ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

INFORMATION TECHNOLOGY

DOI: 10.20998/2079-0023.2023.02.10
УДК 004.627

Я. М. КЛЯТЧЕНКО, кандидат технічних наук, доцент кафедри системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ, Україна; e-mail: y.kliatchenko@kpi.ua; ORCID: https://orcid.org/ 0000-0003-4236-4059
В. В. ГОЛУБ, студент кафедри системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ, Україна; e-mail: holub.volodymyr@lll.kpi.ua

ЕФЕКТИВНІСТЬ МОДИФІКАЦІЇ АЛГОРИТМУ УЩІЛЬНЕННЯ ДАНИХ БЕЗ ВТРАТ

Сучасний рівень розвитку інформаційних технологій обумовлює стрімке збільшення обсягів інформації, що зберігається, передається та оброблюється в комп'ютерних системах. Забезпечення повноцінного та ефективного використання цієї інформації вимагає застосування новітніх удосконалених алгоритмів для ущільнення та оптимізації її зберігання. Подальше зростання технічного рівня апаратних та програмних засобів тісно пов'язано з проблемами нестачі пам'яті для зберігання, що, також, актуалізує задачу ефективної компресії даних. Покращені алгоритми стиснення дозволяють ефективніше використовувати ресурси для зберігання та зменшують час пересилання даних через мережу. Щороку програмісти, вчені та науковці шукають способи удосконалення існуючих алгоритмів, а також винаходять нові, оскільки кожен алгоритм, навіть якщо він є простим, має свій потенціал удосконалення. Широке коло технологій, пов'язаних зі збором, обробкою, зберіганням і передачею інформації, значною мірою орієнтується на розвиток систем, в яких графічне подання інформації має перевагу над іншими типами представлення. Розвиток сучасних комп'ютерних систем і мереж вплинув на широке розповсюдження засобів, що оперують цифровими зображеннями. Зрозуміло, що зберігання і передача великої кількості зображень у первісному, необробленому вигляді є досить затратною по ресурсам задачею. В свою чергу сучасні мультимедійні системи набули значної популярності завдяки, насамперед, ефективним засобам компресії графічної інформації. Стиснення зображень є ключовим чинником для підвищення ефективності передачі даних та використання обчислювальних ресурсів. Робота присвячена дослідженню модифікації алгоритму стиснення даних The Quite OK Image Format, або QOI, що оптимізовано за швидкодією для стиснення графічної інформації. Тестування тих реалізацій алгоритму, які були запропоновані його автором, демонструє такі обнадійливі результати, що можуть зробити його конкурентоспроможним щодо вже відомого алгоритму PNG, забезпечуючи більшу швидкодію стиснення та націленість на роботу з архівами. В статті проведено порівняння результатів роботи двох запропонованих модифікацій алгоритму з оригінальною реалізацією та показано їхні переваги. Оцінено ефективність модифікацій та особливості їхнього застосування для різних випадків. Також проведено порівняння коефіцієнтів стиснення файлів, що були ущільнені оригінальним алгоритмом QOI з такими коефіцієнтами, які було отримано в результаті застосування модифікацій його початкової версії.

Ключові слова: стиснення даних, алгоритм QOI, модифікація алгоритму, ущільнення даних без втрат.

Вступ. Прогрес в галузі технічних засобів відтворення, передачі і зберігання інформації не встигає за потребами людства, і тому важливо ефективно використовувати наявні засоби зберігання і передачі інформації. Одним із чинників, що це забезпечує, може бути ефективна подача наявної інформації з використанням ущільнення (компресії) даних без втрат [1].

Значний потік інформації споживачі мультимедіа контенту отримують завдяки мережі Інтернет, переглядаючи вебсайти [2]. Для більшої наочності, доступності та розуміння змісту таких сайтів, часто ця інформація супроводжується графічними зображеннями, які підкреслюють чи висвітлюють основну її суть [3]. Зображення є таким типом інформації, що в недалекому минулому могло поступитися популярністю тільки текстовим даним, але завдяки можливостям сучасних мультимедійних засобів ця популярність вже давно перевершила символьне подання інформації.

