

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Кафедра програмування

Звіт

до індивідуального завдання №1
з курсу
«Обробка зображень та мультимедіа»

Виконав:
студент ПМІ-41
Юрій Лісовський

Львів 2019

1. Завантажив 24-бітне зображення *24bit_images_5.bmp* без стиснення (розмір — 151,374 байти):



2. Для стиснення виданого зображення методом *RLE* реалізував структуру *bmp* файлу та сам *RLE* алгоритм, оскільки, вбудований алгоритм не змінював розмір файла, а для стиснення методами *LZW* та *JPEG* — використав вбудовані бібліотеки *System.Drawing* та *System.Drawing.Imaging* у мові програмування *C#*. Перетворив запропоноване зображення у наступні формати:

- у формат *bmp* зі стисненням за допомогою методу *RLE* у файл *rle_encoded_8bit_images_5.bmp*; оскільки, алгоритм *RLE* застосовується до *bmp* зображень з глибиною кольору 8-bit, попередньо конвертував оригінальне зображення у *8bit_images_5.bmp*;
- у формат *tiff* зі стисненням за методом *LZW* у файл *lzw_encoded_images_5.tiff*;
- у формат *jpeg* зі стандартним кодуванням (Standard Encoding) у файл *jpeg_encoded_images_5.jpeg*.

Дані про час кодування та декодування зображень, їх читання та запису, а також про зміну розмірів подав у наступній таблиці:

Назва методу стиснення	Час кодування, μs	Час декодування, μs	Час читання, μs	Час запису, μs	Розмір, байтів	Різниця з оригіналом, байтів
Без стиснення (8-bit)	—	—	—	—	51 518	—
Без стиснення (24-bit)	—	—	—	—	151 374	—
RLE	10 331	2 144	105	150	84 014	-32 496
LZW	8 296	7 159	115	207	150 890	484
JPEG	1 451	1 559	131	87	22 465	128 909

Оригінал (мал. 1) та зображення, отримане за допомогою стиснення методом JPEG, (мал. 2):



мал. 1 (розмір — 151,374 байти)



мал. 2 (розмір — 22,465 байт)

RLE — це алгоритм стиснення без втрат, який працює з послідовностями, у яких один і той ж символ повторюється декілька разів підряд. При кодуванні, замість символу, який повторюється, пишеться кількість повторів символу і цей символ. Для застосування цього методу, необхідно, щоб глибина кольору *bmp* зображення була 8 біт (тобто на кожен піксель припадає по одному байту). При такій глибині кольору піксель — це індекс, який вказує на таблицю кольорів (палітру) зображення. Оскільки, видане зображення не містить великої кількості однакових пікселів, які розташовані підряд, розмір не

зменшився, а, навіть, збільшився на 32,496 байтів. Час виконання алгоритму — 10,331 мікросекунда.

Стиснення методом **LZW** полягає у тому, що алгоритм динамічно створює таблицю перетворення рядків: певним послідовностям символів ставить у відповідність групи бітів фіксованої довжини (зазвичай 12-бітні). Великою перевагою цього алгоритму є те, що таблиця створюється динамічно, як під час кодування, так і під час декодування, що дозволяє не зберігати її. Цей алгоритм, як і попередній, не призводить до втрат даних. Для ефективного кодування цим методом, потрібно, щоб зображення мало якнайбільше однакових послідовностей пікселів. Оскільки, видане зображення не містить багато однакових послідовностей, кодування методом *LZW* зменшило розмір зображення лише на 484 байти і час виконання становить 8,296 мікросекунди, що є меншим, ніж у попереднього методу.

Формат **JPEG** базується на алгоритмі дискретного косинусного перетворення. Мета цього алгоритму — видалити кольори, які не сприймає людське око, що дозволить суттєво зменшити розмір файла. Кроки алгоритму:

- перетворення *RGB* пікселя у формат *YCbCr*, де *Y* — компонента яскравості, *Cb* та *Cr* — канали синього та червоного кольору відповідно;
- до блоків 8x8 пікселів застосовується дискретне косинусне перетворення;
- квантування блоків 8x8;
- перетворення матриці у вектор;
- кодування вектора, одним із відомих алгоритмів, наприклад, методом *RLE*, оскільки він досить швидкий.

На даному зображені, алгоритм *JPEG* працював найшвидше — 1,451 мікросекунда і найбільше зменшив розмір файла. За допомогою цього вдалося зменшити розмір аж на 128,909 байтів, оскільки, наше зображення містить плавні переходи між тонами. Цей алгоритм призводить до безповоротних втрат даних, а тому якість зображення втрачається.

3. Зробив різницю зображення у форматі JPEG та початкового зображення спочатку для всіх кольорів. Для цього, використав вбудовані засоби для читання зображень і реалізував власний алгоритм попіксельної різниці. Інвертував зображення та збільшив його інтенсивність, щоб краще було видно втрати. Кількість втрачених пікселів становить 47,112:

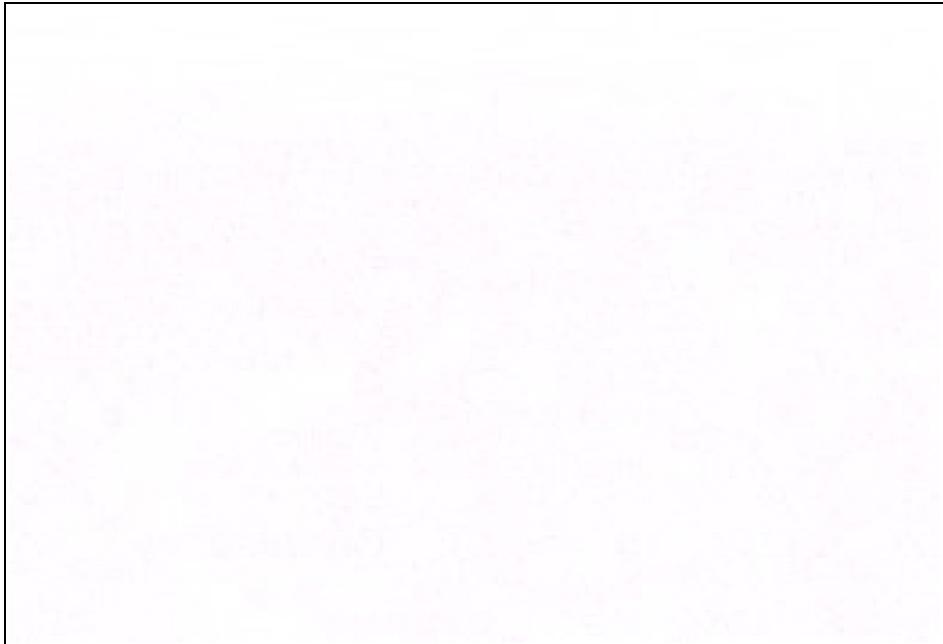


Також зробив різницю для кожного каналу окремо:

- для червоного, втрати — 41,436:



- для зеленого, втрати — 40,783:



- для синього, втрати — 41,931:



Отже, найбільші втрати даних для синього каналу пікселя, бо зображення найчастіше змінює значення цього кольору, а найменші — для зеленого. Оскільки, методи стиснення RLE та LZW — без втрат, то різниці з оригіналом не буде.