Відбір і обробка інформації. Вісник ФМІ, Львів; № 13, 2008. – С.101–110. 6. Шаховська Н.Б. Інтеграція, консолідація та федералізація даних для інформаційних технологій туристичного бізнесу // Д.І. Угрин. – Відбір і обробка інформації. ФМІ, № 16, 2008. – С.98–108. 7. Шаховська Н.Б. Альтернативні рішення ситуацій опрацювання даних з розрізнених джерел в просторах даних туризму // Д.І. Угрин. – Інтернет-конференція РусНаука, [Електронний ресурс]: http://www. rusnauka. com/25_DN_2008/Matemathics/28431.doc.htm.

УДК 004.043

О.В. Шпортько
Рівненський державний гуманітарний університет,
кафедра інформатики та прикладної математики

ОПТИМІЗАЦІЯ БЛОКІВ СТИСНУТИХ ДАНИХ У ГРАФІЧНОМУ ФОРМАТІ PNG

Запропоновано алгоритм генерування альтернативних стиснутих блоків, вибору найкоротшого стиснутого блока з альтернативних та ітеративного зменшення його розмірів для покращання компресії зображень у форматі PNG. Як показують експерименти, реалізація цього алгоритму дає змогу підвищити коефіцієнт стиснення переважної більшості зображень на 2 – 6 %.

Ключові слова – компресія зображень, формат PNG, коефіцієнт стиснення.

This algorithm of the generation of alternative compressed blocks, choice of the shortest compressed block from alternative one and iterative diminishing of its size for the improvement of compression images in the format of PNG is offered in the following article. As the experiments show, the realization of this algorithm allows to raise the compression factor of the majority of images in 2 – 6 percent.

Keywords – compression images, format PNG, compression factor.

Вступ

Формат графічних файлів PNG був створений 1 жовтня 1996 року для ефективного збереження растрових зображень без втрат після того, як компанія Unisys почала вимагати плату за використання формату GIF [1]. Сьогодні дизайнери та розробники Web-сайтів найчастіше зберігають фотореалістичні зображення у форматі JPEG, а дискретно-тонові – у форматі PNG. Крім цього, формат PNG найчастіше використовується для зберігання зображень, де втрати недопустимі (наприклад, для рентгенівських знімків). Саме тому цей формат підтримує більшість сучасних програм для перегляду і створення зображень. Проблема підвищення ефективності стиснення зображень у форматі PNG є актуальною сьогодні і буде актуальною в найближчому майбутньому, оскільки навіть наближення до її вирішення дасть змогу зменшити розміри відповідних файлів, що, своєю чергою, сприяє підвищенню ефективності використання дискового простору та прискорення завантаження з мережі. У цій статті описується один із способів часткового вирішення вказаної проблеми.

Принципи та оптимізація стиснення зображень у форматі PNG. Аналіз останніх досліджень. Постановка задачі

Будь-яке стиснення даних можливе за рахунок зменшення надлишковостей. Чим більше видів надлишковостей виявляє й опрацьовує компресор і чим краще він ці надлишковості усуває – тим

краще він зможе стиснути націлені на таку обробку дані. Формат PNG орієнтований на зменшення надлишковостей чотирьох типів:

- між сусідніми пікселами, що мають, як правило, близькі кольори;
- між однаковими фрагментами зображення;
- між однаковими змінами кольорів;
- між переважаючими кольорами пікселів зображення.

Пікселі зображення записуються у PNG-файли найчастіше по рядках зверху вниз, а у кожному рядку – послідовно зліва направо (як символи у текстах), формуючи тим самим вхідний потік. Перед кодуванням ці пікселі можуть бути попередньо оброблені предикторами, принцип дії яких розглядається наприкінці цього розділу. У PNG-файлах, що використовуються сьогодні, стиснуті дані зберігаються у відокремлених блоках згідно з форматом словникового стиснення DEFLATE [1, 2]. У блоках стиснутих даних містяться результати застосування до вхідного потоку алгоритму LZH [2], згідно з яким результати контекстно-залежного словникового алгоритму LZ77 [3] стискаються контекстно-незалежними кодами Хафмана.

