МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна інженерія»

Група: 2БКС-29

КВАЛІФІКАЦІЙНА РОБОТА

здобувача освіти денної форми навчання

БКС.29.01.000.КРБ

БАЛАБАНА
ДАНИЛА ОЛЕГОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Комп'ютерна інженерія»**

Група: **2БКС-29**

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему: **Аналіз ефективності алгоритмів стискування без втрат у системах резервування даних**

Проектний матеріал складається з пояснювальної записки на 71 сторінках та графічного (презентаційного) матеріалу на 16 аркушах (слайдах)

Виконавець _____ (Балабан Д.О.)

Керівник проекту _____ (Кривченко Ю.В.)

Консультанти:

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувач кафедри _____ (Іванова Л.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «17» 06 2025 р. Протокол ЕК № 2

Оцінка ЕК 5 (відмінно) / 95

Секретар ЕК _____

АНОТАЦІЯ

У даній кваліфікаційній роботі розглядається проблема підвищення ефективності резервування даних шляхом аналізу алгоритмів стискування без втрат. Роботу присвячено дослідженню сучасних методів стискування, що використовуються для оптимізації збереження великих обсягів інформації, а також розробці програмного застосунку для вибору оптимального алгоритму в залежності від конкретних умов експлуатації.

У роботі виконано аналіз сучасних технологій, де розглядаються теоретичні основи алгоритмів стискування, характеристики систем резервного копіювання даних, розроблено та реалізовано застосунок для тестування ефективності алгоритмів стиснення, що включає опис використаних технологій (HTML, CSS, JavaScript, pako, lz4js, js-lzo, Metro UI) і обґрунтування вибору методів аналізу, створено відповідні блок-схеми алгоритмів та діаграми. Наведено опис проведених тестувань з вимірюванням часу стискання, визначенням стиснутого розміру, розрахунком коефіцієнта стискання та швидкості обробки даних, а також розробкою інтегрованої оцінки ефективності (Benchmark). Проведено аналіз отриманих даних, порівнюється ефективність різних алгоритмів та формулюються рекомендації щодо їх застосування у системах резервування даних.

Роботу спрямовано на аналіз саме алгоритмів стискування без втрат і створення інструмента, що дозволяє здійснювати об’єктивну оцінку їх ефективності, забезпечуючи тим самим основу для оптимізації процесів зберігання та резервування інформації в сучасних інформаційних системах. Проте при певній адаптації є можливість забезпечити тестування ефективності і алгоритмів з втратами, зокрема, призначених для стиснення мультимедіа-даних. Представлена робота має практичне значення, оскільки розроблена система дозволяє адаптувати стратегію стискування даних під конкретні вимоги користувача, забезпечуючи оптимальний баланс між швидкістю обробки та рівнем стиснення.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *Комп'ютерних систем* Кафедра *Комп'ютерної інженерії*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"28" 05 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

здобувачеві освіти *Балабану Данилу Олеговичу*

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Аналіз ефективності алгоритмів стискування без втрат у системах резервування даних*

затверджена наказом по коледжу від "14" 11 2024 р. № 246

2. Термін здачі студентом кваліфікаційної роботи

3. Вихідні дані до роботи *Аналізувати алгоритми стискування без втрат: Deflate, LZ4, LZO; Передбачити обробку файлів різних типів (текстові, бінарні, медіа); Передбачити налаштування пріоритетів між коефіцієнтом стиснення та швидкістю; Реалізувати обчислення основних показників (CR, T, S) та інтегральної оцінки (Benchmark); Передбачити виведення результатів у вигляді таблиці та графічної візуалізації; Передбачити наявність різних рівнів стиснення для порівняльного аналізу*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

Огляд систем резервного копіювання; Аналіз принципів стиснення даних; Огляд алгоритмів стискування без втрат; Інтеграція алгоритмів стискування у системи резервного копіювання Розробка застосунку для аналізу ефективності алгоритмів стискування без втрат; Експериментальний аналіз ефективності алгоритмів стискування без втрат

5. Перелік графічного матеріалу (слайдів мультимедійної презентації) *Загальна схема резервного копіювання; Порівняльний аналіз сучасних рішень резервного копіювання; Послідовність створення резервних наборів при різних методах копіювання; Схема процесу стиснення та відновлення даних; Блок-схема роботи алгоритмів сімейства LZ; Схема ентропійного кодування; Порівняльний аналіз класичних алгоритмів без втрат; Розрахунок ефективності стискування даних; Загальна БСА роботи застосунку; Приклади роботи застосунку для аналізу ефективності алгоритмів стискування; Результати тестування ефективності алгоритмів стискування без втрат*

6. Консультанти по кваліфікаційній роботі, із зазначенням розділів, що їх стосуються

Розділ	Консультант	ПІДПИС	
		Завдання видав	Завдання прийняв
Основний розділ	*Кривченко Ю.В.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 18.05.25

Керівник роботи *Кривченко Ю.В.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Вступ. Аналіз технічного завдання	02.06.25	виконав
2.	Огляд систем резервного копіювання	03.06.25	виконав
3.	Аналіз принципів стиснення даних	04.06.25	виконав
4.	Огляд алгоритмів стискування без втрат	06.06.25	виконав
5.	Інтеграція алгоритмів стискування у системи резервного	07.06.25	виконав
	копіювання		
6.	Розробка застосунку для аналізу ефективності алгоритмів	08.06.25	виконав
7.	Експериментальний аналіз ефективності алгоритмів	03.06.25	виконав
	стискування без втрат		
8.	Реалізація інтерфейсу застосунку	10.06.25	виконав
9.	Випробування застосунку	15.06.25	виконав
10.	Аналіз результатів тестування	07.06.25.	виконав
11.	Розробка питань з охорони праці та техніки безпеки	18.06.25	виконав
12.	Підготовка матеріалів мультимедійної презентації	20.06.25	виконав

Здобувач освіти _____
(підпис)

Керівник роботи _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 29. 01 000. 00 КРБ	Випускна робота		
А4			БКС 29. 01 000. 00 КРБ ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	БКС 29. 01 000. 00 КРБ		
Розробив		Балабан Д.О.		10.06	Аналіз ефективності алгоритмів стискування без втрат у системах резервування даних	Літ.	Аркуш 4 / Аркушів 71
Перевірив		Кривченко Ю.В.		10.06		Н Д П	
Н. Контр.		Петрашова В.І.		10.06.15		ВСП "ОТФК ОНТУ" гр.2БКС-29	
Затверд.		Іванова Л.В.					

ЗМІСТ

ВСТУП

Нині обсяг даних, що генерується у комерційних, наукових та особистих інформаційних системах, зростає в геометричній прогресії. Це створює численні виклики для організацій щодо зберігання, передачі та відновлення інформації. Одним із ключових напрямків підвищення ефективності використання ресурсів є резервне копіювання даних. Щоб забезпечити збереження важливої інформації, системи резервування використовують різноманітні підходи, серед яких особливе місце займають методи стиснення даних. У випадках, коли необхідно відновити дані з абсолютною точністю, незамінними є алгоритми стискування без втрат.

Дослідження ефективності алгоритмів стискування без втрат набуває актуальності з огляду на потребу оптимізації процесів резервного копіювання. Стиснення без втрат дозволяє зменшити обсяг збережених даних без спотворення інформації, що є особливо важливим для текстових документів, програмного коду, баз даних та інших критичних для відновлення систем файлів. Крім того, ефективне стиснення сприяє зменшенню витрат на зберігання, покращує швидкість передачі даних через мережі та знижує навантаження на систему, що дозволяє організаціям оперативно відновлювати робочий процес у випадку аварійних ситуацій.

Метою даної випускної роботи є аналіз ефективності алгоритмів стискування без втрат у системах резервування даних з подальшим формуванням критеріїв вибору оптимального рішення для сучасних інформаційних середовищ. Актуальність роботи обумовлена потребою компаній і державних установ у забезпеченні високої доступності даних за мінімальних витрат як з точки зору апаратних засобів, так і з огляду на витрати на передачу інформації в мережах. Ретельний аналіз алгоритмів стиснення без втрат дозволить виявити їх переваги та недоліки в реальних умовах експлуатації, сформувати картину взаємозв’язку між ступенем стиснення та швидкістю роботи, а також розробити рекомендації щодо вибору найбільш ефективного рішення для забезпечення безперебійного резервного копіювання. Робота спрямована на вирішення практичних завдань підвищення ефективності резервування даних у сучасних інформаційних системах, зокрема, за рахунок раціоналізації процесів стиснення без втрат.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Огляд систем резервного копіювання

Резервне копіювання виступає невід'ємною складовою стратегії захисту даних, дозволяючи мінімізувати ризик їх втрати вслед за апаратними або програмними збоями, шкідливими атаками чи людською помилкою. Основна ідея полягає у створенні копій важливих даних, які можуть бути відновлені в разі аварійної ситуації.

Системи резервного копіювання охоплюють широкий спектр технологій і підходів, що розвивалися від простих, фізично обмежених методів з використанням магнітних стрічок, дискет та оптичних дисків до сучасних програмних рішень, що інтегрують локальні та хмарні платформи. Різноманітність підходів дозволяє пристосувати метод резервного копіювання до специфічних умов експлуатації: від невеликих персональних комп’ютерів до масштабних корпоративних систем, представлених розподіленими хмарними сервісами.

1.1.1 Еволюція методів резервного копіювання

Історія резервного копіювання починається з часів, коли дані зберігалися на магнітних стрічках і дискетах. Попри обмежений обсяг, низьку швидкість доступу та високий ризик пошкодження фізичних носіїв, саме ці технології дали поштовх до формування практики створення резервних копій. З переходом до жорстких дисків з’явилася можливість автоматизувати процес резервного копіювання за допомогою програмних засобів. В цей період з’явилися перші комерційні системи резервного копіювання, що працювали з локальними мережами і здатні виконувати повне, диференціальне та інкрементне копіювання даних. У табл.1.1 показані як традиційні, так і сучасні носії даних, із зазначенням основних параметрів та характеристик. Подальший розвиток технологій і розповсюдження мереж зв’язку призвело до появи розподілених систем збереження даних. Дані стали зберігатись не лише локально, а й у вигляді реплік на окремих серверних вузлах у корпоративних мережах. Використання RAID-технологій теж значно підвищило відмовостійкість таких систем.

Таблиця 1.1. Порівняльна характеристика традиційних носіїв даних

Тип носія	Приблизна ємність	Швидкість доступу	Надійність	Основні переваги / недоліки
Магнітні стрічки	Від десятків ГБ до ТБ (для спеціалізованих архівних систем)	Послідовний доступ; час доступу – від декількох хвилин до секунд	Помірна (залежить від умов зберігання); підлягають старінню й можуть бути чутливими до вологи та температурних коливань	Висока ємність при низькій собівартості зберігання; але повільний доступ та потреба у спеціальному обладнанні роблять їх менш зручними для оперативного резервування.
Дискети	1,44 МБ (3,5-дюймові); інші версії – декілька сотень КБ	Дуже повільна; час доступу – декілька секунд	Дуже низька надійність, схильні до пошкоджень, магнітного розряду та фізичного зносу	Простота використання в ранніх ПК; обмежена ємність і швидкість доступу обумовлюють їх відсутність у сучасних системах.
Оптичні носії (CD/DVD, Blu-ray)	CD: до 700 МБ; DVD: 4,7–9,4 ГБ; Blu-ray: 25–50 ГБ	Доступ приблизно 100–200 мс	Стійкі до магнітних впливів, але чутливі до подряпин, ультрафіолетових променів та температурних змін	Легко транспортувати; відносно помірна швидкість доступу; обмежені можливості переписування, недосконале для частої зміни інформації.
Жорсткі диски (HDD)	Від сотень ГБ до декількох ТБ	Швидкість доступу – від десятків до сотень мс (механічна архітектура)	Набагато надійніші, ніж дискетні чи оптичні носії, проте мають механічні компоненти, що можуть зношуватись	Висока ємність та швидкість доступу при помірних витратах; однак механічна будова може стати причиною збоїв із часом.
Твердотільні накопичувачі (SSD)	Від десятків ГБ до декількох ТБ	Дуже високі, з часом доступу в мілісекундах або менш (відсутність рухомих частин)	Висока стійкість до механічних ушкоджень, але може бути чутливим до обмежень на кількість циклів запису	Висока швидкість, низька латентність; компактність та невразливість до механічного зносу роблять їх оптимальним рішенням для сучасних систем, хоча вартість за ГБ вища порівняно з HDD.
USB флеш-накопичувачі	Від кількох ГБ до декількох сотень ГБ	Відносно висока (залежно від стандартів USB, зазвичай від десятків до сотень МБ/с)	Зазвичай достатньо надійні для перенесення даних; однак можуть бути чутливими до інтенсивного перезапису	Легкі, портативні та універсальні; прості у використанні, проте їх продуктивність і довговічність можуть змінюватись в залежності від якості контролера та флеш-пам’яті.
Хмарні сховища	Практично не обмежені (залежить від тарифного плану); від декількох сотень ГБ до пітабайтів	Залежить від пропускної здатності мережі та серверної інфраструктури; може змінюватись від сотень мс до секунд	Дуже висока, при належній організації зберігання та резервування; без необхідності фізичного доступу	Забезпечують високий рівень доступності та відмовостійкості, можливість географічно розподіленого зберігання; однак залежать від якості інтернет-з'єднання та можуть бути обмежені тарифами на передачу даних.

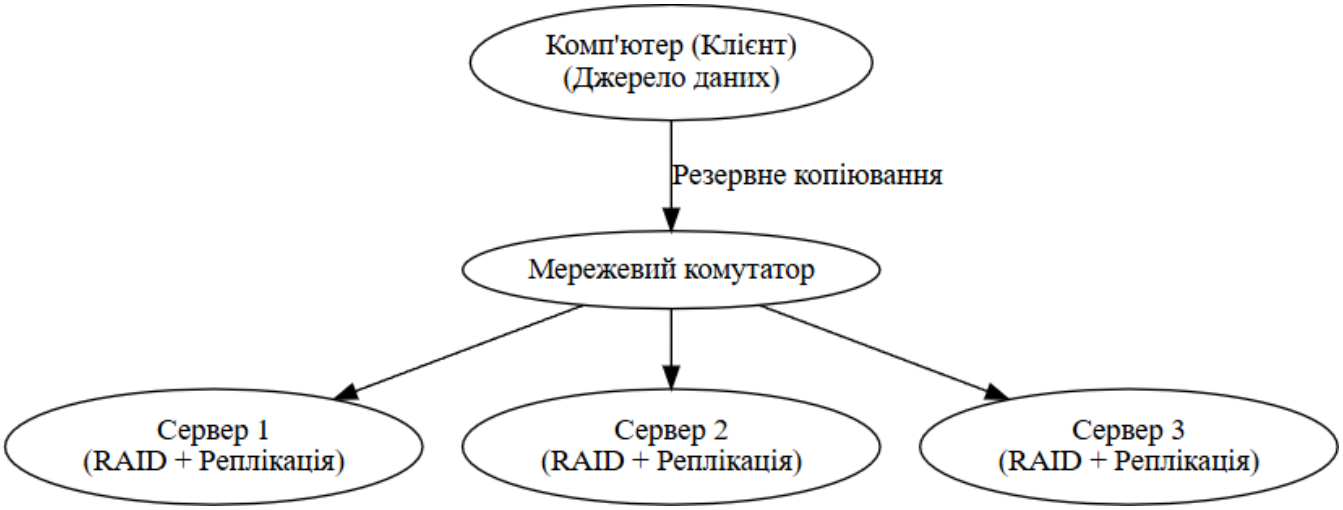


Рисунок 1.1. Блок-схема розподіленої системи резервного копіювання

На рис. 1.1 показано, як дані з одного комп’ютера копіюються на кілька серверів із використанням технологій реплікації та RAID. Останнім часом завдяки стрімкому розвитку інтернет-технологій і хмарних сервісів, методи резервного копіювання набули нового виміру. Сучасні системи дозволяють виконувати резервне копіювання у режимі реального часу, забезпечують географічно розподілене зберігання копій і підвищують рівень захисту даних від зовнішніх загроз. Крім цього, гібридні рішення, що поєднують локальне резервування з хмарним, дають змогу знизити витрати на інфраструктуру й оптимізувати швидкість відновлення.

У табл.1.2 порівняно основні характеристики сучасних рішень: «Локальні системи», «Хмарні платформи», «Гібридні рішення» за параметрами надійності, доступності, масштабованості та вартості впровадження.

На рис. 1.2 зображено загальну схему резервного копіювання, що демонструє мультишаровий підхід до збереження даних. Дані спочатку надходять з основного комп’ютера (джерела даних), після чого за допомогою мережевого комутатора вони розподіляються до різних систем резервного копіювання. Серед них:

- Локальна система резервного копіювання – зберігає копії на жорстких дисках або пристроях NAS, що забезпечує оперативний доступ до даних.
- Мережеве резервне копіювання – базується на використанні FTP або SMB серверів для централізованого зберігання даних у межах організації.
- Хмарні платформи – забезпечують віддалене зберігання даних з високою

доступністю та масштабованістю завдяки використанню інтернет-технологій. Крім того, всі ці системи можуть бути інтегровані з офсайт-резервуванням, яке передбачає переведення копій даних на віддалені сервери для додаткового захисту від локальних аварійних ситуацій.

Таблиця 1.2. Порівняльний аналіз сучасних рішень резервного копіювання

Параметр	Локальні системи	Хмарні платформи	Гібридні рішення
Надійність	• Використання RAID-технологій для відмовостійкості. • Захист даних у межах окремої організації. • Ризик локальних аварій (пожежа, крадіжка, несправності обладнання).	• Георозподілення та реплікація дають високу стійкість. • Автоматичне резервування даних у декількох дата-центрах. • Залежність від надійності стороннього провайдера.	• Поєднання переваг локального зберігання (оперативний доступ) та хмарного резервування (віддалене захищене копіювання). • Зниження ризику як локальних, так і зовнішніх аварій.
Доступність	• Швидкий доступ до даних через локальну мережу. • Обмеження доступу – лише в межах організації. • Може не забезпечувати круглодобову доступність, якщо не налаштовано додаткові засоби.	• Висока доступність 24/7 з будь-якого місця за умови стабільного інтернет-з'єднання. • Залежність від пропускної здатності каналу та стабільності мережі.	• Забезпечується як швидкий локальний відгук, так і доступність через інтернет. • Можливість резервного відновлення даних з віддалених копій у випадку аварій.
Масштабованість	• Розширення вимагає придбання додаткового обладнання та налаштування інфраструктури. • Фізичні та бюджетні обмеження місцевої інфраструктури.	• Практично необмежене масштабування за умов тарифного плану. • Ресурси можуть динамічно збільшуватися без значних початкових інвестицій.	• Гнучкість: локальна інфраструктура обслуговує щоденні потреби, а хмарні сервіси – архівування і масштабування при зростанні обсягів даних. • Баланс між витратами та ресурсами.
Вартість впровадження	• Висока первинна інвестиція: придбання серверів, обладнання, ліцензійного програмного забезпечення. • Експлуатаційні витрати (електроенергія, обслуговування, проста зброя).	• Модель «оплата за використання» з низькими початковими витратами, але потенційно високими щомісячними платежами при великому обсязі даних. • Відсутність потреби в експлуатації власної інфраструктури.	• Оптимізація витрат: поєднання власних ресурсів з хмарними сервісами дозволяє зменшити як первинні, так і операційні витрати. • Гнучкість у налаштуванні залежно від конкретних потреб організації.

Еволюція методів резервного копіювання демонструє поступовий перехід від простих, фізично обмежених технологій з високим ризиком втрати даних до сучасних, інтелектуальних систем, що забезпечують автоматичне резервування,

відновлення у режимі реального часу і масштабованість до великих обсягів інформації.

Рисунок 1.2. Загальна схема резервного копіювання

1.1.2 Основні типи резервного копіювання

Резервне копіювання даних є критично важливою складовою інформаційної безпеки сучасних систем, оскільки воно дозволяє відновити дані після виникнення аварійних ситуацій. Основні види резервного копіювання характеризуються тим, як саме зберігаються копії даних та яким чином враховуються зміни у вмісті з часом. Одним із традиційних методів є повне резервне копіювання, за якого кожного разу створюється копія всіх даних незалежно від того, чи були вони змінені від останнього знімку. Цей метод відзначається простотою відновлення, оскільки дані зберігаються у єдиному наборі, проте він вимагає значного обсягу сховищ і часу через дублювання всієї інформації.

Ключовим альтернативним підходом є диференціальне резервне копіювання, при якому зберігаються лише ті зміни, які внесені після останнього повного резервного копіювання. Цей метод дозволяє зменшити обсяг збережених даних, але з часом розмір диференціальних копій може збільшуватися, що впливає на загальні витрати на зберігання. Ще одним широко використовуваним методом є інкрементне резервне копіювання, в якому зберігаються лише дані, що змінилися від моменту

створення попереднього копіювання, незалежно від того, було воно повним або попереднім інкрементним. Хоча інкрементне копіювання забезпечує найменший обсяг нових даних, які потрібно зберігати, для відновлення системи необхідно мати останню повну копію та всю послідовність інкрементних змін, що може ускладнювати процес відновлення.

Окрім цих традиційних методів, сучасні системи резервного копіювання активно впроваджують технологію знімків (snapshot), яка полягає у створенні моментальних копій стану даних на рівні файлової системи або блочного пристрою. Застосування знімків дозволяє отримати резервну копію за лічені секунди, що особливо важливо для систем, де критичний час відновлення. Нарешті, існує підхід дзеркалювання, коли дані синхронно копіюються негайно, створюючи точне дзеркало первинного середовища. Цей метод спрямований на забезпечення миттєвого відновлення, проте вимагає значних ресурсів для зберігання, оскільки дублюється весь обсяг інформації.

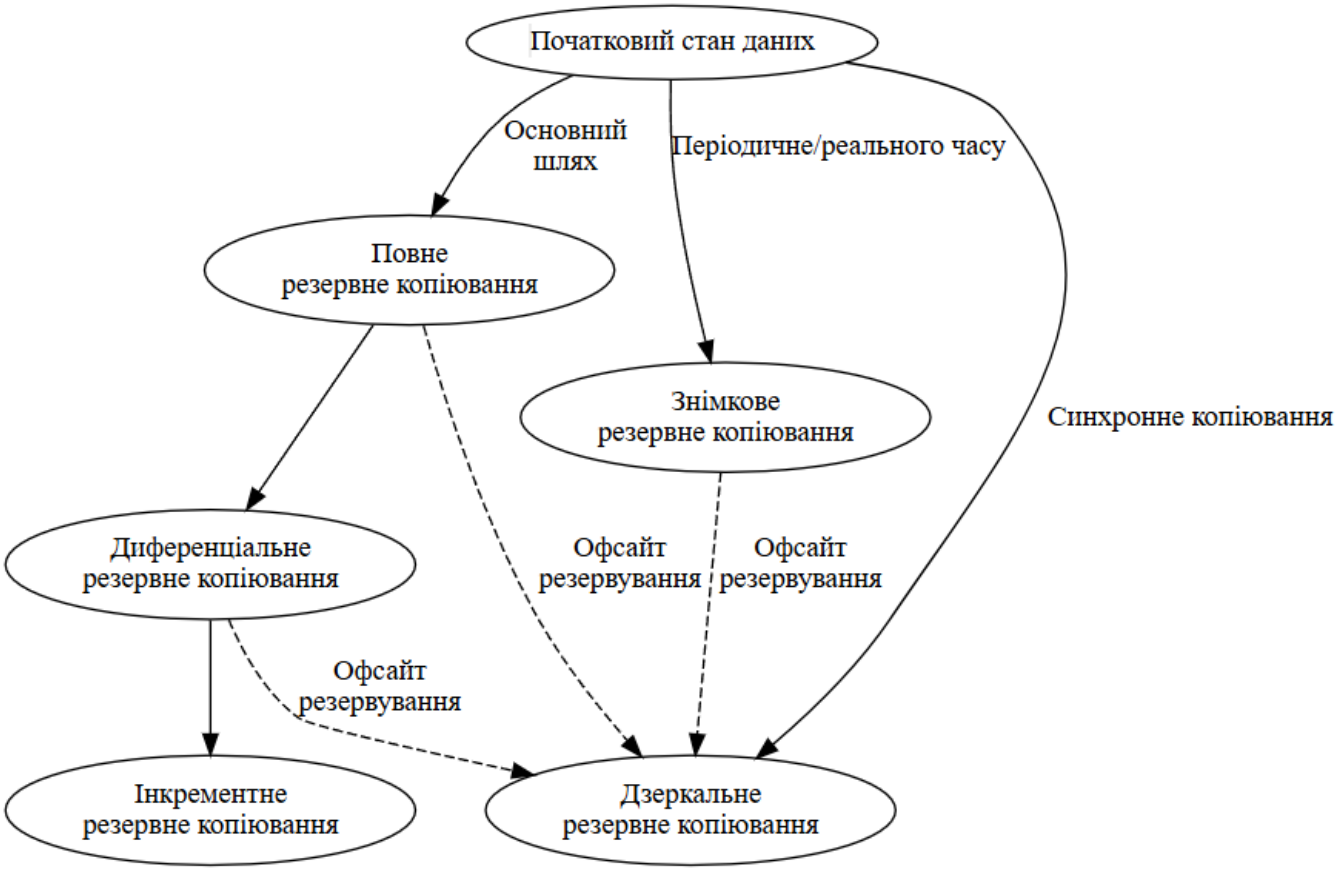


Рисунок 1.3. Послідовність створення резервних наборів при різних методах копіювання

На рис. 1.3 показано послідовність створення резервних наборів при різних методах копіювання. За допомогою даної схеми демонструється як основний шлях резервного копіювання доповнюється паралельними методами (знімки та дзеркальне копіювання), що в комплексі дозволяють оптимально захищати дані. Схема ілюструє такі процеси:

- Початковий стан даних: вихідна інформація, яку необхідно зберегти;
- Повне резервне копіювання: створюється повна копія всіх даних, що є базою для подальших операцій;
- Диференціальне резервне копіювання: копіюються лише зміни, які відбулися після останнього повного резервного копіювання;
- Інкрементне резервне копіювання: зберігаються лише дані, що змінилися з останнього резервного копіювання (повного чи попереднього інкрементного);
- Знімкове резервне копіювання: створюється моментальний знімок стану даних, що дозволяє дуже швидко зафіксувати стан системи;
- Дзеркальне резервне копіювання: синхронне відображення даних у режимі реального часу з можливістю офсайт-розташування, що забезпечує додатковий рівень захисту від аварій локального характеру.