Мультимедійні засоби дозволяють ефективно вирішити широкий спектр завдань − ми зустрічаємо їх коли починаємо роботу з комп'ютером, відкриваємо будь-яку програму, натрапляємо на піктограми, що теж є видом графіки. Протягом доби робиться незліченна кількість фотографій, якими діляться з колегами, друзями, які публікуються в соцмережах тощо [4]. Характеристики фотографій, якість передачі кольору, розміри зображень, їх розширення покращуються для відображення об'єктів ближче до природніх, щоб точно передати їх сутність [5]. З одного боку це

покращує сприйняття людиною картини, але з підвищенням якості фото зростає і вміст надлишкової інформації, яка є небажаною з точки зору зберігання, обробки та пересиланням зображення між комп'ютерними засобами. Надлишковість впливає на швидкість передачі даних по мережі та використання ресурсів мережі [6].

Для боротьби з надлишковістю або зменшення розміру того чи іншого типу файлів, використовують алгоритми стиснення зображення [7]. Завдяки стисненню даних можемо отримати файли меншого розміру, а отже і скоротити час передачі таких даних та звільнити додаткове місце на пристрої для зберігання. Це дозволяє оптимізувати простір для зберігання та навантаження на мережу.

Проблема стиснення зображень завжди буде актуальною та потребуватиме удосконалення. Популярні алгоритми стиснення, які набули широкого застосування сьогодні, були винайдені десятки років тому. Багато з них вже були неодноразово модифіковані та покращені для використання із обмеженим колом типів зображень. Попри це, пошук єдиного універсального алгоритму стиснення ще триває. Кожного року завдяки розробникам програмного забезпечення ми отримуємо нові алгоритми стиснення, які ще потрібно удосконалити та перевірити їх ефективність на практиці [8]. З часом, можливо, вони замінять поширені нині алгоритми. У даній статті пропонується розглянути та оцінити ефективність модифікацій відносно нового алгоритму стиснення зображень QOI, який створив Домінік Щаблевські [9].

Постановка задачі. Задача полягає в розробці удосконаленого алгоритму стиснення даних без втрат, що має покращити результати існуючого алгоритму [9], а також проведення порівняння оригінального алгоритму із модифікованими версіями та із популярними форматами.

Опис алгоритму QOI. Алгоритм QOI стискає зображення за один прохід по всіх пікселях зображення [10].

Основні складові :

- Заголовок – 14 байтів
- Тіло (будь яка кількість фрагментів даних), розмір фрагментів - кратне 8 бітам
- Заключний маркер

Структура заголовку:

- char magic; – "магічні" байти "qoif",
- uint32_t width; – ширина зображення в пікселях,
- uint32_t height; – висота зображення в пікселях,
- char magic; – "магічні" байти "qoif",
- int8_t channels; – 3 = RGB, або 4 = RGBA,
- uint8_t colorspace; – 0 = sRGB з лінійною альфою ; 1 = всі канали лінійні,
- channels та colorspace – тільки інформативні поля.

На початку роботи алгоритм визначає наявність чи відсутність альфа-каналу. В залежності від цього використовується один із варіантів кодування піксельних послідовностей (фрагменту) [11].

Якщо альфа-канал присутній, то фрагмент QOI_OP_RGBA кодується як показано на рис. 1.

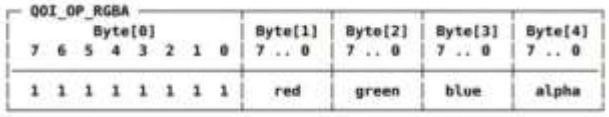

Рис. 1 – Фрагмент QOI_OP_RGBA

Без альфа каналу фрагмент QOI_OP_RGB кодується як на рис.2.

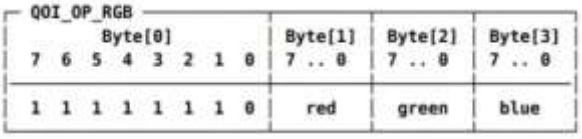

Рис. 2 – Фрагмент QOI_OP_RGB

Проходячи по пікселях, алгоритм може кодувати поточний піксель одним із чотирьох варіантів. Також, під час проходження по поточних пікселях зображення в проміжному масиві завжди зберігається 64 попередньо пройдених значень пікселів.