Описуючи словникові алгоритми, фіксовану кількість попередніх закодованих елементів вхідного потоку називають словником, а наступних незакодованих – буфером. Алгоритм LZ77 ґрунтується на заміні у вихідному потоці послідовності чергових елементів буфера посиланням на аналогічну послідовність елементів словника у вигляді пари чисел *<довжина; зміщення від закінчення словника>*. У разі відсутності аналогічної послідовності елементів у словнику перший елемент буфера переноситься у вихідний потік без змін. Після цього закодовані елементи переносяться з початку буфера в кінець словника і кодування продовжується аналогічно аж до закінчення елементів вхідного потоку. Наприклад, потік кодів бітів пікселів *36 38 35 35 36 38 35 36 36 38 35 38 36 38 35 35 28* в закодованому вигляді записується як *36 38 35 35 <3; 4> 36 <3; 4> 38 <4; 12> 28*. Алгоритм LZ77 використовується у форматі PNG насамперед для зменшення надлишковостей між однаковими фрагментами зображення.

Під час декодування кодів алгоритму LZ77 окремі елементи копіюються у вихідний потік без змін. Пари ж *<довжина; зміщення>* декодуються послідовним копіюванням з кінця вихідного потоку за вказаним зміщенням у кінець вихідного потоку необхідної кількості елементів. Природно, що алгоритм декодування повинен розрізняти окремі елементи та групи *<довжина; зміщення>*. Згідно з алгоритмом LZH у форматі DEFLATE з цією метою довжини замін та окремі елементи кодуються разом числами в межах [0; 285]. При цьому числа з діапазону [0; 255] відповідають кодам окремих елементів, 256 позначає закінчення блока, а числа з діапазону [257; 285] вказують на базові значення довжин. Після базових значень довжин міститься визначена форматом кількість бітів, що разом з базовим значенням однозначно визначає довжину заміни. Зміщення зберігається після відповідної довжини аналогічно – у вигляді базового значення та додаткових бітів. Базове значення зміщення лежить в межах [0; 29]. У форматі DEFLATE максимальне значення довжини закодованої послідовності може сягати 258, а зміщення – 32768.

Ідея використання кодів Хафмана, що застосовуються для кодування окремих елементів і базових значень довжин та базових значень зміщень після виконання алгоритму LZ77, полягає у заміні чисел з більшою частотою кодами меншої кількості бітів, ніж для чисел з меншою частотою. Коди Хафмана для елементів/довжин та зміщень визначаються для кожного блока стиснутих даних, як правило, окремо, що сприяє покращанню стиснення. Автономне кодування Хафмана у форматі PNG насамперед зменшує надлишковості між переважаючими кольорами пікселів зображення.

Згідно з теоремою Шеннона елемент s_i з ймовірністю появи $p(s_i)$ найвигідніше кодувати $-\log_2 p(s_i)$ бітами. Тоді середня довжина коду елемента після застосування контекстно-незалежного алгоритму має наближатися до *ентропії джерела* [2]:

$$H = -\sum_i p(s_i) \times \log_2 p(s_i). \quad (1)$$

Ентропія джерела зменшується у разі збільшення нерівномірності розподілу ймовірностей між елементами.

Зменшити ентропію джерела при обробці зображень у форматі PNG намагаються за допомогою предикторів. *Предиктор* – це функція, що прагне, використовуючи значення відомих суміжних елементів, спрогнозувати (змоделювати) значення чергового елемента. Якщо піксел зображення характеризується декількома компонентами (наприклад, R, G, B), то предиктор кожної компоненти прогнозує значення згідно з відповідними компонентами сусідніх пікселів. Використовуючи цю технологію, обчислюють і надалі кодують відхилення чергової компоненти від прогнозованого предиктором значення. Тому загалом застосування предикторів до кожної компоненти піксела у вузлі *(i, j)* можна записати формулою

$$\Delta_{ij} = C_{ij} - predict_{ij}, \qquad (2)$$

де C_{ij} – значення компоненти до застосування предиктора, Δ_{ij} – значення компоненти після застосування предиктора, $predict_{ij}$ – значення предиктора, обчисленого для вибраної компоненти.

Оскільки сусідні пікселі зображення мають, як правило, близькі кольори і тому близькі значення відповідних компонентів, то часто значення предиктора буде збігатися зі значенням чергового елемента, найчастіше – буде близьким до цього значення і рідко значно відрізнятиметься від нього. Тобто більшість значень Δ_{ij} будуть близькими до 0. Такий перерозподіл частот значень (а отже, і ймовірностей) значно підвищує нерівномірність розподілу [4] і тому зменшує ентропію джерела (згідно з (1)), а отже, і довжину закодованої послідовності. Отже, предиктори використовуються насамперед для зменшення надлишковостей між сусідніми пікселами зображення, що мають близькі кольори. Крім цього, застосування алгоритму LZ77 та кодування Хафмана до результатів дії предикторів дає змогу зменшити надлишковості між однаковими змінами кольорів.