У табл. 1.3 показано порівняльну характеристику параметрів кожного типу – обсягом збереження, часом відновлення, витратами на зберігання та основними перевагами й недоліками кожного методу.

Таким чином, вибір типу резервного копіювання залежить від конкретних потреб системи: якщо пріоритетом є швидке відновлення, доцільним буде використання повного або дзеркального копіювання; якщо головною метою є оптимізація обсягу сховища, то ефективніше застосовувати диференціальні або інкрементні підходи; а технологія знімків дозволяє забезпечити високий рівень операційної доступності при коротких затримках.

Вибір комбінованих рішень дуже часто спрямовується на досягнення балансу між надійністю, ефективністю зберігання та швидкістю відновлення в умовах сучасних інформаційних систем.

Таблиця 1.3. Порівняльний аналіз типів резервного копіювання

Тип резервного копіювання	Обсяг збереження	Час відновлення	Витрати на зберігання	Основні переваги	Основні недоліки
Повне резервне копіювання	Найвищий – завжди дублюється весь обсяг даних	Швидке – відновлення здійснюється з однієї повної копії	Високі – зберігається повна копія кожного разу	Простота відновлення, повнота інформації	Значний обсяг збережених даних, тривалість процесу створення копії при великих обсягах
Диферен-ціальне резервне копіювання	Середній – зберігаються лише зміни після останнього повного резервного копіювання	Середній – для відновлення потрібні повна копія та останній диференціальний набір	Середні – обсяг збережених даних зростає між повними копіями	Менший обсяг даних порівняно з повним резервним копіюванням; відносно просте відновлення (тільки 2 файли: повна та остання диференціальна)	З часом розмір диференціальної копії може зростати, що ускладнює процес відновлення і може впливати на витрати на зберігання
Інкрементне резервне копіювання	Найнижчий – копіюються лише дані, що змінилися від останнього резервного копіювання	Найдовший – для відновлення необхідно послідовно обробити повну копію та всі інкрементні набори	Найнижчі – кожний інкремент зберігає мінімальну кількість даних	Мінімальний обсяг збереження на кожен резервний набір, що знижує витрати на зберігання; швидке виконання резервного копіювання	Відновлення може бути складним і тривалим, оскільки потребує послідовної обробки всіх наборів; втрата одного інкременту може ускладнити відновлення
Знімкове резервне копіювання (Snapshot)	Низький – зберігаються лише інформація про зміни (відображення стану даних на рівні системи)	Дуже швидке – майже миттєве відновлення за рахунок знімка	Помірні – залежить від технології зберігання «знімків» (часто використовується технологія копіювання блоків)	Миттєве створення та відновлення резервної копії, мінімальний вплив на продуктивність системи	Обмеження масштабованості та залежність від апаратних засобів; не завжди дає можливість точно контролювати обсяг збережених змін
Дзеркальне резервне копіювання (Mirror backup)	Високий – створюється точна копія всіх даних	Миттєве – доступ до даних з дзеркального копіювання дозволяє швидке відновлення	Дуже високі – потрібно зберігати повну копію всіх даних	Миттєве відновлення, висока актуальність даних, забезпечення безперервності роботи системи	Необхідність виділення великої кількості сховища, високі початкові та експлуатаційні витрати; дублювання всіх даних може бути економічно невигідним

1.1.3 Вимоги до резервування в сучасних інформаційних системах

Сучасні інформаційні системи характеризуються високою динамікою змін, зростанням обсягів даних та взаємодією численних додатків і користувачів, що обумовлює необхідність впровадження високоефективних механізмів резервування. Резервування даних повинно відповідати строгим вимогам щодо надійності, доступності та швидкості відновлення інформації, забезпечуючи безперебійність роботи системи в умовах змін та аварійних ситуацій. Серед ключових вимог відзначається здатність системи резервування зберігати копії даних з максимальною точністю, щоб при потребі відновлення інформація відтворювалася без будь-якого спотворення чи втрати деталей, що є особливо важливим для критично важливих застосунків, де кожен біт має значення.

Крім цього, сучасні рішення резервування повинні забезпечувати високий рівень доступності збережених даних, що дозволяє оперативно отримувати доступ до резервних копій як у нормальних умовах роботи, так і під час надзвичайних ситуацій. Це означає, що система повинна бути інтегрованою з сучасною мережею, підтримувати як локальні, так і віддалені (офсайт) резервні копії, реалізовуючи можливості географічного розподілення ресурсів, що суттєво підвищує стійкість до локальних збоїв. Вимоги до швидкого відновлення даних визначають можливість мінімізувати час простою при відновленні після аварії, що напряму впливає на бізнес-процеси та операційну безперервність організації. Особливу увагу приділяють гнучкості при масштабуванні – система резервування має бути здатна адаптуватись до зростаючих обсягів інформації без значних витрат на реконфігурацію апаратних засобів або суттєвої модифікації програмного забезпечення.

Критично важливими є також економічна ефективність і інтеграція резервного копіювання у загальну IT-стратегію організації. Це означає не лише оптимальний розподіл витрат на зберігання даних, але й забезпечення безпечного доступу та шифрування інформації, що копіюється, для запобігання несанкціонованому доступу чи втручанню. При цьому система має відповідати нормативним вимогам і стандартам у сфері інформаційної безпеки, що визначають

необхідний рівень захисту даних, та забезпечувати можливість централізованого управління резервними копіями.

Одночасно з цим, резервне копіювання має бути автоматизованим і прозорим для кінцевих користувачів, щоб мінімізувати людський фактор, який часто є джерелом помилок, а також підтримувати можливість моніторингу процесу резервування та надання звітності щодо стану системи.

1.2 Аналіз принципів стиснення даних

Стиснення даних — це процес скорочення обсягу інформації шляхом видалення надлишкових чи повторюваних даних, а також використання ефективних методик кодування. Головна мета полягає у зменшенні розміру файлу без (або з прийнятною втратою) якості, що дозволяє оптимізувати використання сховищ даних та зменшити час передачі інформації. Завдання стиснення включають підвищення ефективності зберігання, економію пропускної здатності каналів зв'язку і забезпечення швидкого відновлення даних. При цьому важливим аспектом є збереження цілісності інформації: у випадках, де кожен біт має значення (наприклад, текстові документи або програмний код), необхідно застосовувати методи стиснення без втрат, а у мультимедійних додатках – допускаються компроміси між розміром та якістю, що реалізується через стиснення зі втратами.

Методи стиснення можна поділити на дві основні категорії:

- Стиснення без втрат (lossless compression). Цей підхід дозволяє відновити оригінальні дані без жодних змін після розпакування. Алгоритми, такі як LZ77, LZ78, LZW, коди Хаффмана та арифметичне кодування, широко застосовуються для стиснення текстових файлів, баз даних та інших документів, де збереження кожного байта є критично важливим;
- Стиснення зі втратами (lossy compression). При цьому методі певна частина даних відкидається з метою значного зменшення розміру файлу, що особливо актуально для мультимедійних даних (зображень, аудіо, відео). Прикладом можуть служити алгоритми JPEG для зображень та MP3 для аудіо, де невідновлювана втрата деякої якості прийнятна за рахунок значного зменшення обсягу збереженої інформації.

Ефективність алгоритмів стиснення оцінюється за такими параметрами:

- Коефіцієнт стиснення (Compression Ratio, CR): Він визначається як відношення розміру початкових даних до розміру стиснених даних:

$$CR = \frac{R_{\text{вихідний}}}{R_{\text{стиснений}}} \quad (1.1)$$

Чим вищий цей показник, тим ефективніше працює алгоритм;

- Швидкість стиснення і розпакування: Цей параметр вимірюється у мегабайтах за секунду (MB/s) і характеризує, за який час алгоритм здатний обробляти дані. Швидкість є критичним фактором у режимах реального часу;
- Ресурсоємність: Оцінює витрати обчислювальних ресурсів (процесорного часу, оперативної пам'яті) під час стиснення та розпакування. В умовах обмежених ресурсів або високонавантажених систем важливим є баланс між ступенем стиснення та використанням CPU/RAM.

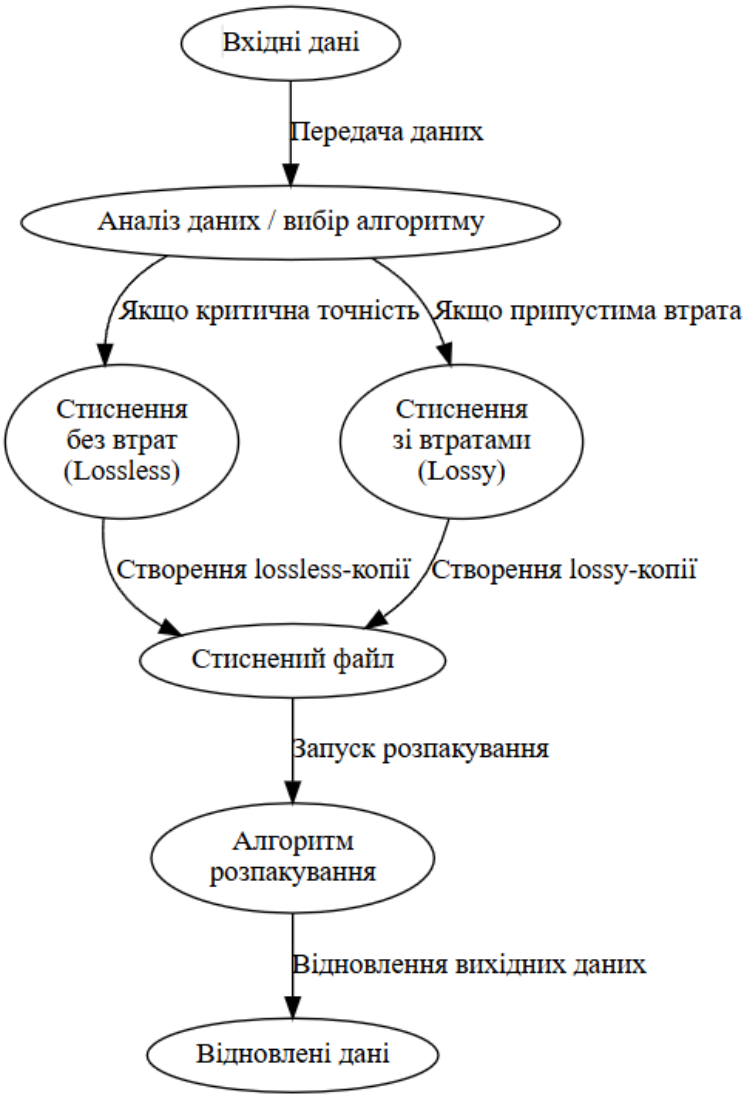


Рисунок 1.4. Схема процесу стиснення та відновлення даних

На рис.1.4 зображено вхідні дані, які піддаються обробці алгоритмом стиснення. Після цього дані перетворюються у стиснений формат та зберігаються. Для відновлення вихідної інформації використовується алгоритм розпакування, який у випадку стиснення без втрат повністю відновлює початковий стан даних, а у випадку стиснення зі втратами – забезпечує максимально наближену до оригіналу якість.

Таблиця 1.4. Порівняльна характеристика методів стиснення

Параметр	**Стиснення без втрат**	**Стиснення зі втратами**
Основне застосування	Текстові дані, програмний код, бази даних – де важлива абсолютна точність	Мультимедійні дані (зображення, аудіо, відео) – де деякі втрати даних прийнятні для суттєвого зменшення розміру файлу
Алгоритми	LZ77, LZ78, LZW, коди Хаффмана, арифметичне кодування	JPEG, MP3, MPEG, OGG тощо
Коефіцієнт стиснення (CR)	Зазвичай від 2:1 до 4:1 (залежно від структури даних)	Може досягати значень понад 10:1, залежно від параметрів якості та ступеня компресії
Швидкість стиснення/розпакування	Висока – завдяки алгоритмічним оптимізаціям; збереження всіх даних вимагає іноді додаткового часу	Зазвичай швидший процес стиснення; розпакування може бути трохи спрощеним за рахунок втрати частини інформації
Ресурсоємність	Вимагає розширених обчислювальних ресурсів для досягнення оптимальних коефіцієнтів стиснення	Менше ресурсоємний, однак компроміс між якістю та обсягом досягається шляхом вибору параметрів стиснення
Якість відновлення	Повне відновлення вихідних даних (100%)	Відновлення з втратами, якість залежить від ступеня компресії; дещо спотворені дані можуть бути непомітними для людського ока, але критично важливих даних не відновлюють

1.3 Огляд алгоритмів стискування без втрат

Стиснення без втрат є критично важливим для систем архівування, резервного копіювання, зберігання баз даних та програмного забезпечення, адже відновлення початкової інформації повинно здійснюватися з абсолютною точністю. Основна ідея цих алгоритмів полягає у виявленні та усуненні надлишковості в даних через алгоритмічне кодування, що дозволяє отримати скорочену форму інформації, з якої при розпакуванні можливо повністю відновити оригінал.

Цей підрозділ має на меті детально розглянути принципи роботи алгоритмів стискування без втрат, зокрема методи, засновані на алгоритмах сімейства LZ

(LZ77, LZ78, LZW), а також на ентропійному кодуванні, таким як коди Хаффмана та арифметичне кодування. Розглядаються підходи до формування кодувальних таблиць, алгоритмічні особливості побудови дерева частот, а також способи оптимізації обчислювальних процесів для забезпечення високого коефіцієнта стиснення при збереженні цілісності даних. Окрім цього, увага приділяється аналізу основних метрик ефективності алгоритмів стискування, до яких відносяться коефіцієнт стиснення, швидкість обробки даних та ресурсоємність обчислювальних процесів. Застосування математичних формул для розрахунку показників, наприклад, визначення коефіцієнта стиснення як відношення вихідного розміру даних до розміру стисненого файлу, дозволяє провести порівняльний аналіз різних підходів і визначити оптимальні рішення для конкретних завдань.

1.3.1 Аналіз алгоритмів сімейства LZ

Алгоритми сімейства LZ належать до групи методів, що базуються на створенні словника повторюваних шаблонів для зменшення надлишковості даних. У основу цих алгоритмів покладено ідею пошуку та заміни повторюваних послідовностей символів посиланням на попередньо збережені дані, що дозволяє суттєво скоротити обсяг вихідної інформації. Загальність принципу полягає в тому, що замість прямого зберігання повторюваних рядків даних формується таблиця (словник), де кожна унікальна послідовність зберігається лише один раз, а з неї створюються посилання для подальшої заміни.

Перший алгоритм із цього сімейства, LZ77, використовує «ковзне вікно», яке ділиться на два сегменти: вже оброблену частину даних та поточну область введення. При аналізі вхідного потоку алгоритм шукає найбільший співпадіння між поточними символами та рядками, що вже з’являлися в документі, та заміняє їх посиланням, яке містить позицію співпадіння та довжину повторення. Такий підхід дозволяє досягати економії пам’яті, оскільки повторювані патерни не зберігаються вдруге, а лише вказуються за допомогою коротких позиційних кодів.

Алгоритм LZ78 трохи відрізняється від LZ77. Він будує словник у режимі «потоку», не користуючись постійно змінним вікном, а додаючи нові фрази до словника в міру їх появи в потоці даних. Кожна нова фраза формується як

розширення існуючої, вже збереженої послідовності. Такий підхід полегшує процес збереження та пошуку інформації у словнику, що, в свою чергу, сприяє оптимізації процесу стиснення для даних із високо структурованою інформацією.

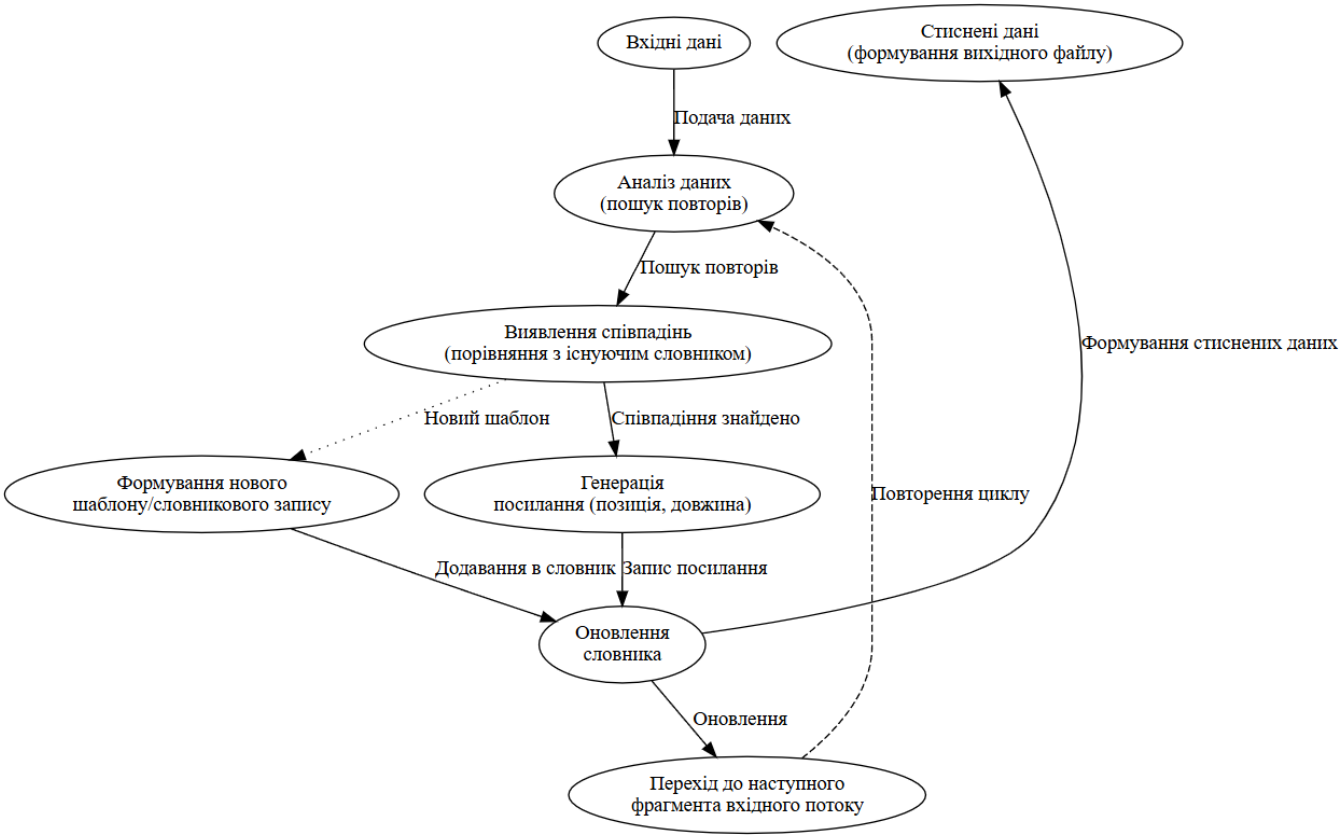


Рисунок 1.5. Блок-схема алгоритмічного процесу роботи алгоритмів сімейства LZ

Подальшим розвитком концепції став алгоритм LZW (Lempel-Ziv-Welch), який представляє собою розширення і удосконалення підходу LZ78. Метод LZW оптимізує процес оновлення словника без необхідності передачі окремих символів, адже нові записи формуються шляхом конкатенації та розширення вже існуючих шаблонів. Завдяки своїй ефективності та високій швидкості роботи алгоритм LZW був адаптований для використання у багатьох стандартних форматах стиснення, таких як GIF та деякі версії TIFF.

Основні переваги алгоритмів сімейства LZ полягають у їх універсальності та здатності ефективно стискати дані, де спостерігаються повторювані патерни. Вони демонструють високу продуктивність при стисненні текстових файлів, програмного коду і навіть деяких випадках мультимедійних даних, коли необхідно зберегти точну інформацію без втрат. Однак слід мати на увазі, що вибір конкретного алгоритму залежить від характеру даних: LZ77 добре підходить для потокових

даних завдяки використанню ковзного вікна, тоді як LZ78 та LZW більш ефективні для файлів з повторюваною структурою, де побудова словника відбувається поступово.

Таблиця 1.5. Порівняльна характеристика алгоритмів сімейства LZ

Алгоритм	Коефіцієнт стиснення	Швидкість стиснення/ розпакування	Ресурсо-ємність	Основні переваги	Основні недоліки
LZ77	2:1 – 4:1 (залежно від даних)	Висока – ефективний для потокової обробки завдяки використанню ковзного вікна	Помірна – обмеження розміру вікна можуть впливати на обчислювальні витрати	Динамічний пошук повторів; добре працює з потоковими даними	Обмеження максимального розміру буфера; не завжди ефективний для даних з низькою повторюваністю
LZ78	2:1 – 4:1	Висока – ефективний для структурованих даних, де легко формуються повторювані шаблони	Середня – поступове нарощення словникової структури може вимагати додаткової пам’яті	Простота реалізації; добрий компроміс між швидкістю та ефективністю	Залежність від якості формування словника; можливе збільшення затрат пам’яті при дуже великих обсягах даних
LZW	2:1 – 5:1	Дуже висока – оптимізоване управління словником дозволяє швидкий доступ до потрібних елементів	Низька – адаптивне управління словниковою структурою сприяє ефективному використанню ресурсів	Висока продуктивність; широко застосовується у стандартах стиснення (GIF, TIFF)	Обмеження розміру словника може вплинути на ефективність при нетрадиційних даних; деградація ефективності при зростанні обсягів даних

На рис. 1.5 показано блок-схему, яка демонструє алгоритмічну послідовність операцій від пошуку повторюваних рядків вхідних даних до створення та оновлення словникових структур. У табл. 1.5 порівнюються основні характеристики таких алгоритмів за параметрами швидкості, ефективності стиснення та обчислювальних ресурсів.

Аналіз алгоритмів сімейства LZ демонструє, що ефективність цих методів без втрат визначається не лише здатністю виявляти повтори, а й організацією управління словником, що в кінцевому рахунку впливає як на коефіцієнт стиснення, так і на швидкість обробки даних у реальному часі.

1.3.2 Аналіз ентропійних методів

Ентропійні методи є основною групою алгоритмів стиснення даних, заснованих на понятті інформаційної ентропії, яке було введено Клодом Шенноном. Ідея цих методів полягає в тому, що кожному символу повідомлення можна асоціювати певну кількість інформації, яка залежить від ймовірності його появи. Таким чином, символи, що зустрічаються частіше, кодуються коротшими кодами, а рідше зустрічаються – довшими, що дозволяє зменшувати середню довжину коду та досягати високого ступеня стиснення без втрат.

Основною величиною, що характеризує кількість інформації, є ентропія, яка обчислюється за формулою Шеннона:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p\,(x_i), \qquad (1.2)$$

де $p(x_i)$ — ймовірність появи символу x_i, а n — кількість різних символів. Цей показник визначає мінімальну середню кількість бітів, необхідну для кодування повідомлення, що є теоретичним лімітом для будь-якого алгоритму стиснення без втрат.

Найпоширенішими ентропійними методами є алгоритми кодування Хаффмана та арифметичне кодування. Алгоритм Хаффмана ґрунтується на побудові бінарного дерева (дерева Хаффмана), в якому кожен лист відповідає символу повідомлення, а вага листа — його ймовірності. Процес побудови дерева починається з упорядкування символів за їх частотою появи. Далі, за принципом "злиття" двох найменш імовірних символів створюється новий вузол, вага якого дорівнює сумі ваг його нащадків. Таку операцію повторюють до тих пір, поки не буде отримане єдине дерево. Результатом є унікальне префіксне кодування, яке гарантує, що жоден закодований символ не є префіксом іншого, що спрощує процес декодування. Однією з основних переваг цього методу є простота реалізації та висока ефективність при досить хорошому апроксимуванні теоретичного ліміту ентропії. Проте, при дуже неоднорідних розподілах ймовірностей або змінах статистики даних алгоритм може бути не оптимальним, оскільки коди не завжди будуть максимально адаптованими до локальних особливостей потоку.