Для першого варіанту перевіряється значення кольору поточного пікселя. Якщо воно дорівнює значенню попереднього пікселя зображення, то при цьому збільшується значення змінної пройденої довжини зображення − змінну run збільшуємо на одиницю та переходимо до наступного пікселя. Таку послідовність дій виконуємо до тих пір, поки поточний піксель не буде відрізнятися від попереднього. Після знаходження чергової відмінності від попередніх пікселів, у фрагмент QOI_OP_RUN заноситься значення run – кількість повторів у бітовому представленні (рис.3).

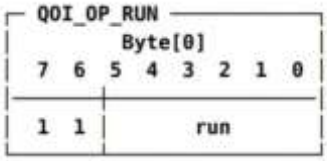

Рис. 3 – Фрагмент QOI_OP_RUN

Другий варіант. Перевіряється попередні 64 пікселя, і в разі знаходження співпадіння у фрагмент QOI_OP_INDEX заноситься індекс пікселя з буфера (рис.4).

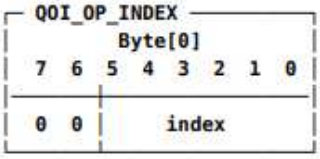

Рис.4 – Фрагмент QOI_OP_INDEX

Третій варіант. Оскільки у більшості випадків різниця між попереднім і поточним значенням пікселів

невелика, то фрагмент дорівнюватиме різниці по модулю 255 (8-бітний канал) між поточним пікселем і попереднім (рис. 5).

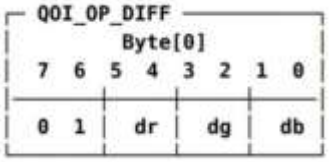

Рис. 5 – Фрагмент QOI_OP_DIFF

Також можливим варіантом фрагменту є QOI_OP_LUMA (див. рис. 6). Він представляє собою різницю між поточним пікселем і попереднім, в даному прикладі різниця по зеленому каналу використовується як базова різниця для інших кольорів. Цей метод, як було доведено, збільшує ступінь стиснення в тестах бенчмарку.

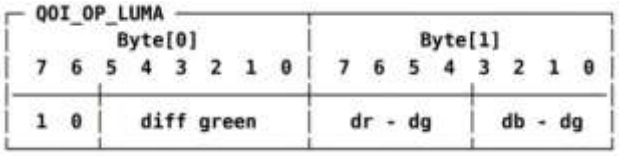

Рис. 6 – Фрагмент QOI_OP_LUMA

Четвертий варіант. У випадку, коли жоден із попередніх варіантів не підходить, запам'ятовується останнє значення без змін та переходимо до наступного пікселя і виконуємо одну із чотирьох можливий дій.

Для запам'ятовування всієї інформації використовується фрагмент QOI_OP_RGB (див. рис.2) або QOI_OP_RGBA (див. рис. 1).

Модифікації алгоритму QOI. Для початку модифікації алгоритму визначимо ключові елементи алгоритму (буфер, або те, як кодуються символи тощо) та проведемо аналіз можливості модифікації кожного елементу окремо чи разом із іншими.

У алгоритмі стиснення QOI можемо виділити такі етапи:

- Буфер із попередніми значеннями 64 пройдених пікселів.
- Чотири методи кодування пікселів:

1. Запам'ятовування, якщо поточний піксель дорівнює попередньому (QOI_OP_RUN),
2. Запам'ятати різницю значень колірної моделі по попереднім пікселям(QOI_OP_DIFF або QOI_OP_LUMA),
3. Віднайти індекс такого ж пікселя у буфері пройдених пікселів(QOI_OP_INDEX),
4. Записати оригінальні значення колірної моделі поточного пікселя (QOI_OP_RGB або QOI_OP_RGBA).

Кожен із методів кодування записується у фрагменти. Розмір одного фрагменту залежить від його типу (табл. 1).

Для того, щоб вірно розкодувати ущільнене зображення, всі фрагменти мають свої теги, або краще сказати, біти-ідентифікатори.

Ідентифікація фрагментів QOI_OP_RGBA та QOI_OP_RGB особлива, їх можна ідентифікувати, коли значення байту дорівнює 11111111 або 11111110 відповідно. Інші фрагменти ідентифікуються тільки по двом старшим бітам, оскільки цього достатньо тому, що всього лишається 4 види фрагментів (для множини варіацій значень двох бітів цілком достатньо).

Таблиця 1 – Таблиця розміру кожного із фрагментів, які містить алгоритм QOI.