У блоках стиснутих даних формату PNG даним кожного рядка передує окремий байт, що визначає предиктор, який застосовується до компонент всіх його пікселів. Сьогодні форматом передбачено п'ять можливих значень цього байта [1], що визначає чотири різні предиктори: 0 – дані рядка не обробляються предикторами; 1 – предиктор дорівнює значенню відповідної компоненти зліва; 2 – предиктор дорівнює значенню компоненти зверху; 3 – предиктор дорівнює середньому арифметичному значень ліворуч та зверху; 4 – предиктор Піфа, що прогнозує значення у напрямку найменшого приросту. Опис предикторів формату PNG міститься в [1], класифікація цих та інших предикторів наведена в [4].

Сьогодні у спеціалізованому програмному забезпеченні для покращання стиснення зображень у форматі PNG найчастіше використовують метод повного перебору різних варіантів предикторів, розмірів блоків стиснутих даних та стратегій стиснення. Ми ж пропонуємо алгоритм оптимізації блоків стиснутих даних за допомогою аналізу ефективності замін алгоритму LZ77.

Метою статті є опис алгоритму для покращання компресії зображень у форматі PNG за рахунок генерування альтернативних блоків стиснутих даних, вибору найкоротшого блока серед альтернативних та ітеративного зменшення його розміру.

Алгоритм генерування, зменшення розмірів та вибору найкоротшого серед альтернативних блоків стиснутих даних

Для виявлення типів стиснутих блоків проаналізуємо різні варіанти розподілів частот (тут і надалі – абсолютних) їх елементів/довжин у форматі PNG на прикладі перших 64 Кб зображення Lena.bmp. Значення компонент пікселів цього блока мають слабко виражену нерівномірність розподілу (рис. 1, a), адже найбільша частота серед окремих елементів дорівнює 1650 (2.5 % від загальної кількості). Ця нерівномірність значно посилюється після застосування предиктора Піфа (рис. 1, b), який дає змогу кардинально збільшити частоту нульового значення до 11405 (17.4 %), збільшити частоти елементів навколо нуля до понад 4000 (6.1 %, від'ємним результатам дії предикторів відповідають значення, збільшені на 256) та зменшити частоти всіх інших елементів.

Збільшує нерівномірність розподілу і застосування замін послідовностей (рис. 1, c), що збігаються. Але у фотореалістичних зображеннях, до яких належить і Lena.bmp, як правило, трапляються лише короткі повторення послідовностей елементів. Наприклад, у першому блоці цього зображення віднайдено 8016 повторень по 3 елементи (відповідає значенню 257) та 933 повторення по 4

елементи (відповідає значенню 258). При цьому короткі послідовності, що збігаються, трапляються, як правило, серед елементів з найбільшою частотою, і тому застосування таких замін зменшує нерівномірність розподілу окремих елементів. Наприклад, частота елемента 96 зменшилася з 1650 до 513. Ще більше коротких послідовностей, що збігаються, зустрічається у фрагментах зображення після застосування предикторів (рис. 1, d). Наприклад, після застосування предиктора Піфа кількість триелементних послідовностей, що збігаються, зросла до 11066, а 4-елементних – до 3312. Водночас зменшилася кількість найдовших послідовностей, що збігаються, з 23 до 22, оскільки застосування предикторів реалізує вплив сусідніх елементів і здатне фрагментувати такі послідовності.

Зовсім інші властивості мають розподіли елементів/довжин у випадку застосування лише довгих замін. Такі заміни, як правило, зменшують максимальні частоти елементів, але не порушують характеру нерівномірності розподілу. Наприклад, застосування замін від 9 елементів зменшило максимальну частоту розподілу з 1650 (рис. 1, a) до 1538 (рис. 1, e), а після застосування предиктора – з 11405 (рис. 1, b) до 5266 (рис. 1, f).

