Рівненський державний гуманітарний університет

ВОЛИНСЬКИЙ МАТЕМАТИЧНИЙ ВІСНИК

СЕРІЯ ПРИКЛАДНА МАТЕМАТИКА

Збірник наукових праць

Випуск 7 (16)

Рівне-2010

"Волинський математичний вісник. Серія прикладна математика" публікує результати досліджень з математичного моделювання і обчислювальних методів та суміжної проблематики в галузі математики, інформатики, механіки. Розрахований на наукових працівників, викладачів ВНЗ, аспірантів та студентів старших курсів.

"Волынский математический вестник. Серия прикладная математика".
The "Volyn Mathematical Bulletin. Applied Mathematics Series".

Редакційна колегія

Барановський С.В.
Бейко І.В.
Бомба А.Я. *(головний редактор)*
Булавацький В.М.
Бурак Я.Й.
Власюк А.П.
Войтович М.М.
Гаращенко Ф.Г.
Гарбарчук В.І.
Джунь Й.В.
Каштан С.С.
Климюк Ю.Є. *(технічний секретар)*
Кратко М.І.
Кузьменко А.П.
Кундрат М.М.
Ляшенко І.М.
Недашковський М.О.
Новіков О.М.
Петрівський Я.Б.
Пономаренко Л.А.
Пригорницький Д.О. *(секретар)*
Присяжнюк І.М.
Савула Я.Г.
Свідзинський А.В.
Скопецький В.В. *(консультант)*
Сяський А.О.
Турбал Ю.В.
Чикрій А.О.
Шваб'юк В.І.
Янчук П.С.

Видається у Рівненському державному гуманітарному університеті при сприянні Інституту кібернетики ім. В. М. Глушкова НАН України, Інституту прикладних проблем механіки і математики ім. Я. С. Підстригача НАН України, навчальних закладів та наукових товариств Волинського регіону. Друкується за ухвалою Вченої ради РДГУ (протокол № 3 від 29 жовтня 2010 р.).

Адреса редакції: 33028, Україна, м. Рівне, вул. Остафова, 31,
Рівненський державний гуманітарний університет,
кафедра інформатики та прикладної математики, редакція ВМВ.
Тел.: +380362260444. E-mail: vmvspm@gmail.com.

ISBN 966-7281-02-8
© РДГУ, 2010

Зміст

Пам'яті Скопецького Василя Васильовича 5

Бігун Я. Й., Левицька О. І., Сергєєва Л. М. *Побудова просторової структури для моделі поширення епідемії зі змінним коефіцієнтом ліквідації* 8

Бомба А. Я., Пеньковський С. О., Савюк Є. В. *Метод фіктивної фільтрації моделювання одного класу квазіідеальних процесів руху рідин* 20

Бомба А. Я., Теребус А. В. *Комплексно спряжені многочлени і крайові задачі на конформні відображення* 30

Бомба А. Я., Шпортько О. В., Шпортько Л. В. *Використання статичного арифметичного кодування у растровому графічному форматі PNG* 43

Булавацький В. М. *Математичне моделювання процесу консолідації деформівних пористих середовищ за умов підземного вилуговування* 59

Климюк Ю. Є. *Числово-асимптотичне наближення розв'язку просторової задачі моделювання процесу видалення залишкових катіонів алюмінію при фільтруванні через окислювально-відновні завантаження із врахуванням зміни фільтраційних властивостей середовища* 71

Климюк Ю. Є., Пригорницький Д. О. *Просторові аналоги крайових задач на конформні відображення для одного класу двозв'язних областей* 84

Климюк Ю. Є., Сівак В. М. *Моделювання процесу доочистки води від залишкових катіонів алюмінію фільтруванням через аніоноактивні завантаження із врахуванням зміни фільтраційних властивостей середовища* 93

Кузьменко А. П., Гладка О. М. *До розв'язування початково-крайової задачі для параболічного рівняння методом прямих* 110

Мусурівський В. І. *Про оцінку параметрів динамічної тралової системи* 116

Петрик М.Р. *Математичне моделювання та аналіз умов і параметрів масопереносу в двовимірних неоднорідних середовищах* 124

Рогаль І. В. *Застосування методу прямих до розв'язування крайових задач конвективної дифузії солей або гіпсів, що залягають у фільтраційному потоці у вигляді включення* .. 147

Романюк В. В. *Зведення 15 випадків розв'язку однієї строго опукло-вгнутої неперервної антагоністичної гри до шести перших розв'язків*..... 168

Сафоник А. П. *Математичне моделювання динамічного режиму магнітного фільтра на основі передатної функції* ... 193

Сяський А. О., Шинкарчук Н. В. *Мішана контактна задача для пластинки з криволінійним отвором і жорсткого диска* 199

Фурсачик О. А. *Числово–асимптотичне наближення розв'язків одного класу обернених сингулярно збурених задач типу «конвекція-дифузія»* 210

Шпортько О. В., Шпортько Л. В. *Підвищення ефективності стиснення зображень у форматі PNG за допомогою їх розбиття на блоки однорідних рядків*..... 217

Янчук П. С. *Застосування квазіспектральних поліномів до розв'язування задачі Коші* 242

Янчук П. С., Собко В. Г. *Квазіспектральні поліноми на базисі поліномів Чебишова* 260

Ярощак С. В. *Один метод математичного моделювання еволюції границі розділу різнокольорових рідин у неоднорідному пласті*..... 281

УДК 004.043

Шпортько О. В., Шпортько Л. В.

