УДК 681.327

ПОРЄВ В. М.

СТОСОВНО МОДИФІКАЦІЇ МЕТОДА КОДУВАННЯ ДОВЖИН ПОВТОРЕНЬ

В статті проаналізовані модифікації класичного метода кодування довжин повторень для компресії растрових даних. У результаті аналізу відзначається необхідність побудування різноманітних кодових послідовностей, які найбільш адекватні рядкам конкретних растрових зображень. Зроблено пропозиції щодо організації прямого доступу для прискорення декодування фрагментів растрових даних великих обсягів.

The article analyzes modifications of the classical method of run length encoding for compression of raster data. As a result of the analysis, the need to construct the corresponding code sequences, which are most adequate to the lines of specific raster images. Proposals for organizing direct access are made to accelerate the decoding of large-scale raster data fragments.

Ключові слова: Растрові зображення, кодова послідовність, методи компресії.

1. Вступ

Растрові дані, які складаються з матриці елементів (пікселів), наприклад, зображення, є популярним видом представлення даних у багатьох цифрових технологіях. Від початку використання растрів і на сьогоднішній день актуальним є пошук та реалізація ефективних алгоритмів компресії таких даних задля зменшення обсягів пам'яті для їхнього зберігання, пересилання по мережам та (або) прискорення кодування-декодування. У даній статті увагу зосереджено на методі кодування довжин повторень та його модифікаціях.

2. Аналіз досліджень і публікацій

Для компресії растрів, починаючи з середини минулого сторіччя розроблялися різноманітні методи і алгоритми, більшість з яких описані у книзі [1]. Як виявляється, методи та алгоритми компресії растрів можна розділити, по-перше, на компрессію із втратами і компрессію без втрат (*loseless compression*). По-друге, методи компресії виявляються різними для різних кількостей кольорів у растрах. Зазвичай методи компресії із втратами використовують для растрів типу фотографій, а методи компресії без втрат є ефективними для растрів з невеличкою кількістю кольорів – зазвичай до 256.

Одним з найпростіших методів компресії без втрат є метод RLE (*Run Length Encoding*) – кодування довжин повторень. Цей метод привертає увагу розробників систем завдяки високій швидкодії декодування. Основна ідея метода RLE – кожне числове значення елементів вздовж рядка растру замінюється на пару чисел: (кількість повторів, значення). Наприклад, послідовність з 16 числових значень: 15 15 15 15 15 15 237 4 4 4 4 4 4 4 4 4 кодується 3-ма парами: (6,15) (1,237) (9,4). Якщо первісні та кодовані значення кодуються однаковою кількістю бітів, наприклад 8, то при кодуванні 16 чисел шістьма ступінь компресії дорівнює 16/6 = 2.66. Алгоритми RLE ефективно працюють з растрами, у яких в рядках є багато повторюваних значень – чим довше ланцюжки однакових пікселів, тим вища компресія. Позначимо довжину рядка растру як W. Якщо у такому рядку усі піксели однакові, то цей рядок кодується двома числами – парою (W, значення). У цьому випадку ступінь компресії максимальна і оцінюється як W/2 (потрібно ще враховувати, яка кількість бітів буде використовуватися для запису коду).

Як виконувати кодування, у випадку, коли у рядку немає ланцюжків повторюваних пікселів (наприклад у цифровій фотографії)? Буквальне використання базового метода RLE призведе до запису W пар виду (1,значення$_i$), тобто для кожного *i*-го неповторюваного піксела записується кількість 1 і кодоване значення кольору. Це найгірший випадок – розмір коду збільшується вдвічі. Загалом, це є основною причиною того, що растри типу фотографій не можуть бути суттєво компресовані не тільки методом RLE, а й іншими відомими методами компресії без втрат, наприклад, словниковими LZ-подібними [1].

Основна сфера використання методів і алгоритмів класу *loseless compression* і,

зокрема, RLE – компресія растрів зображень, у яких багато фрагментів суцільного кольору, наприклад, схем, діаграм, карт. Для покращення властивостей кодування, і в першу чергу, для підвищення компресії, потрібна модифікація базового метода RLE.