Назва	Довжина
QOI_OP_RUN	1 байт
QOI_OP_DIFF	1 байт
QOI_OP_LUMA	2 байти
QOI_OP_INDEX	1 байт
QOI_OP_RGBA	5 байтів
QOI_OP_RGB	4 байти

Зауважимо, що внаслідок особливості ідентифікації фрагментів QOI_OP_RGBA та QOI_OP_RGB, для фрагменту QOI_OP_RUN існують обмеження. Ці обмеження пов'язані з тим, що для ідентифікації фрагменту QOI_OP_RUN два старших біта встановлюються в 1, а решта 6 бітів використовуються для запам'ятовування кількості повторів. І якщо у нас повторів буде 62 або 63, у бітовому представленні це виглядає як 1111110 або 111111 відповідно і до цих значень потрібно ще додати старші біти, які дорівнюють 11. У результаті матимемо двійкові значення для QOI_OP_RGB і QOI_OP_RGBA. Під час модифікації потрібно враховувати таку особливість, щоб уникнути помилок при декодуванні.

Модифікація фрагменту QOI_OP_INDEX. Візьмемо до уваги QOI_OP_INDEX, довжина якого вісім бітів, два з яких відносять до заголовку і шість бітів на запам'ятовування (64 різних значень). Хеш-функція для визначення індексу позиції [12] виглядає як:

$$index = (r*3 + g*5 + b*7 + a*11)\,\%\,64 .$$

Після ознайомлення з ключовими етапами алгоритму, розглянемо можливість зміни довжини буферу пройдених пікселів. Розглянемо одну із ситуацій, коли, наприклад, у нас після проходження є 80 відмінних між собою пікселів. Припустимо, що ми знаходимо піксель, який був 14-м пікселем із пройдених 80 пікселів. Дана ситуація не підходить до жодного із попередніх варіантів кодування, окрім варіанту запису значень колірної моделі без змін у фрагменти QOI_OP_RGBA або QOI_OP_RGBA. У такій ситуації було б зручно мати більший буфер пройдених пікселів, що дає виграш від трьох до чотирьох байтів. Враховуючи те, що сучасні зображення мають доволі великі розміри – наприклад, зображення розміром 2 560 × 1 280 пікселів містить в собі 3 276 800 пікселів. З цього випливає, що вищезгадана ситуація цілком можлива і при більше ніж трьох мільйонах пікселів може зустрітись часто. У такій ситуації три або чотири байти різниці можуть зіграти вирішальну роль, вплинувши позитивно на вихідний розмір зображення.

Для того, щоб збільшити розмір буфера, потрібно збільшити розмір фрагменту QOI_OP_INDEX (див. рис. 4) з 1 байту до 2 байт. При збільшенні розміру фрагменту потрібно врахувати те, що треба зменшити можливий виграш у розмірі з трьох–чотирьох до двох–трьох байтів. Кількість виграшних байтів залежить від типу зображення, та від того чи містить він альфаканал чи ні. Для цього випадку максимально можливе значення індексу дорівнює 16384 (нагадуємо, що для попереднього варіанту максимальна кількість символів не перевищувала 64).

Після зміни довжини буфера, потрібно змінити хеш-функцію, яка записує індекс і в залежності від значення R G B видавала лише одне відповідне значення індексу буфера. Щоб переробити функцію так, щоб використовувались цілочисельні коефіцієнти потрібно знайти такі значення коефіцієнтів *r*, *g*, *b* та *a*, щоб вони були цілими числами і за можливості були найменшими спільними множниками для коефіцієнтів 3, 5, 7 і 11. Найменший спільний множник для цих чисел дорівнює 1 155. Отже, будемо використовувати коефіцієнти, рівні 1 155, 2 310, 8 085 і 12 615 відповідно. Хеш-функція для визначення індексу матиме вигляд:

$$index = (r*1155 + g*2310 + b*8085 + a*12615)\%16384.$$

Модифікація фрагменту QOI_OP_RUN. Розглянемо інший варіант – фрагмент QOI_OP_RUN. Він фіксує кількість повторів попереднього пікселя. Отже, якщо у нас виникне ситуація, коли буде знайдено більше, ніж 62 однакових пікселя підряд, то для наступного пікселя потрібно буде використати один із 3 інших варіантів. Наступний піксель теж буде закодовано за допомогою фрагменту QOI_OP_RUN. Внаслідок чого отримаємо 3 байти інформації. І для кожних наступних 62 однакових пікселів цей варіант буде додавати 2 байти.