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ СТИСНЕННЯ ЗОБРАЖЕНЬ У ФОРМАТІ PNG ЗА ДОПОМОГОЮ ЇХ РОЗБИТТЯ НА БЛОКИ ОДНОРІДНИХ РЯДКІВ

Пропонується спосіб розбиття зображень на блоки однорідних рядків пікселів з близькими значеннями ентропії та встановлення параметрів їх стиснення перед кодуванням у форматі PNG. Наведено фрагменти програм мовою С для оцінки ентропії рядків пікселів безпосередньо та після застосування контекстно-залежного алгоритму. Як показують експерименти, описаний аналіз дозволяє, наприклад, зменшити коефіцієнти стиснення природних зображень набору Kodak True Color Image у форматі PNG в середньому до 12,41 bpp.

Вступ. У сучасному світі формат PNG [1] є одним з основних для збереження зображень без втрат. В Інтернеті, наприклад, нараховується понад 15 млн. сторінок, що містять зображення у цьому форматі. Популярності формату PNG сприяють, насамперед, прийнятні коефіцієнти стиснення (КС) та висока швидкість декодування. Проблема зменшення розмірів файлів зображень у цьому форматі є актуальною на сьогодні і залишатиметься такою в найближчому майбутньому, оскільки навіть часткове її розв'язання дає змогу прискорити завантаження PNG-файлів з мережі та підвищити ефективність використання дискового простору. У цій статті описується спосіб розбиття зображень на блоки однорідних рядків та визначення прогнозованих параметрів компресії кожного блоку перед стисненням у форматі PNG, що дозволяє суттєво покращити КС.

Принципи стиснення зображень у форматі PNG. Постановка задачі. 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. У PNG-файлах, що використовуються на сьогодні, стиснуті дані зберігаються у відокремлених блоках згідно формату словникового стиснення DEFLATE [2]. Відповідно до цього формату, у стиснутих блоках

© О. В. Шпортько, Л. В. Шпортько, 2010

містяться результати застосування до вхідного потоку алгоритму LZH [3], згідно з яким результати контекстно-залежного словникового алгоритму LZ77 [4] стискаються контекстно-незалежними кодами Хафмана [5]. Розглянемо етапи обробки даних зображень у форматі PNG детальніше.

1. Контекстно-залежний алгоритм LZ77. Описуючи словникові алгоритми, фіксовану кількість попередніх закодованих неподільних елементів (літералів) вхідного потоку називають словником, а наступних незакодованих – буфером. Алгоритм LZ77 базується на заміні у вихідному потоці послідовності чергових літералів буфера посиланням на аналогічну послідовність літералів, що починається у словнику, у вигляді пари чисел <*довжина; зміщення від закінчення словника*>. У випадку відсутності аналогічної послідовності літералів у словнику, перший літерал буфера переноситься у вихідний потік без змін. Після цього закодовані літерали переносяться з початку буфера в кінець словника і кодування продовжується аналогічно аж до закінчення літералів вхідного потоку. Наприклад, потік кодів байтів пікселів *36 38 35 35 36 38 35 36 36 38 35 38 36 38 35 35 28* в закодованому вигляді можна записати так: *36 38 35 35 <3; 4> 36 <3; 4> 38 <4; 12> 28.* Очевидно, що для досягнення стиснення довжина співпадаючої послідовності має перевищувати 2. Оскільки більші зміщення кодуються, як правило, більшою кількістю бітів, то співпадаючі послідовності шукаються у словнику з кінця справа наліво.

Під час декодування кодів, отриманих за алгоритмом LZ77, окремі літерали копіюються у вихідний потік без змін. Пари ж <*довжина; зміщення*> декодуються шляхом послідовного копіювання з кінця вихідного потоку за вказаним зміщенням в кінець вихідного потоку необхідної кількості літералів. Природно, що алгоритм декодування має розрізняти окремі літерали та групи <*довжина; зміщення*>. Згідно з алгоритмом LZH, у форматі DEFLATE з цією метою довжини замін та окремі літерали кодуються разом числами в межах [0; 285]. При цьому числа з діапазону [0; 255] відповідають кодам окремих літералів, 256 позначає закінчення блоку, 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 визначають довжину заміни. Зміщення зберігається після відповідної довжини аналогічно – у вигляді базового значення та додаткових бітів. Базове значення зміщення знаходиться в межах [0; 29]. У форматі DEFLATE максимальне значення довжини закодованої послідовності може сягати 258, а зміщення – 32768.

На практиці для формування розкладу LZ77 найчастіше використовують такі підходи:

1.1. Жадібний розклад (greedy parsing). При цьому підході на кожному кроці алгоритму для чергових літералів буфера шукають *найдовшу* співпадаючу послідовність, що починається у словнику. Цей підхід забезпечує максимальну швидкість формування розкладу, оскільки зі словника в буфер щоразу переноситься максимальна кількість літералів, але не найкращі коефіцієнти стиснення, так як максимальні довжини замін призводять до використання максимальних зміщень та, крім цього, не відслідковується можливість використання літерала і заміни замість послідовності двох замін. Ми використовуємо жадібний розклад лише під час попереднього аналізу зображень.