Відома версія RLE для файлів формату PCX [1]. Байт коду складається з двох бітів префіксу (pp) і решти бітів (xxxxxx). При декодуванні спочатку аналізуються біти префіксу. Якщо pp = 11, то наступні біти (xxxxxx) репрезентують довжину повторень. Інші значення бітів префікса: pp = 00, 01, 10 означають, що увесь цей байт репрезентує числове значення кольору одиночного пікселя растру (рис. 1)

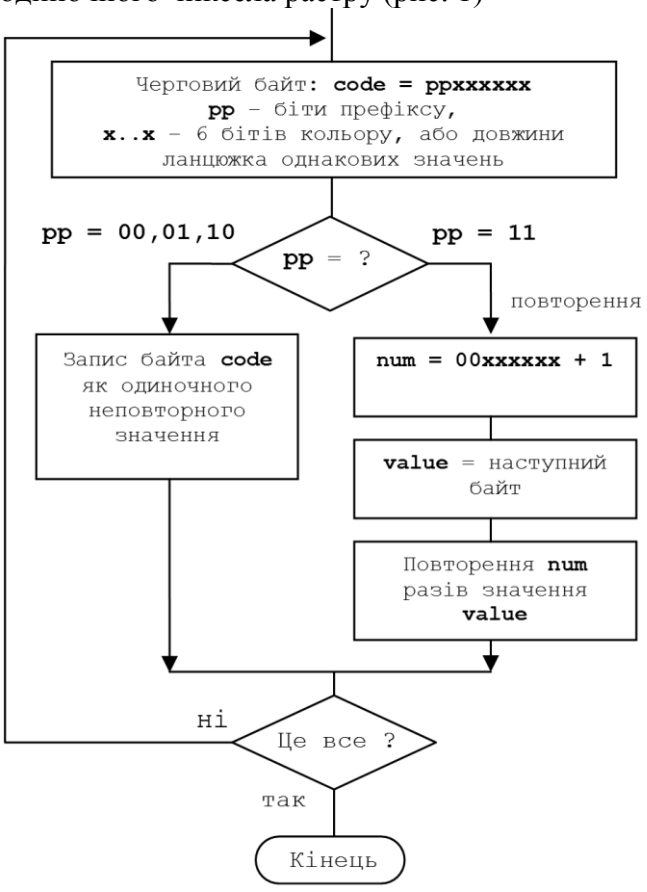

Рис. 1. Алгоритм декодування RLE PCX

Максимальна ступінь компресії визначається кількістю бітів довжини. Шістьма бітами представляється довжина ланцюжка до 64 однакових значень, які кодуються парою байтів (11111111, значення), що означає компресію у 32 рази.

Чому для префіксу використовуються саме два біти, а не один (тоді для кодування довжини було б сім бітів, що давало б кращу компресію ланцюжків – у 64 рази)? Наявність двох бітів префіксу забезпечує для одиночних пікселів кодування 75% можливих значень індексів кольорів – у діапазоні від 00000000 до 10111111. Ідеальним для PCX є випадок, коли 256-кольорова палітра впорядкована так, що тільки перші 192 індексів репрезентують кольори одиночних пікселів. Одиночні пікселі з індексами кольорів 11xxxxxx повинні кодуватися вже парою байтів (11000000, 11xxxxxx), тобто, розмір коду може зрости вдвічі. Це суттєвий недолік PCX.

Інша відома версія RLE була розроблена для файлів формату TGA. Байт коду складається з одного біту префіксу (p) та семи бітів довжини (nnnnnnn). Значення біту p=0 означає, що далі буде послідовність будь-яких значень (літерал). Довжина літералу – від 1 до 128. Цим передбачено кодування неповторних значень у рядку, а також запобігання суттєвого зростання коду. Значення біту p=1 означає ланцюжок однакових значень з довжиною до 128. Алгоритм декодування на рис. 2.