Далі знову збільшимо довжину фрагмента на 8 біт. Так можна закодувати до 15 872 однакових пікселів, враховуючи обмеження: що у 16-бітовому числі в старших бітах не можна використовувати комбінації, що дорівнюють значенням 255 та 254 відповідно. Застосовуючи такий прийом можна виграти декілька десятків, а, можливо, сотень байт.

Оскільки в більшості зображень важко зустріти ситуацію, коли однакових пікселів буде більше 62 поспіль, то зручним варіантом було б те, щоб фрагмент QOI_OP_RUN був адаптивним. Адаптивність має полягати в тому, що у випадку, якщо кількість повторюваних символів більше ніж 62, то потрібно використовувати збільшений фрагмент QOI_RUN_OP. Якщо ця кількість менша, то використовується звичайний фрагмент розміром в один байт. Таку реалізацію не можна відкидати, але в такому разі потрібно вводити новий тип фрагменту, що спричинить появу нового ідентифікатора. У поточному стані всі чотири фрагменти кодуються набором із 2 біт, а якщо буде введено п’ятий фрагмент, то потрібно додавати ще один біт до варіацій і така зміна вплине на весь алгоритм. Таким чином це недоцільно для алгоритму.

Інші фрагменти змінити складно тому, що в них міститься інформація про колір і вона залежить від поточного або попереднього значення пікселя.

На рис. 7 показано скріншот результату порівняння стиснення фалів із розширенням *qoi* та *png* [12] в *zip*-архіві.

Результати стиснення одноколірного зображення представлені в табл. 2.

Ім'я	Тип	Розмір стиснутого файлу	Розмір	Коефіцієнт стискання
amazon.com	Файл PNG	6 166 КБ	6 232 КБ	2%
amazon.qoi	Файл QOI	3 886 КБ	5 019 КБ	23%
amazon_run.qoi	Файл QOI	3 938 КБ	5 216 КБ	25%
amazonindex.qoi	Файл QOI	4 195 КБ	5 303 КБ	21%
apple.com	Файл PNG	2 246 КБ	2 306 КБ	3%
apple.qoi	Файл QOI	1 518 КБ	2 104 КБ	28%
apple_run.qoi	Файл QOI	1 541 КБ	2 189 КБ	30%
appleindex.qoi	Файл QOI	1 630 КБ	2 287 КБ	29%
cnn.com	Файл PNG	2 590 КБ	2 685 КБ	4%
cnn.qoi	Файл QOI	1 528 КБ	2 302 КБ	34%
cnn_run.qoi	Файл QOI	1 547 КБ	2 343 КБ	34%
cnnindex.qoi	Файл QOI	1 536 КБ	2 330 КБ	35%
creativecommons.org	Файл PNG	6 102 КБ	6 146 КБ	1%
creativecommons.qoi	Файл QOI	3 943 КБ	4 908 КБ	20%
creativecommons_run.qoi	Файл QOI	4 003 КБ	5 120 КБ	22%
creativecommonsindex.qoi	Файл QOI	4 176 КБ	5 275 КБ	21%

Рис. 7 – Порівняння стиснення фалів із розширенням *qoi* та *png* у *zip*-архів

Таблиця 2 – Результат стиснення одноколірного зображення

Модифікація алгоритму	Зображення	Розмір, байт	Стиснутий розмір, байти	Відсоток стиснення	Час стиснення, секунди	Час декодування, секунди
ORIGINAL	black_1920_1080	8 294 457	367 46	99,56	5,78	4,35
RUN	black_1920_1080	8 294 457	383	100,00	5,82	4,48
INDEX	black_1920_1080	8 294 457	367 46	99,56	5,80	4,61

Таблиця 3 – Результати стиснення зображення із розміром 1 313 × 6 097

Модифікація алгоритму	Зображення	Розмір, байт	Стиснутий розмір, байти	Відсоток стиснення	Час стиснення, секунди	Час декодування, секунди
ORIGINAL	amazon	32 021 501	5 169 981	83,85	32,91	22,86
RUN	amazon	32 021 501	5 816 227	81,84	31,82	23,09
INDEX	amazon	32 021 501	5 816 227	81,84	32,10	24,39

Результати стиснення зображення із розміром 1 313 × 6 097 наведені в табл. 3.