1.2. Лінивий розклад (lazy parsing). У випадку використання цього підходу максимальну довжину співпадаючої послідовності для чергової позиції *порівнюють* з аналогічною довжиною для наступної позиції. Якщо максимальна довжина співпадаючої послідовності з наступної позиції довша, то з чергової позиції кодують не заміну, а літерал. Цей розклад формується довше від жадібного, оскільки виконує вдвічі більше пошуків співпадаючих послідовностей, хоча й забезпечує кращі КС. Ми використовуємо модифікований лінивий розклад для стиснення зображень у форматі PNG, порівнюючи між собою не максимальні довжини замін, а прогнозований коефіцієнт стиснення чергової заміни з прогнозованим КС літерала та заміни від наступної позиції (КС розраховуються з відношення прогнозованої кількості бітів для кодування до загальної кількості закодованих бітів). Якщо другий з цих КС менший від першого, то з чергової позиції кодуємо літерал, інакше – кодуємо заміну.

1.3. Майже оптимальний розклад (almost optimum parsing). При цьому підході для кожної позиції потоку *j* обчислюють *мінімальну* прогнозовану *вартість* (кількість бітів) кодування *Vj* від початку потоку до цієї позиції включно за формулою $V_j = \min_{l<j}\left(V_l + price_{l+1,j}\right)$, де $price_{l+1,j}$ – прогнозована вартість зберігання пари <*довжина; зміщення*> для кодування літералів з *l+1* по *j*-й, якщо *l<j-1* (вартість нескінченна, коли пари заміни для кодування цих елементів не існує) або прогнозована вартість зберігання літерала s_j, якщо *l=j-1*. Після визначення мінімальної вартості кодування останнього літерала потоку відбирають позиції та зміщення, що дозволяють досягнути цієї мінімальної вартості, і саме з них створюється закодований потік. Цей розклад формується найдовше з розглянутих, оскільки потребує пошуків

співпадаючих послідовностей з усіх позицій потоку, але й забезпечує найкращі КС. Ми використовуємо цей розклад як альтернативу лінивому для стиснення зображень у форматі PNG.

Зрозуміло, що для обчислення прогнозованих КС лінивого розкладу та вартостей кодувань майже оптимального розкладу необхідно знати приблизні кількості біт, що будуть витрачені для кодування літералів, довжин замін та зміщень. Вгадати наперед ці значення неможливо. Більше того, як буде показано далі, прогнозовані кількості бітів кодів можуть відрізнятися для різних блоків рядків. Тому ці значення розраховуються в процесі попереднього аналізу зображень.

2. Контекстно-незалежне кодування Хафмана. Ідея використання динамічних кодів Хафмана, що найчастіше застосовуються після виконання алгоритму LZ77 для стиснення літералів і базових значень довжин а також для відокремленого стиснення базових значень зміщень, полягає у заміні чисел з більшою частотою (тут і надалі – абсолютною) кодами меншої кількості бітів, ніж для чисел з меншою частотою. Коди Хафмана для літералів/довжин та зміщень визначаються для кожного блоку стиснутих даних окремо, що сприяє покращенню стиснення.

Згідно теореми Шеннона, елемент s_i з ймовірністю появи $p(s_i)$ найвигідніше кодувати $-\log p(s_i)$ бітами (тут і надалі логарифм береться за основою 2). Тоді середня довжина коду елемента після застосування контекстно-незалежного алгоритму має наближатися до *ентропії джерела* [3]

$$H = -\sum_i p(s_i) \times \log\left(p(s_i)\right). \qquad (1)$$

Ентропія джерела зменшується при збільшенні нерівномірності розподілу ймовірностей між елементами. Середня довжина коду Хафмана співпадає з ентропією джерела лише тоді, коли для всіх елементів s_i довжини їх оптимальних кодів $-\log p(s_i)$ цілі. Чим більше $-\log p(s_i)$ відхиляються від цілих чисел, тим більшою стає різниця між середньою довжиною коду Хафмана і ентропією джерела.

Алгоритм генерування динамічних кодів Хафмана відомий ще з середини минулого століття [5] і зменшити його КС не видається можливим. Ми лише намагатимемося розбити зображення на блоки рядків пікселів з близькими значеннями ентропії. Це дозволить відокремити і локалізувати рядки з високою ентропією (межі фрагментів зображення або значні перепади яскравостей пікселів), підвищивши тим самим нерівномірності розподілу в суміжних блоках, і тому зменшити КС.

3. Різновиди предикторів у форматі PNG. Зменшити ентропію при обробці зображень у форматі PNG намагаються також 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 відповідних компонент сусідніх пікселів. У процесі використання цієї технології обчислюють і надалі кодують відхилення чергової компоненти від прогнозованого предиктором значення. Тому, у загальному випадку, процес 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 обраної компоненти.

Оскільки сусідні пікселі зображення мають, як правило, близькі кольори і тому близькі значення відповідних компонент, то, згідно (2), часто значення предиктора буде співпадати 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 і тому зменшує ентропію джерела (згідно (1)), а, отже, і довжину закодованої послідовності. Таким чином, предиктори хоча й не виконують безпосереднє стиснення даних, але зменшують, насамперед, КС контекстно-незалежного алгоритму [6]. Крім цього, окремі предиктори можуть розбити існуючі або згенерувати нові повтори фрагментів зображення і цим вплинути на КС алгоритму LZ77, про що буде сказано далі.