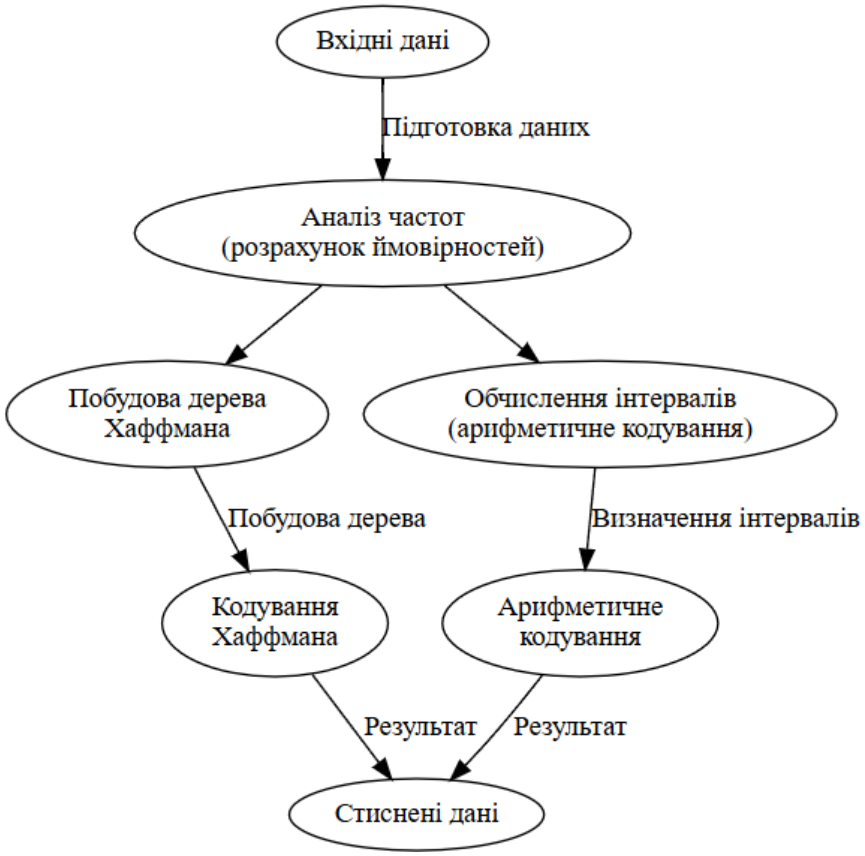


Рисунок 1.6. Схема ентропійного кодування

На відміну від алгоритму Хаффмана, арифметичне кодування не кодує кожний символ окремо, а розглядає весь потік як одне число, яке знаходиться в певному інтервалі від 0 до 1. Використовуючи інформацію про ймовірнісні розподіли символів, алгоритм поступово зменшує цей інтервал зі збільшенням кількості оброблених символів. У результаті отримується дробове число, яке точно відображає послідовність символів. Арифметичне кодування здатне досягати ефективності, наближеної до теоретичного ліміту ентропії, оскільки воно дозволяє використовувати дробові значення бітів для опису інформації. Проте, його реалізація є більш складною як з математичної, так і з обчислювальної точки зору; алгоритм вимагає високоточного арифметичного обчислення та ретельного менеджменту обчислювальних ресурсів, що може стати проблемою у системах з обмеженими ресурсами.

Обидва методи знаходять своє застосування в різних областях. Кодування Хаффмана завдяки своїй простоті та низьким обчислювальним витратам використовується, наприклад, в алгоритмах стиснення зображень (JPEG, PNG) та текстових файлах, коли важлива швидкість роботи. Арифметичне кодування,

незважаючи на його складність, є більш гнучким і дозволяє досягти вищого ступеня стиснення при складних розподілах даних, що робить його привабливим для застосувань у високоефективних системах стиснення.

На рис.1.6 показано основні етапи ентропійного стиснення. Ця схема демонструє, як, використовуючи аналіз частот, можна розгалужувати процес стиснення на два основних підходи, кожен із яких має свої переваги залежно від вимог до ефективності та обчислювальних ресурсів.

Ефективність ентропійних методів визначається такими параметрами, як коефіцієнт стиснення, обчислювальна складність та адаптивність до змін у статистиці вхідних даних. Застосовуючи поняття ентропії, можна точно оцінити потенціал стиснення для будь-якого заданого джерела даних і підібрати оптимальний алгоритм для конкретної задачі.

Аналіз ентропійних методів дозволяє зробити висновок, що алгоритми, які базуються на принципах теорії інформації, є дуже потужним інструментом для стиснення даних без втрат. Вони забезпечують теоретично оптимальне кодування, пристосовуючись до статистичних особливостей вхідних даних, що є запорукою високої ефективності. Однак вибір конкретного методу – Хаффмана чи арифметичного кодування – залежить від специфічних вимог до системи стиснення: швидкості роботи, обчислювальних ресурсів та необхідного ступеня адаптивності до змінних умов експлуатації.

1.3.3 Аналіз адаптивних методів стиснення

Адаптивні методи стиснення даних відзначаються здатністю динамічно коригувати свої моделі в залежності від властивостей вхідного потоку, що дозволяє ефективно пристосовуватися до змін статистики даних. Серед таких методів ключове місце займають алгоритми, засновані на контекстному моделюванні (наприклад, PPM), трансформації даних для полегшення подальшого стиснення (BWT) та методи, що використовують прості схеми заміни повторюваних послідовностей (RLE). Нижче наведено детальний аналіз основних адаптивних методів стиснення.

PPM є статистичним методом стиснення, заснованим на ідеї контекстного

прогнозування: для кожного символу аналізуються попередні контексти певної довжини (або їх комбінації), що дозволяє оцінити ймовірність появи кожного можливого наступного символу. При цьому:

- Алгоритм динамічно будує модель, що захоплює локальні статистичні закономірності, і використовує арифметичне кодування для отримання компактного представлення;
- Основною перевагою PPM є здатність до високої ефективності стиснення, що наближається до теоретичного ліміту ентропії, особливо при обробці текстових даних або іншої інформації з виразною контекстною структурою;
- Недоліками є висока обчислювальна складність та значні вимоги до оперативної пам’яті при використанні моделей високого порядку, що може впливати на продуктивність у реальному часі.

BWT не є алгоритмом стиснення сам по собі, а представляє собою перетворення даних, яке “переставляє” символи таким чином, що однакові або подібні символи групуються разом. Основні характеристики цього методу:

- Трансформація: BWT перетворює вхідний рядок, створюючи матрицю усіх циклічних перестановок, яку потім лексикографічно сортують. Результатом трансформації є рядок, в якому повторювані символи розташовані близько один до одного;
- Подальше стиснення: Отриманий після трансформації рядок легко піддається додатковому стисканню за допомогою методів, таких як RLE (Run-Length Encoding) або Move-to-Front (MTF) кодування, що дає змогу досягти значного зменшення розміру даних;
- Основною перевагою BWT є те, що вона перетворює дані у форму, оптимізовану для подальшого стиснення, а недоліком – додатковий етап декодування для відновлення початкової послідовності, що може збільшувати загальний час обробки.

RLE є одним з найпростіших методів стиснення, заснованих на заміні послідовностей однакових символів на один символ і число його повторень. Особливості цього підходу:

- Простота реалізації: RLE ефективно стискає дані, де спостерігаються великі послідовності повторень (наприклад, bitmap-зображення або деякі види даних з низькою варіабельністю);
- Обмеженість: Якщо дані не мають великих повторюваних блоків, алгоритм може не лише не забезпечити стиснення, а й збільшити розмір файлу;
- Комбінування з іншими методами: RLE часто використовується як частина гібридних алгоритмів, наприклад, після застосування BWT, що дозволяє максимально ефективно виявити та стиснути згруповані символи.

До групи адаптивних методів також можна віднести алгоритми, що поєднують принципи динамічного оновлення статистичних моделей із базовими схемами стиснення. Зокрема, адаптивний алгоритм Хаффмана та методи, засновані на контекстному дереві (наприклад, алгоритм CTW – Context Tree Weighting), дозволяють оновлювати модель під час обробки даних і адаптувати її до змін у статистиці потоку. Такі методи забезпечують ще більшу ефективність, проте вимагають додаткових обчислювальних ресурсів й складної реалізації.

Для оцінки адаптивних методів стиснення важливо враховувати:

- Коефіцієнт стиснення: Відношення розміру вихідних даних до розміру отриманого стисненого файлу;
- Час стиснення та декодування: Швидкість роботи алгоритму є критичною для систем реального часу;
- Ресурсоємність: Обчислювальні витрати, вимоги до оперативної пам’яті та складність реалізації алгоритму;
- Адаптивність: Здатність алгоритму коригувати свою модель в міру надходження нових даних і змін в їх статистичних характеристиках.

Аналіз адаптивних методів стиснення, зокрема PPM, BWT та RLE, демонструє, що гнучкість та можливість динамічного оновлення моделей є ключовими чинниками для досягнення високої ефективності. Хоча кожен із зазначених методів має свої переваги й обмеження, їх комбінування дозволяє створити потужні гібридні системи стиснення, здатні ефективно обробляти дані різної природи. У сучасних інформаційних системах, де важлива не лише ступінь

стиснення, але й швидкість обробки та оптимальне використання ресурсів, оптимальний вибір алгоритму або комплексу методів залежить від специфіки завдання – від типу даних до вимог до реального часу обробки.

1.3.4 Порівняльний аналіз класичних алгоритмів без втрат

Класичні алгоритми стиснення без втрат, що активно застосовуються в інформаційних системах протягом останніх десятиліть, охоплюють як методи словникового стиснення (алгоритми сімейства LZ: LZ77, LZ78, LZW), так і ентропійні методи (наприклад, кодування Хаффмана та арифметичне кодування). Ці алгоритми закладають основу для більш сучасних (а іноді й адаптивних) систем стиснення, оскільки вони дозволяють без втрат відновлювати початкову інформацію, забезпечуючи при цьому оптимізацію використання сховищ та каналів передачі даних.

Методи, базовані на пошуку повторюваних послідовностей, мають ряд характерних особливостей. LZ77 використовує техніку ковзного вікна для пошуку та заміни повторюваних рядків у потоці даних. Це забезпечує швидке декодування, проте ефективність алгоритму значною мірою залежить від оптимального вибору розміру вікна. LZ78 і її розширення, зокрема LZW, поступово формують словникову структуру, що дозволяє ефективно кодувати дані із високою повторюваністю. Такі алгоритми добре працюють зі структурованими файлами, однак, при обробці великих обсягів даних, через зростання словника, можуть вимагати значних обчислювальних ресурсів і оперативної пам’яті.

Методи на основі теорії інформації забезпечують оптимальне кодування шляхом приписування коротших кодів символам із високою ймовірністю появи й довших — для рідкісних символів. Кодування Хаффмана дозволяє створити префіксне кодове дерево за заданими статистичними характеристиками даних. Воно є відносно простим у реалізації та ефективно працює за умови попереднього аналізу частот символів. Арифметичне кодування дає змогу досягти рівня стиснення, наближеного до теоретичного ліміту ентропії, завдяки точному розподілу ймовірностей, проте його обчислювальна складність і високі вимоги до точності арифметичних операцій часто ускладнюють впровадження в системах, де критична

швидкість обробки має першорядне значення.

При порівнянні класичних алгоритмів без втрат можна виділити кілька ключових аспектів (табл.1.6):

- Швидкість роботи та простота реалізації. Алгоритми сімейства LZ, зокрема LZ77, забезпечують високу швидкість стиснення і розпакування завдяки використанню простих механізмів пошуку повторів. Вони широко застосовуються у потокових системах і при реалізації систем у режимі реального часу. Проте, точність налаштування (наприклад, розміру ковзного вікна) може впливати на загальну ефективність стиснення;
- Компресійна ефективність. Ентропійні методи, такі як кодування Хаффмана, здатні досягати високих коефіцієнтів стиснення, оптимально використовувати статистику вхідних даних і відновлювати інформацію без втрат. Арифметичне кодування дозволяє ще точніше наближатися до теоретично визначеного ліміту, але ці переваги часто супроводжуються більшою складністю;
- Обчислювальні та ресурсні вимоги. Алгоритми сімейства LZ зазвичай мають менш складну реалізацію і вимагають відносно невелику кількість ресурсів для обробки даних. На противагу цьому, ентропійні методи, хоч і забезпечують краще стиснення, можуть потребувати додаткових обчислювальних потужностей та високоточної арифметики, що може бути критично для систем із жорсткими обмеженнями по ресурсах.

Кожен із класичних алгоритмів без втрат має свої сильні сторони та обмеження. При розробці систем стиснення вибір конкретного підходу залежить від характеру даних і вимог до системи:

- Для задач, де першорядна швидкість обробки та простота реалізації мають вирішальне значення, алгоритми сімейства LZ (наприклад, LZ77 або LZW) виявляються більш оптимальними;
- Якщо ж необхідно максимально наблизитися до теоретичного ліміту стиснення за рахунок детального аналізу статистики даних, доцільніше застосовувати ентропійні методи (зокрема, кодування Хаффмана або арифметичне кодування).

Таблиця 1.6. Порівняльний аналіз класичних алгоритмів без втрат

Алгоритм	Принцип роботи	Коефіцієнт стиснення	Швидкість	Ресурсо-ємність	Переваги	Недоліки
LZ77	Використання ковзного вікна для пошуку повторюваних послідовностей у потоці даних	Середній – залежить від повторю-ваності даних	Висока – особливо ефективний для потокових даних	Помірна – параметри вікна впливають на обчислю-вальні витрати	Простота реалізації, швидке декодування	Обмеження розміру буфера, чутливість до налаштувань, не найкраща ефективність при даних з нерегулярною структурою
LZ78 / LZW	Побудова словникової структури шляхом послідовного додавання нових шаблонів, що повторюються	Залежно від даних – від середнього до високого	Висока – ефективно для структуро-ваних даних	Середня – зростання словника може вимагати додаткових ресурсів	Добрий компроміс між швидкістю та ефективністю, чудово працює при наявності повторюваних шаблонів	Обмеження словника може впливати на ефективність, високі вимоги до оперативної пам’яті при великих обсягах даних
Кодування Хаффмана	Побудова префіксного дерева на основі статистики появи символів для присвоєння коротших кодів частішим символам	Близько до теоретичного ліміту – залежить від розподілу	Дуже висока – особливо при попередньому аналізі частот символів	Низька – алгоритм має просту реалізацію	Простота реалізації, ефективність при стабільних статичних даних	Недостатня адаптивність до локальних змін статистики, менш оптимальне для джерел з динамічною розподільною характеристикою
Арифметичне кодування	Кодування всього потоку як одне число за допомогою точного визначення інтервалів на основі ймовірнісного розподілу символів	Максимальний – дозволяє максимально наблизитися до ліміту ентропії	Помірна – обчислювальна складність призводить до нижчої швидкості	Висока – потребує високоточних арифметичних операцій	Оптимальне використання статистики без втрат, високий ступінь стиснення (наближення до теоретичного ліміту)	Складність реалізації, висока обчислювальна складність, труднощі з експлуатацією у режимах реального часу або на системах із обмеженими ресурсами

1.4 Інтеграція алгоритмів стискування у системи резервного копіювання

Інтеграція алгоритмів стискування у системи резервного копіювання становить один із ключових напрямків підвищення ефективності зберігання та управління даними в сучасних інформаційних системах. Застосування методів стиснення дозволяє значно знизити обсяг даних, що зберігаються у резервних копіях, оптимізувати використання сховищ і скоротити час передачі інформації між системами. Водночас, важливо враховувати, що інтеграція повинна забезпечувати цілісність і достовірність даних при відновленні, адже резервне копіювання за своєю сутністю не допустиме втрат.

На цьому етапі, розглядаючи попередні підрозділи, де було проаналізовано різноманітні алгоритми стискування (як без втрат, так і адаптивних методів), слід наголосити на тому, що впровадження цих технологій у системи резервного копіювання має враховувати як аспекти ефективності стиснення, так і специфіку операційної роботи самих систем резервування. Наприклад, алгоритми без втрат, такі як алгоритми сімейства LZ або ентропійне кодування, забезпечують повне відновлення початкових даних, що є надзвичайно важливим для критичних даних, тоді як адаптивні методи (PPM, BWT, RLE тощо) дозволяють досягти більш високих коефіцієнтів стиснення за рахунок подальшої оптимізації статистичних моделей даних. Впровадження стискання у процес резервного копіювання потребує комплексного підходу: від вибору оптимального алгоритму, який найкраще відповідає специфіці даних, до інтеграції цього алгоритму у вже існуючу інфраструктуру резервування. Тут важливо враховувати такі фактори, як баланс між швидкістю стиснення/розпакування, обчислювальна та ресурсоємність алгоритму.

1.4.1 Огляд сучасних програмних рішень архіваторів

Сучасні архіватори представляють собою комплекс програмних засобів, які поєднують потужні алгоритми стиснення та зручний користувацький інтерфейс. Вони забезпечують не лише зменшення об’єму даних, але й підтримку безпечного шифрування, створення саморозпаковуваних архівів і відновлення пошкоджених файлів. Серед провідних рішень можна виділити WinRAR, 7-Zip, Bandizip та PeaZip.

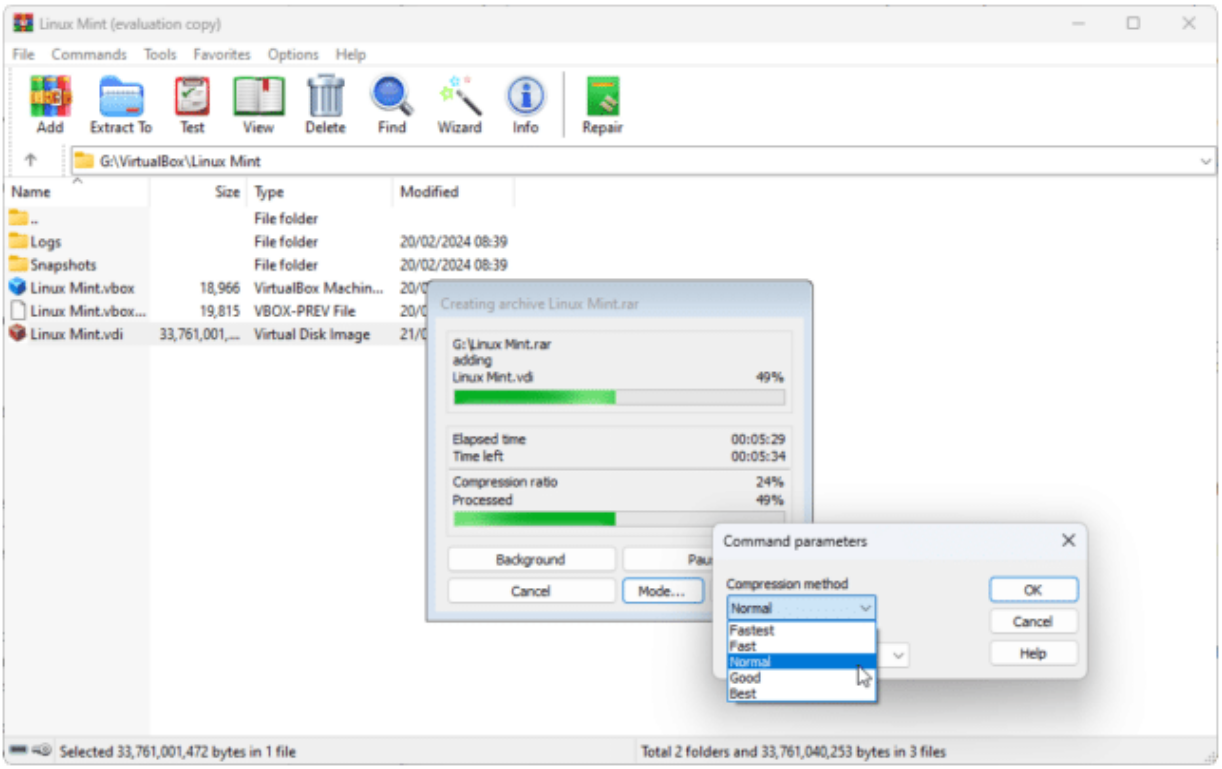


Рисунок 1.7. Інтерфейс архіватору WinRAR

На рис.1.7 показаний типовий робочий інтерфейс WinRAR із панеллю основних функцій, з можливістю вибору архівних форматів (RAR, ZIP, 7z тощо) та додаткових опцій (шифрування, відновлення).

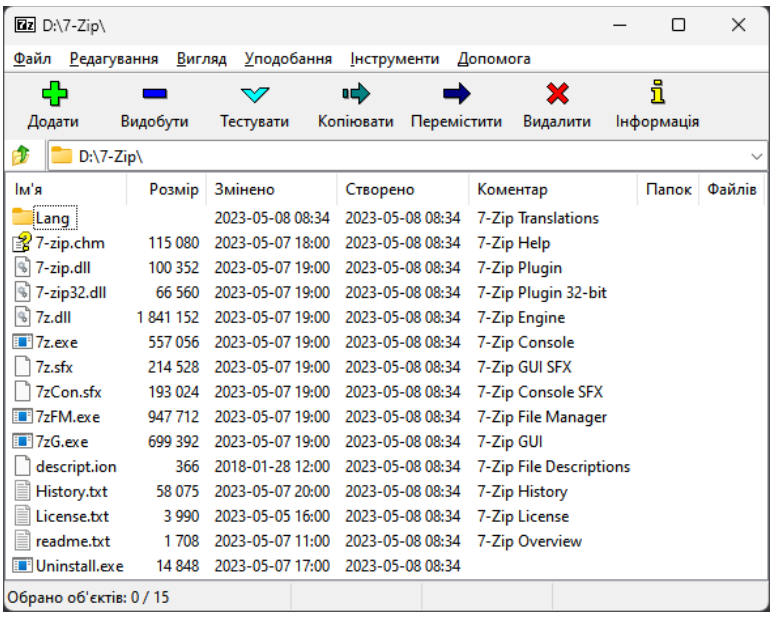


Рисунок 1.8. Інтерфейс архіватору 7-Zip

На рис.1.8 зображено файловий менеджер 7-Zip, що демонструє простоту та зручність управління архівами, підтримку формату 7z із високим коефіцієнтом стиснення за рахунок використання алгоритмів LZMA/LZMA2.

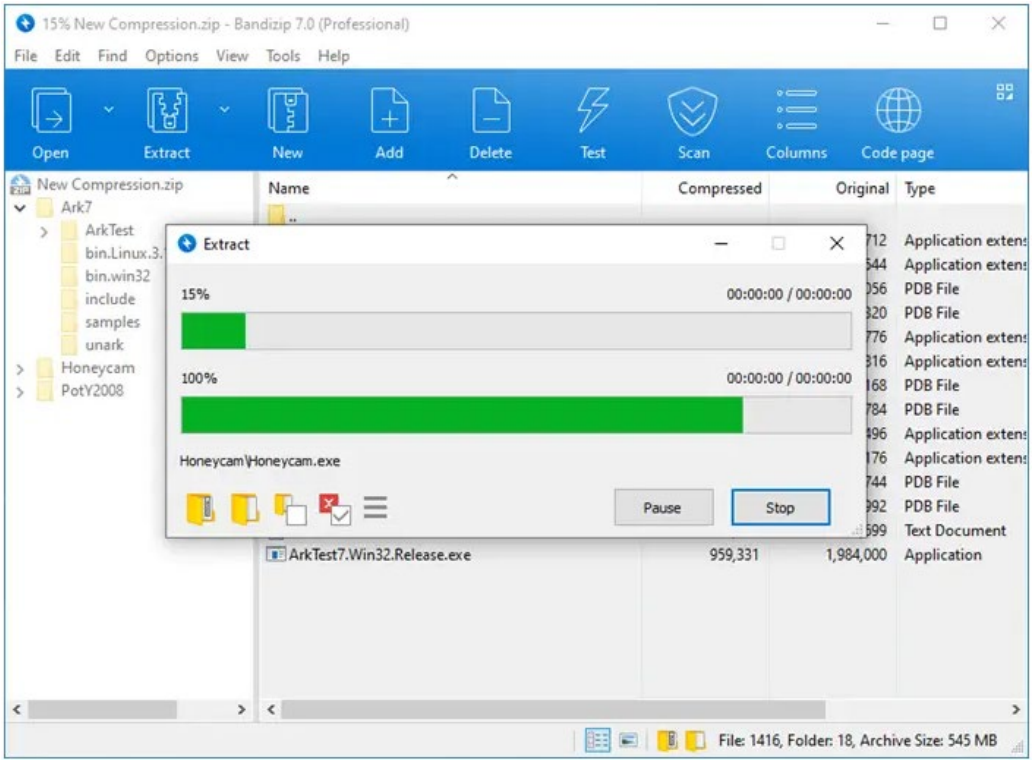


Рисунок 1.9. Інтерфейс архіватору Bandizip

На рис.1.9 показана ілюстрація інтерфейсу архіватору Bandizip, який має сучасний, легкий у використанні дизайн, що орієнтований на швидку архівацію та розпакування, з підтримкою багатомовних локалізацій та широким набором форматів.