Висновки. Проаналізувавши отримані результати, можна зробити наступні висновки :

- Модифіковані алгоритми QOI не завжди дають покращення показників стиснення.
- Якщо порівнювати оригінальний алгоритм QOI та дві його модифікації, то можна зробити висновок, що модифікація QOI зі збільшенням буферу для запам'ятовування пройдених пікселів значно програє модифікації QOI, в якій ми розширювали фрагмент QOI_OP_RUN.
- Алгоритм із модифікацією фрагменту QOI_OP_RUN можна використовувати замість оригінального алгоритму. Оскільки різниця в часі кодування, декодування, та вихідним розміром стиснутого зображення практично однакова, в випадках коли зображення має більше, ніж 62 однакових пікселя підряд, він виграє порівняно із оригінальним алгоритмом.
- У випадку порівняння алгоритмів PNG та QOI при стисненні зображення, що представляє собою скріншот екрану, то перший програє всім модифікаціям QOI та оригінальному алгоритму QOI, на що вказують наведені вище результати тестів.

Список використаної літератури

1. Fu S., Wang L., Cheng Y., Chen, G. Intelligent compression for synchrotron radiation source image. *25th International Conference on Computing in High Energy and Nuclear Physics*. 2021. Вип. 251. С. 1–7. DOI: https://doi.org/10.1051/epjconf/202125103073.
2. Шкіль Л. *63% людей зараз онлайн. Великий звіт Digital 2022 про користувачів інтернету*. URL: https://ain.ua/2022/04/30/zvit-\ digital-2022/ (дата звернення: 13.11.2023).
3. *Use images and media to enhance understanding*. URL: https://accessibility.huit.harvard.edu/use-images-and-media-\ enhance-understanding (дата звернення: 12.11.2023).
4. Ріхтер Ф. *Smartphones Cause Photography Boo*. URL: https://www.statista.com/chart/10913/number-of-photos-taken-\ worldwide (дата звернення: 13.11.2023).
5. *Характеристики зображення та засобів його відтворення*. URL: https://library.vpuhlukhiv.com.ua/subjects:basic:informatika:\ graph:k%D0%B0r%D0%B0kterystyky_zobr%D0%B0zhennia_t%\ D0%B0_z%D0%B0sobiv_ioho_vidtvorennia (Дата звернення: 13.11.2023).
6. *What are different types of redundancies in digital image? Explain in detail*. URL: https://www.ques10.com/p/7293/what-are-\ different-types-of-redundancies-in-digi-1 (дата звернення: 13.11.2023).
7. *Compression algorithms*. URL: https://www.prepressure.com/library\ /compression-algorithm (дата звернення: 02.11.2023).
8. *WebP*. URL: https://uk.wikipedia.org/wiki/WebP (дата звернення: 13.11.2023).
9. Szablewski D. *The Quite OK Image Format for Fast, Lossless Compression*. URL: https://qoiformat.org/ (дата звернення: 1.11.2023).
10. Szablewski D. *QOI - The Quite OK Image Format*. URL: https://qoiformat.org/benchmark (дата звернення: 7.11.2023).
11. Szablewski D. *QOI-Specification*. URL: https://qoiformat.org/qoi-specification.pdf (дата звернення: 25.10.2023).
12. *Portable Network Graphics (PNG) Specification (Second Edition)*. URL: http://www.w3.org/TR/PNG/ (дата звернення 8.11.2023).

References (transliterated)