Для однозначності декодування предиктори не можуть використовувати значення, що обходяться після чергового елемента. Оскільки елементи зображення найчастіше обходять по рядках зверху вниз, а в кожному рядку – зліва направо, то предиктори не можуть використовувати значення правіше та нижче від чергового елемента. У стиснутих блоках формату PNG даним кожного рядка передує окремий

байт, що визначає предиктор, який застосовується до компонент всіх його пікселів. На сьогодні форматом передбачено п'ять можливих значень цього байта [6], що визначає чотири різні предиктори: 0 – дані рядка не обробляються предикторами; 1 – LeftPredict; 2 – AbovePredict; 3 – AveragePredict; 4 – PaethPredict. Для опису цих предикторів позначимо значення сусідніх елементів для елемента X згідно схеми:

LeftAbove	Above
Left	X

Тоді предиктори формату PNG мовою C записуються так:

```
char LeftPredict(char Left, char Above, char LeftAbove)
 {return Left; }
char AbovePredict(char Left, char Above, char LeftAbove)
 {return Above; }
char AveragePredict(char Left, char Above, char LeftAbove)
 {return (Left+Above)/2; }
char PaethPredict(char Left, char Above, char LeftAbove)
 {int pp=Left+Above-LeftAbove;
  int pa,pb,pc;
  pa=abs(pp-Left);
  pb=abs(pp-Above);
  pc=abs(pp-LeftAbove);
  if (pa<=pb && pa<=pc) return Left;
  else if (pb<=pc) return Above;
       else return LeftAbove; }
```

Проблема вибору предикторів для окремих рядків пікселів залишалася **невирішеною** до останнього часу [6]. У цій статті ми пропонуємо в процесі аналізу зображення обирати ті предиктори, які мінімізують коефіцієнт стиснення кожного блоку, а не лише ентропію компонент пікселів.

Аналіз впливу різних варіантів предикторів на стиснення зображень у форматі PNG. Алгоритми, що використовуються у форматі PNG для стиснення даних, орієнтовані на зменшення різних видів надлишковостей: якщо LZ77 усуває надлишковості між однаковими фрагментами даних, то кодування Хафмана орієнтоване на зменшення надлишковостей між переважаючими елементами розподілів. Алгоритм LZ77 у цьому форматі може стискувати дані максимум у 129, а кодування Хафмана – максимум у 8 разів. Саме тому алгоритм LZ77 і

використовується у форматі PNG, хоча він і зменшує нерівномірність розподілу між окремими елементами і може зменшити ефективність кодування Хафмана. З іншого боку, у природних зображеннях яскравості компонент сусідніх пікселів, як правило, близькі, але не однакові. Це робить малоефективним використання для них алгоритму LZ77, і тому кодування Хафмана у цьому випадку забезпечує хоча б задовільні КС. Ключову роль в процесі стиснення різних зображень і навіть окремих їх фрагментів можуть відігравати як алгоритм LZ77, так і кодування Хафмана. Різні предиктори (чи відмова від їх використання) формату PNG орієнтуються на різні алгоритми стиснення. Відповідно, для кожного рядка зображення слід обирати той предиктор, який дозволить підсилити дію ключового алгоритму стиснення і допоможе цим забезпечити найменший загальний КС.

Дослідимо вплив різних варіантів предикторів на прикладі стиснення 24 різнотипних 24-бітних природних зображень стандартного набору файлів Kodak True Color Images (KTCI) у форматі PNG. Завантажити ці файли можна, наприклад, з http://r0k.us/graphics/kodak/-index.html. Тестування проводилося на комп'ютері з процесором AMD-K6 з частотою 300 МГц за допомогою програми з CD до [6] з жадібним розкладом LZ77, у яку були внесені такі модифікації: забезпечена можливість виходу зі словника в буфер під час кодування повторів; запроваджений розглянутий алгоритм аналізу альтернативних стиснутих блоків; застосований 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, де показником компресії файлів обрано КС, виражений в bpp, тобто у кількості біт, що в середньому витрачаються для кодування одного піксела зображення.
Стиснення зображень у форматі PNG без застосування предикторів відбувається за рахунок повторень та наявності нерівномірності розподілу значень яскравостей пікселів. Але в природних зображеннях аналогічні фрагменти та однакові значення яскравостей компонент трапляються порівняно рідко, тому й таке стиснення призводить до високих КС (стовпець 2 в табл. 1). Стиснення зображень без застосування предикторів ефективне, насамперед, для дискретно-тонових штучних зображень з високою роздільною здатністю. Середній час компресії файлів набору KTCI (тут і надалі – на комп'ютері з процесором AMD-K6 з частотою 300 МГц) цим способом складає 8 с.