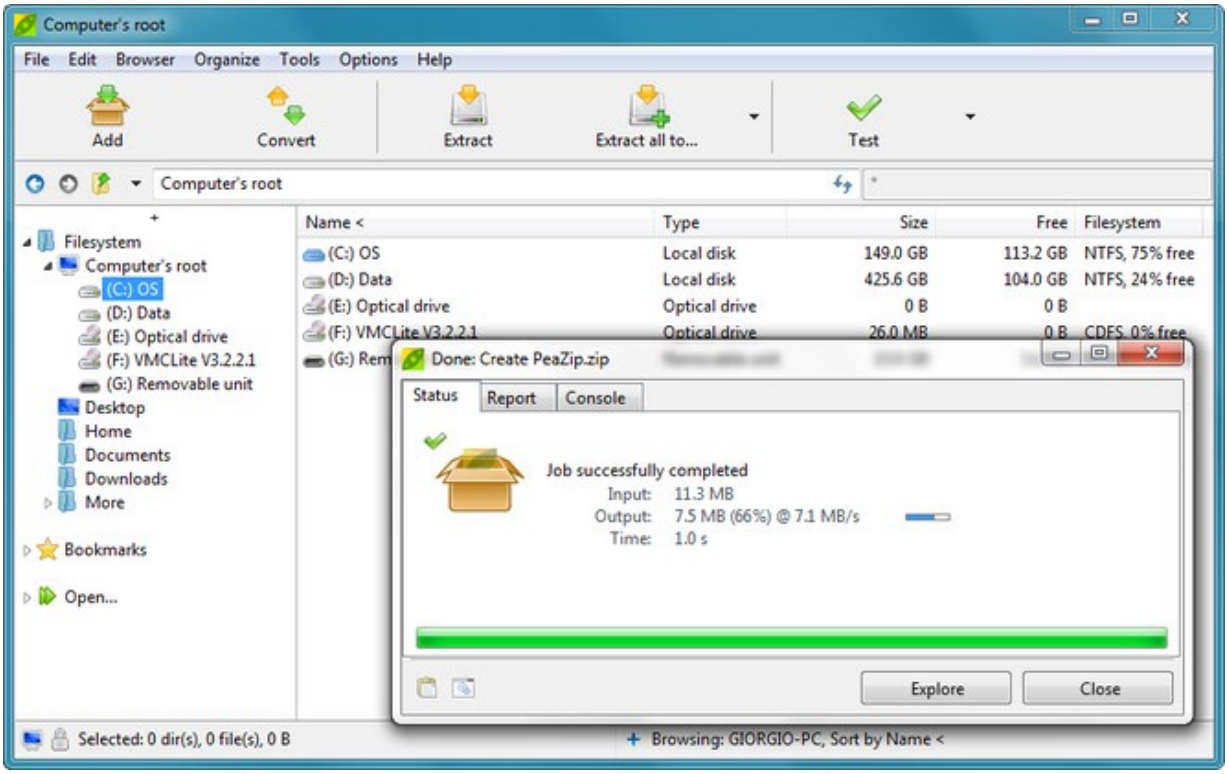


Рисунок 1.10. Інтерфейс архіватору PeaZip

Таблиця 1.7. Порівняльний аналіз архіваторів

Архіватор	Підтримувані формати	Алгоритми стиснення	Особливості інтерфейсу	Додаткові функції	Примітки
WinRAR	RAR, ZIP, 7z та ін.	Комбіновані методи (на базі LZ77, багаторівневе стиснення)	Інтуїтивно зрозумілий, налаштовуваний інтерфейс, підтримка самоархівів	Шифрування, створення SFX-архівів, відновлення пошкоджених архівів	Комерційний продукт (з режимом пробного використання)
7-Zip	7z, ZIP, GZIP, BZIP2, TAR, RAR (режим читання)	LZMA, LZMA2	Простий, функціональний файловий менеджер з відкритим кодом	Високий коефіцієнт стиснення, інтеграція з Windows Explorer	Вільне програмне забезпечення (open source)
Bandizip	ZIP, 7z, RAR, TAR та ін.	Оптимізовані алгоритми для швидкої архівації	Сучасний, легкий та адаптивний інтерфейс із багатомовною підтримкою	Швидка архівація/ розпакування, розширені налаштування шифрування	Популярний серед користувачів завдяки високій швидкості
PeaZip	7z, ARC, PAQ, UPX, ZIP, RAR та ін.	Гібридні алгоритми, підтримка численних форматів	Гнучкий та налаштовуваний інтерфейс, відкритий код	Підтримка великої кількості форматів, додаткові засоби безпеки	Повністю безкоштовний, орієнтований на розширену функціо-нальність

На рис.1.10 представлено інтерфейс PeaZip – відкритого та вільного архіватора, який забезпечує підтримку великої кількості форматів, широкий спектр налаштувань і додаткових засобів безпеки.

У табл.1.7 показане порівняння основних характеристик сучасних програмних рішень архіваторів. Сучасні архіватори не лише забезпечують високий ступінь стиснення даних, але й пропонують широкий спектр функцій для забезпечення безпеки, зручності використання та інтеграції в корпоративні системи резервного копіювання.

1.4.2 Огляд сучасних програмних рішень резервного копіювання

Програмні рішення для резервного копіювання розробляються з урахуванням високих вимог до надійності, гнучкості налаштувань, масштабованості та інтеграції з різними системами. Сучасні резервні платформи охоплюють як традиційні локальні архітектури, так і Гібридні або хмарні рішення, що дозволяють

організаціям оперативно реагувати на зростаючі обсяги даних і мінімізувати ризик їх втрати.

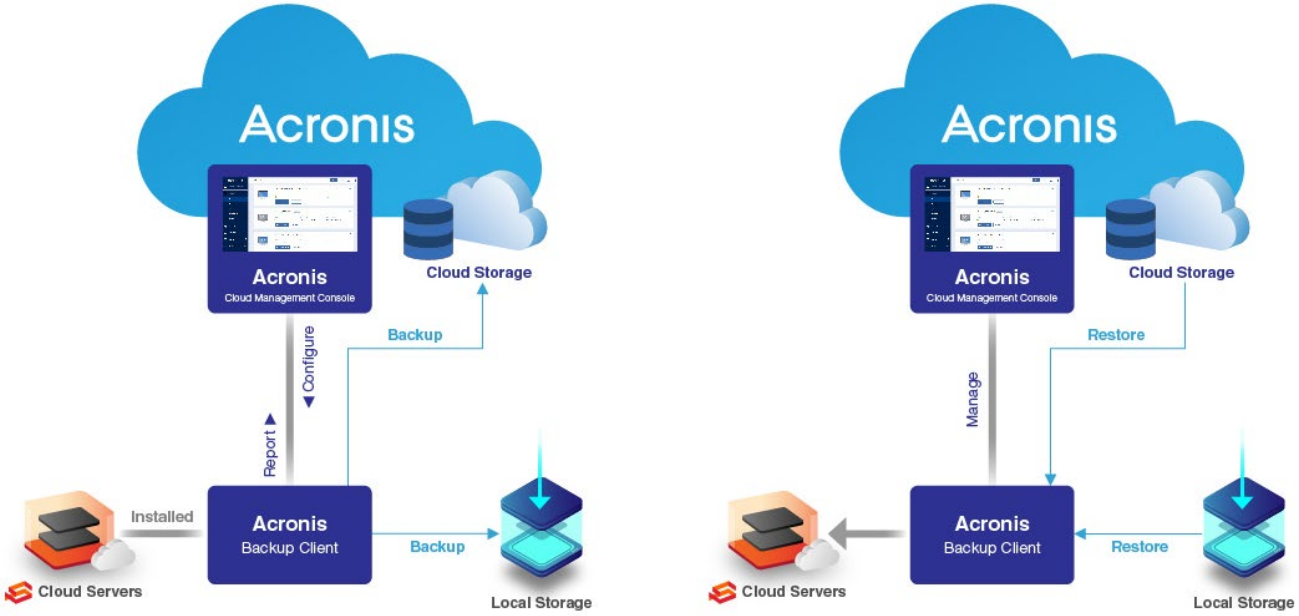


Рисунок 1.11. Архітектура програмного забезпечення Acronis Cyber Protect

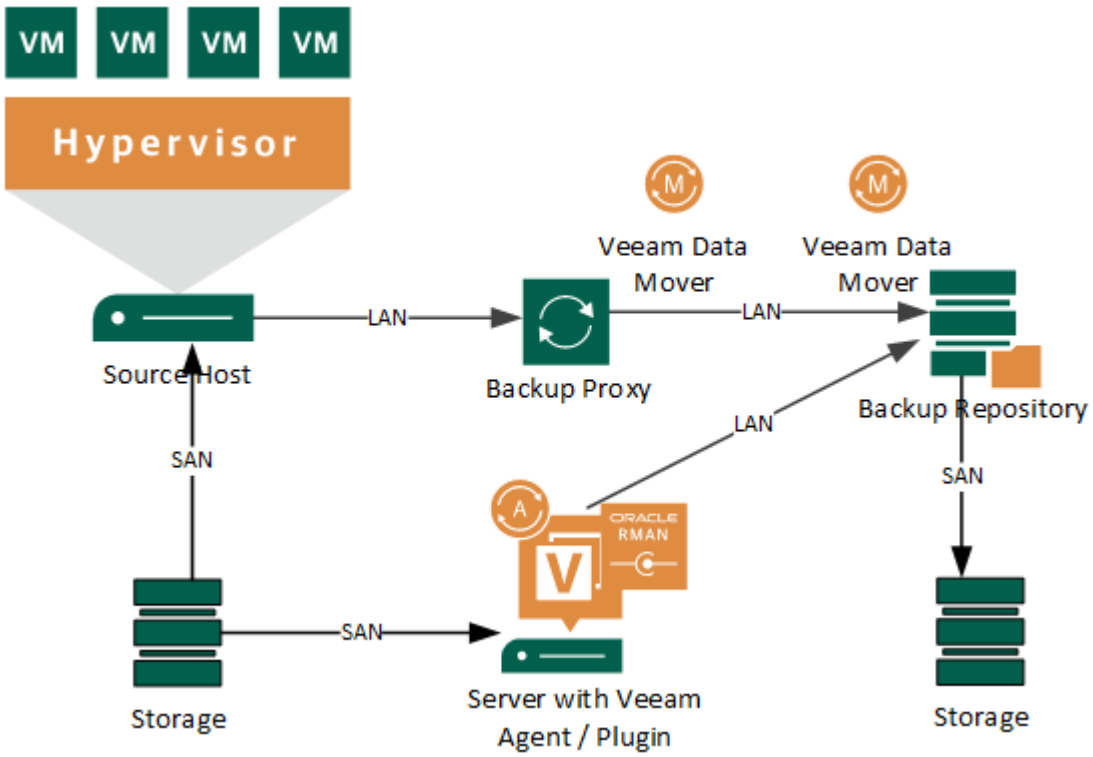


Рисунок 1.12. Архітектура програмного забезпечення Veeam Backup & Replication

Основні тенденції сучасних рішень резервного копіювання:

1. Інтеграція з хмарними технологіями. Багато сучасних продуктів резервного копіювання підтримують синхронізацію локальних копій із хмарними сервісами

(наприклад, Microsoft Azure, Amazon AWS, Google Cloud). Це дозволяє забезпечити географічну розподіленість даних та оперативне відновлення навіть у разі масштабних системних збоїв;

2. Автоматизація та планування процесів. Резервне копіювання організовується за допомогою автоматизованих розкладів, що дозволяють регулярно виконувати інкрементні, диференціальні або повні копії даних. Сучасні рішення підтримують моніторинг стану резервних копій, сповіщення про невдалі спроби архівації та автоматичну перевірку цілісності збережених даних;

3. Інтеграція алгоритмів стискування та шифрування. Для зменшення обсягу використовуваного сховища резервні рішення часто використовують алгоритми стискування (як без втрат, так і адаптивні методи) разом із функціями шифрування даних. Це дозволяє не лише економити ресурси, а й гарантувати безпеку конфіденційної інформації в архівах;

4. Модульність і масштабованість. Сучасні технології резервного копіювання передбачають можливість легкої інтеграції з існуючими корпоративними інформаційними системами. Вони дозволяють масштабувати ресурси відповідно до зростання обсягів даних та змін вимог бізнесу, забезпечуючи гнучке управління як локальними, так і віддаленими копіями.

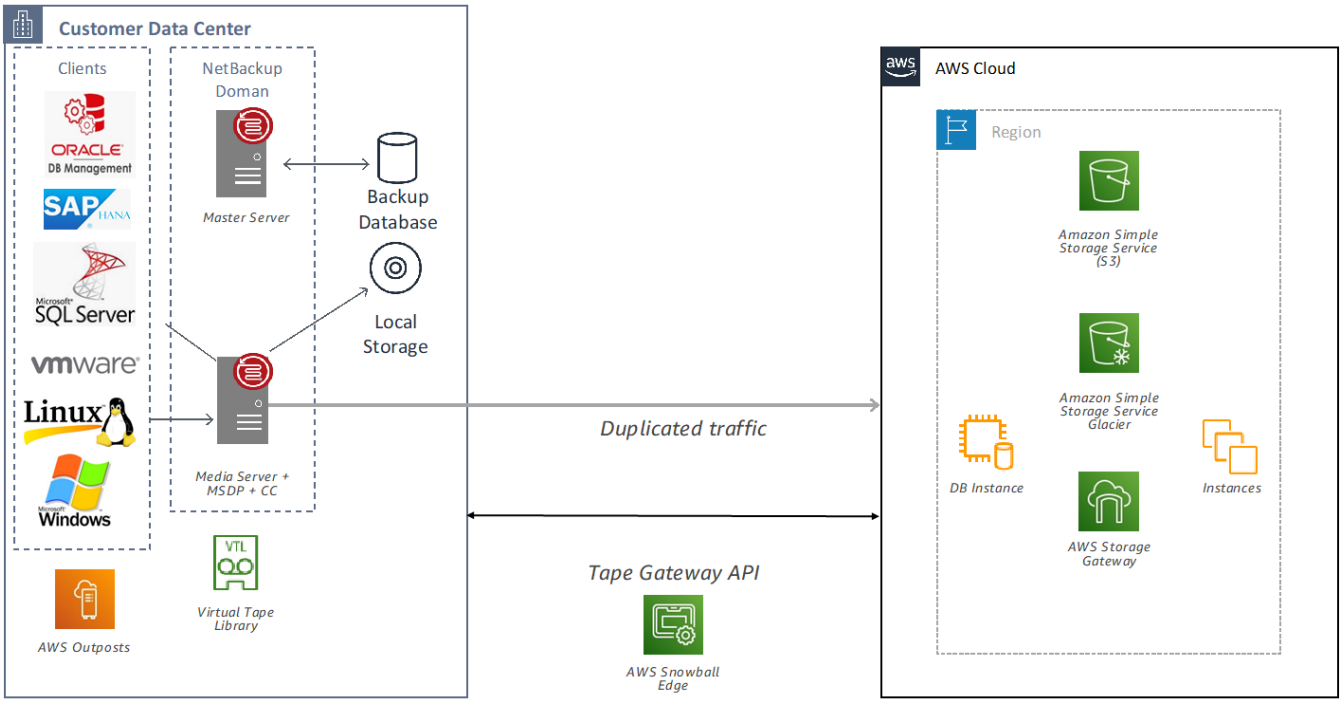


Рисунок 1.13. Архітектура програмного забезпечення Veritas NetBackup

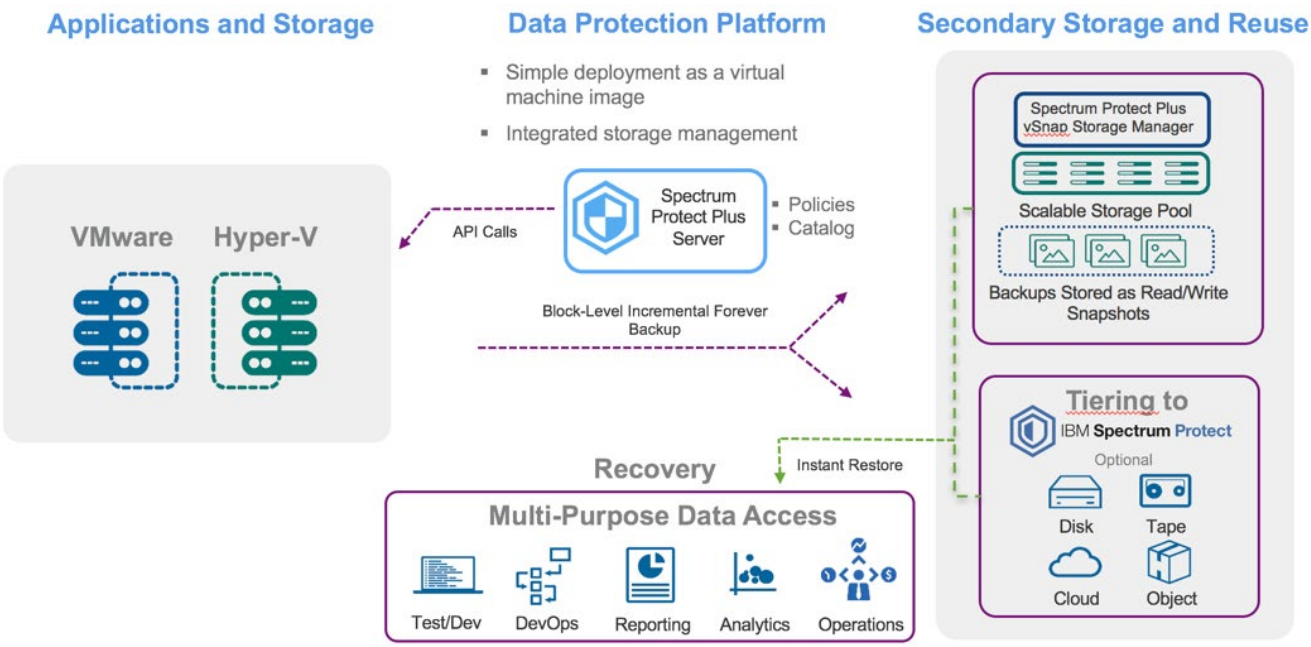


Рисунок 1.14. Архітектура програмного забезпечення IBM Spectrum Protect

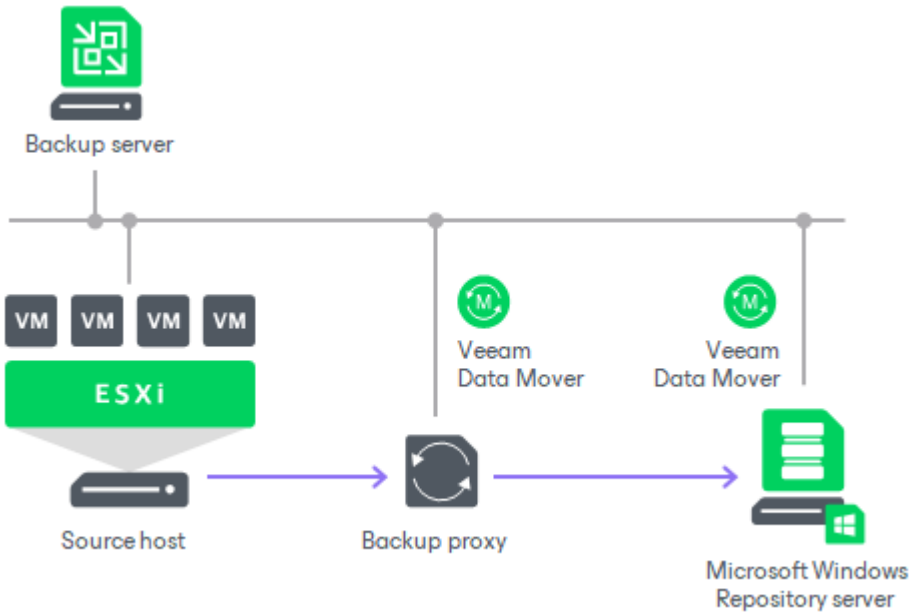


Рисунок 1.15. Архітектура програмного забезпечення Windows Backup

Серед популярних на ринку програмних рішень варто відзначити такі продукти:

- Acronis Cyber Protect (рис.1.11): Забезпечує комплексний захист, поєднуючи резервне копіювання даних із кібербезпекою (антивірус, антималвар). Підтримує як локальні, так і хмарні варіанти зберігання, має інтуїтивно зрозумілий інтерфейс та можливість централізованого управління копіями для відновлення системи;

- Veeam Backup & Replication (рис.1.12): Орієнтований переважно на віртуалізовані середовища (VMware, Hyper-V), пропонуючи високі показники швидкості створення та відновлення резервних копій. Вбудовані можливості оптимізації мережевого трафіку та використання адаптивного стискання дозволяють знизити витрати на сховища збереження;

Таблиця 1.8. Порівняльний аналіз сучасних рішень резервного копіювання

Продукт	**Підтримка платформ**	**Тип резервування**	**Ключові функції**	**Стиснення/ Шифрування**	**Особливості інтерфейсу**
Acronis Cyber Protect	Windows, Linux, Mac, мобільні платформи	Локальне, хмарне, гібридне	Централізоване управління, кібербезпека, відновлення після аварій	Вбудовані алгоритми стискання, стандартне шифрування	Інтуїтивно зрозумілий, з широкими можливостями налаштування
Veeam Backup & Replication	VMware, Hyper-V, фізичні сервери, хмарні платформи	Інкрементне, диферен-ціальне, повне резервне копіювання	Миттєве відновлення, оптимізація мережевого трафіку, реплікація даних	Адаптивне стиснення, шифрування даних під час передачі	Сучасний, орієнтований на корпоративний сегмент
Veritas NetBackup	Різні ОС, віртуальні та фізичні середовища	Повне, диферен-ціальне, інкрементне	Масштабованість, централізоване управління, підтримка багатьох типів сховищ	Використання класичних алгоритмів стискання із захистом даних	Богатий функціонал, оптимізований для великих підприємств
IBM Spectrum Protect	Різні платформи та середовища	Повне резервне копіювання, реплікація	Басейнове зберігання, інтеграція з корпоративними системами, оптимізація копій	Комбіноване стиснення з адаптивними алгоритмами	Централізована консоль управління, високий рівень автоматизації
Time Machine / Windows Backup	Вбудовані ОС (Mac OS, Windows)	Автоматичне локальне резервне копіювання	Простота використання, автоматичне архівування, версіонування даних	Стандартні засоби стискання та захисту інформації	Інтегрований інтерфейс, мінімальні налаштування

- Veritas NetBackup (рис.1.13): Підходить для великих підприємств і підтримує багаторівневе резервне копіювання даних, включаючи інтегроване відновлення систем та роботу з декількома типами сховищ. Рішення відзначається високою масштабованістю та гнучкістю конфігурації;
- IBM Spectrum Protect (рис.1.14): Забезпечує централізоване управління резервним копіюванням даних для різних платформ та середовищ. Продукт

відомий своєю здатністю зменшувати витрати на зберігання за рахунок використання диференціальних стратегій копіювання та алгоритмів стиснення даних;

- Time Machine (Mac OS) та Windows Backup (вбудовані засоби ОС): Ці інструменти, попри свою спрощену функціональність, забезпечують базовий рівень резервного копіювання для особистих та малих бізнес-користувачів (рис.1.15). Вони інтегровані у відповідні операційні системи та пропонують зручний автоматизований процес без додаткових налаштувань.

У табл.1.8 порівняні основні характеристики деяких сучасних програмних продуктів резервного копіювання. Сучасні програмні рішення резервного копіювання пропонують широкий функціонал від автоматичного локального резервування до комплексних гібридних систем із підтримкою хмарних рішень. Інтеграція передових технологій стискання, шифрування і централізованого управління дозволяє забезпечити високий рівень безпеки та оперативності відновлення критичних даних у різних умовах роботи інформаційних систем.

1.4.3 Роль апаратного забезпечення та оптимізація процесів стиснення

У сучасних інформаційних системах процес стиснення відіграє ключову роль у збереженні, передачі та відновленні даних. При цьому апаратне забезпечення – від центральних процесорів до спеціалізованих прискорювачів – стає визначальним чинником у забезпеченні високої швидкості, ефективності та надійності здійснення компресії. Розглянемо основні аспекти ролі апаратного забезпечення та методи оптимізації процесів стиснення.

Сучасні алгоритми стиснення, незалежно від того, чи використовують вони методи без втрат (наприклад, алгоритми сімейства LZ, кодування Хаффмана) або адаптивні методи, є обчислювально інтенсивними. Використання багатоядерних процесорів і технологій паралельної обробки дозволяє розподілити обчислювальне навантаження між різними ядрами, що значно прискорює як процес стиснення, так і розпакування. Наприклад, багато сучасних реалізацій адаптивного стиснення використовують інструкції SIMD (Single Instruction, Multiple Data), що дозволяють виконувати обчислювальні операції одночасно для декількох блоків даних.

Крім традиційних ЦПУ, спеціалізовані апаратні засоби, такі як графічні процесори (GPU), тензорні процесори (TPU) та програмовані логічні інтегральні схеми (FPGA), набувають популярності для вирішення завдань стиснення. Вони дозволяють реалізувати конкретні алгоритми стиснення у вигляді апаратних модулів, оптимізованих для обробки великих обсягів даних у режимі реального часу. Такі рішення особливо корисні у системах з високими вимогами до пропускної здатності та енергоефективності, наприклад, у дата-центрах.

Архітектура пам’яті також суттєво впливає на продуктивність процесу стиснення. Сучасні системи оснащені великими кеш-пам’яттю (L1, L2, L3), що забезпечують швидкий доступ до даних, зокрема для повторних операцій зчитування під час аналізу блоків даних. Ефективне використання кешу дозволяє зменшити затримки під час виконання алгоритмів, значно оптимізувати роботу з великими обсягами даних та скоротити час стиснення.

Окрім апаратного забезпечення, значний вплив на ефективність процесів стиснення має оптимізація алгоритмів на програмному рівні. Це включає:

- Вибір алгоритму стиснення: Підбір методу, який не лише забезпечить високий коефіцієнт стиснення, а й дозволить задовольнити вимоги до швидкості та ресурсозабезпечення. Наприклад, для систем, де критична оперативність, можуть бути застосовані методи, що працюють за принципом LZ77 або алгоритми на базі кодування Хаффмана;
- Паралелізація процесів: Розбиття роботи алгоритму на незалежні частини дозволяє використовувати багатоядерні процесори та спеціальні прискорювачі. Використання потокової обробки дозволяє обробляти дані «на льоту», знижуючи затримки та підвищуючи загальну продуктивність;
- Оптимізація пам’яті: Зведення до мінімуму доступу до оперативної пам’яті за допомогою ефективного кешування і використання локальних буферів сприяє зменшенню затримок при читанні та записі даних.

Сучасні бібліотеки, такі як zlib, LZ4, Snappy та інші, розробляються з урахуванням сучасних архітектур процесорів і дозволяють суттєво прискорити процес стиснення та розпакування. Ці рішення включають оптимізовані версії

алгоритмів на рівні асемблера, що використовують спеціальні інструкції для обробки великих обсягів даних із суворими часовими обмеженнями.

Оптимізація процесу стиснення завжди є компромісом між ступенем стиснення (тобто економією місця) та швидкістю обробки даних. У багатьох практичних застосуваннях, зокрема у системах резервного копіювання, важливо знайти оптимальний баланс, щоб зменшення обсягу даних не приводило до значного збільшення часу на стиснення та розпакування. Для цього часто застосовуються гібридні підходи, які комбінують алгоритми зі швидким стисненням (наприклад, RLE або LZ4) з алгоритмами, що забезпечують максимальну ефективність (наприклад, арифметичне кодування чи LZMA).