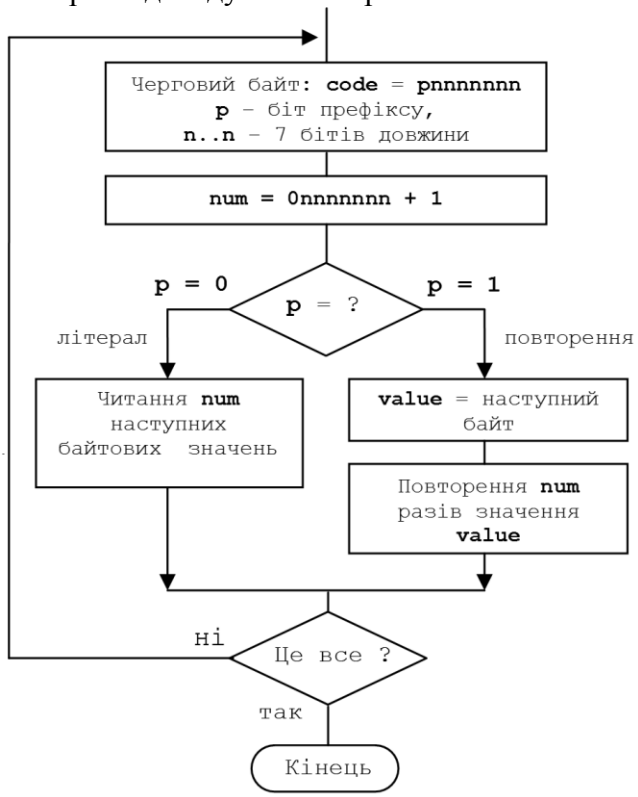

Рис. 2. Алгоритм декодування RLE TGA

Таким чином, максимальна компресія RLE TGA становить 64 рази. Збільшення коду рядків неповторних значень максимум до 129/128 (а не вдвічі, як у RLE PCX) [1].

Може скластися враження, що версія RLE TGA краще версії RLE PCX. Це не зовсім так, оскільки можна навести приклади растрів, для яких розмір коду PCX менший, аніж TGA. Наприклад, послідовність з 12 значень

0 15 15 15 1 15 15 15 2 15 15 15

алгоритм RLE PCX запише у 9 байтів коду:

0 195 15 1 195 15 2 195 15
а алгоритм RLE TGA запише у 12 байтів:
0 0 130 15 0 1 130 15 0 2 130 15

Зауваження. Якби різні рядки растру можна було б компресувати різними алгоритмами сімейства RLE, то можливо було б досягти більшої компресії тільки за рахунок поєднання вже відомих алгоритмів.

У роботі [2] запропонований комплексний метод компресії без втрат, який адаптовано для стискування томографічних даних і дозволяє зменшити обсяг таких даних понад 5 разів. Особливістю томографічних зображень є суцільний чорний фон за межами контуру об'єкту (рис. 3)

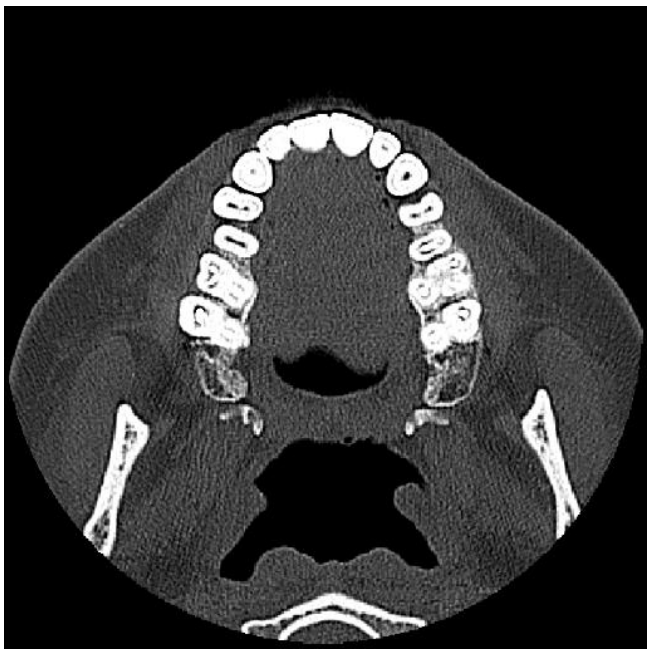

Рис. 3. Приклад томографічного зображення. Джерело – [2]

Запроваджено модифікацію класичного метода RLE, згідно якій кожний ланцюжок чорного (0-го) кольору кодується парою байтів (0, довжина), а пікселі кольорів від 1 до 255 записуються власними значеннями. Такий спосіб дає якусь компресію лише у випадку, коли ланцюжки чорного кольору довгі. Необхідно зазначити, що основний ефект компресії без втрат для томографічних знімків досягається за рахунок поєднання RLE із попереднім дельта-кодуванням [2].