1. Fu S., Wang L., Cheng Y., Chen G. Intelligent compression for synchrotron radiation source image. *25th International Conference on Computing in High Energy and Nuclear Physics*. 2021. vol. 251 pp. 1–7. DOI : https://doi.org/10.1051/epjconf/202125103073.
2. Skil L. *63% liudei zaraz onlain. Velykyi zvit Digital 2022 pro korystuvachiv internetu*. Available at: https://ain.ua/2022/04/30/zvit-digital-2022/ (accessed 13.11.2023).
3. *Use images and media to enhance understanding*. Avaible at: https://accessibility.huit.harvard.edu/use-images-and-media-\ enhance-understanding (accessed: 12.11.2023).
4. Richter F. *Smartphones Cause Photography*. Available at: https://www.statista.com/chart/10913/number-of-\ photos-taken-worldwide (accessed: 13.11.2023).
5. *Kharakterystyky zobrazhennia ta zasobiv yoho vidtvorennia*. Available at: https://library.vpuhlukhiv.com.ua/subjects:basic:informatika:graph:k%D0%B0r%D0%B0kterystyky_zobr%D0%B0zhen\nia_t%D0%B0_z%D0%B0sobiv_ioho_vidtvorennia (accessed: 13.11.2023).
6. *What are different types of redundancies in digital image? Explain in detail*. Available at: https://www.ques10.com/p/7293/what-are-\ different-types-of-edundancies-in-digi-1 (accessed: 13.11.2023).
7. *Compression algorithms*. Available at: https://www.prepressure.com/library/compression-algorithm accessed: 02.11.2023).
8. *WebP*. Available at: https://uk.wikipedia.org/wiki/WebP (accessed: 13.11.2023).

9. Szablewski D. *The Quite OK Image Format for Fast, Lossless Compression*. Available at: https://qoiformat.org (accessed: 1.11.2023).
10. Szablewski D. *QOI - The Quite OK Image Format*. Available at: https://qoiformat.org/benchmark (accessed: 7.11.2023).
11. Szablewski D. *QOI-Specification.* Available at: https://qoiformat.org/qoi-specification.pdf (accessed: 25.10.2023).
12. *Portable Network Graphics (PNG) Specification (Second Edition)*. Available at: http://www.w3.org/TR/PNG/ (accessed: 8.11.2023).

Надійшла (received) 17.11.2023

UDC 004.627

Y. M. KLYATCHENKO, PhD, associate professor of Department of System Programming and Specialized Computer Systems of Igor Sikorsky Kyiv Polytechnic Institute; Kyiv, Ukraine, e-mail: y.kliatchenko@kpi.ua; ORCID: https://orcid.org/ 0000-0003-4236-4059
V. V. HOLUB, student of Department of System Programming and Specialized Computer Systems of Igor Sikorsky Kyiv Polytechnic Institute; Kyiv, Ukraine, e-mail: holub.volodymyr@lll.kpi.ua

EFFICIENCY OF LOSSLESS DATA COMPRESSION ALGORITHM MODIFICATION

The current level of development of information technologies causes a rapid increase in the amount of information stored, transmitted and processed in computer systems. Ensuring the full and effective use of this information requires the use of the latest improved algorithms for compaction and optimization of its storage. The further growth of the technical level of hardware and software is closely related to the problems of lack of memory for storage, which also actualizes the task of effective data compression. Improved compression algorithms allow more efficient use of storage resources and reduce data transfer time over the network. Every year, programmers, scientists, and researchers look for ways to improve existing algorithms, as well as invent new ones, because every algorithm, even if it is simple, has its potential for improvement. A wide range of technologies related to the collection, processing, storage and transmission of information are largely oriented towards the development of systems in which graphical presentation of information has an advantage over other types of presentation. The development of modern computer systems and networks has influenced the wide distribution of tools operating with digital images. It is clear that storing and transferring a large number of images in their original, unprocessed form is a rather resource-intensive task. In turn, modern multimedia systems have gained considerable popularity thanks, first of all, to effective means of compressing graphic information. Image compression is a key factor in improving the efficiency of data transfer and the use of computing resources. The work is devoted to the study of the modification of the data compression algorithm The Quite OK Image Format, or QOI, which is optimized for speed for the compression of graphic information. Testing of those implementations of the algorithm, which were proposed by its author, shows such encouraging results that it can make it competitive with the already known PNG algorithm, providing a higher compression speed and targeting work with archives. The article compares the results of the two proposed modifications of the algorithm with the original implementation and shows their advantages. The effectiveness of the modifications and the features of their application for various cases were evaluated. A comparison of file compression coefficients, which were compressed by the original QOI algorithm, with such coefficients, which were obtained as a result of the application of modifications of its initial version, was also carried out.

Keywords: data compression, QOI algorithm, algorithm modification, lossless data compression.

Повні імена авторів / Author's full names

Автор 1 / Author 1: Клятченко Ярослав Михайлович, Klyatchenko Yaroslav Mykhailovich
Автор 2 / Author 2: Голуб Володимир Володимирович, Holub Volodymyr Volodymyrovych