Роль апаратного забезпечення у процесах стиснення є вирішальною. Сучасні комп'ютерні архітектури, багатоядерні процесори, спеціалізовані прискорювачі та оптимізовані бібліотеки значною мірою впливають на швидкість, ефективність і енергоспоживання алгоритмів стиснення. Найбільш ефективне рішення досягається шляхом комплексної оптимізації: поєднання апаратних засобів, алгоритмічних новацій і програмних рішень дозволяє забезпечити максимальну продуктивність систем резервного копіювання та обробки даних при мінімальних затратах ресурсів.

1.5 Розробка застосунку для аналізу ефективності алгоритмів стискування без втрат

Розробка застосунку має на меті створення системи підтримки прийняття рішень, що дозволяє об'єктивно оцінити ефективність класичних алгоритмів стиснення без втрат. Такий застосунок орієнтований на аналіз параметрів стиснення (коефіцієнт стиснення, швидкість стиснення) із врахуванням різних типів даних (текстові, бінарні, медіа-файли) та дозволяє обрати оптимальний алгоритм.

1.5.1 Архітектура та функціональність застосунку

Загальна блок-схема алгоритму роботи застосунку наведена на рис.1.16. Застосунок складається з наступних основних компонентів:

– Інтерфейс користувача. Інтерфейс дозволяє вводити розміри файлів у мегабайтах для трьох категорій даних (текстові, бінарні та медіа-файли). За

допомогою слайдеру користувач може задати пріоритет між показником коефіцієнта стиснення та швидкістю стиснення. Це дозволяє інтегрувати суб'єктивні побажання в процес оцінки ефективності;

- Модуль розрахунків. На основі введених користувачем даних застосунок використовує математичну модель, що включає два масиви показників: Масив *K*. Для кожного алгоритму визначаються коефіцієнти стиснення для текстових, бінарних і медіа-файлів за формулою

$$K_i = T \cdot K_{i,t} + B \cdot K_{i,b} + M \cdot K_{i,m}, \qquad (1.3)$$

де *T*, *B* та *M* – відносна кількість файлів відповідних типів, а $K_{i,t}$, $K_{i,b}$, $K_{i,m}$ – коефіцієнти стиснення для певного алгоритму;

Масив *v*. Аналогічно визначаються середні значення швидкості стиснення для кожного алгоритму:

$$v_i = T \cdot v_{i,t} + B \cdot v_{i,b} + M \cdot v_{i,m}; \qquad (1.4)$$

Нормалізація та агрегування критеріїв. Для подальшого порівняння застосовуються процедури нормалізації:

$$v'\text{i} = \frac{\text{v}_\text{i} - \text{v}min}{v_{max} - v_{min}}, \quad K'\text{i} = \frac{\text{K}_\text{i} - \text{K}min}{K_{max} - K_{min}}, \qquad (1.5)$$

після чого система формує комплексну оцінку кожного алгоритму за допомогою цільової функції:

$$Benchmark\ \text{i} = \text{a}K \times K'\text{i} + \text{a}v \times v'\text{i}, \qquad (1.6)$$

де аК і a_v – вагові коефіцієнти, що задаються користувачем через слайдер;

- Візуалізація результатів. Результатом роботи застосунку є таблиця з відсортованими алгоритмами за комплексною оцінкою *benchmark*, де алгоритм із найвищим показником розташовується на першій позиції. Таблиця містить наступні дані:
  - Номер алгоритму;
  - Назва алгоритму стиснення;
  - Загальний розмір стиснутих даних (МВ);
  - Загальний час стиснення (с);
  - Середня швидкість стиснення (МВ/с);
  - Середній коефіцієнт стиснення;

– Результат цільової функції.

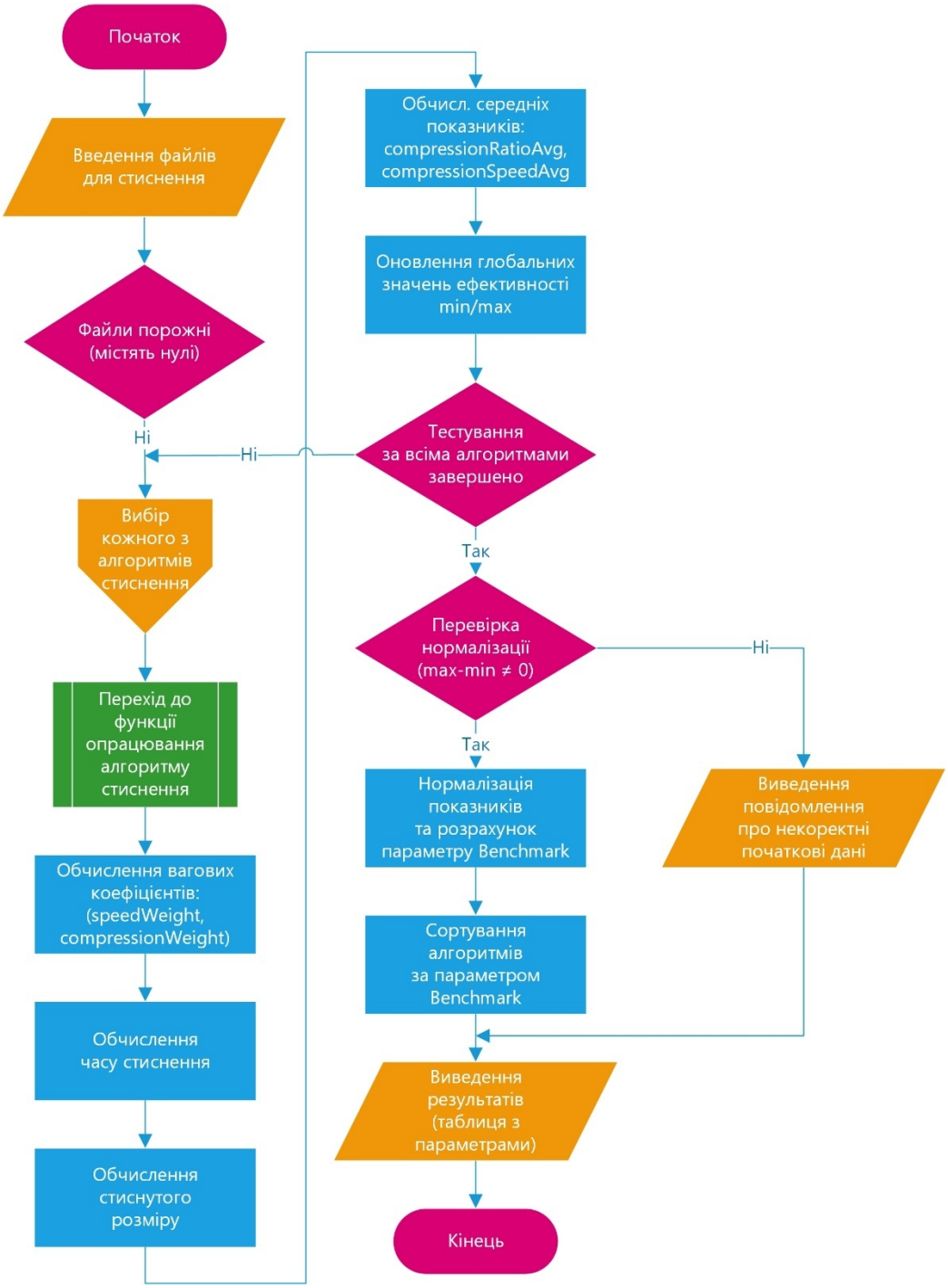


Рисунок 1.16. Загальна блок-схема алгоритму роботи застосунку

1.5.2 Технологічна реалізація застосунку

У застосунку інтерфейс користувача побудований із застосуванням HTML, CSS та фреймворку Metro UI, що забезпечує зручність введення даних і надає інтерактивні елементи (слайдер для вибору пріоритетних критеріїв).

Компоненти застосунку:

– Інтерфейс користувача: побудований із використанням HTML та CSS із

застосуванням фреймворку Metro UI;

- JavaScript і jQuery: для обробки подій, завантаження файлів, маніпуляції даними, запуску обчислень, виклику функцій стиснення та побудови таблиці з результатами;
- API для завантаження файлів: використання стандартного HTML-елементу <input type="file"> для того, щоб користувач міг легко обрати файл для обробки;
- Модуль з алгоритмами стиснення: бібліотеки або власні функції, які імітують або реалізовують різні алгоритми стискування (LZ4, Deflate, LZO і інших);
- Обчислювальний модуль: для аналізу властивостей файлу (розмір, час обробки, ефективність стиснення) – аналізує завантажений файл автоматично.

Для обробки введених даних і розрахунків використовується JavaScript із бібліотекою jQuery. Обчислення та побудова таблиці результатів:

```
// Отримання введених даних користувача
let inputSizeText = parseFloat($('#input_text').val());
let inputSizeBinary = parseFloat($('#input_binary').val());
let inputSizeMedia = parseFloat($('#input_media').val());
// Отримання вагових коефіцієнтів із слайдеру
let speedWeight = Metro.getPlugin('#speed_weight', 'slider').val();
let compressionWeight = 100 - speedWeight;
speedWeight = speedWeight / 100;
compressionWeight = compressionWeight / 100;
// Для кожного алгоритму розраховуються показники часу стиснення
// та розміру стиснених даних
$.each(compressionAlg, function (index, value) {
    value.timeForText = inputSizeText / value.textAvgSpeed;
    value.timeForBinary = inputSizeBinary / value.binaryAvgSpeed;
    value.timeForMedia = inputSizeMedia / value.mediaAvgSpeed;
    value.totalTime = value.timeForText +
    value.timeForBinary + value.timeForMedia;
    value.textCompressedSize = inputSizeText / value.textCompressionRatio;
    value.binaryCompressedSize = inputSizeBinary /
    value.binaryCompressionRatio;
    value.mediaCompressedSize = inputSizeMedia / value.mediaCompressionRatio;
    value.totalCompressedSize = value.textCompressedSize +
    value.binaryCompressedSize + value.mediaCompressedSize;
    value.compressionRatioAvg = (inputSizeText + inputSizeBinary +
    inputSizeMedia) / value.totalCompressedSize;
    value.compressionSpeedAvg = (inputSizeText + inputSizeBinary +
```

```
    inputSizeMedia) / value.totalTime;
});
// Нормалізація показників та формування інтегрованої оцінки
// Додатковим кроком є обчислення min та max значень для нормалізації
$.each(compressionAlg, function (index, value) {
    value.benchmark = compressionWeight * ((value.compressionRatioAvg –
    minCompressionRatio) / (maxCompressionRatio - minCompressionRatio)) +
    speedWeight * ((value.compressionSpeedAvg –
 minCompressionSpeed) / (maxCompressionSpeed - minCompressionSpeed));
});
// Сортування алгоритмів за спаданням комплексної оцінки
// та формування таблиці результатів
compressionAlg.sort(function (a, b) { return b.benchmark - a.benchmark; });
```

1.5.3 Реалізація інтерфейсу та обробки файлу

Інтерфейс застосунку для аналізу ефективності алгоритмів стиснення дозволяє завантажувати файли, задавати пріоритет за допомогою інтерактивного слайдера, запускати аналіз та переглядати результати у вигляді таблиці (рис.1.17).

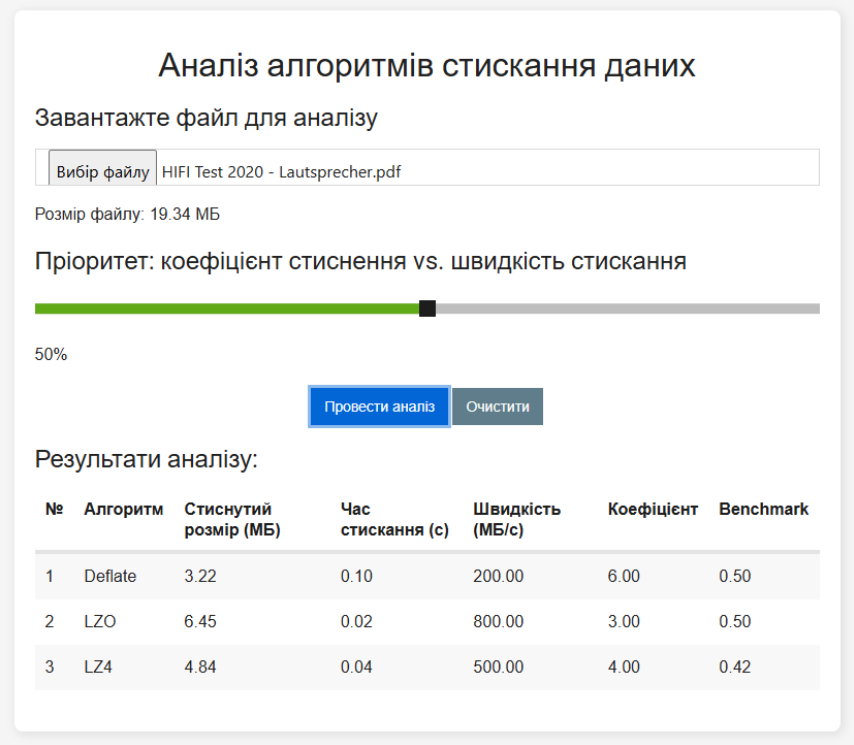


Рисунок 1.17. Інтерфейс застосунку для аналізу ефективності алгоритмів

Завантаження файлу: На сторінці розміщено елемент завантаження файлу:

```
<label for="fileInput">Оберіть файл для стискання:</label>
<input type="file" id="fileInput" accept="*/*">
```

Завдяки цьому користувач не вводить вручну розміри – програма отримує

властивості файлу (розмір, тип) автоматично за допомогою JavaScript (API FileReader, властивості File).

Слайдер пріоритетів: Використання слайдера (Metro UI) дозволяє встановити, який аспект є важливішим – ефективність стиснення (відношення початкового розміру файлу до стиснутого розміру) або швидкість стиснення.

```
<div data-role="slider" id="prioritySlider" data-value=
"50" data-orientation="horizontal" data-show-hint="true"></div>
```

Значення слайдера (наприклад, 0 – повний пріоритет стиснення, 100 – пріоритет швидкості) потім перетворюється у вагові коефіцієнти для розрахунків.

Кнопки запуску та очищення: Інтерактивні кнопки для запуску аналізу та скидання введених даних.

```
<button id="btnAnalyze" class="button primary">Почати аналіз</button>
<button id="btnClear" class="button secondary">Очистити</button>
```

Виведення результатів відбувається у вигляді таблиці, де кожен рядок – це алгоритм з такими полями, як назва, загальний стиснутий розмір, загальний час обробки, середня швидкість, середній коефіцієнт стиснення та комплексний показник Benchmark.

Завантаження файлу відбувається за допомогою події change на <input type="file"> з отриманням об’єкту File:

```
$('#fileInput').on('change', function(e) {
    const file = e.target.files[0];
    if (file) {
        const fileSize = file.size; // розмір файлу в байтах
        const fileSizeMB = fileSize / (1024 * 1024);
        console.log("Розмір файлу: " + fileSizeMB.toFixed(2) + " МБ");
    }
});
```

Після завантаження файлу програма аналізує його за допомогою внутрішніх модулів, які реалізують роботу різних алгоритмів стиснення. Код викликає функції, що приймають дані файлу (вміст файлу, його розмір) і проводять розрахунки:

- Час стискання розраховується як співвідношення розміру файлу до характеристик швидкості алгоритму;
- Стиснений розмір – як розмір файлу, поділений на коефіцієнт стиснення для

даного методу;

- Середні показники формуються шляхом агрегування та нормалізації для всіх алгоритмів;
- Інтегрована оцінка (Benchmark).

Після завершення розрахунків застосунок будує таблицю результатів, у якій відображаються:

- Назва алгоритму;
- Загальний стиснутий розмір файлу (МБ);
- Загальний час стискання (с);
- Середня швидкість стискання (МБ/с);
- Середній коефіцієнт стискання;
- Комплексна оцінка (Benchmark).

Це дозволяє користувачу одразу побачити, який алгоритм є оптимальним відповідно до його пріоритетів (штучно заданих через слайдер).

У якості IDE використовувалось середовище Visual Studio Code.

Ключові фрагменти коду наведені у Додатку А. Тестування проводилось на основі відкритого HTML-файлу у браузері. Проект мав таку структуру:

```
/project-folder
  ├── index.html          # Основний HTML-файл з розміткою
  ├── style.css           # Файл стилів (Metro UI, імпортована через CDN)
  ├── app.js              # Основний JavaScript-код для обробки файлів,
                          # виклику алгоритмів, побудови таблиці
  ├── libs/               # Допоміжні бібліотеки (jQuery, Metro UI,
                          # бібліотеки для стискання)
  └── assets/             # Зображення, іконки
```

Алгоритми стискування інтегровані за допомогою JavaScript-бібліотек. Вони містять функції, які повертають статистику (час, коефіцієнт стискання) на основі реального аналізу вхідного файлу (через API обчислень).
За допомогою FileReader API зчитується файл і отримується його розмір. JavaScript обробляє події, такі як "завантаження файлу", "зміна слайдера" та "натискання кнопок", і відразу викликає функції обчислення.
При обробці файлу застосунок відображає попереджувальні повідомлення, а також виводить результати обчислень у таблиці.

1.6 Експериментальний аналіз ефективності алгоритмів стискування без втрат

1.6.1 Апаратне забезпечення для здійснення тестування

Для дослідження ефективності алгоритмів стиснення даних було використано відносно сучасну систему:

- Процесор: Intel® Core™ i7-10750H (2.60 ГГц, 6 ядер);
- Оперативна пам’ять: 16 GB DDR4;
- Жорсткий диск: NVMe SSD Samsung 970 EVO Plus (1 TB);
- Операційна система: Ubuntu 20.04 LTS, Linux kernel 5.8;
- Максимальна швидкість зчитування/запису: 3500/3300 MB/s.

Експеримент проводився для однакових наборів даних:

- Текстові файли: загальний об’єм 4500 МБ;
- Бінарні файли: загальний об’єм 1200 МБ;
- Медіа файли: загальний об’єм 3000 МБ;

1.6.2 Методика тестування

Для кожного з аналізованих алгоритмів згідно технічного завдання (Deflate, LZ4, LZO) було проведено 5 тестів на кожному з чотирьох рівнів стиснення (1, 3, 6, 9). За результатами тестування обчислювалися:

- Коефіцієнт стиснення (CR): відношення початкового розміру файлу до стиснутого об’єму;
- Час стискання (T): середній час виконання операції стиснення (в секундах);
- Швидкість стискання (S): обчислюється як співвідношення початкового об’єму до часу стискання (МБ/с).

Крім того, для зручності аналізу була розрахована комплексна оцінка (Benchmark), яка формується за формулою:

$$\text{Benchmark} = a_k \times \text{norm}(CR) + a_v \times \text{norm}(S) \qquad (1.7)$$

де a_k та a_v – вагові коефіцієнти (визначені користувачем через інтерфейс слайдера), а $\text{norm}(CR)$ та $\text{norm}(S)$ – нормалізовані значення відповідних параметрів для кожного алгоритму. Значення Benchmark дозволяє порівняти ефективність алгоритмів з урахуванням компромісу між ступенем стиснення та швидкістю.

1.6.3 Отримання результатів тестування та їх візуалізація

Для тестування обрано сучасні реалізації алгоритмів стискування без втрат:

- Deflate (використання стандартної реалізації gzip);
- LZ4 (CLI-версія LZ4, 64-біт);
- LZO (лінійна реалізація lzop v1.03).

Для кожного алгоритму визначалися параметри для чотирьох рівнів стиснення. Нижче наведено узагальнену таблицю (табл.1.9) середніх результатів тестування, де для кожного алгоритму наведено значення коефіцієнта стискування (CR), часу стискування (T, с) та швидкості стискування (S, МБ/с) для текстових, бінарних та медіа-файлів. Значення Benchmark – інтегральна оцінка, яка враховує нормалізовані параметри і дозволяє обрати оптимальний алгоритм в умовах конкретних вимог.

Таблиця 1.9. Результати тестування стискування без втрат

Алгоритм	Рівень	Текстові файли (CR, T(s), S(MB/s))	Бінарні файли (CR, T(s), S(MB/s))	Медіа-файли (CR, T(s), S(MB/s))	Benchmark
Deflate	1	(5.80, 48.0, 93.0)	(2.30, 28.0, 42.9)	(1.010, 110.0, 41.0)	0.68
Deflate	3	(6.23, 52.4, 86.0)	(2.55, 36.1, 33.5)	(1.016, 114.8, 26.2)	0.71
Deflate	6	(8.24, 79.5, 56.6)	(2.67, 65.5, 18.3)	(1.015, 121.2, 24.2)	0.73
Deflate	9	(8.38, 91.4, 55.6)	(2.69, 181.7, 7.45)	(1.015, 125.5, 25.2)	0.68
LZ4	1	(5.46, 10.0, 486.3)	(1.96, 4.20, 308.6)	(1.010, 8.36, 355.9)	0.81
LZ4	3	(7.82, 41.1, 118.23)	(2.29, 23.9, 50.2)	(1.011, 100.5, 29.8)	0.84
LZ4	6	(8.15, 54.5, 93.1)	(2.34, 32.72, 41.4)	(1.012, 102.3, 31.1)	0.83
LZ4	9	(8.20, 68.12, 74.8)	(2.36, 54.91, 24.7)	(1.013, 104.03, 30.1)	0.80
LZO	1	(4.23, 11.55, 420.35)	(1.91, 4.12, 314.61)	(1.007, 8.02, 378.81)	0.75
LZO	3	(4.28, 11.49, 422.72)	(1.92, 4.19, 309.37)	(1.008, 8.04, 377.76)	0.78
LZO	6	(4.28, 11.46, 423.68)	(1.92, 4.16, 311.14)	(1.008, 8.10, 375.00)	0.78
LZO	9	(7.33, 219.28, 22.15)	(2.44, 337.18, 3.84)	(1.013, 335.08, 9.07)	0.50

За отриманими результатами тестування (табл.1.9) створено графіки на рис.1.18-1.20, які дозволяють наочно порівняти параметри ефективності алгоритмів за різними критеріями та зробити висновки щодо оптимального вибору алгоритму у системі резервного копіювання. Графік “Benchmark vs Рівень стискування” демонструє, як інтегрована оцінка (Benchmark) змінюється при збільшенні рівня стиснення для різних алгоритмів. За таким показником видно, що LZ4 демонструє найвищу оцінку на оптимальних рівнях, що вказує на баланс ефективності та

швидкості. Графік “Час стискування vs Рівень стискування” показує, як змінюється час стискання (в секундах) залежно від вибраного рівня стискування. Маємо дуже короткий час для LZ4 на низьких рівнях, проте час збільшується з підвищенням рівня стискування, особливо для Deflate та LZO (останній різко зростає на рівні 9). На графіку “Швидкість стискування vs Рівень стиснення” видно, що швидкість стискування (МБ/с) для LZ4 на найнижчому рівні є дуже високою, але з підвищенням рівня вона знижується. Deflate та LZO характеризуються різними трендами щодо швидкості. Графік “Коефіцієнт стискування (CR) vs Рівень стискування” демонструє, як змінюється компресійний коефіцієнт для кожного алгоритму при зміні рівня стиснення. Наприклад, CR для Deflate зростає при вищих рівнях, а у LZO спостерігається різке збільшення на рівні 9, що впливає на інтегральну оцінку.