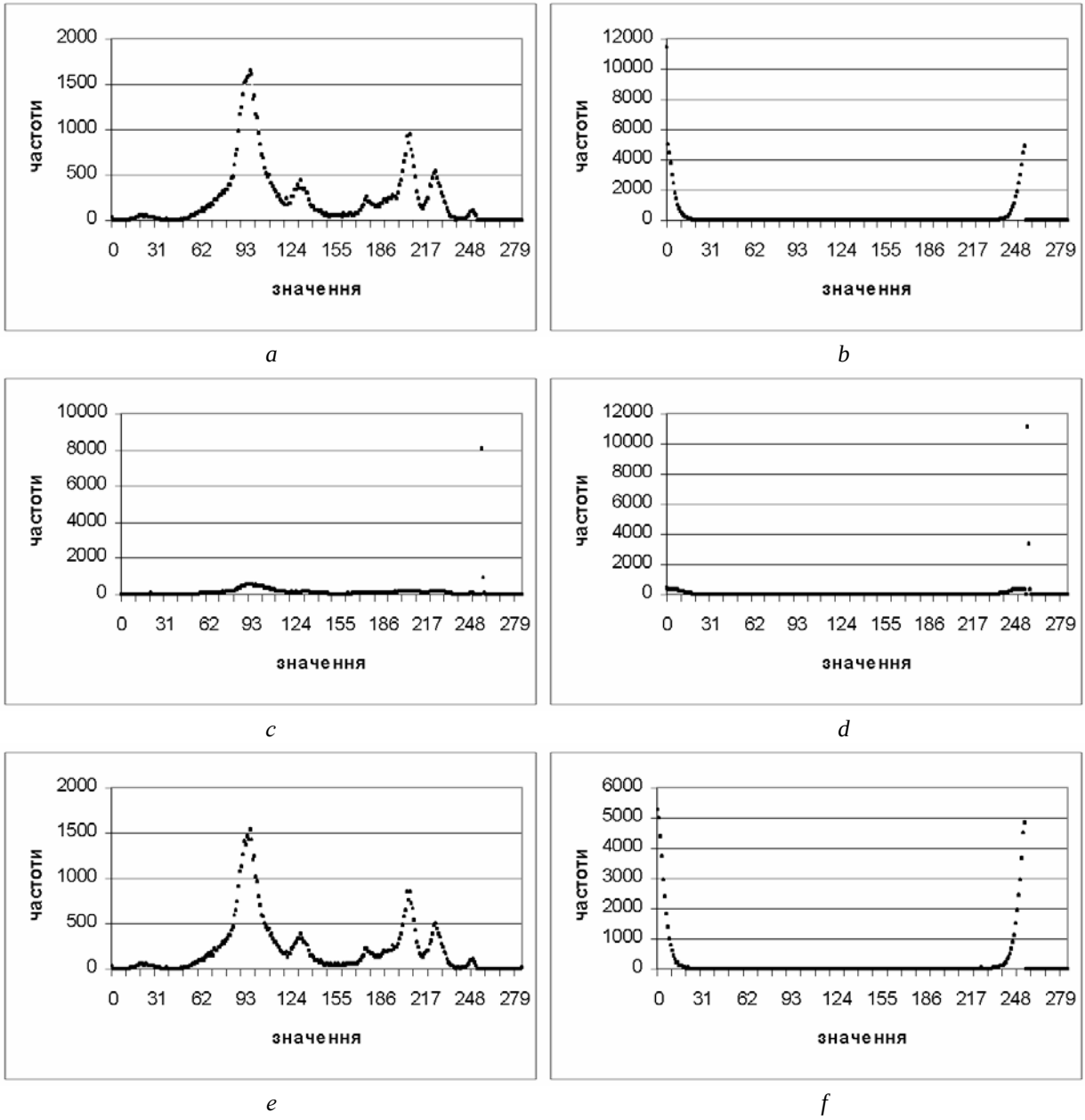

Рис. 1. Розподіл частот елементів/довжин перших 64 Кб зображення Lena.bmp: a – без застосування предикторів та замін; b – після застосування предиктора Піфа; c – після застосування замін; d – після застосування предиктора Піфа та замін; e – після застосування замін від 9 елементів; f – після застосування предиктора Піфа та замін від 9 елементів

Заміни вважатимемо ефективними, якщо вони кодуються не більшою кількість бітів, ніж окремі елементи, які вони замінюють. Оскільки окремі елементи і базові значення довжин та зміщення у форматі DEFLATE записуються кодами Хафмана, то заміна j довжини len_j, що виконується починаючи з елемента s_k за зміщенням $offset_j$, ефективна лише тоді, коли

$$\sum_{i=0}^{len_j-1} l_{s_{k+i}} \geq l_{len_j} + d_{len_j} + \lambda_{offset_j} + \delta_{offset_j}, \quad (3)$$

де l_m – довжина коду Хафмана елемента/довжини з кодом m, d_m – кількість додаткових бітів для запису довжини заміни m, λ_m – довжина коду Хафмана зміщення з кодом m, δ_m – кількість додаткових бітів для запису зміщення m.