У роботі [3] запропоновано модифікацію класичного алгоритму RLE задля збільшення максимальної довжини ланцюжків однакових значень. У коді спочатку йде службовий байт1, у якого старший біт=0 означає, що йде одиночний піксел, а якщо старший біт=1, то буде ланцюжок повторюваних значень. Значення наступного біта службового байту1 є індикатором короткого або довгого ланцюжка повторень. Якщо другий біт службового байту1 дорівнює 0, то у молодші шість бітів записано довжину короткого ланцюжка (не більше 64). А якщо у службовому байті1 другий біт = 1, то далі йде службовий байт2, у який записується довжина (не більше 256). У такий спосіб можна закодувати довжини до $320 = 2^6 + 2^8$. Необхідно відзначити, що з тексту джерела [3] не зовсім зрозуміло, як використовуються решта бітів службового байту1 – як здається, код у деяких випадках може бути збитковим.

У публікації [4] повідомляється про розробку модифікації метода RLE, яка враховує вірогідності повтору елементів у рядках. В ході виконання алгоритму заповнюється таблиця, у якій кожному значенню довжини ланцюжка відповідає кількість таких ланцюжків. Крім того, вводиться спеціальний символ, який приймає значення 1, якщо піксел повторюється тричі і 0 – якщо третього повторення немає [4].

Аналіз публікацій щодо різноманітних модифікацій метода RLE дозволяє сказати, що у результаті досліджень розробляється деяке спеціальне бітове представлення кодованих елементів, яке оптимізується для

- зменшення впливу вкраплень одиночних (неповторюваних) пікселів;
- збільшення допустимої довжини ланцюжків повторюваних пікселів;
- врахування статистики використання пікселів різних кольорів;
- врахування інших факторів

Трохи історії. Свого часу Бліновою Т.О. та Порєвим В.М. було запропоновано, розроблено і реалізовано, а пізніше, у 2007 році загалом оприлюднено модифікацію метода RLE, яку названо RLE-БП [5]. Така модифікація дозволяє досягти рівня компресії значно більшого, порівняно з класичним RLE та іншими відомими (у тому числі тими, які пропонуються значно пізніше) модифікаціями цього методу. Метод RLE-БП може конкурувати зі словниковими LZ-подібними методами у багатьох застосуваннях, наприклад, діловій графіці, для зберігання електронних растрових карт у геоінформаційних системах тощо [6, 7].

Оглянемо основні моменти для RLE-БП. Введемо позначення.

M – кількість бітів двійкового коду, яка необхідна для кодування кольору, тобто, *M* = $\log_2$(КількістьКольорів). Врахуємо те, що у деякому зображенні можуть бути присутні не усі кольори з 256-колірної палітри – тоді *M* може бути менше 8.

*C*1 – кількість бітів, яка потрібна для кодування індексів множини **головних кольорів**. Якщо виконати сортування усіх кольорів палітри відповідно частоті їхнього використання у даному зображенні, то *M*-бітний код **00..00** означає номер найбільш популярного кольору, код **00..01** – для наступного по популярності кольору, і так далі. Будемо вважати, що кодер і декодер обробляють окремо множину з 2^{C1} головних кольорів – тоді *C*1 < *M*. Якщо з *M* бітів виділити *C*1 молодших бітів, то це і буде індекс головного кольору. Наприклад, якщо *C*1=2, то це означає чотири головних кольори, індекси яких кодуються як **00**, **01**, **10** и **11** в порядку зменшення частоти використання.

Спосіб кодування 1.

Використовуються три різновиди кодових послідовностей.

0c...c (усього *M* бітів) − для одиночних пікселів, у яких старший біт кольору є 0. Одиночні піксели кольорів **1c...c** будуть кодуватися вже як ланцюжки довжини 1. Для кодування ланцюжків пікселів будемо використовувати наступні послідовності:

10n...nc...c – спочатку префікс (біти **10**), потім *N*1 бітів коду довжини ланцюжка (біти **n**). Завершують послідовність *C*1 бітів **c**, які означають індекс головного кольору.

11n...nc...c – спочатку префікс (**11**), потім *N*2 бітів довжини ланцюжка, і потім *M* бітів кольору.

Спосіб кодування 2.

Використовуються кодові послідовності двох типів.

0c...c – для одиночних пікселов будь-якого кольору. Спочатку префікс (**0**), а потім *M* бітів кольору.

1c...cnn...n – для ланцюжків пікселів головного кольору. Спочатку префікс (**1**), потім *C*1 бітів індексу головного кольору (тобто, так можна закодувати тільки 2^{C1} найбільш популярних кольорів). Завершують кодову послідовність біти довжини ланцюжка (**nn...n**). Кожному *i*-му головному кольору відповідає своя кількість бітів *n* – *Ni*. Надамо приклад кодування ланцюжків головних кольорів для *C*1 = 2:

100nnn – *N*0 = 3 для найбільш популярного кольору (індекс **00**);

101nnnnnnn – *N*1 = 7 для головного кольору з індексом **01**;

110nnnn – *N*2 = 4 для головного кольору з індексом **10**;

111nnnnn – *N*3 = 5 для головного кольору з індексом **11**.

