\n",
"        const base64 = btoa(String.fromCharCode.apply(null, chunk));\n",
"        yield {\n",
"            response: {\n",
"                action: 'append',\n",
"                file: file.name,\n",
"                data: base64,\n",
"            },\n",
"        };\n",
"\n",
"        let percentDone = fileData.byteLength === 0 ?\n",
"            100 :\n",
"            Math.round((position / fileData.byteLength) * 100);\n",
"        percent.textContent = `${percentDone}% done`;\n",
"\n",
"    } while (position < fileData.byteLength);\n",
"    }\n",
"\n",
"    // All done.\n",
"    yield {\n",
"        response: {\n",
"            action: 'complete',\n",
"        },\n",
"    };\n",
"}\n",
"\n",
"scope.google = scope.google || {};\n",
"scope.google.colab = scope.google.colab || {};\n",

```

```

        "scope.google.colab._files = {\n",
        "  _uploadFiles,\n",
        "  _uploadFilesContinue,\n",
        "};\n",
        "))(self);\n",
        "</script> "
    ]
},
"metadata": {}
},
{
    "output_type": "stream",
    "name": "stdout",
    "text": [
        "Saving ukraine240_F_143279703_vhfkoLstISCTmw2Uzj01vYhcUohe8Xgu.jpg to\n",
        "ukraine240_F_143279703_vhfkoLstISCTmw2Uzj01vYhcUohe8Xgu.jpg\n"
    ]
},
{
    "output_type": "display_data",
    "data": {
        "text/plain": [
            "<Figure size 600x600 with 1 Axes>"
        ],
        "image/png":
        "iVBORw0KGgoAAAANSUheUgAAAEQAAAFgCAYAAABqjwo3AAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLjguMCwgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy81sbWrAAAACXBIWXMAAA9hAAAPYQGoP6dpAAEAAE1EQve5DviLXvSi3bEf//Efd8Bvv/323bHHPe5xDvirX/3q3bH3ve99v1qt/Fu+5Vt2x7bbrddar7vH7bff7qvVyr/r\n",
        "u77rumOA//iP//gHfOZXvepVDvjP/dzP3efZvu7rv53rJTtH/MxH+Mi4t/7vd+7037p0iXf29vz5zznOdedOwz\n",
        "Ddfe5dOmSP+IRj/Cv/uqv3h37z//5PzvgL3nJS3bHaq3+t/7W37pP2//23/7b/rSnPc232+3umJn5Z3zGZ/jHfd\n",
        "zHfcBnXLDgwxWLhrzgIw7Xrl0D4OzZsx/\n",
        ".....\n",
        "wvWgoSGnmr4ATf2FOBZTveep4HOpjzaVOy5ML7Q3ukhqdbhXnFSh9EHiniK11oX3blq/fuFneSqneW0sAOHR42F\n",
        "CWqJeAakdrUyFsmKbVOSEkTLnUcLTHVklKS9iExCdyKRlSS3pRqRE1AqYm0pV7bKhmp0p5gkkGjEVSadL8Od5Oe\n",
        "fC9t36Pabq5LP7yfZNRpyK1H9fp06PCgoASiVARTgdQpOkp8Sv9uWngUbLSAJyCI2Ia0ykBTRTlRi0iTpqdtDFR\n",
        "OSuOzjXOjMTSsoP00wDqRU1hUQ+ozOg/4Q8Eqkv6D0EZ+EH5Dhw4PH230Uw3zohaLRFEBvZSRKvQ10VDqaYtdMM\n",
        "/eaNcPLLBRxx8RmqoYHHu1FyKrWm3Czra4L3iU2kjRR+lsOnTo0KFDhw4dOnTo0KFDhw4dOnwo0LmkOnTo0KFDh\n",
        "w4dOnTo0KFDhw4dOjxwdKRUhw4dOnTo0KFDhw4dOnTo0KFDhweOjpTq0KFDhw4dOnTo0KFDhw4dOnTo8MDRkVid\n",
        "OnTo0KFDhw4dOnTo0KFDhw4dHjg6UqpDhw4dOnTo0KFDhw4dOnTo0KHDA0dHSnXo0KFDhw4dOnTo0KFDhw4dOnR\n",
        "44OhIqQ4dOnTo0KFDhw4dOnTo0KFDhw4PHB0p1aFDhw4dOnTo0KFDhw4dOnTo0OGBoyOlOnTo0KFDhw4dOnTo0K\n",
        "FDhw4dOjxw/P8BMtv2ZICBrBwAAAAASUVORK5CYII=\n"
    ]
},
"metadata": {}
},
{
    "output_type": "stream",
    "name": "stdout",
    "text": [
        "Original Image Size: 30.23 KB\n"
    ]
}
]
}
]
}

```