Але короткі заміни виявляються ефективними (рис. 1, c, 1, d), як правило, для стиснутих блоків, в яких інші короткі заміни теж враховуються (назвемо їх *орієнтованими на заміни*). Це пов'язано з тим, що короткі заміни істотно зменшують частоти окремих елементів, а отже, збільшують довжини їх кодів Хафмана. З іншого боку, короткі заміни виявляються, як правило, неефективними для стиснутих блоків, у яких інші короткі заміни теж не враховуються (рис. 1, e, 1, f), оскільки довгі заміни найчастіше не порушують характеру розподілу окремих елементів (назвемо такі блоки *орієнтованими на елементи*).

Частоти окремих елементів у блоках, орієнтованих на елементи, завжди будуть не меншими від частот цих самих елементів у блоках, орієнтованих на заміни (рис. 2). Розподіли ж замін будуть розміщуватися навпаки.

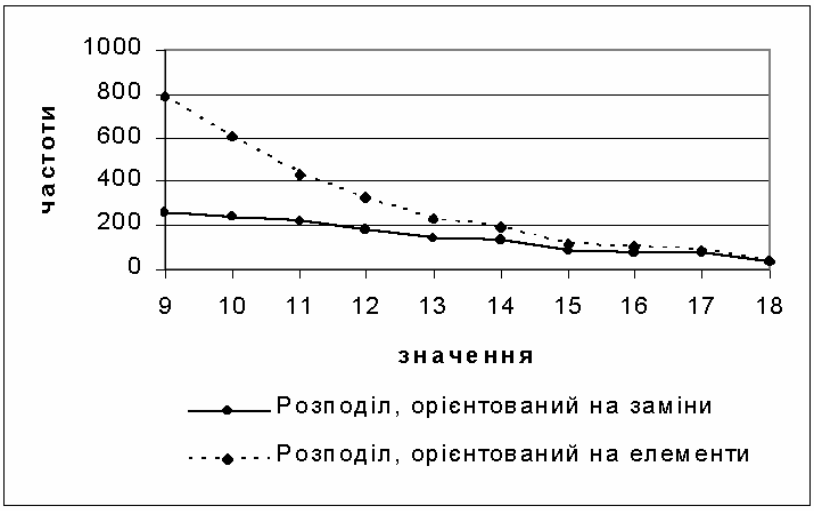

Рис. 2. Фрагмент альтернативних розподілів частот елементів перших 64 Кб зображення Lena.bmp

Для окремого блока даних коротшим може виявитися як стиснутий блок, орієнтований на заміни, так і стиснутий блок, орієнтований на елементи – переважно це залежить від характеру розподілу його окремих елементів: якщо блок даних має виражену нерівномірність, то для нього, як правило, коротшим виявляється блок стиснутих даних, орієнтований на елементи, у разі ж відсутності вираженої нерівномірності коротшим, як правило, виявляється блок, орієнтований на заміни. Крім цього, на довжини стиснутих блоків істотно впливають зміщення між однаковими фрагментами блока даних, адже більші зміщення кодуються більшою кількістю додаткових бітів. Для чергового блока даних характер розподілу його елементів і, тим більше, зміщень наперед визначити неможливо, тому **для кожного блока даних доцільно створити два альтернативні**

стиснуті блоки: орієнтований на заміни та орієнтований на елементи, зменшити їх довжини, застосовуючи (3), **вибрати з них найкоротший, ітеративно зменшити його довжину та використати для зберігання даних** у форматі DEFLATE (див. рис. 3).

У форматі DEFLATE мінімальна довжина кодів Хафмана дорівнює 1, максимальна – 15, максимальна кількість додаткових бітів довжини становить 5, а зміщення – 13. Тому, згідно з (3), для будь-яких розподілів ефективними будуть заміни довжиною від 48 елементів. Ефективність решти замін залежить не лише від їх елементів, а й загалом від розподілу стиснутого блока.