Характеристики способу кодування 2 визначаються множиною параметрів *C*1 и *Ni* (*i* від 0 до 2^{C1}-1).

На відміну від попереднього, згідно способу 2 кодування одиночних пікселів виконується не більш ніж (*M*+1)-бітовим кодом незалежно від кольору. Так враховується вірогідність великої кількості одиночних пікселів другорядних кольорів.

Також можна відмітити, що спосіб кодування 2 враховує не тільки популярність кольорів, але і те, що один з головних кольорів може бути представлений переважно у коротких ланцюжка пікселів, а інший – у довгих. На відміну традиційного RLE-кодування ланцюжків, коли спочатку записується довжина ланцюжка, а потім колір, в кодах способу 2 спочатку колір, а потім довжина ланцюжка. Це потрібно для того, щоб декодер був здатен правильно розпізнати код довжини ланцюжка.

У способі 2 кількість початкових бітів (**1c...c**) однакова для кодів ланцюжків усіх головних кольорів. У наступному способі запропоновано кольори кодувати префіксами різної кількості бітів, причому ця кількість зворотна популярності кольору. Можна сказати, що це поєднання RLE і метода Хаффмана [1].

Спосіб кодування 3.

0c...c − для одиночних пікселів довільного кольору. Спочатку префікс (**0**), потім *M* бітів кольору.

1p...pnn...n – для ланцюжків пікселів. Спочатку префікс (**1p...p**), потім біти довжини ланцюжка (**nn...n**). Як і в способі 2, кожному *i*-му головному кольору відповідає індивідуальна кількість (*Ni*) бітів **n**.

Кодування префіксними кодами згідно способу 3 розглянемо на такому прикладі

0c...c – одиночні піксели;

10nn...n – ланцюжок кольору 0 (*N*0 бітів **n**);

`110nn...n` – ланцюжок кольору 1 (*N*1 бітів **`n`**);

`1110nn...n` – ланцюжок кольору 2 (*N*2 бітів **`n`**);

`1111nn...n` – ланцюжок кольору 3 (*N*3 бітів **`n`**).

В даному прикладі довжина коду ланцюжка кольору 0 менше, аніж для способу 2, проте, для кольорів 2 та 3 коди ланцюжків довше.

Параметрами способу кодування 3 є кількість головних кольорів і множина значень *Ni*.

У наступному способі передбачено розвинені можливості кодування ланцюжків повторень для широкого діапазону довжин.

Спосіб кодування 4.

`0c...c` – для одиночних пікселів довільного кольору. Спочатку префікс (**`0`**), потім *M* бітів кольору.

`1c...cxx...x` – для ланцюжків пікселів головного кольору. Спочатку префікс (**`1`**), потім *C*1 бітів для індексу головного кольору. Завершують кодову послідовність біти довжини ланцюжка (**`xx...x`**). Кожному значенню індексу головного кольору (біти **`c...c`**) відповідає власний формат коду довжини ланцюжка, який обирається з трьох наступних форматів (*а, б, в*):

формат *а*:

`1c...cnn...n` – *N*1 бітів **`n`** довжини ланцюжка (*N*1 від 0 до 15);

формат *б*:

`1c...c0nn...n` – *N*1 бітів **`n`** (*N*1 від 0 до 15),

`1c...c1nn...n` – *N*2 бітів **`n`** (*N*2 від *N*1+1 до *N*1+16);

формат *в*:

`1c...c0nn...n` – *N*1 бітів **`n`** (*N*1 від 0 до 7);

`1c...c10nn...n` – *N*2 бітів **`n`** (*N*2 від *N*1+1 до *N*1+8);

`1c...c11nn...n` – *N*3 бітів **`n`** (*N*3 від *N*2+1 до *N*2+8).

Параметрами коду для способу 4 є значення *C*1, *N*1, *N*2, *N*3 і тип формату коду довжини.

Теоретично, різновиди форматів коду у вигляді **`1c...cy..ynn...n`** можна розвивати і нарощувати шляхом збільшення кількості бітів **`y..y`**, але на практиці трьох форматів (*а, б, в*) для способа 4 цілком достатньо для забезпечення ефективного кодування усіх можливих ланцюжків з довжинами до десятків тисяч пікселів.