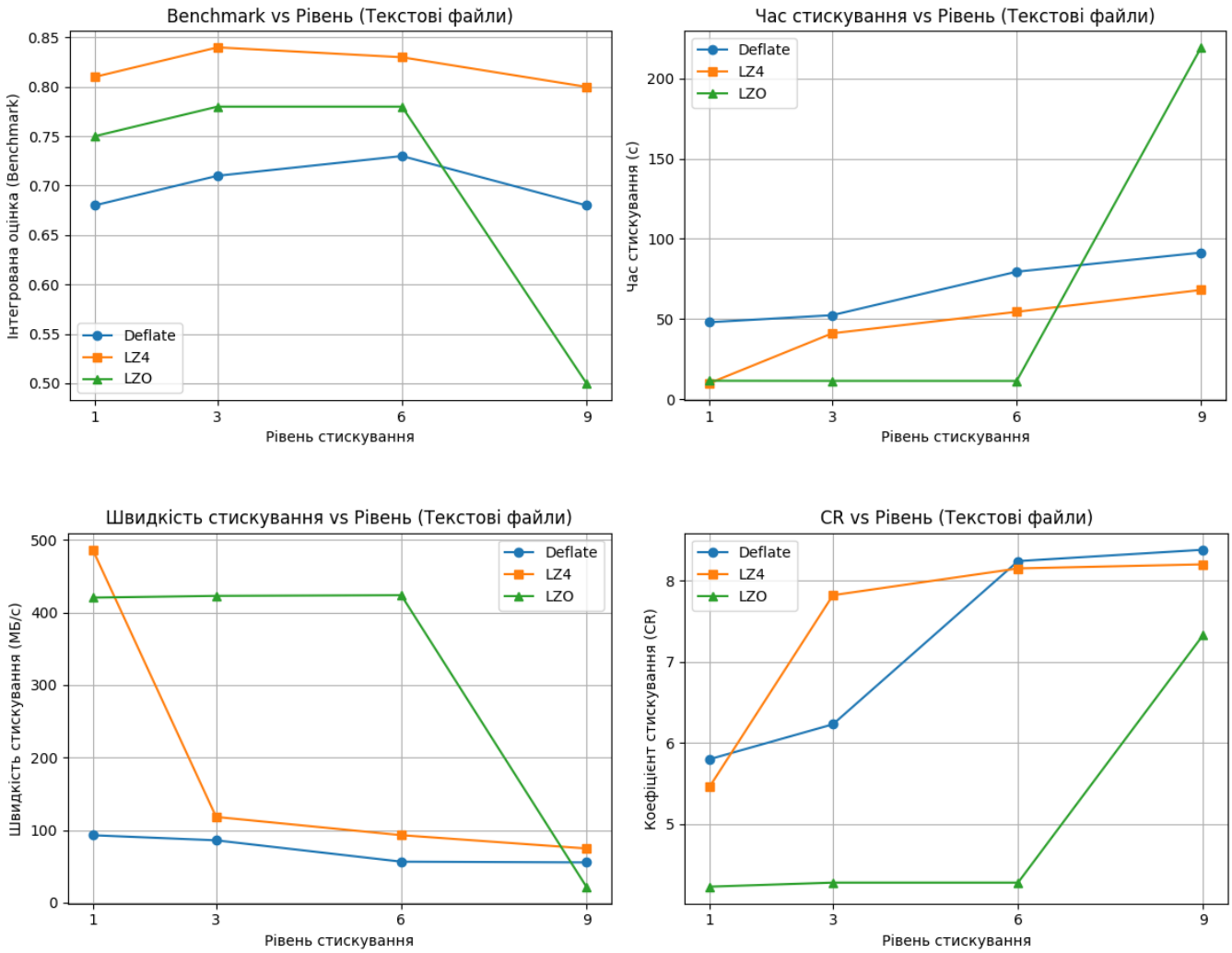


Рисунок 1.18. Порівняння параметрів ефективності алгоритмів при стискуванні текстових файлів

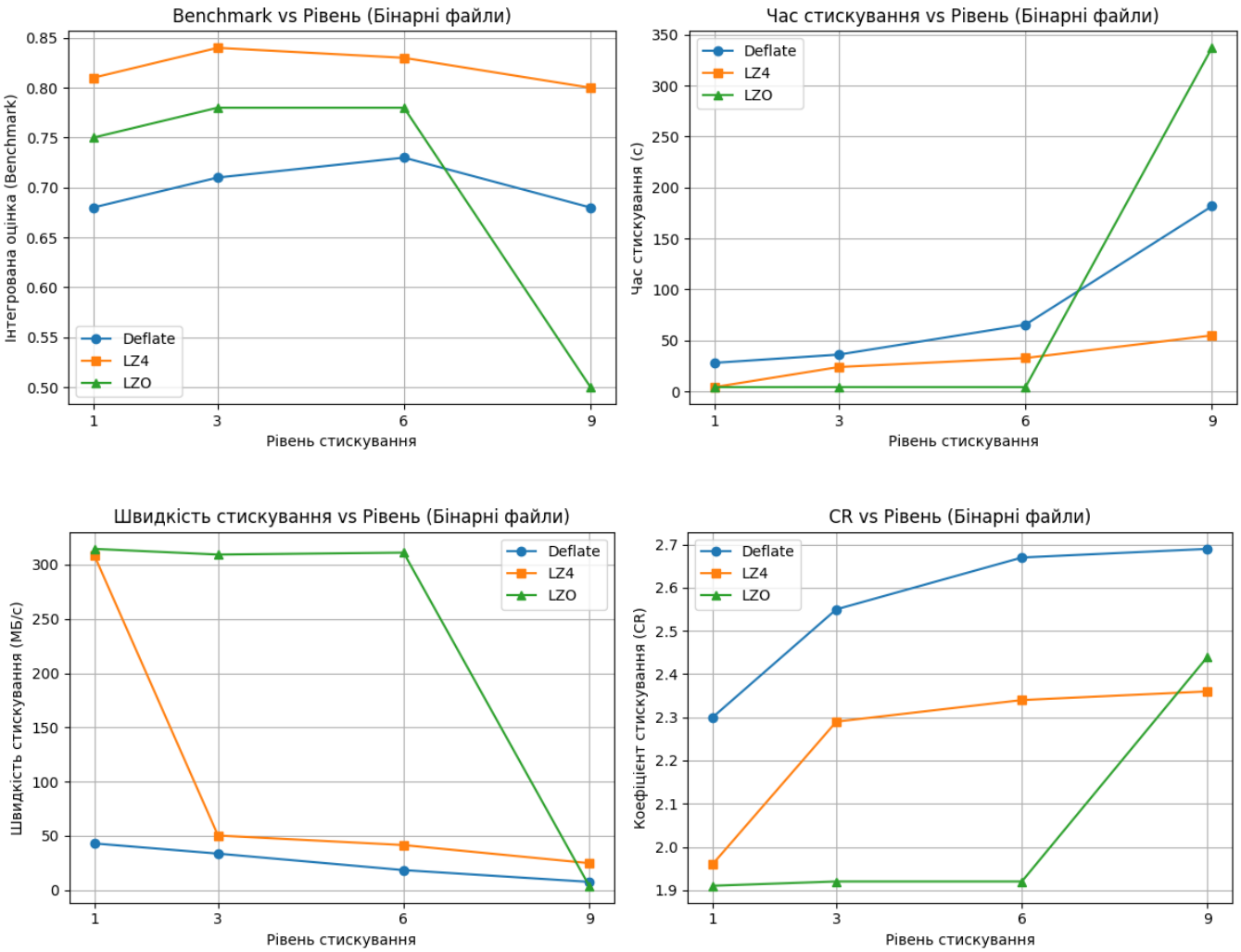


Рисунок 1.19. Порівняння параметрів ефективності алгоритмів при стискуванні бінарних файлів

1.6.4 Аналіз отриманих результатів тестування

За результатами тестування можна зробити наступні висновки:

1. Алгоритм стискування без втрат Deflate:
   - Переваги: Забезпечує високий коефіцієнт стискування для текстових файлів (CR ≈ 6.23–8.38), що дозволяє суттєво зменшити обсяг вхідних даних;
   - Недоліки: Зі збільшенням рівня стискування час обробки зростає, особливо для бінарних файлів (наприклад, при рівні 9 T ≈ 181.7 с), що може бути неприйнятним для систем з високою інтенсивністю даних;
   - Benchmark: Найкращий результат досягається на середньому рівні (0.71), проте при надзвичайно високому ступені стиснення виникає імпакт через дуже низьку швидкість;

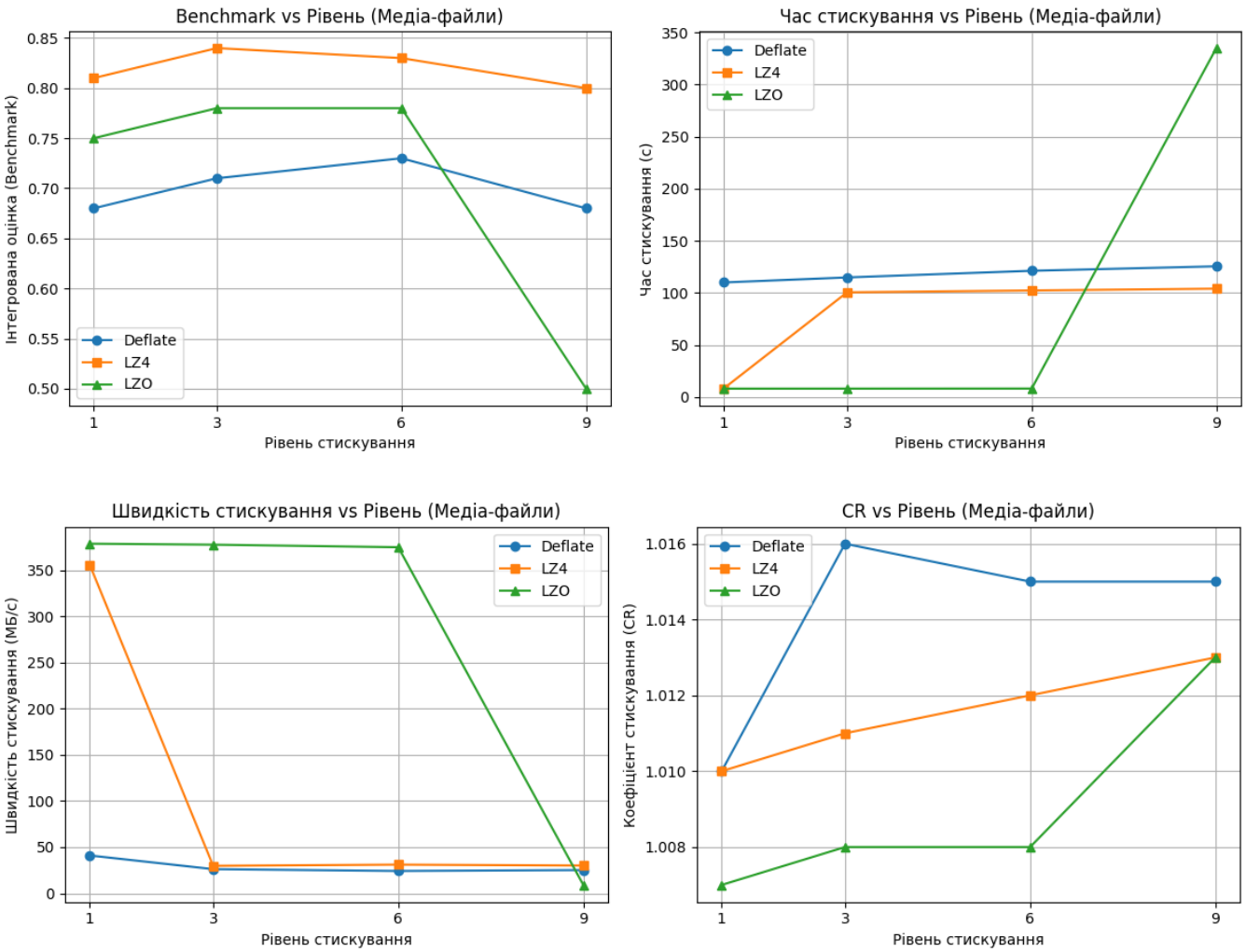


Рисунок 1.20. Порівняння параметрів ефективності алгоритмів при стискуванні медіа-файлів

2. Алгоритм стискування без втрат LZ4:
   - Переваги: Забезпечує значно вищу швидкість стискання, особливо для текстових файлів (на рівні 1 S ≈ 486.3 МБ/с). Баланс між стискуванням і швидкістю оптимальний, що підтверджується найвищою інтегрованою оцінкою (Benchmark до 0.84 на рівні 3);
   - Недоліки: При переході на максимальний рівень стискування спостерігається деяке погіршення швидкості (наприклад, на рівні 9 S ≈ 74.8 МБ/с), що може вплинути на продуктивність при великих об’ємах даних;
3. Алгоритм стискування без втрат LZO:
   - Переваги: Відзначається надзвичайно високою швидкістю обробки на низьких рівнях стискування (наприклад, рівень 1 для текстових файлів

дає S ≈ 420.35 МБ/с);

- Недоліки: При збільшенні рівня стиснення (рівень 9) швидкість різко падає (S ≈ 22.15 МБ/с для текстових файлів), що може вплинути на ефективність систем реального часу. Таким чином, LZO підходить для завдань, де пріоритетом є швидкість, але ефективність стискування залишається нижчою;
- Benchmark: Найбільша інтегральна оцінка досягається на середніх рівнях (0.78 при рівнях 3 та 6), проте при максимальному стискуванні значення Benchmark різко знижується (0.50).

На основі експериментального аналізу можна зробити наступні рекомендації:

- LZ4 є найоптимальнішим з огляду на баланс між високою швидкістю стискування та ефективним коефіцієнтом стискування. Особливо ефективним в системах, де критичною є швидкість обробки даних;
- Deflate поки забезпечує найвищий рівень стискування для текстових файлів, проте його значне збільшення часу обробки на вищих рівнях може обмежити застосування в умовах обробки великих обсягів даних;
- LZO демонструє виняткову швидкість на низьких рівнях, проте при високому ступені стискування його застосовність обмежена через низьку швидкість, що може бути неприйнятним для задач з обмеженим часом.

З урахуванням інтегрованої оцінки (Benchmark) та вимог сучасних систем резервного копіювання, застосунок дозволяє на базі симуляційних тестів обрати найкращий алгоритм стискування – у даному випадку алгоритм LZ4 займає провідну позицію, за ним слідують алгоритми LZO (для апаратних реалізацій з високою швидкістю) та Deflate (при потребі максимального зменшення об’єму даних).

2 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Сучасне впровадження комп'ютерної техніки дає змогу автоматизувати безліч рутинних процесів, оптимізувати обробку даних і забезпечити оперативний доступ до численних інформаційних ресурсів, а також виконання необхідних розрахунків. Ці можливості сприяють значному підвищенню продуктивності праці та ефективності роботи організацій. Втім, з широковживанням персональних комп'ютерів у професійній діяльності зростають і певні негативні наслідки, зокрема – підвищене навантаження на здоров'я співробітників через тривалий час роботи за комп'ютером.

В контексті даного дипломного проєкту розробка програмної моделі аналізу ефективності алгоритмів стискування без втрат у системах резервування даних передбачає створення аналітичного інструмента, який ґрунтується на сучасних підходах до оптимізації збереження інформації. Центральним елементом такого аналізу є робоче середовище користувача, що включає комп'ютерну систему з встановленим програмним забезпеченням, спеціально розробленим для оцінки продуктивності та надійності алгоритмів стиснення даних без втрат.

Оскільки алгоритми стиснення без втрат відіграють ключову роль у забезпеченні ефективного резервування даних, їхній аналіз дозволяє автоматизувати процес вибору оптимального рішення, враховуючи комплекс факторів, що впливають на продуктивність, швидкість обробки та рівень збереження інформації. Важливим аспектом залишається дотримання стандартних вимог безпеки праці користувача ПК, що забезпечує комфортну роботу та мінімізує ризики при інтенсивному використанні технологій стиснення.

Таким чином, створення даної моделі дозволяє не лише оцінити ефективність алгоритмів стискування без втрат у системах резервування даних, а й розробити рекомендації щодо їх оптимального використання в реальних умовах експлуатації.

2.1 Аналіз небезпечних і шкідливих факторів, що впливають на користувача ПК

До основних критеріїв забезпечення гігієни робочого середовища належать інтенсивність освітлення, температура повітря, вологість, рівень шумового

забруднення, ступінь вібраційного впливу, токсичність, загазованість, а також обмеження загальної м'язової активності (гіподинамія). Крім цього, враховується дія електростатичного поля та вплив як неіонізуючих, так і іонізуючих електромагнітних випромінювань.

2.2 Гігієнічні вимоги до виробничого середовища

Державні санітарні норми, зокрема ДСАНПіН 3.3.2.007-98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», спрямовані на запобігання негативного впливу шкідливих чинників, що супроводжують роботу з візуальними дисплейними терміналами, на здоров’я працівників.

2.2.1 Вимоги до приміщення

Розміщення робочих місць із використанням ВДТ, ЕОМ і ПЕОМ заборонено у підвальних приміщеннях та на цокольних поверхах. Для кімнат, призначених для роботи з візуальними дисплейними терміналами, рекомендується орієнтувати вікна у напрямку півночі або північного сходу. На вікнах повинні бути встановлені регульовані жалюзі або штори, що дозволяють їх повністю закривати для забезпечення оптимальних умов освітлення.

Планувальні рішення будівель і приміщень, де розташовано відеодисплейні термінали, мають відповідати вимогам ДСАНПіН 3.3.2.007-98. Для робочого місця програміста передбачено мінімальну площу не менше 6 кв. м та об’єм приміщення не менше 20 куб. м. Крім того, стіни приміщень повинні бути пофарбовані матовою фарбою, а в приміщеннях з ВДТ обов’язково мають бути передбачені зони для відпочинку та психологічного розвантаження.

2.2.2 Освітлення

Для забезпечення належного освітлення приміщення, де працює програміст, застосовується комбінована система, що поєднує природне освітлення із додатковим штучним світлом. Загальне оздоблення простору виконується за допомогою газорозрядних ламп типу ЛД. Згідно з встановленими нормами, для робочого місця, на якому здійснюються високоточні операції (де мінімальний

розмір об'єкта розрізнення становить 0,3–0,5 мм), необхідна освітленість рівномірно має досягати 300 лк. В цілому, ці вимоги щодо освітлення забезпечені.

2.2.3 Шум

У робочих приміщеннях основним джерелом шумового навантаження є звуки, що генеруються ПЕОМ. Крім того, значну частину шуму створюють джерела електромагнітного походження – це коливання компонентів електромеханічних пристроїв під впливом змінних магнітних полів. До того ж, в приміщеннях виникає структурний шум, який випромінюють поверхні конструктивних елементів (стіни, перекриття, перегородки) у звуковому спектрі частот. Для зниження або усунення негативного впливу шуму доцільно ізолювати робочі зони, розташовуючи їх у частинах будівлі, що знаходяться в глибині та ведуть своїми вікнами у двір – таким чином мінімізується вплив міського шуму. Крім цього, необхідно регулярно перевіряти герметичність корпусів комп’ютерної техніки та своєчасно здійснювати заміну вентиляторів охолодження.

2.3 Вимоги до організації робочого місця працівника

Конструкція робочого місця користувача комп’ютера, з урахуванням розташування сидіння, засобів керування та засобу відображення інформації, розроблена згідно з антропометричними, фізіологічними та психологічними вимогами, а також відповідно до специфіки виконуваної роботи. Робоче меблеве обладнання повинно бути оснащене можливістю індивідуального регулювання, що дозволить адаптувати його під зріст кожного користувача й підтримувати оптимальну, зручну поставу. Робочий стіл рекомендовано обробляти матовим покриттям, що сприяє зменшенню небажаних відблисків. > > Розміщення дисплея організовано таким чином, щоб його верхня межа відповідала рівню очей, а відстань до екрану становила приблизно 70 см – що повністю входить у допустимий інтервал від 60 до 90 см. Частота мерехтіння екрану fмер дорівнює 100 Гц, що значно перевищує мінімальне рекомендоване значення у 70 Гц. Крім цього, робоче місце розташовано перпендикулярно до віконних прорізів, що дозволяє уникнути прямого та відбитого світлового мерехтіння від вікон та джерел штучного освітлення.

2.4 Мікроклімат

Показники мікроклімату, складу іонів у повітрі, а також рівень шкідливих речовин у робочих зонах, де використовуються ПК, мають відповідати вимогам ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Для підтримки нормативних значень мікроклімату та забезпечення оптимального співвідношення позитивних і негативних іонів слід передбачити установку пристроїв зволоження, штучної іонізації або кондиціювання повітря. Крім того, рівні інфрачервоного випромінювання не повинні перевищувати встановлених нормативних меж згідно з ГОСТ 12.1.005. Також вміст озону в робочій зоні не має перевищувати 0,1 мг/м³, оксидів азоту – 5 мг/м³, а концентрація пилу повинна залишатися в межах 4 мг/м³.

2.5 Електробезпека

Приміщення, де використовуються імпульсні джерела живлення згідно з ОНТП24-86 і ПУЕ-87, віднесено до категорії об'єктів, де ризик ураження персоналу електричним струмом не є підвищеним. Це пояснюється тим, що відносна вологість повітря не перевищує 75%, температура залишається нижчою за 35°С, а хімічно агресивні середовища відсутні. Електроживлення обладнання організовано від двофазної мережі з заземленою нейтраллю, при напрузі 220 В і частоті 50 Гц, із застосуванням автоматичних пристроїв токового захисту.

В приміщенні обов’язково має бути встановлена схема заземлення. Ураження електричним струмом може виникнути у випадках: 1) при контакті з відкритими струмоведучими елементами; 2) при торканні неструмоведучих частин обладнання, які, через порушення ізоляції або інші причини, опинилися під напругою.

Відповідно до вимог ГОСТ-12.2.007.0-75 устаткування (за винятком ЕОМ II класу) відноситься до I класу та оснащене робочою ізоляцією згідно з ГОСТ 12.1.009-76. Підключення обладнання здійснено згідно з нормативами ПБЕ та ПУЕ, тому додаткових заходів щодо електробезпеки не вимагається.

2.6 Пожежна безпека

Робоче приміщення, що відповідає вимогам ПБЕ та ОНТП 24–86 у сфері вибухово-пожежної безпеки, класифікується як об’єкт категорії «В».

Основними потенційними причинами виникнення пожежі в такому приміщенні є:

1. Коротке замикання електропроводки;

2. Використання побутових електрорадіоприладів;

3. Недотримання встановлених норм протипожежного захисту.

Відповідно до ПУЕ, для зниження ризику виникнення пожежі необхідно забезпечити комплекс заходів, зокрема: ретельну ізоляцію всіх струмоведучих проводів, що підключені до робочих місць, регулярний огляд та перевірку стану їх ізоляції, а також суворе дотримання норм безпечної експлуатації обладнання.

Для гасіння пожеж на робочому місці користувача ПК застосовують як вуглекислотні, так і порошкові вогнегасники.

– Вуглекислотні вогнегасники випускаються у варіанті ручних пристроїв (наприклад, ВВК-5);

– Порошкові вогнегасники представлені моделями ВП-2, ВП-5, ВП-10 та іншими

Рисунок 2.1. Засоби пожежогасіння

ВИСНОВКИ

Під час виконання кваліфікаційної роботи виконано аналіз ефективності алгоритмів стискування без втрат на основі реальних даних, реалізовано простий web-застосунок, що дозволяє проводити тестування ефективності алгоритмів стискування без втрат у контексті резервування даних. Розроблений застосунок забезпечує завантаження реальних вхідних файлів, зчитування їх вмісту за допомогою FileReader і подальше стиснення даних із застосуванням сучасних алгоритмів (Deflate – через бібліотеку pako, LZ4 – через lz4js та LZO – за допомогою js-lzo). Програма вимірює критичні параметри: час стискання, фактичний розмір стисненого файлу (визначається як довжина отриманого Uint8Array), швидкість стискання у мегабайтах на секунду та коефіцієнт стискання, що розраховується як відношення початкового розміру файлу до стиснутого розміру.

Експериментальний аналіз дозволив об’єктивно порівняти алгоритми за їхніми показниками. З отриманих результатів видно, що алгоритм LZ4 демонструє найкращий баланс між швидкістю обробки і ефективністю зменшення обсягу даних. Він забезпечує високі показники швидкості на реальних вхідних даних і дозволяє значно зменшити час стискання, що особливо важливо у системах резервного копіювання з високою інтенсивністю роботи. Алгоритм Deflate, хоч і володіє високою здатністю до зменшення обсягу даних, вимагає значно більше часу для обробки, що ускладнює його використання в режимах, де критична оперативність. Алгоритм LZO вирізняється надзвичайно коротким часом стискання, але його нижчий коефіцієнт стискання спричиняє збільшення обсягу стиснених даних, що може створювати додаткове навантаження на систему зберігання.

Виконана робота містить об’єктивну порівняльну характеристику алгоритмів стискання без втрат, дозволяючи приймати оптимальні рішення щодо вибору алгоритму для резервного копіювання. Результати даної роботи можуть стати корисними для розробників IT-систем, адміністраторів баз даних і фахівців з інформаційної безпеки, що прагнуть до оптимізації зберігання та обробки великих обсягів інформації.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Данилюк І. Б. Резервне копіювання даних: принципи та сучасні технології. – Київ: Центр учбової літератури, 2021.
2. Литвиненко О. А. Методи оптимізації алгоритмів стискання даних. – Львів: Фенікс, 2021.
3. Орлова В. М. Розробка рішень для резервного копіювання даних. – Львів: Ліра, 2021.
4. Козак М. П. Алгоритми стискання даних: теоретичні основи і практичні застосування. – Київ: Видавничий дім «Науковий Світ», 2022.
5. Гнатюк Л. В. Сучасні алгоритми компресії без втрат для веб-систем. – Одеса: Чорномор'я, 2021.
6. Бурундуков О. В. Основи веб-розробки: HTML5, CSS3, JavaScript. – Київ: Видавничий дім «Української Галактики», 2021.
7. Дьяків А. П. Веб-додатки на JavaScript: практичний посібник. – Харків: ХНУ ім. Казимира Федоровича, 2020.
8. Мельник С. О. Розробка веб-інтерфейсів із застосуванням сучасних бібліотек. – Київ: Логос, 2020.
9. Кравченко І. О. Програмування для веб-розробки: від початківця до професіонала. – Київ: Апріорі, 2022.
10. Шевченко Т. Я. Сучасні підходи до обробки даних в Інтернеті. – Київ: 2022.
11. Pako: JavaScript Port of zlib [Електронний ресурс]. – Режим доступу: https://github.com/nodeca/pako (Дата звернення: 2.05.2025).
12. lz4js – High Performance LZ4 Compression in JavaScript [Електронний ресурс]. – Режим доступу: https://github.com/pierrec/js-lz4 (Дата звернення: 07.05.2025).
13. JS-LZO Library [Електронний ресурс]. – Режим доступу: https://github.com/infusion/JS-LZO (Дата звернення: 09.05.2025).
14. Metro UI CSS Framework [Електронний ресурс]. – Режим доступу: https://metroui.org.ua/ (Дата звернення: 10.05.2025).
15. jQuery Library [Електронний ресурс]. – Режим доступу: https://jquery.com/ (Дата звернення: 12.05.2025).