Табл. 1.Коефіцієнти стиснення файлів зображень набору KTCI у форматі PNG після застосування різних варіантів предикторів рядків, с

№ файла	Без предикторів	Left Predict	Right Predict	Entropi Predict	EntropiLZ77 Predict
01	14.95	14.90	15.22	14.70	14.51
02	12.80	13.03	12.84	12.70	12.53
03	11.03	10.35	11.20	11.16	10.97
04	14.32	13.18	12.95	13.07	12.99
05	17.65	16.03	16.19	15.82	15.82
06	13.70	12.70	14.34	12.88	12.76
07	12.78	11.49	12.30	11.82	11.80
08	18.03	17.09	16.13	15.55	15.55
09	12.14	11.93	12.11	11.93	11.72
10	13.36	12.43	12.39	12.14	11.93
11	13.55	12.72	13.55	12.72	12.39
12	11.61	10.95	11.91	11.61	11.30
13	17.88	16.69	17.32	16.80	16.76
14	15.86	14.15	15.11	14.59	14.40
15	13.59	12.80	12.30	12.57	12.24
16	11.30	10.99	12.61	11.39	11.30
17	13.26	12.36	12.47	12.68	12.47
18	17.76	16.15	15.80	15.69	15.72
19	14.38	13.78	13.74	13.38	13.38
20	10.72	10.16	10.66	10.20	10.26
21	13.78	13.07	14.13	13.28	13.18
22	16.80	15.01	14.40	14.30	14.26
23	15.17	12.14	11.59	11.49	11.47
24	15.22	14.53	13.99	14.20	14.03
Average	**14.24**	**13.28**	**13.55**	**13.19**	**13.07**

Предиктор LeftPredict генерує горизонтальні прирости яскравостей між компонентами сусідніх пікселів. Однакові прирости яскравостей трапляються в природніх зображеннях, як правило, частіше, ніж однакові фрагменти, що покращує КС алгоритму LZ77. Крім цього, застосування предиктора LeftPredict підвищує нерівномірність розподілу навколо нуля, зменшуючи цим самим КС кодування Хафмана. Саме тому середній КС зображень набору KTCI з застосуванням LeftPredict зменшився порівняно з варіантом без застосування предикторів майже на 1 bpp (стовпець 3 з табл. 1). Використання предиктора LeftPredict ефективне також для штучних зображень з рівномірними приростами

яскравостей компонент пікселів по горизонталі (наприклад, у фоні). Але застосування LeftPredict здатне зменшувати довжини повторень, що були між фрагментами оригіналу зображення, тому може і підвищити загальний КС (дані зображення № 2 в табл. 1). Отже, однозначно стверджувати, що стиснення з використанням LeftPredict ефективніше від стиснення без застосування предикторів неможна. Середня тривалість компресії файлів набору KTCI цим способом складає 9 с.

Аналогічно впливає на стиснення зображень і застосування предиктора RightPredict, який генерує вертикальні прирости яскравостей компонент сусідніх пікселів. Різні зображення мають різний рівень відмінностей пікселів по горизонталі та вертикалі. Саме тому КС в стовпцях 3 і 4 табл. 1 різні. Гірший середній КС предиктора RightPredict стосовно LeftPredict пояснюється тим, що однакові прирости по вертикалі зустрічаються, як правило, в сусідніх рядках. Зміщення ж між сусідніми пікселами по вертикалі при обробці даних у форматі PNG дорівнює довжині рядка. Тобто вони значно більші від зміщень між сусідніми пікселами по горизонталі і тому кодуються значно більшою кількістю додаткових біт.

Предиктор AveragePredict врівноважує вплив сусідніх пікселів по горизонталі та вертикалі. Він добре прогнозує яскравості компонент пікселів для зображень великих та розмитих об'єктів, хоча й дуже чутливий до різких перепадів кольорів (наприклад, зображень променів світла на об'єктах). Ще кращі в середньому прогнози значень пікселів генерує предиктор PaethPredict, який однаково добре працює як на дискретно-тонових, так і на неперервно-тонових зображеннях. Але, по-перше, він обчислюється найдовше з усіх розглянутих предикторів, що негативно впливає як на час кодування так і декодування, і, по-друге, цей предиктор, як і попередній, часто зменшує довжини повторень, що були між фрагментами оригіналу зображення.

Отже, алгоритм LZ77 досягає кращих КС зображень без дії предикторів або після дії предикторів LeftPredict чи RightPredict. Кодування ж Хафмана найкраще стискає розподіли зображень після дії предикторів (особливо після AveragePredict та PaethPredict), оскільки вони підвищують нерівномірність розподілу. Обрати найкращий з трьох варіантів використання предикторів стосовно алгоритму LZ77 для кожного рядка пікселів неможливо без виконання трьох попередніх

розкладів зображення з застосуванням кожного з цих варіантів, оскільки однакові фрагменти даних можуть зустрічатися в різних рядках. **Найкращий же предиктор стосовно розкладу Хафмана для кожного рядка пікселів оберемо з умови мінімуму ентропії серед результатів їх дій**, до якої прямує середня довжина коду цього розкладу. Справді, нехай кожен з елементів S_i зустрічається N_i разів в рядку довжиною $N = \sum_i N_i$. Тоді $p(s_i) = N_i / N$ і загальна довжина закодованих даних, враховуючи (1), наближається до значення

$$N \times H = N \log(N) - \sum_i N_i \log(N_i). \qquad (3)$$

Мовою С підпрограма для такого наближеного обчислення ентропійного розміру блоку кодів Хафмана за частотами елементів з визначенням прогнозованого КС має вигляд:

```
double  sizeEntropiCode(long * freq, short countAllFreq,
          double &entropiKC)
 {// freq - масив частот елементів
  // countAllFreq - загальна кількість частот в масиві
  double size=0; long n=0;
  for (long i=0; i<countAllFreq; i++)
   {n+=freq[i]; // сумуємо частоти
    if (freq[i]>1) // для існування log
     size+=freq[i]*log(freq[i]); }
  if (n) {size=(n*log(n)-size)/log(2); entropiKC=size/(8*n); }
  else {size=0; entropiKC=1; }
  return size; }
```