Наведені вище чотири способи кодування були опубліковані у [5, 6]. Необхідно відзначити, що кожний спосіб кодування не є універсальним – максимальну компресію для одних даних дає один спосіб, а для інших даних ефект дає інший. Авторами було запропоновано кожний рядок зображення кодувати індивідуально одним із цих чотирьох способів. При кодуванні адаптивний кодер знаходить оптимальні параметри відповідного способу кодування для забезпечення найменшого обсягу коду кожного рядка. Адаптивний кодер реалізовано у програмному забезпеченні геоінформаційної системи [5].

3. Постановка задачі

Позитивною рисою метода RLE є висока швидкість декодування. Потрібно шукати шляхи організації растрових кодованих даних задля ефективної реалізації цього метода компресії. Це дозволить досягти більшої зручності користувачам різноманітних інформаційних систем, які працюють із растровими даними.

4. Виклад основного матеріалу

Спочату наведемо приклади зображень, які придатні для компресії без втрат методом RLE (рис. 4, 5). Далі у таблиці 1 надані порівняльні відомості компресії зображень різними методами

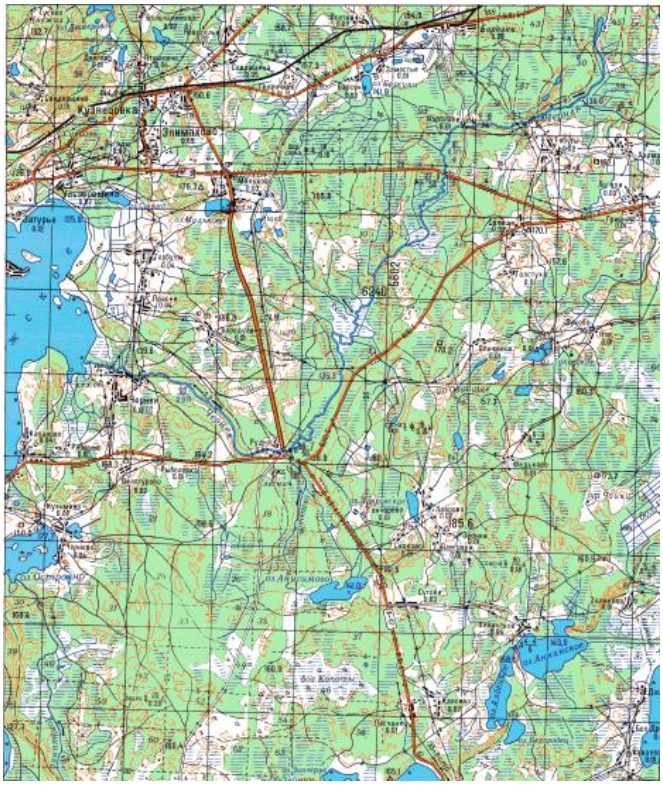

Рис.4. Фрагмент растрової карти

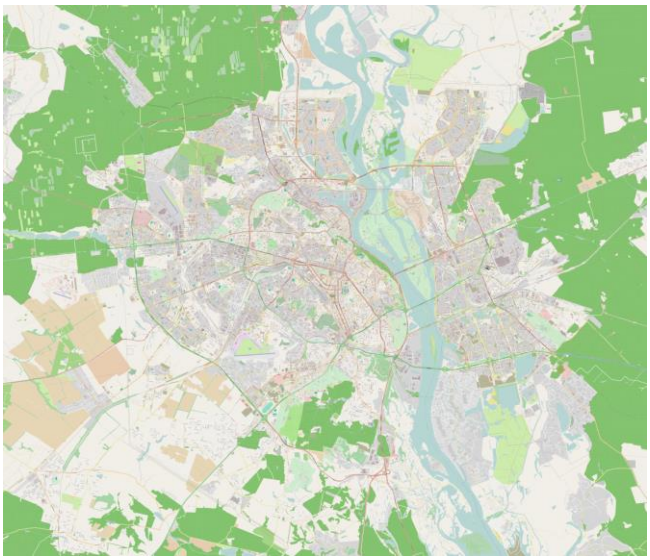

Рис. 5. Картосхема

Таблиця 1

Зображення, розміри растру, обсяг файлу без компресії	Обсяг компресованого коду, ступінь компресії для різних методів та відповідних форматів файлів			
	RLE PCX	RLE TGA	RLE-БП	LZW GIF
Томограма рис. 3 512×512 258 КБ	145 КБ 1.78	151 КБ 1.71	125 КБ 2.06	128 КБ 2.01
Карта рис. 4 1861×2204 4014 КБ	2272 КБ 1.77	2458 КБ 1.63	1953 КБ 2.05	2015 КБ 1.98
Картосхема рис. 5 11680×10000 114064 КБ	23004 КБ 4.96	23963 КБ 4.76	7635 КБ 14.94	7311 КБ 15.60