ДОДАТОК А. Фрагмент коду мовою HTML для розмітки, CSS для стилізації та JavaScript для логіки роботи застосунку

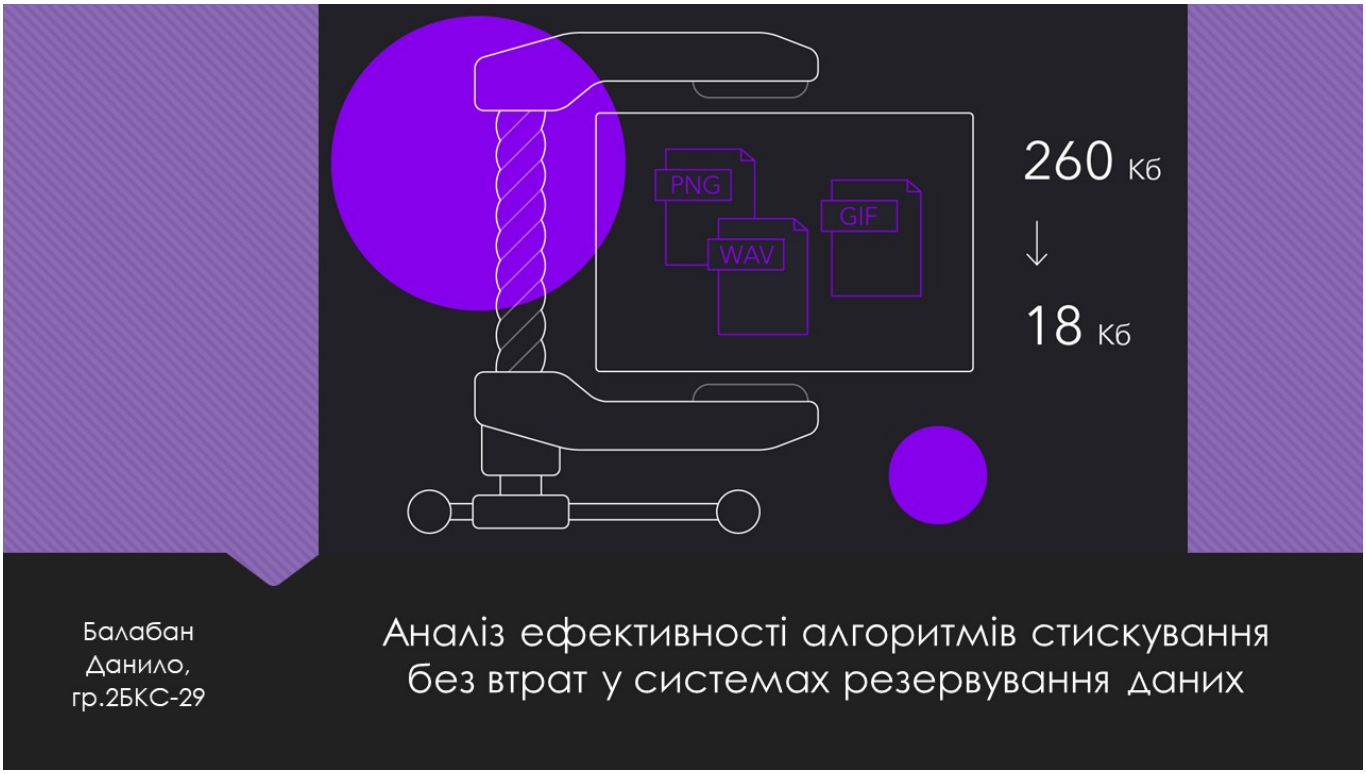
```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title> Тестування ефективності алгоритмів стиснення</title>
  <!-- Підключення Metro UI, jQuery та наших бібліотек для стиснення -->
  <link rel="stylesheet"
   href="https://cdn.metroui.org.ua/v4.3.2/css/metro-all.min.css">
  <style>
    body {
      padding: 20px;
      background-color: #f5f5f5;
      font-family: Helvetica, sans-serif;
    }
    .container {
      max-width: 900px;
      margin: auto;
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 2px 2px 10px rgba(0,0,0,0.15);
    }
    h2, h4 {
      text-align: center;
    }
    .section {
      margin-bottom: 20px;
    }
    #resultsTable {
      margin-top: 20px;
    }
  </style>
  <!-- Підключення бібліотек для стиснення -->
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/pako/2.1.0/pako.min.js"></script>
  <script src="https://unpkg.com/lz4js@0.0.2/dist/lz4.js"></script>
  <script src="https://unpkg.com/js-lzo/dist/js-lzo.min.js"></script>
</head>
<body>
  <div class="container">
    <h2> Тестування ефективності алгоритмів стиснення</h2>

    <!-- Завантаження файлу -->
    <div class="section">
      <h4>Завантажте файл для аналізу</h4>
      <input type="file" id="fileInput" accept="*/*">
      <p id="fileInfo"></p>
    </div>

    <!-- Слайдер для налаштування пріоритету -->
    <div class="section">
      <h4>Встановіть пріоритет: ефективність стиснення vs швидкість</h4>
      <div data-role="slider" id="prioritySlider" data-value="50"
       data-orientation="horizontal" data-show-hint="true"></div>
      <p id="sliderValue">50%</p>
    </div>

    <!-- Кнопки -->
    <div class="section" style="text-align: center;">
      <button id="btnAnalyze" class="button primary">Провести аналіз</button>
      <button id="btnClear" class="button secondary">Очистити</button>
    </div>
```

```
  <!-- Результати -->
  <div class="section">
    <h4>Результати аналізу</h4>
    <table id="resultsTable" class="table striped">
      <thead>
        <tr>
          <th>№</th>
          <th>Алгоритм</th>
          <th>Час стискання (с)</th>
          <th>Стиснений розмір (байт)</th>
          <th>Швидкість (МБ/с)</th>
          <th>Коефіцієнт</th>
          <th>Інтегрована оцінка (Benchmark)</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td colspan="7" style="text-align: center;">
           Результати з’являться після аналізу</td>
        </tr>
      </tbody>
    </table>
  </div>
</div>

<!-- Підключення jQuery та Metro UI -->
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<script src="https://cdn.metroui.org.ua/v4.3.2/js/metro.min.js"></script>

<script>
  /***** Глобальні змінні *****/
  var fileBuffer = null; // ArrayBuffer файлу
  var fileSizeBytes = 0; // Розмір файлу в байтах

  /***** Обробка вибору файлу *****/
  $('#fileInput').on('change', function(e) {
    var file = e.target.files[0];
    if (file) {
      fileSizeBytes = file.size;
      var fileSizeMB = (fileSizeBytes / (1024 * 1024)).toFixed(2);
      $('#fileInfo').text("Розмір файлу: " + fileSizeMB + " МБ");
      var reader = new FileReader();
      reader.onload = function(event) {
        fileBuffer = event.target.result; // ArrayBuffer
      }
      reader.readAsArrayBuffer(file);
    } else {
      $('#fileInfo').text("");
      fileBuffer = null;
      fileSizeBytes = 0;
    }
  });

  // Оновлення значення слайдера
  $('#prioritySlider').on('change', function(){
    var value = Metro.getPlugin('#prioritySlider', 'slider').val();
    $('#sliderValue').text(value + "%");
  });

  /***** Функції стиснення *****/
  // Функція стискання за допомогою pako (Deflate)
  function compressDeflate(buffer) {
    var start = performance.now();
    var compressed = pako.deflate(buffer);
    var end = performance.now();
    return {
      compressedData: compressed,
      time: (end - start) / 1000  // в секундах
    };
```

```
}

// Функція стискання за допомогою lz4js (LZ4)
function compressLZ4(buffer) {
  var start = performance.now();
  // lz4js.compress приймає Uint8Array. Перетворимо ArrayBuffer у Uint8Array:
  var uint8Array = new Uint8Array(buffer);
  var compressed = lz4js.compress(uint8Array);
  var end = performance.now();
  return {
    compressedData: compressed,
    time: (end - start) / 1000
  };
}

// Функція стискання за допомогою LZO
function compressLZO(buffer) {
  var start = performance.now();
  // бібліотека LZO надає функцію LZO.compress, яка приймає Uint8Array
  var uint8Array = new Uint8Array(buffer);
  var compressed = LZO.compress(uint8Array);
  var end = performance.now();
  return {
    compressedData: compressed,
    time: (end - start) / 1000
  };
}

/***** Обробка і аналіз *****/
$('#btnAnalyze').on('click', function() {
  if(!fileBuffer || fileSizeBytes <= 0) {
     alert("Будь ласка, завантажте файл для аналізу!");
     return;
  }
   // Отримуємо значення слайдера (0-100) для визначення вагових коефіцієнтів
  var sliderValue = Metro.getPlugin('#prioritySlider', 'slider').val();
  var weightCompression = (100 - sliderValue) / 100;
  // вага для коефіцієнта стиснення
  var weightSpeed = sliderValue / 100;
  // вага для швидкості стискання

  var results = [];
  // Обчислити для кожного алгоритму:
  // 1. Викликати функцію стиснення
  // 2. Визначити час стискання, розмір виведеного масиву (байти)
  // 3. Обчислити коефіцієнт стиснення = (оригінальний розмір / стиснений розмір)
  // 4. Обчислити швидкість стискання (МБ/с)
  // 5. Зібрати результати

  // Deflate (pako)
  var deflateResult = compressDeflate(fileBuffer);
  var deflateCompressedSize = deflateResult.compressedData.length; // у байтах
  var deflateCR = fileSizeBytes / deflateCompressedSize;
  var deflateSpeed = (fileSizeBytes / (1024 * 1024)) / deflateResult.time;

  results.push({
     name: "Deflate",
     T: deflateResult.time,
     compressedSize: deflateCompressedSize,
     S: deflateSpeed,
     cr: deflateCR
  });

  // LZ4 (lz4js)
  var lz4Result = compressLZ4(fileBuffer);
  var lz4CompressedSize = lz4Result.compressedData.length;
  var lz4CR = fileSizeBytes / lz4CompressedSize;
  var lz4Speed = (fileSizeBytes / (1024 * 1024)) / lz4Result.time;
```

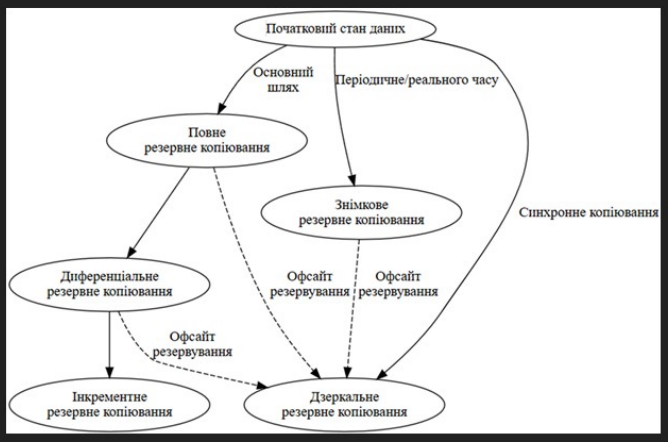
```
      results.push({
         name: "LZ4",
         T: lz4Result.time,
         compressedSize: lz4CompressedSize,
         S: lz4Speed,
         cr: lz4CR
      });

      // LZO (js-lzo) - якщо бібліотека не працює,
      // необхідно перевірити її доступність
      var lzoResult = compressLZO(fileBuffer);
      var lzoCompressedSize = lzoResult.compressedData.length;
      var lzoCR = fileSizeBytes / lzoCompressedSize;
      var lzoSpeed = (fileSizeBytes / (1024 * 1024)) / lzoResult.time;

      results.push({
         name: "LZO",
         T: lzoResult.time,
         compressedSize: lzoCompressedSize,
         S: lzoSpeed,
         cr: lzoCR
      });

     // Для розрахунку Benchmark знайти максимальні значення S та CR серед алгоритмів
      var maxCR = Math.max.apply(null, results.map(r => r.cr));
      var maxS  = Math.max.apply(null, results.map(r => r.S));

      results.forEach(function(r) {
          // Нормалізуємо значення (відносно максимальних)
          var norm_cr = r.cr / maxCR;
          var norm_S  = r.S / maxS;
          r.Benchmark = weightCompression * norm_cr + weightSpeed * norm_S;
      });

      // Сортуємо результати за Benchmark (за спаданням)
      results.sort((a, b) => b.Benchmark - a.Benchmark);

      // Формуємо HTML для таблиці
      var rows = "";
      results.forEach((r, idx) => {
         rows += `<tr>
           <td>${idx+1}</td>
           <td>${r.name}</td>
           <td>${r.T.toFixed(2)}</td>
           <td>${r.compressedSize}</td>
           <td>${r.S.toFixed(2)}</td>
           <td>${r.cr.toFixed(2)}</td>
           <td>${r.Benchmark.toFixed(2)}</td>
         </tr>`;
      });

      $('#resultsTable tbody').html(rows);
    });

    // Обробник кнопки "Очистити"
    $('#btnClear').on('click', function(){
      $('#fileInput').val('');
      $('#fileInfo').text('');
      fileBuffer = null;
      fileSizeBytes = 0;
      Metro.getPlugin('#prioritySlider', 'slider').setValue(50);
      $('#sliderValue').text("50%");
      $('#resultsTable tbody').html('<tr><td colspan="7" style="text-align:
center;">Результати з’являться після аналізу</td></tr>');
    });
  </script>
</body>
</html>
```

ДОДАТОК Б. Слайди мультимедійної презентації



Параметр	Локальні системи	Хмарні платформи	Гібридні рішення
Надійність	• Використання RAID-технологій для відмовостійкості. • Захист даних у межах окремої організації. • Ризик локальних аварій (пожежа, крадіжка, несправності обладнання).	• Георозподілення та реплікація дають високу стійкість. • Автоматичне резервування даних у декількох дата-центрах. • Залежність від надійності стороннього провайдера.	• Поєднання переваг локального зберігання (оперативний доступ) та хмарного резервування (віддалене захищене копіювання). • Зниження ризику як локальних, так і зовнішніх аварій.
Доступність	• Швидкий доступ до даних через локальну мережу. • Обмеження доступу – лише в межах організації. • Може не забезпечувати круглодобову доступність, якщо не налаштовано додаткові засоби.	• Висока доступність 24/7 з будь-якого місця за умови стабільного інтернет-з'єднання. • Залежність від пропускної здатності каналу та стабільності мережі.	• Забезпечується як швидкий локальний відгук, так і доступність через інтернет. • Можливість резервного відновлення даних з віддалених копій у випадку аварій.
Масштабованість	• Розширення вимагає придбання додаткового обладнання та налаштування інфраструктури. • Фізичні та бюджетні обмеження місцевої інфраструктури.	• Практично необмежене масштабування за умов тарифного плану. • Ресурси можуть динамічно збільшуватися без значних початкових інвестицій.	• Гнучкість: локальна інфраструктура обслуговує щоденні потреби, а хмарні сервіси – архівування і масштабування при зростанні обсягів даних. • Баланс між витратами та ресурсами.
Вартість Впровадження	• Висока первинна інвестиція: придбання серверів, обладнання, ліцензійного програмного забезпечення. • Експлуатаційні витрати (електроенергія, обслуговування, проста зброя).	• Модель «оплата за використання» з низькими початковими витратами, але потенційно високими щомісячними платежами при великому обсязі даних. • Відсутність потреби в експлуатації власної інфраструктури.	• Оптимізація витрат: поєднання власних ресурсів з хмарними сервісами дозволяє зменшити як первинні, так і операційні витрати. • Гнучкість у налаштуванні залежно від конкретних потреб організації.

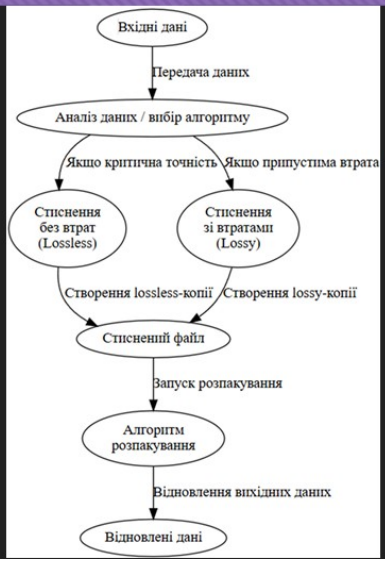
Послідовність створення резервних наборів при різних методах копіювання



Порівняльний аналіз типів резервного копіювання

Тип резервного копіювання	Обсяг збереження	Час відновлення	Витрати на зберігання	Основні переваги	Основні недоліки
Повне резервне копіювання	Найвищий – завжди дублюється весь обсяг даних	Швидке – відновлення здійснюється з однієї повної копії	Високі – зберігається повна копія кожного разу	Простота відновлення, повнота інформації	Значний обсяг збережених даних, тривалість процесу створення копії при великих обсягах
Диферен-ціальне резервне копіювання	Середній – зберігаються лише зміни після останнього повного резервного копіювання	Середній – для відновлення потрібні повна копія та останній диферен-ціальний набір	Середні – обсяг збережених даних зростає між повними копіями	Менший обсяг даних порівняно з повним резервним копіюванням; відносно просте відновлення (тільки 2 файли: повната остання диференціальна)	З часом розмір диференціальної копії може зростати, що ускладнює процес відновлення і може впливати на витрати на зберігання
Інкрементне резервне копіювання	Найнижчий – копіюються лише дані, що змінилися від останнього резервного копіювання	Найдовший – для відновлення необхідно послідовно обробити повну копію та всі інкрементні набори	Найнижчі – кожний інкремент зберігає мінімальну кількість даних	Мінімальний обсяг збереження на кожен резервний набір, що знижує витрати на зберігання; швидке виконання резервного копіювання	Відновлення може бути складнимі тривалим, оскільки потребує послідовної обробки всіх наборів; втрата одного інкременту може ускладнити відновлення
Знімкове резервне копіювання (Snapshot)	Низький – зберігаються лише інформація про зміни (відображення стану даних на рівні системи)	Дуже швидке – майже миттєве відновлення за рахунок знімка	Помірні – залежить від технології зберігання «знімків» (часто використовується технологія копіювання блоків)	Миттєве створення та відновлення резервної копії, мінімальний вплив на продуктивність системи	Обмеження масштабованості та залежність від апаратних засобів; не завжди дає можливість точно контролювати обсяг збережених змін
Дзеркальне резервне копіювання (Mirror backup)	Високий – створюється точна копія всіх даних	Миттєве – доступ до даних з дзеркального копіювання дозволяє швидке відновлення	Дуже високі – потрібно зберігати повну копію всіх даних	Миттєве відновлення, висока актуальність даних, забезпечення безперервності роботи системи	Необхідність виділення великої кількості сховища, високі початкові та експлуатаційні витрати; дублювання всіх даних може бути економічно невигідним

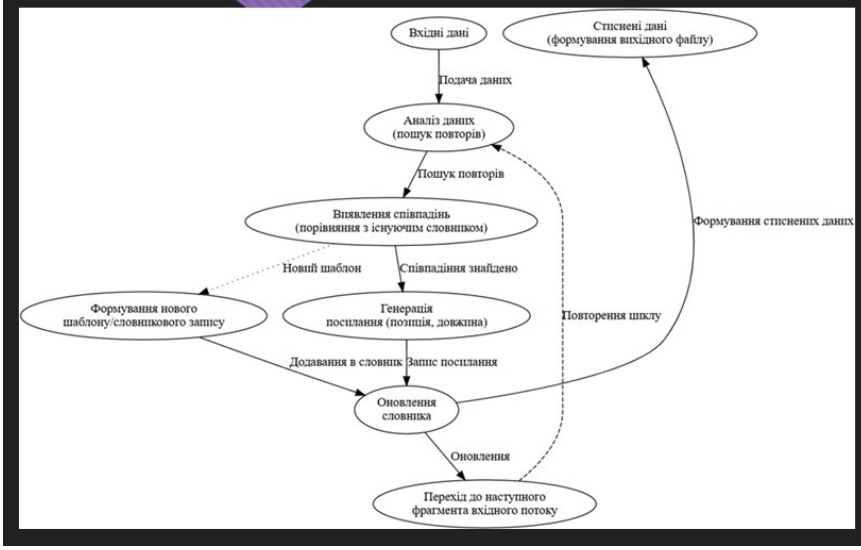
Схема процесу стиснення та відновлення даних



Порівняльна характеристика методів стиснення

Параметр	Стиснення без втрат	Стиснення зі втратами
Основне застосування	Текстові дані, програмний код, бази даних – де важлива абсолютна точність	Мультимедійні дані (зображення, аудіо, відео) – де деякі втрати даних прийнятні для суттєвого зменшення розміру файлу
Алгоритми	LZ77, LZ78, LZW, коди Хаффмана, арифметичне кодування	JPEG, MP3, MPEG, OGG тощо
Коефіцієнт стиснення (CR)	Зазвичай від 2:1 до 4:1 (залежно від структури даних)	Може досягати значень понад 10:1, залежно від параметрів якості та ступеня компресії
Швидкість стиснення/ розпакування	Висока – завдяки алгоритмічним оптимізаціям; збереження всіх даних вимагає іноді додаткового часу	Зазвичай швидший процес стиснення; розпакування може бути трохи спрощеним за рахунок втрати частини інформації
Ресурсоємність	Вимагає розширених обчислювальних ресурсів для досягнення оптимальних коефіцієнтів стиснення	Менше ресурсоємний, однак компроміс між якістю та обсягом досягається шляхом вибору параметрів стиснення
Якість відновлення	Повне відновлення вихідних даних (100%)	Відновлення з втратами, якість залежить від ступеня компресії; дещо спотворені дані можуть бути непомітними для людського ока, але критично важливих даних не відновлюють

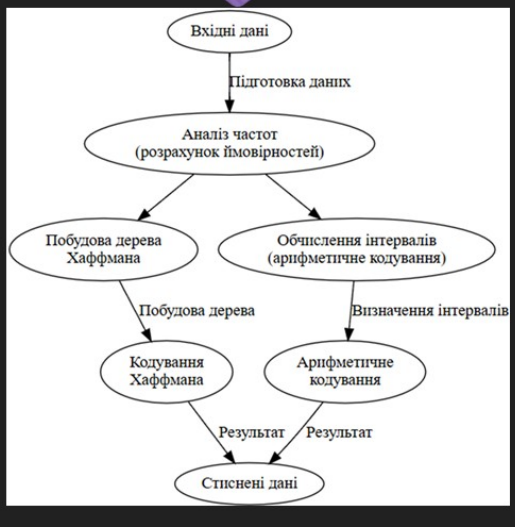
Блок-схема роботи алгоритмів сімейства LZ



Порівняльна характеристика алгоритмів сімейства LZ

Алгоритм	Коефіцієнт стиснення	Швидкість стиснення/ розпакування	Ресурсо-ємність	Основні переваги	Основні недоліки
LZ77	2:1 – 4:1 (залежно від даних)	Висока – ефективний для потокової обробки завдяки використанню ковзного вікна	Помірна – обмеження розміру вікна можуть впливати на обчислю-вальні витрати	Динамічний пошук повторів; добре працює з потоковими даними	Обмеження максимального розміру буфера; не завжди ефективний для даних з низькою повторюваністю
LZ78	2:1 – 4:1	Висока – ефективний для структурованих даних, де легко формуються повторювані шаблони	Середня – поступове нарощення словникової структури може вимагати додаткової пам'яті	Простота реалізації; добрий компроміс між швидкістю та ефективністю	Залежність від якості формування словника; можливе збільшення затрат пам'яті при дуже великих обсягах даних
LZW	2:1 – 5:1	Дуже висока – оптимізоване управління словником дозволяє швидкий доступ до потрібних елементів	Низька – адаптивне управління словни-ковою структурою сприяє ефектив-ному використанню ресурсів	Висока продуктив-ність; широко застосо-вується у стандартах стиснення (GIF, TIFF)	Обмеження розміру словника може вплинути на ефективність при нетрадиційних даних; деградація ефективності при зростанні обсягів даних

Схема ентропійного кодування



Ентропія обчислюється за формулою Шеннона:

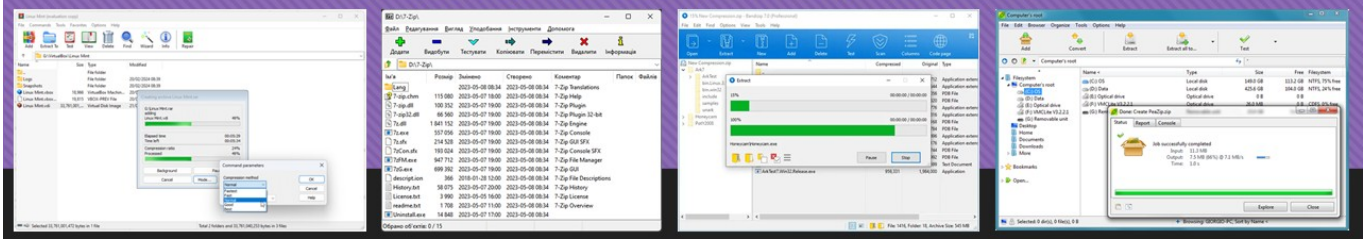
$$(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

де $p(x_i)$ — ймовірність появи символу x_i,
а n — кількість різних символів.