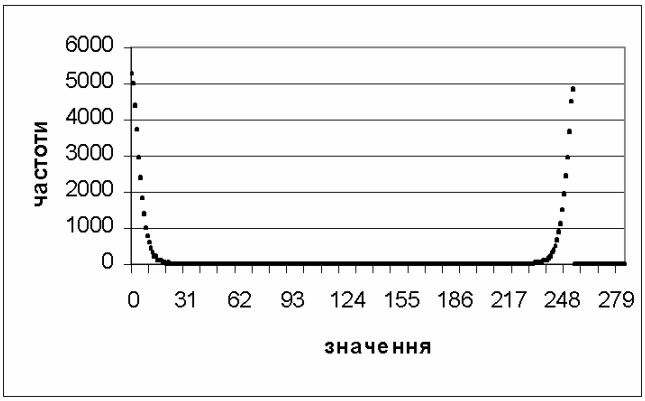

Рис. 3. Розподіл частот елементів/довжин перших 64 Кб зображення Lena.bmp після застосування алгоритму вибору найкоротшого стиснутого блока

Зменшити довжини стиснутих блоків, орієнтованих на заміни, можна за рахунок відкидання неефективних замін. Згідно з (3) такими вважатимемо заміни, для яких

$$\sum_{i=0}^{len_j-1} l'_{s_{k+i}} < l'_{len_j} + d_{len_j} + \lambda'_{offset_j} + \delta_{offset_j}, \tag{4}$$

де штрих над літерами вказує на розрахунок параметрів з блока, орієнтованого на заміни.

Зменшити довжини стиснутих блоків, орієнтованих на елементи, можна за рахунок урахування ефективних замін. Згідно з (3) такими вважатимемо заміни, для яких

$$\sum_{i=0}^{len_j-1} l''_{s_{k+i}} \geq l''_{len_j} + d_{len_j} + \lambda''_{offset_j} + \delta_{offset_j}, \tag{5}$$

де два штрихи над літерами вказують на розрахунок параметрів з блока, орієнтованого на елементи.

Заміни, ефективні в блоках, орієнтованих на елементи, виявляються ефективними і в блоках, орієнтованих на заміни. І навпаки: заміни, неефективні в блоках, орієнтованих на заміни, виявляються неефективними і в блоках, орієнтованих на елементи. До того ж урахування ефективних замін в блоках, орієнтованих на елементи, зменшує частоти окремих елементів і, відповідно, може збільшити після перерахунку довжини їх кодів Хафмана. Тому серед замін, неефективних з попередніми кодами Хафмана, можуть виявитися ефективні з перерахованими такими кодами. І навпаки, вилучення неефективних замін у блоках, орієнтованих на заміни, збільшує частоти окремих елементів і, відповідно, може зменшити після перерахунку довжини їх кодів Хафмана. Тому серед замін, ефективних з попередніми кодами Хафмана, можуть виявитися неефективні з перерахованими такими кодами. Більшість замін виявляються, як правило, ефективними в блоках, орієнтованих на заміни і неефективними в блоках, орієнтованих на елементи. Після кожного перерахунку кодів розподіли частот альтернативних стиснутих блоків будуть зближуватися.

Враховуючи те, що довжина стиснутого блока складається з суми довжин закодованих елементів, довжин замін та зміщень, загальну довжину блока, орієнтованого на заміни, будемо розраховувати за формулою

$$L' = \sum_{i=0}^{255} n_i' l_i' + l_{256}' + \sum_{i=257}^{285} n_i' \left(l_i' + d_i \right) + \sum_{i=0}^{29} \eta_i' \left(\lambda_i' + \delta_i \right), \tag{6}$$

де n_i – частота елемента чи базової довжини i; η_i – частота базового зміщення i. Довжину блока, орієнтованого на заміни, будемо розраховувати аналогічно:

$$L'' = \sum_{i=0}^{255} n_i'' l_i'' + l_{256}'' + \sum_{i=257}^{285} n_i'' \left(l_i'' + d_i \right) + \sum_{i=0}^{29} \eta_i'' \left(\lambda_i'' + \delta_i \right). \tag{7}$$

Покроково алгоритм використання альтернативних стиснутих блоків для кожного блока даних записується так:

