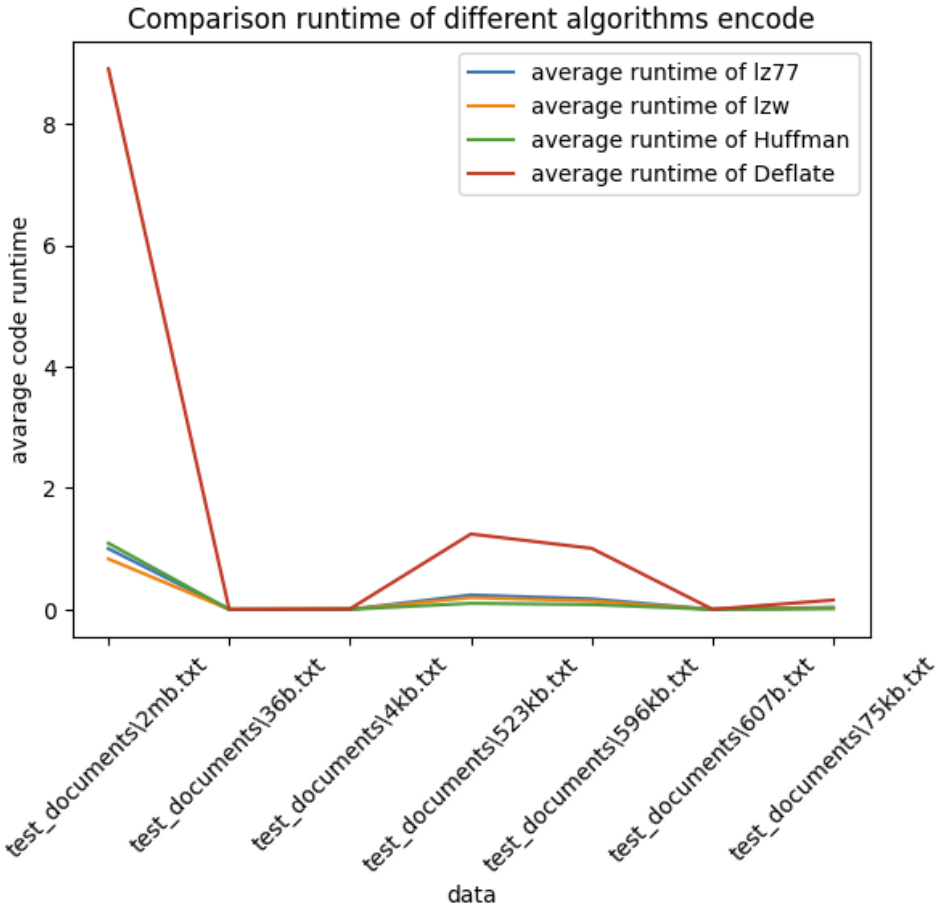
Лабораторна робота №2

Павлюк Богдан
Ільницький Давид

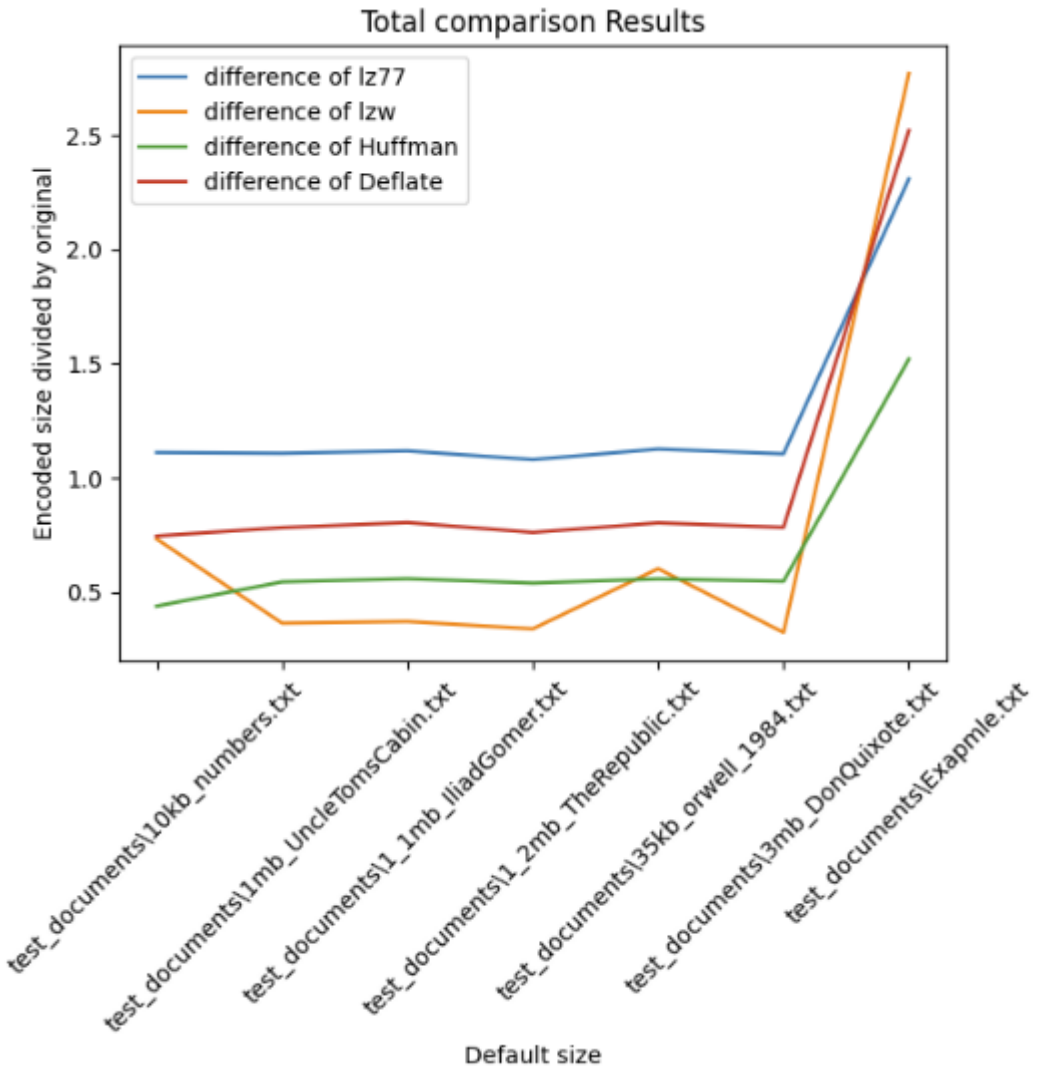
1. Розподіл роботи:
Алгоритм Гаффмана - Ільницький Давид
Lzw - Ільницький Давид
Декодування Lzw - Богдан Павлюк
Lz77 - Богдан Павлюк
Deflate - Богдан Павлюк
Створення ipynb, написання всіх допоміжних функцій для перевірки швидкості алгоритмів та перевірка швидкості алгоритмів, пошук файлів, внесення правок до звіту та ipynb - Богдан Павлюк
Написання коду для порівняння розмірів початкових та закодованих файлів, оцінка стиснення, пошук файлів, написання звіту - Давид Ільницький