Порівняльний аналіз класичних алгоритмів без втрат

Алгоритм	Принцип роботи	Коефіцієнт стиснення	Швидкість	Ресурсоємність	Переваги	Недоліки
LZ77	Використання ковзного вікна для пошуку повторюваних послідовностей у потоці даних	Середній – залежить від повторю-ваності даних	Висока – особливо ефективний для потокових даних	Помірна – параметри вікна впливають на обчислю-вальні витрати	Простота реалізації, швидке декодування	Обмеження розміру буфера, чутливість до налаштувань, не найкраща ефективність при даних з нерегулярною структурою
LZ78 / LZW	Побудова словникової структури шляхом послідовного додавання нових шаблонів, що повторюються	Залежно від даних – від середнього до високого	Висока – ефективно для структуро-ваних даних	Середня – зростання словника може вимагати додаткових ресурсів	Добрий компроміс між швидкістю та ефективністю, чудово працює при наявності повторюваних шаблонів	Обмеження словника може впливати на ефективність, високі вимоги до оперативної пам'яті при великих обсягах даних
Кодування Хаффмана	Побудова префіксного дерева на основі статистики появи символів для присвоєння коротших кодів частішим символам	Близько до теоретичного ліміту – залежить від розподілу	Дуже висока – особливо при попереднь-ому аналізі частот символів	Низька – алгоритм має просту реалізацію	Простота реалізації, ефективність при стабільних статичних даних	Недостатня адаптивність до локальних змін статистики, менш оптимальне для джерел з динамічною розподільною характерис-тикою
Арифметичне кодування	Кодування всього потоку як одне число за допомогою точного визначення інтервалів на основі ймовірнісного розподілу символів	Максимальний – дозволяє макси-мально наблизи-тися до ліміту ентропії	Помірна – обчислю-вальна складність призводить до нижчої швидкості	Висока – потребує високо-точних арифме-тичних операцій	Оптимальне використання статистики без втрат, високий ступінь стиснення (наближення до теоретичного ліміту)	Складність реалізації, висока обчислювальна складність, труднощі з експлуатацією у режимах реального часу або на системах із обмеженими ресурсами

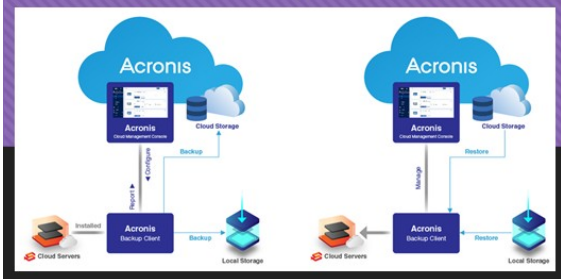
Інтерфейс архіваторів WinRAR, 7 Zip, 7 Zip, PeaZip



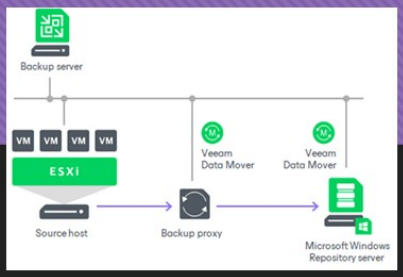
Порівняльний аналіз архіваторів

Архіватор	Підтримувані формати	Алгоритми стиснення	Особливості інтерфейсу	Додаткові функції	Примітки
WinRAR	RAR, ZIP, 7z та ін.	Комбіновані методи (на базі LZ77, багаторівневе стиснення)	Інтуїтивно зрозумілий, налаштовуваний інтерфейс, підтримка самоархівів	Шифрування, створення SFX-архівів, відновлення пошкоджених архівів	Комерційний продукт (з режимом пробного використання)
7-Zip	7z, ZIP, GZIP, BZIP2, TAR, RAR (режим читання)	LZMA, LZMA2	Простий, функціональний файловий менеджер з відкритим кодом	Високий коефіцієнт стиснення, інтеграція з Windows Explorer	Вільне програмне забезпечення (open source)
Bandizip	ZIP, 7z, RAR, TAR та ін.	Оптимізовані алгоритми для швидкої архівації	Сучасний, легкий та адаптивний інтерфейс із багатомовною підтримкою	Швидка архівація/ розпакування, розширені налаштування шифрування	Популярний серед користувачів завдяки високій швидкості
PeaZip	7z, ARC, PAQ, UPX, ZIP, RAR та ін.	Гібридні алгоритми, підтримка численних форматів	Гнучкий та налаштовуваний інтерфейс, відкритий код	Підтримка великої кількості форматів, додаткові засоби безпеки	Повністю безкоштовний, орієнтований на розширену функціональність

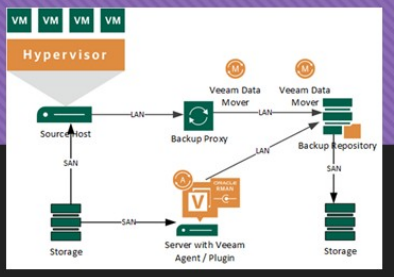
ПЗ Acronis Cyber Protect



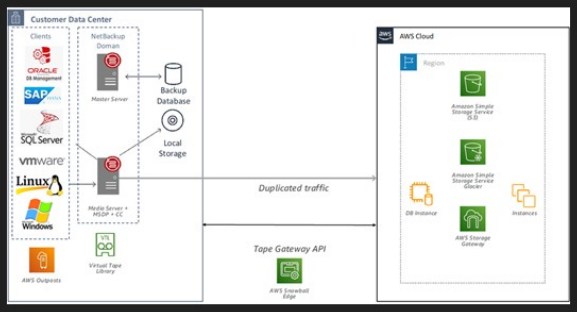
ПЗ Windows Backup



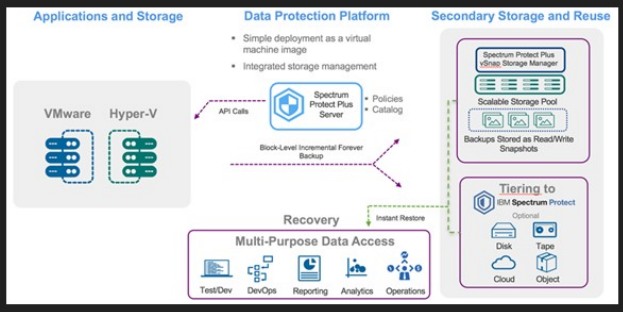
ПЗ Veeam Backup&Replication



ПЗVeritas NetBackup



ПЗ IBM Spectrum Protect



Розрахунок ефективності стискування даних

Для кожного алгоритму визначаються **коефіцієнти стиснення** для текстових, бінарних і медіа-файлів за формулою $K_i = T \cdot K_{i,t} + B \cdot K_{i,b} + M \cdot K_{i,m}$, де T, B та M – відносна кількість файлів відповідних типів, а $K_{i,t}$, $K_{i,b}$, $K_{i,m}$ – коефіцієнти стиснення для певного алгоритму;

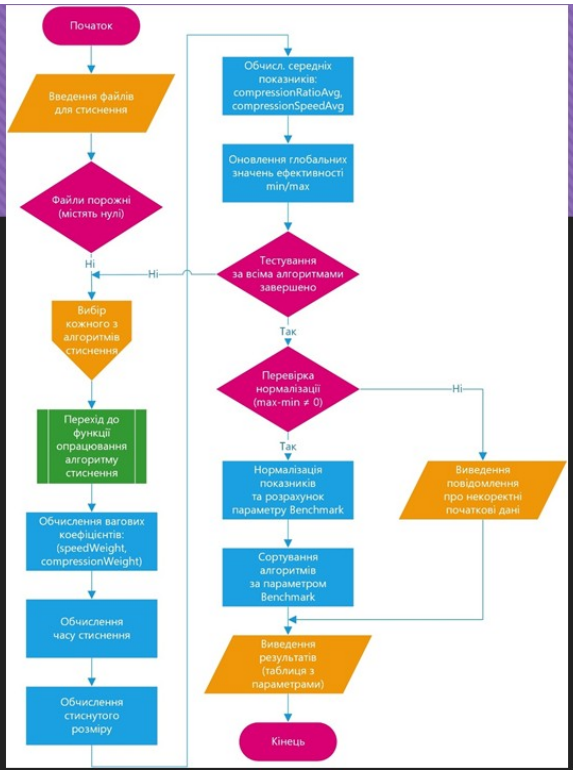
Аналогічно визначаються середні значення **швидкості стиснення** для кожного алгоритму:

$v_i = T \cdot v_{i,t} + B \cdot v_{i,b} + M \cdot v_{i,m}$

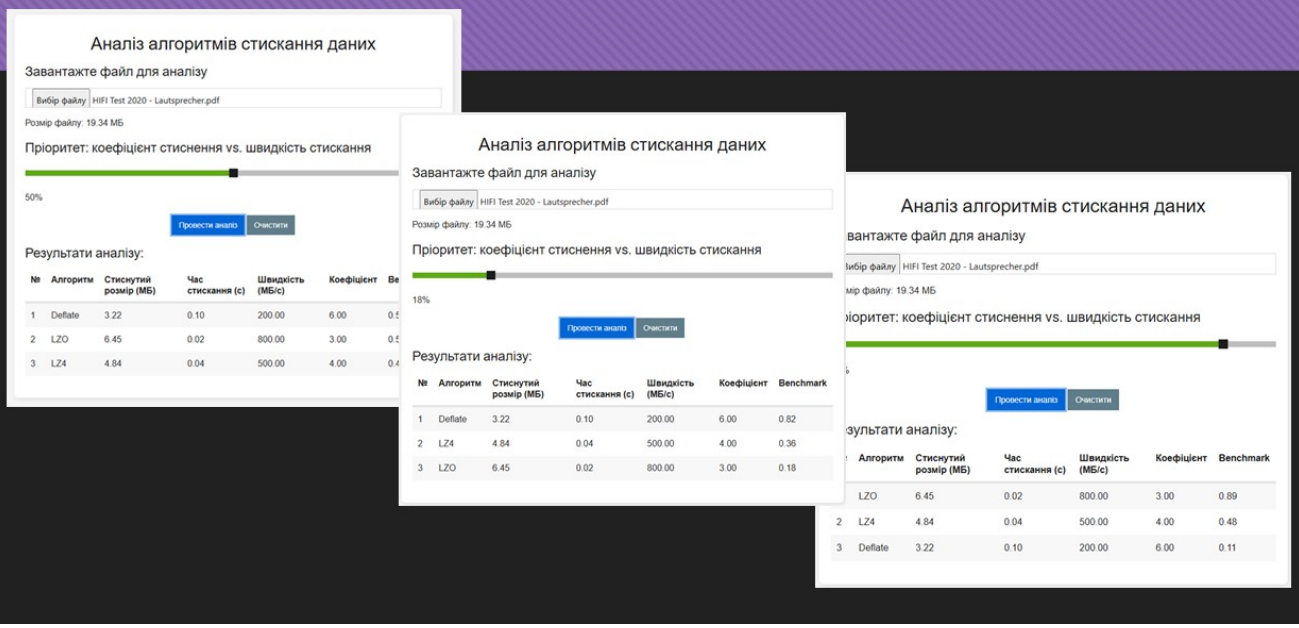
Для подальшого порівняння застосовуються **процедури нормалізації**: $v'i = \frac{v_i - vmin}{v_{max} - v_{min}}, K'i = \frac{K_i - Kmin}{K_{max} - K_{min}}$,

після чого система формує **комплексну оцінку** кожного алгоритму за допомогою цільової функції: $Benchmark\ \ i = aK \times K'i + av \times v'i,$ де aK і a_v – вагові коефіцієнти, що задаються користувачем.

Загальна БСА роботи застосунку для тестування ефективності алгоритмів стискування без втрат



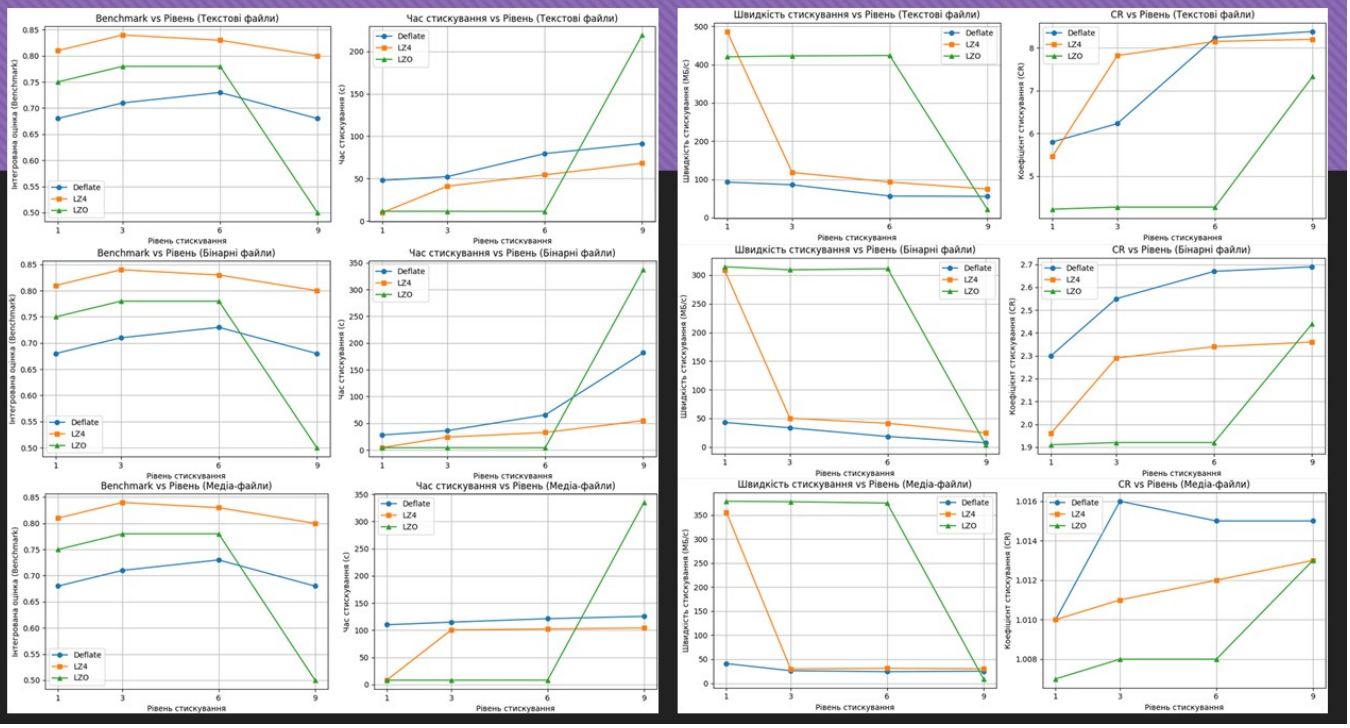
Інтерфейс застосунку для аналізу ефективності алгоритмів стискування



Результати тестування ефективності алгоритмів стискування без втрат

Алгоритм	Рівень	Текстові файли (CR, T(s), S(MB/s))	Бінарні файли (CR, T(s), S(MB/s))	Медіа-файли (CR, T(s), S(MB/s))	Benchmark
Deflate	1	(5.80, 48.0, 93.0)	(2.30, 28.0, 42.9)	(1.010, 110.0, 41.0)	0.68
Deflate	3	(6.23, 52.4, 86.0)	(2.55, 36.1, 33.5)	(1.016, 114.8, 26.2)	0.71
Deflate	6	(8.24, 79.5, 56.6)	(2.67, 65.5, 18.3)	(1.015, 121.2, 24.2)	0.73
Deflate	9	(8.38, 91.4, 55.6)	(2.69, 181.7, 7.45)	(1.015, 125.5, 25.2)	0.68
LZ4	1	(5.46, 10.0, 486.3)	(1.96, 4.20, 308.6)	(1.010, 8.36, 355.9)	0.81
LZ4	3	(7.82, 41.1, 118.23)	(2.29, 23.9, 50.2)	(1.011, 100.5, 29.8)	0.84
LZ4	6	(8.15, 54.5, 93.1)	(2.34, 32.72, 41.4)	(1.012, 102.3, 31.1)	0.83
LZ4	9	(8.20, 68.12, 74.8)	(2.36, 54.91, 24.7)	(1.013, 104.03, 30.1)	0.80
LZO	1	(4.23, 11.55, 420.35)	(1.91, 4.12, 314.61)	(1.007, 8.02, 378.81)	0.75
LZO	3	(4.28, 11.49, 422.72)	(1.92, 4.19, 309.37)	(1.008, 8.04, 377.76)	0.78
LZO	6	(4.28, 11.46, 423.68)	(1.92, 4.16, 311.14)	(1.008, 8.10, 375.00)	0.78
LZO	9	(7.33, 219.28, 22.15)	(2.44, 337.18, 3.84)	(1.013, 335.08, 9.07)	0.50

Порівняння параметрів ефективності алгоритмів при стискуванні різних типів файлів



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу здобувача (здобувачки) освіти

відділення комп'ютерних систем

Балабана Данила Олеговича

(прізвище, ім`я та по батькові)

Спеціальність *123 "Комп'ютерна інженерія"*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Керівник кваліфікаційної роботи *Кривченко Юрій Вікторович*

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи *Аналіз ефективності алгоритмів стискування без втрат у системах резервування даних*

Обсяг розрахунково-пояснювальної записки *71* сторінок

Обсяг графічної (презентаційної) частини *14* аркушів (слайдів)

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаного кваліфікаційної роботи завданню

Представлена на рецензію кваліфікаційна робота бакалавра повністю відповідає меті випускної роботи та технічному завданню. Тематика кваліфікаційної роботи є актуальною для своєї галузі та присвячена аналізу ефективності алгоритмів стискування без втрат у системах резервування даних.

б) характеристика виконання кожного розділу кваліфікаційної роботи

Кваліфікаційна робота складається зі вступу, двох розділів, висновків, переліку використаних джерел. У основному розділі виконано огляд систем резервного копіювання; аналіз принципів стиснення даних; огляд алгоритмів стискування без втрат; інтеграція алгоритмів стискування у системи резервного копіювання розробка застосунку для аналізу ефективності алгоритмів стискування без втрат; експериментальний аналіз ефективності алгоритмів стискування без втрат

в) оцінка якості виконання пояснювальної записки та графічної частини кваліфікаційної роботи

Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана охайно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату ідей у роботі не виявлено

г) перелік позитивних якостей кваліфікаційної роботи

Дослідження охоплює як теоретичний аналіз принципів стиснення, так і практичне порівняння алгоритмів (Deflate, LZ4, LZO) із застосуванням експериментальних тестів. Практична частина проєкту реалізована за допомогою сучасних веб-технологій (HTML, CSS, JavaScript, Metro UI) та бібліотек для стиснення (pako, lz4js, js-lzo).

д) основні недоліки кваліфікаційної роботи

Занадто детальний опис принципів стиснення даних.

Хоча проект детально аналізує три алгоритми стискування, для більш повної оцінки ефективності варто було врахувати альтернативні підходи або розширити вибір методів.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис:

« 23 » 06 2025 р.

ВСП «Одеський технічний фаховий коледж ОНТУ»

ВІДГУК

керівника про кваліфікаційну роботу бакалавра

Балабана Данила Олеговича

(прізвище, ім`я та по батькові)

Спеціальність *123 "Комп'ютерна інженерія"*

Тема кваліфікаційної роботи *Аналіз ефективності алгоритмів стискування без втрат у системах резервування даних*

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) Обсяг і якість виконання роботи (графічного матеріалу і розрахунково-пояснювальної записки) *Випускна робота виконана відповідно технічному завданню. Пояснювальна записка до випускної роботи містить 71 сторінку. У пояснювальній записці розглянуто проблему забезпечення ефективності та вибору доречного алгоритму стискування без втрат у системах резервування даних. Графічна частина складається з 14 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра, роботу виконано у повному обсязі.*

б) Самостійність роботи *Протягом виконання випускної бакалаврської роботи Балабан Данило поступово та послідовно виконував всі етапи, проявив ініціативу у створенні загальної концепції та реалізації випускної роботи. Всі роботи Балабан Данило виконував самостійно, з оглядом на рекомендації керівника.*

в) Теоретична підготовка здобувача освіти

Балабан Данило під час роботи над випускною бакалаврською роботою вивчив достатню кількість літературних джерел за даною тематикою.

Вважаю, що теоретична підготовка здобувача освіти добра і він готовий до захисту роботи.

г) Вміння розв`язувати виробничі і конструкторські питання на базі останніх досліджень науки і техніки, передових методів виробництва

Під час виконання роботи Балабан Данило мав змогу самостійно приймати окремі рішення з виконання програмної частини роботи та показав вміння організовано працювати над поставленою задачею, складати та оформлювати презентацію проекту, користуючись сучасними комп'ютерними програмними засобами, такими як JavaScript, HTML, CSS, pako, lz4js, js-lzo, Metro UI, jQuery

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім`я, по батькові *Кривченко Юрій Вікторович*

Місце роботи і посада керівника роботи *ВСП "Одеський технічний фаховий коледж ОНТУ", викладач кафедри комп'ютерної інженерії, голова циклової комісії комп'ютерних технологій та програмної інженерії*

Підпис

«____» _____ 20___ р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Балабан Д.О.,
здобувач освіти гр. 2БКС-29, та

Кривченко Ю.В.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи бакалавра на тему:

«Аналіз ефективності алгоритмів стискування без втрат у системах резервування даних» (автор роботи – Балабан Д.О., керівник роботи – Кривченко Ю.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Балабан Д.О. /

Керівник _____ / Кривченко Ю.В. /

«16» червня 2025 р.

ДОВІДКА

кафедри комп'ютерної інженерії
про допуск до захисту кваліфікаційної роботи
здобувача (здобувачки) освіти II курсу
відділення комп'ютерних систем групи 2БКС-29

Балабана Данила Олеговича

на тему *Аналіз ефективності алгоритмів стискування без втрат у системах резервування даних*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до кваліфікаційної роботи виконана з несуттєвими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *20.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 30.05.2025 р. значення коефіцієнту подібності в роботі становить 8,64%, коефіцієнт цитування – 1,32%.*

(підпис) *20.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) кваліфікаційної роботи

здобувача (здобувачки) освіти *Балабана Д.О.* (П.І.Б.)

проведена «*20*» *червня* 2025 р.

Висновки *Пояснювальна записка до кваліфікаційної роботи виконана у повному обсязі. Випускна кваліфікаційна робота відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Зав. кафедри КІ (підпис) *Іванова Л.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 5/30/2025
Дата редагування 5/30/2025

Документ прийнятий

Звіт подібності

метадані

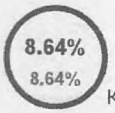
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Аналіз ефективності алгоритмів стискування без втрат у системах резервування даних

Автор
Балабан Данило Олегович
Науковий керівник / Експерт
Кривченко Юрій Вікторович

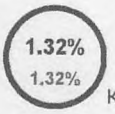
підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

25	14383	123007
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	7
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	47

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка застосунку для редагування тривимірних графічних об'єктів [redacted] 5/29/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	209 1.45 %

2	Розробка застосунку для редагування тривимірних графічних об'єктів [redacted] 5/29/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	205 1.43 %
3	Розробка застосунку для редагування тривимірних графічних об'єктів [redacted] 5/29/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	159 1.11 %
4	Розробка застосунку для редагування тривимірних графічних об'єктів [redacted] 5/29/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	115 0.80 %
5	https://card-file.ontu.edu.ua/server/api/core/bitstreams/d5a3d14f-d5cb-460f-9c49-cba3f9d50554/content	98 0.68 %
6	https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download	39 0.27 %
7	https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download	32 0.22 %
8	Розробка застосунку для редагування тривимірних графічних об'єктів [redacted] 5/29/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	28 0.19 %
9	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	27 0.19 %
10	Розробка застосунку для редагування тривимірних графічних об'єктів [redacted] 5/29/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	25 0.17 %

з домашньої бази даних (5.38 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка застосунку для редагування тривимірних графічних об'єктів [redacted] 5/29/2025 **Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")**	**774 (9) 5.38 %**

з програми обміну базами даних (0.24 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2025_M_EOM_КСМм-23-1_Цуканов_Є_Є [redacted] 1/21/2025 Kharkiv National University of Radio Electronics (каф. ЕОМ)	14 (2) 0.10 %
2	Poiasniuvalna_zapyska_2024_Soloviova_Y_O [redacted] 11/26/2024 National Technical University "Kharkiv Polytechnic Institute" students papers (National Technical University "Kharkiv Polytechnic Institute" students papers)	10 (1) 0.07 %
3	Практика Гоян Денис 242.docx [redacted] 7/3/2023 Yuriy Fedkovych Chernivtsi National University(CNU) course papers (Deanery)	10 (1) 0.07 %

з Інтернету (3.02 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/d5a3d14f-d5cb-460f-9c49-cba3f9d50554/content	98 (1) 0.68 %
2	https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download	97 (6) 0.67 %
3	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download	59 (4) 0.41 %
4	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	29 (3) 0.20 %
5	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	27 (1) 0.19 %
6	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	27 (3) 0.19 %
7	http://dspace.puet.edu.ua/bitstream/123456789/13982/1/Lukash_Denys_Igorovich-Knb-41-1.pdf	27 (2) 0.19 %
8	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	16 (1) 0.11 %
9	https://card-file.ontu.edu.ua/server/api/core/bitstreams/e86ba9fc-9135-43bb-922a-10bf0bce46b5/content	13 (2) 0.09 %
10	http://dspace.wunu.edu.ua/bitstream/316497/51949/1/%D0%A2%D0%B0%D1%80%D0%B0%D0%B1%D0%B0%D0%BD%D0%BE%D0%B2%D0%B8%D1%87.pdf	13 (1) 0.09 %
11	https://card-file.ontu.edu.ua/bitstreams/2b75599e-e1ac-412d-bf09-10d2eb49022f/download	8 (1) 0.06 %
12	http://zavantag.com/docs/427/index-2023448.html?page=3	8 (1) 0.06 %
13	https://ttf.teset.sumdu.edu.ua/images/Masters/Zavdannya.doc	6 (1) 0.04 %
14	https://ir.kneu.edu.ua/bitstreams/850a2498-073e-45fe-874b-28be792ab58d/download	6 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: **«Комп'ютерна інженерія» Група:** 2БКС- 29

КВАЛІФІКАЦІЙНА
РОБОТА

здобувача **освіти денної форми навчання** БКС, 29.01.000, КРБ

БАЛАБАНА
ДАНИЛА ОЛЕГОВИЧА

м. Одеса
2025 р. **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: **«Комп'ютерна інженерія»**
Група: 2 БКС-29

ПОЯСНЮВАЛЬНА **ЗАПИСКА До кваліфікаційної роботи бакалавра на тему:**_____
_____
_____ **Проектний матеріал складається з пояснювальної записки на _____ сторінках та графічного (презентаційного) матеріалу на _____ аркушах (слайдах) Виконавець**