Примітка. Обсяги пам’яті оцінювалися розмірами відповідних файлів. Для оцінки обсягу некомпресованого растру вибрано файли формату BMP. Для оцінки RLE-БП наведено розміри файлів власних форматів GGF4 та RPF, хоча в них міститься також і додаткова інформація, що трохи збільшує обсяг файлів.

Наведені у таблиці дані ілюструють можливості компресії зображень різних типів. Для зображення томограми (рис. 3) компресія невеличка і становить два рази (для досягнення більшої компресії потрібно використати ще й дельта-кодування, як і зазначається автором публікації [2]). Приблизно такий самий ступінь компресії для фрагмента карти, насиченої дрібними елементами (рис. 4). В обох випадках RLE-БП випереджає не тільки RLE PCX, TGA, але й представника іншого класу методів компресії – словниковий метод LZW для формату GIF. Для растру картосхеми високої роздільності з великими (до декількох сотень) ланцюжками повторюваних пікселів (рис. 5), перевага RLE-БП над RLE PCX, TGA майже трикратна, проте, він трохи поступається LZW GIF.

Можна поставити запитання: а навіщо взагалі використовувати RLE, якщо існують такі методи, зокрема словникові LZ-подібні, які забезпечують, загалом, вищу ступінь компресії? Доцільність використання RLE обумовлена такими факторами:

- висока швидкість декодування;
- відсутність потреби додаткової пам’яті для декодування;
- простота реалізації прямого доступу до фрагментів растрових даних великих розмірів.

Стосовно прямого доступу. Під цим розуміється можливість читання будь-якого фрагменту растру без декодування того, що передує фагменту. Необхідно відзначити, що класичний метод RLE, загалом, не вимагає розділення масиву даних і кодування окремими порціями, наприклад, рядками растру. Більше того, якщо витягнути двовимірний растр у одновимірний вектор, то ступінь компресії може бути більшою, порівняно із компресією окремими рядками – це активно використовують словникові LZ-подібні алгоритми компресії, для яких кодування окремими рядками суттєво зменшить компресію. Можна сказати, що більшість відомих реалізацій метода RLE традиційно орієнтовані на достатньо зручний для декодування запис двовимірного растру послідовністю рядків. Це дещо сприяє організації прямого доступу.

Оскільки кожний рядок растру кодується, загалом, різним обсягом байтів коду, то для забезпечення швидкого прямого доступу можна запропонувати такі способи:

1) у заголовку файлу записувати масив вказівників на початки кодованих рядків;
2) на початку кожного кодованого рядка записувати обсяг коду рядка.

Обидва ці способи призводять до збільшення обсягу коду на величину (кількість рядків) × (кількість бітів значення), проте таке збільшення не суттєве. Щодо прискорення доступу, то тут перевага за першим способом, оскільки перейти до кожного рядка можна одною операцією **seek**, проте для найшвидшої реалізації декодеру потрібно використовувати додаткову оперативну пам'ять. Другий спосіб реалізує доступ до початку будь-якого рядка по

ланцюжку, пропускаючи без декодування коди непотрібних рядків.

Для растрових зображень одним з популярних видів прямого доступу є перегляд частини зображення у вікні зі скролінгом

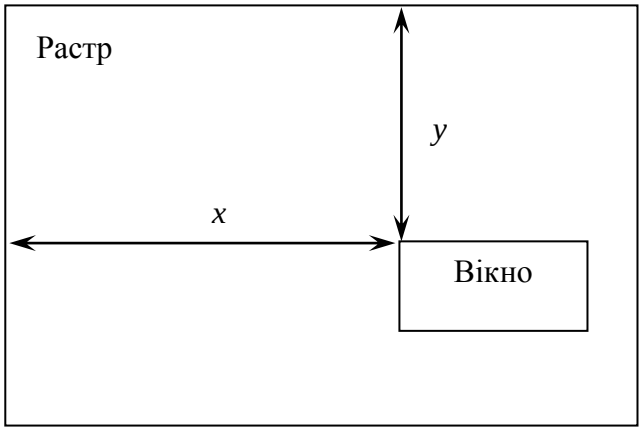

Рис. 6. Перегляд частини зображення у вікні

На рис. 7 наведено графік залежності часу доступу від *y*-координати розташування прямокутного фрагменту (вікна) розмірами 1000×500 для зображення картосхеми рис. 5 розмірами 11680×10000