2. Графіки часу роботи відносно розміру вхідних даних для кожного з алгоритмів



Домовленості про стиск файлів:
Ми не записували результати алгоритмів у файли і не порівнювали їх на пряму. Для кожного алгоритму був свій підхід.
Гаффман: тут ми отримуємо повідомлення з 0 і 1 та дерево. Довжину повідомлення ми ділимо на 8, а з дерева робимо словник. В словнику для ключів рахуємо довжину і ділимо на 8, а значення сприймаємо відразу як байти. Суму отриманого ділимо на розмір файла
Дефлейт: робився за таким ж принципом, як і Гаффман
LZ77: тут ми отримуємо список, що складається з трьох-елементних кортежів. Ми вважаємо, що кожен кортеж це 3 байти
LZW: тут ми отримуємо код і початковий словник. Кожне значення коду ми вважаємо, що закодоване 2-ма байтима. А словник має вагу по 2 байта на кожен ключ і відповідне йому значення

3. Графік оцінки стиску розміру вхідних даних



Таблиця стиску даних

Розмір закодованого файлу поділений на розмір початкового

	10kb_numbers.txt	1mb_uncle_tomas.txt	3mb_DonQuixote.txt
Lz77	1.11	1.10	1.10
Lzw	0.73	0.36	0.32
Huffman	0.44	0.54	0.55
Deflate	0.74	0.78	0.78

4.

Lzw:

Алгоритм є найоптимальніших з порівняних(на наших даних). Найкраще використовувати коли багато символів(слів) часто повторюються. А найбільш ефективний на файлах дуже великих розмірів

Lz77:

Графік є найменш оптимальним серед описаних. Можна сказати, що він потребує більшої показника споріднених символів, для того щоб працювати оптимально.

Huffman:

У нашому випадку цей алгоритм найкраще з алгоритмів працює з файлом, що складається з рандомізованих чисел. Тобто, це означає, що для його оптимальності потрібно найменший показник спорідненості даних серед інших алгоритмів.

Deflate:

Оскільки цей алгоритм залежить від якості роботи Huffman та Lz77, у нашій реалізації він не показує хороших результатів, адже Lz77 працює найгірше з точки зору оптимальності. Проте, навіть так, Deflate працює краще ніж Lz77 за рахунок алгоритму Гаффмана.

5. У висновку, Lzw є найбільш оптимальним алгоритмом для випадків, коли файл є великим та показник спорідненості є великим(кількість символів(слів), що повторюється є великою). Найгірше ж показує себе LZ77, можливо проблема в тому, як саме ми домовляємось стискати код отриманий в результаті його роботи.

Що до швидкості роботи. То найкраща вона у LZW(кращий при розкодуванні) та Huffman(кращий у закодуванні) і є майже рівною. Найгірше веде себе Deflate, оскільки використовує одночасно 2 алгоритма і використовує не завжди ефективні методи, щоб перетворювати дані з одного алгоритму для іншого.