1. Створити для блока даних два альтернативні стиснуті блоки: блок, орієнтований на заміни, – із застосуванням всіх можливих замін і блок, орієнтований на елементи, – із застосуванням замін з довжиною від 48 елементів. Створити список замін, що ввійшли в перший і не ввійшли в другий стиснутий блок для подальшого аналізу. Визначити для елементів блоків довжини відповідних кодів Хафмана.
2. Вилучити неефективні заміни згідно з (4) зі списку аналізованих та з блока, орієнтованого на заміни.
3. Вилучити ефективні заміни згідно з (5) зі списку аналізованих та врахувати їх в блоці, орієнтованому на елементи.
4. Вибрати найкоротший стиснутий блок серед альтернативних згідно з (6) та (7).
5. Якщо найкоротшим серед альтернативних блоків стиснутих даних виявився блок, орієнтований на заміни, то у разі зменшення кількості замін для аналізу після виконання кроку 2 ітеративно перевизначати для його елементів довжини кодів Хафмана і вилучати з нього та зі списку аналізованих неефективні заміни згідно з (4) доти, доки такі заміни будуть зустрічатися.
6. Якщо найкоротшим серед альтернативних блоків стиснутих даних виявився блок, орієнтований на елементи, то у випадку зменшення кількості замін для аналізу після виконання кроку 3 ітеративно перевизначати для його елементів довжини кодів Хафмана і враховувати в ньому та вилучати зі списку аналізованих ефективні заміни згідно з (5) доти, доки такі заміни будуть зустрічатися.
7. Зберегти отриманий за результатами попередніх кроків блок стиснутих даних згідно з форматом DEFLATE.

Зауважимо, що вибір найкоротшого стиснутого блока серед альтернативних можна виконувати і після ітеративного зменшення їх довжин. Це дає змогу поєднати другий крок алгоритму з п'ятим та третій – з шостим, але майже вдвічі сповільнює його виконання.

Враховуючи те, що блок, орієнтований на заміни, формується зі всіх замін, а блок, орієнтований на елементи, формується з замін довжиною від 48 елементів, додатково можна аналізувати ще й інші альтернативні блоки стиснутих даних, які відразу враховують коротші заміни. Такі альтернативні блоки незначно підвищують показники стиснення (до 0.1 %), але істотно сповільнюють його виконання (на понад 10 %), і тому на практиці не використовуються.

Результати експериментів

На завершення розглянемо результати застосування описаного алгоритму використання альтернативних блоків стиснутих даних для компресії восьми різнотипних 24-бітних зображень стандартного набору файлів ACT у форматі PNG (завантажити їх TIFF-версії можна з http://compression.ru/arctest/act/act-files.html, http://compression.ca/act/ act-files.html чи http://links.uwaterloo.ca/colorset.base.html).

Таблиця 1

Коефіцієнти стиснення файлів зображень набору ACT у форматі PNG після застосування різних варіантів програм, %

Bmp-файл	Photo Editor 2000	Miano без алгоритму	Miano з алгоритмом	OptiPng стандарт	OptiPng макс.
Clegg	62.26	76.30	79.15	77.63	77.82
Frymire	92.57	93.15	93.15	93.21	93.21
Lena	4.81	33.68	39.79	39.66	39.66
Monarch	23.50	45.19	47.44	47.88	47.96
Peppers	8.45	41.74	46.16	46.03	46.16
Sail	13.62	33.82	33.82	33.48	34.17
Serrano	92.35	92.83	92.83	92.90	92.90
Tulips	9.97	38.94	42.15	42.32	42.41
Середній	**38.44**	**56.96**	**59.31**	**59.14**	**59.29**
Сукупний	**54.65**	**67.93**	**69.60**	**69.37**	**69.49**

Таблиця 2

Час стиснення © файлів зображень набору ACT у форматі PNG різними варіантами програм на комп'ютері з частотою 300 МГц

Bmp-файл	Photo Editor 2000	Miano без алгоритму	Miano з алгоритмом	OptiPng стандарт	OptiPng макс.
Clegg	3	14	16	72	996
Frymire	4	18	19	78	684
Lena	2	7	8	14	289
Monarch	3	12	17	33	733
Peppers	2	7	9	16	408
Sail	3	12	13	21	486
Serrano	2	8	8	23	336
Tulips	3	11	14	26	562
Разом	**22**	**89**	**104**	**283**	**4494**

Тестування виконували за допомогою програми з CD [1], у яку було внесено такі модифікації: забезпечена можливість виходу зі словника в буфер під час кодування повторів; реалізований вибір предиктора для рядка пікселів зображення на основі порівняння ефективності стиснення зображення без попередньої обробки з оцінкою нерівномірності розподілу після дії кожного предиктора; запроваджений аналіз кількостей додаткових бітів при записі зміщень; розмір блоків даних збільшений до 64 Кб та відкинуті допоміжні текстові блоки.