А фрагмент програми для визначення номера предиктора, що породжує дані з найменшою ентропією елементів записується так:

```
for (i=0; i<5; ++i) // цикл по предикторах
 {memset(freq, 0, sizeof(freq));
  for (j=0; j<row_width; ++j) // цикл по елементах рядка
   freq[buffers[i][j]]++; // накопичення частот елементів
  size=sizeEntropiCode(freq, 256, entropiKC);
  if (size<minSize)
   {predict1=i; // запам'ятали номер предиктора
    minSize=size; entropiKC1=entropiKC; }}
KC1=entropiKC1;
```

Середній КС зображень набору KTCI з застосуванням ентропійного способу вибору предикторів рядків (EntropiPredict) зменшився порівняно з предиктором LeftPredict на 0.09 bpp (стовпець 5 з табл. 1), хоча покращення спостерігається і не для всіх зображень (№№ 4, 15, 17, 24). Використання цього способу ефективне, насамперед, для природних зображень декількох великих об'єктів. У фрагментах зображень, в яких ентропійний спосіб вибору предикторів рядків є найефективніший, короткі заміни (3-4 літерали), як правило, не застосовуються. Середня тривалість компресії файлів набору KTCI з застосуванням ентропійного способу вибору предикторів складає 10 с. Оптимальними предикторами при такому способі вибору найчастіше є AveragePredict чи PaethPredict.

Крім ентропії окремих елементів, для кожного рядка пікселів слід оцінити також КС після виконання коротких замін. Адже короткі заміни можуть суттєво збільшити частоти довжин замін, що їм відповідають, та зменшити частоти окремих літералів. А це, в свою чергу, призводить до збільшення нерівномірності розподілу і, відповідно, до зменшення КС. Виконувати попереднє стиснення LZ77 для результатів дії всіх предикторів рядків недоцільно, оскільки це призводить до різкого сповільнення процесу кодування. Ми оцінювали КС рядка пікселів після застосування триелементних замін за допомогою хеш-таблиці по чотирьох останніх бітах трьох суміжних елементів, в якій зберігаються абсолютні позиції останніх входжень подібних груп. Використовуючи значення цієї таблиці, імітується розклад LZ77 з підрахунком частот літералів, триелементних замін та їх зміщень і додаткових бітів, після чого визначається загальний КС рядка. Мовою С фрагмент програми для визначення номера предиктора, що породжує дані з найменшим КС після коротких замін записується, наприклад, так:

```
for (i=0; ii<5; ++i) // цикл по предикторах
 {memset(freq, 0, sizeof(freq)); // частоти літералів/довжин
  memset(freqD, 0, sizeof(freqD)); // частоти баз зміщень
  memset(hash, 0, sizeof(hash)); // елементи хеш-таблиці
  plusBit=0; // додаткові біти зміщень
  j=0; // позиція обробки рядка після дії предиктора i
  while (j<row_width-2)
   if (hash[((buffers[i][j] & 15)<<8)+
```

```
            ((buffers[i][j+1] & 15)<<4)+
            (buffers[i][j+2] & 15)])
   {// подібні три байта вже були в рядку
    freq[257]++; // частота триелементної заміни
    offset=j-hash[((buffers[i][j] & 15)<<8)+
    ((buffers[i][j+1]&15)<<4)+(buffers[i][j+2]&15)]+1;
    // збільшуємо частоту бази зміщення
    freqD[codesDistance[offset]]++;
    // накопичуємо додаткові біти зміщення
    plusBit+=extrasDistance[codesDistance[offset]];
    // дані оброблених триелементних груп
    // записуємо в хеш-таблицю
    hash[((buffers[i][j] & 15)<<8)+
     ((buffers[i][j+1] & 15)<<4)+(buffers[i][j+2] & 15)]=j+1;
    if (j+1<row_width-2)
     hash[((buffers[i][j+1] & 15)<<8)+
     ((buffers[i][j+2] & 15)<<4)+(buffers[i][j+3] & 15)]=j+2;
    if (j+2<row_width-2)
     hash[((buffers[i][j+2] & 15)<<8)+
     ((buffers[i][j+3] & 15)<<4)+(buffers[i][j+4] & 15)]=j+3;
    j+=3; }
   else // подібна група відсутня – заносимо літерал
    {freq[buffers[i][j]]++;
      hash[((buffers[i][j] & 15)<<8)+
       ((buffers[i][j+1] & 15)<<4)+(buffers[i][j+2] & 15)]=j+1;
      j++; }
   for (; j<row_width ; ++j) //останні позиції рядка
    freq[buffers[i][j]]++;
   size=sizeEntropiCode(freq, 256, entropiKC)+
         sizeEntropiCode(freqD, 30, entropiKCD)+plusBit;
  if (size<minSize)
   {predict2=i; // запам'ятали номер предиктора
    minSize=size; entropiKC2=entropiKC; }}
KC2=(double)minSize/(8*row_width);
```

Середній КС зображень набору KTCI з застосуванням ентропійного способу вибору предикторів рядків після коротких замін LZ77 (EntropiLZ77Predict) зменшився порівняно з середнім КС

ентропійного поелементного предиктора на 0.12 bpp (стовпець 6 з табл. 1), хоча покращення спостерігається і не на всіх зображеннях (№№ 18, 20). Це й не дивно, адже розрахунок ентропійного КС після дії коротких замін найкраще з розглянутих варіантів моделює стиснення у форматі PNG. Середня тривалість компресії файлів набору KTCI з застосуванням таких предикторів складає 11 с. Оптимальними предикторами при виборі за мінімальним ентропійним КС після застосування коротких замін LZ77 найчастіше виявляються LeftPredict чи RightPredict. Використання цього способу ефективне, насамперед, для природних зображень з багатьма об'єктами.