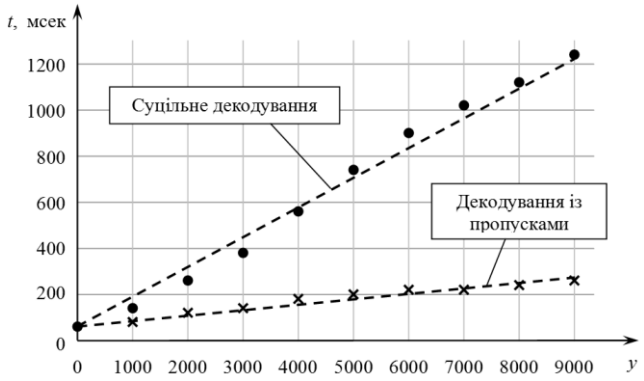

Рис. 7. Швидкодія доступу до фрагменту растру різними способами

Оскільки кількість бітів коду для кожного рядка індивідуальна і визначається конкретним зображенням, то залежність *t(y)* загалом є рандомною, хоч і монотонно зростаючою. Її тренд відображено на рис. 7 пунктирною прямою. Пропуск рядків без декодування дозволяє прискорити майже у 5 разів доступ до фрагментів, максимально віддалених від лівого верхнього кута растру. Для окремих реалізацій час доступу не залежіть від горизонтального зсуву вікна (*x*).

Іншим популярним видом прямого доступу до растру є огляд усього зображення з певним масштабом. Найпростішим є масштабування шляхом проріджування рядків *(interlacing)*. Для огляду зображення наведеної вище картосхеми час декодування усіх рядків на не дуже швидкому комп'ютері становить близько 1250 мсек, проте можливість пропуску-проріджування окремих рядків дає зображення у масштабі 1:25 приблизно у 4.5 разів швидше.

5. Висновки

У результаті аналізу модифікацій класичного метода RLE висвітлено способи побудування кодових послідовностей, які забезпечують адаптування параметрів кодування для врахування особливостей конкретних растрів і, як наслідок, забезпечують підвищення компресії. Проілюстровані переваги модифікації RLE-БП для компресії растрових зображень. Запропоновано підхід до організації прямого доступу до растрових даних високої роздільності, який дозволяє досягти високої швидкості роботи з такими даними.

Список посилань

1. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ-МИФИ, 2003. – 384 с.
2. Шамраева Е.О. Комплексный метод компрессии томограмм // Збірник наукових праць Харківського університету Повітряних Сил, 2014, випуск 2(39) С.160-162
3. Карпенко А.С., Крысова И.В. Повышение эффективности сжатия компьютерной графики с использованием RLE алгоритма // Инженерный вестник Дона. – 2015. – №4. [Електронний ресурс]. – Режим доступу: ivdon.ru/ru/magazine/archive/ n4y2015/3441
4. Аль-Бахдили Х. К., Томилин В. В., Конопелько В. К., Аль-Заиди З. Х. М. Модифицированные алгоритмы кодирования длин серий для сжатия полутоновых изображений / Х. К. Аль-Бахдили [и др.] // Наука – образованию, производству, экономике :

материалы 14-й Международной научно-технической конференции. – Минск : БНТУ, 2016. – Т. 1. – С. 212.

5. Блінова Т.О., Порєв В.М. Деякі способи кодування інформації // Вісник Національного технічного університету України "Київський політехнічний інститут". Інформатика, управління та обчислювальна техніка. – Київ "ВЕК+". – 2007, № 46. – С. 96-104
6. Блинова Т.А., Порев В.Н. Некоторые способы кодирования растров в геоинформационных системах // Электронное моделирование. – 2008. – Т.30, №1. – С. 119-128
7. Blinova T., Porev V. Some Methods Of The Raster Encoding In Geographic Information Systems // Тези доповідей на міжнародній конференції “CODATA`21”, Київ, 2008. – С.153.