Результати тестування наведено в табл. 1, 2, де показником компресії файлів вибрано коефіцієнт стиснення, тобто процент зменшення початкового розміру файла, оскільки він має загальний характер, як і описаний алгоритм. Дані тестування програми без застосування розглянутого алгоритму наведено в третьому, а із застосуванням – у четвертому стовпці. Крім цього, для порівняння ефективності стиснення у другому стовпці таблиць вказано результати тестування програми Microsoft Photo Editor 2000, яка не застосовує предиктори, а в п'ятому та шостому – результати популярної серед Web-дизайнерів програми OptiPng (http://www.optipng.sourceforge.net), яка генерує короткі PNG-файли за результатами відповідно стандартного та максимального перебору предикторів, розмірів стиснутих блоків та стратегій стиснення.

Як свідчать результати тестування, використання описаного алгоритму підвищило коефіцієнт стиснення на 2 – 6 % для 63 % зображень, які, як правило, мають виражену нерівномірність розподілу, та не вплинуло на компресію решти файлів, хоча й сповільнило виконання програми в середньому на 17 %. Крім цього, реалізація розглянутого алгоритму дало змогу наблизитися до коефіцієнта стиснення програми повного перебору, а для деяких зображень і перевищити його, витрачаючи для компресії в 1.75 – 62 рази менше часу.

Висновки

1. Для підвищення ефективності стиснення даних у форматах, що послідовно використовують алгоритми декількох методів, слід враховувати взаємний вплив цих алгоритмів.
2. Розглянутий алгоритм дає змогу істотно підвищити коефіцієнт стиснення більшості зображень за рахунок використання орієнтованого на елементи та орієнтованого на заміни альтернативних стиснутих блоків для кожного блока даних.
3. Описаний алгоритм не вимагає модифікації декодера чи програм перегляду зображень і за рахунок зменшення розмірів файлів лише прискорює їх роботу. Саме тому він може бути ефективно застосований для збереження даних у стандартах, що використовують формат стиснення DEFLATE.

1. Миано Дж. Форматы и алгоритмы сжатия изображений в действии: Учеб. пособ. / Дж. Миано. – М.: Триумф, 2003. – С. 249–318. – (Практика программирования). 2. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М. : ДИАЛОГ-МИФИ, 2003. – С. 17–106. 3. Ziv J., Lempel A. A universal algorithm for sequential data compression / J. Ziv, A. Lempel // IEEE Transactions on Information Theory. – May 1977. – Vol. 23(3). – P. 337–343. 4. Бредихин Д. Ю. Сжатие графики без потерь качества [Електронний ресурс] / Д. Ю. Бредихин. – 2004. – http: // www.compression.ru/ download/articles/i_lless/bredikhin_2004_lossless_image_compression_doc.rar.

УДК 378.147:376.352 (477)

Б.І. Шуневич
Львівський державний університет безпеки життєдіяльності,
кафедра іноземних мов та технічного перекладу

ТЕНДЕНЦІЇ РОЗВИТКУ СКЛАДОВИХ ЧАСТИН ОРГАНІЗАЦІЇ ДИСТАНЦІЙНОГО НАВЧАННЯ

Проаналізовано тенденції розвитку трьох складових частин, необхідних для організації дистанційного навчання у ВНЗ України, а саме: 1) використання віртуальних навчальних середовищ; 2) розроблення дистанційних курсів; 3) підготовки викладацького штату до роботи з новими технологіями навчання. Крім цього, розглянуто використання дистанційнних курсів або їх елементів у традиційному навчанні.

Ключові слова – дистанційне навчання, віртуальні навчальні середовища, дистанційні курси.

The article deals with tendency analysis of three component development needed for distance learning organization at Ukrainian higher schools, namely: 1) application of virtual learning environment; 2) distance course development; 3) faculty personnel training for implementation of new teaching/learning technologies. Besides application of distance courses or their elements in traditional learning is considered.

Keywords – distance learning, virtual learning environment, distance courses.

Як відомо, дистанційне навчання (ДН) може повноцінно розвиватися за наявності принаймні таких її основних складових частин: 1) нормативно-правової бази; 2) контингенту студентів; 3) закладів, які організовують дистанційне навчання; 4) кваліфікованих викладачів; 5) відповідної матеріально-технічної бази (апаратного і програмного забезпечення, можливості організації аудіо- і відеоконференцій) у студента, викладача і навчального закладу; 6) навчальних матеріалів;