Отже, для кожного блоку однорідних рядків зображення предиктори слід обирати з умови мінімуму розрахованих за допомогою (3) КС серед п'яти варіантів компресії: без застосування предикторів, з використанням LeftPredict, з застосуванням RightPredict, з використанням ентропійного поелементного способу вибору предикторів та з застосуванням ентропійного способу вибору предикторів після здійснення коротких замін LZ77. При цьому слід враховувати також відхилення від варіантів компресії, що забезпечують мінімальний КС для попереднього і наступного блоків, адже зміна варіантів компресії між сусідніми блоками призводить до виникнення нового стиснутого блоку (а, отже, і його заголовку) і зменшує ймовірність повторень фрагментів даних для алгоритму LZ77.

Аналіз зображень з розбиттям на блоки однорідних рядків. Звичайно, кожен рядок зображення можна опрацювати найефективнішим для нього предиктором і стиснути в окремий блок. Але загалом таке стиснення не буде найефективнішим, оскільки, по-перше, алгоритм LZ77 в процесі стиснення чергового рядка використовує словник з даних попередніх рядків, тобто вибір предиктора рядка впливає на стиснення не лише цього рядка, а й найближчих наступних рядків, і, по-друге, в заголовку кожного стиснутого блоку динамічних кодів Хафмана міститься опис довжин кодів його розподілів літералів/довжин та зміщень, який може займати понад 1500 бітів, а такі витрати неприпустимі для кожного рядка. Тому ми пропонуємо в процесі попереднього аналізу зображень перед стисненням у форматі PNG **поєднати суміжні рядки з близькими ентропійними КС розподілів у блоки, визначити для кожного блоку**

найефективніший варіант компресії та відповідні предиктори рядків і розрахувати для цих блоків прогнозовані довжини кодів розподілів з метою подальшого ефективного кодування алгоритмом LZ77 з лінивим чи майже оптимальним розкладом.

Кожен однорідний блок рядків в процесі подальшого стиснення може належати одному, а може бути й розбитим на декілька стиснутих блоків, адже розмір стиснутого блоку, як правило, не перевищує 64 Кб. Але різні блоки рядків мають обов'язково належати різним стиснутим блокам, адже вони мають різні ентропійні КС, а отже, і різні нерівномірності розподілів. Визначити наперед, які заміни виявляться ефективними після застосування алгоритму LZ77 неможливо, тому ми в процесі розбиття на блоки рядків зорієнтуємося на ентропійні КС розподілів літералів/довжин. Отримувати цей КС для кожного рядка пікселів будемо за результатами порівняння мінімального поелементного КС (в попередніх фрагментах програм – КС1) та мінімального КС (КС2) після застосування коротких замін LZ77 стосовно результатів дії всіх предикторів, оскільки ці варіанти стиснення забезпечують в середньому найкращі результати. Враховуючи те, що довгі ефективні заміни частково враховані в КС2 і не враховані в КС1, обирати ентропійний КС розподілу літералів/довжин, що забезпечує КС2, будемо лише тоді, коли КС2 менший КС1 більше, ніж на 4%. Інакше обиратимемо ентропійний КС розподілу літералів/довжин, що забезпечує КС1:

```
if (KC2<KC1-0.04) currentEntropiKC=entropiKC2;
else currentEntropiKC=entropiKC1;
```

Для поєднання рядків пікселів у блоки класифікуємо їх за типами залежно від ентропійних КС розподілів літералів/довжин. Аналогічно, для визначення прогнозованих довжин кодів розподілів класифікуємо блоки рядків за типами залежно від прогнозованих КС алгоритму LZ77. Покроково алгоритм попереднього аналізу зображень перед стисненням у форматі PNG записується так:

1. Визначити для кожного рядка пікселів прогнозований ентропійний КС розподілу літералів/довжин, як описано вище.

2. На основі визначених ентропійних КС розділити рядки пікселів на вісім ентропійних типів, віднісши до першого типу рядки з ентропійним КС після коротких замін LZ77 з діапазону (0; 0.25], до

другого – з діапазону (0.25; 0.5], до третього – з діапазону (0.5; 0.66], до четвертого – з діапазону (0.66; 1]. Типи з п'ятого по восьмий мають аналогічні відповідні діапазони, але для ентропійних поелементних КС. Таким чином, з першим діапазоном пов'язуються рядки, що стискуються ентропійним алгоритмом у більш ніж чотири рази, з другим – від чотирьох до двох разів, з третім – від двох разів до 66 % початкового розміру. Останній з цих діапазонів дозволяє локалізувати рядки з високою ентропією (межі фрагментів зображень), що стискаються контекстно-незалежним алгоритмом менше, ніж на 1/3.

3. Виконати згладжування ентропійних типів, підвищивши номери типів для тих рядків, що знаходяться між двома рядками з однаковими більшими типами. Цей крок необхідний, насамперед, для штучних зображень, в яких рядки пікселів дублюються.

4. Поєднати в блоки суміжні рядки, що належать до одного типу. Приєднати рядки, що не належать жодному блоку до того з суміжних блоків, який має ближчий середній ентропійний КС. Поєднати малі та розбити великі блоки на мінімальні блоки рядків з розмірами від 8 до 32 Кб. Нижня межа розміру мінімального блоку рядків обумовлена необхідністю зберігання заголовків стиснутих блоків, а верхня – максимальним розміром словника алгоритму LZ77.

5. Визначити для кожного мінімального блоку рядків i після стиснення кожним з п'яти варіантів компресії j (0 – без застосування предикторів, 1 – з використанням LeftPredict, 2 – з застосуванням RightPredict, 3 – з використанням ентропійного поелементного способу вибору предикторів без врахування коротких замін LZ77 (3-4 елементи), 4 – з застосуванням ентропійного способу вибору предикторів після здійснення коротких замін LZ77) прогнозовані розміри $size_{ij}$ за допомогою алгоритму мінімізації розмірів стиснутих блоків. Запам'ятати також для кожного мінімального блоку рядків і кожного варіанту компресії кількість елементів розподілу літералів/довжин і КС їх кодами Хафмана. Для реалізації цього кроку до пікселів зображення необхідно почергово застосувати кожен з п'яти варіантів предикторів, виконати жадібний розклад LZ77 отриманих результатів та визначити мінімальні розміри стиснутих даних в межах кожного з мінімальних блоків рядків. Виконання цього кроку займає 50-71 % від загального часу стиснення, причому більша частина цього часу витрачається на формування п'яти додаткових розкладів LZ77.

6. Обрати для кожного мінімального блоку рядків i найефективніший варіант компресії $compres_i$ та відповідну кількість елементів розподілу літералів/довжин і ентропійний КС. Цей варіант компресії визначається послідовно для кожного мінімального блоку рядків з умови мінімуму прогнозованого розміру з врахуванням штрафів у випадку відхилення від варіанту компресії попереднього блоку рядків чи прогнозованого варіанту компресії наступного блоку. Дані штрафи обумовлені як необхідністю зберігання заголовка нового стиснутого блоку, що неодмінно виникає у випадку зміни варіанту компресії (біля 1500 бітів), так і змінами словників LZ77, що погіршує КС. Штраф від зміни варіанту компресії попереднього мінімального блоку рядків j на варіант компресії активного мінімального блоку рядків k обчислюється згідно з табл. 2 (значення матриці $fine_{jk}$).

Табл. 2. Штрафи за зміни варіантів компресії між суміжними блоками, бітів

№ варіанта компресії попереднього блоку	№ варіанта компресії активного блоку				
	0	1	2	3	4
0	0	2500	2500	2500	2500
1	8000	0	2000	2000	2000
2	8000	2000	0	2000	2000
3	8000	2000	2000	0	2000
4	8000	2000	2000	2000	0

Штраф у 8000 бітів за перехід від варіанта компресії з предикторами до варіанта без предикторів обумовлений відсутністю потрібного словника LZ77 для стиснення початку блоку рядків без предикторів, за рахунок якого в основному і відбувається його компресія. Штраф за зворотний перехід складає лише 2500 бітів, оскільки стиснення початку блоку рядків з предикторами відбувається, в основному, за рахунок використання кодів Хафмана, хоча такий перехід і змінює кардинально елементи словника LZ77. Штраф же за перехід між варіантами стиснення з предикторами складає лише 2000 бітів, оскільки такі переходи хоча й призводять до створення нових блоків, але не змінюють кардинально елементи словника LZ77.

Під час вибору варіанту компресії для наступного мінімального блоку рядків також будуть враховуватися як штрафи за зміну варіантів компресії, так і відхилення від мінімуму прогнозованого розміру. Тому

штраф за відхилення кожного варіанту чергового блоку від прогнозованого варіанту компресії наступного блоку визначається як мінімум серед сум штрафу за перехід та відхилення прогнозованого розміру від мінімального прогнозованого розміру для кожного варіанту компресії наступного блоку. Отже, для кожного мінімального блоку рядків *i* найефективніший варіант компресії знаходиться з умови:

$$compres_i = m \mid size'_{im} = \min_{j=0,4} size'_{ij},$$

де

$$size'_{ij} = size_{ij} + fine_{compres_{i-1},j} + \min_{k=0,4}\left(fine_{jk} + \left(size_{i+1,k} - \min_{l=0,4} size_{i+1,l} \right) \right).$$

7. Поєднати між собою суміжні мінімальні блоки рядків, що мають однаковий варіант компресії, якщо прогнозована довжина коду внаслідок поєднання цих блоків зростає не більше, ніж на 1500 бітів (у випадку перевищення цієї межі мінімальні блоки рядків доцільно зберігати в різних стиснутих блоках). Поєднання суміжних мінімальних блоків рядків дозволяє уникнути створення додаткових стиснутих блоків, а, отже, і зберігання їх заголовків (біля 1500 бітів).

Для аналізу довжин кодів блоків рядків знову зорієнтуємося на розподіли літералів/довжин замін. Нехай в блоці рядків *i* кількість елементів розподілу літералів/довжин становить LZ_i, а коефіцієнт їх стиснення кодами Хафмана дорівнює H_i. Прогнозована сумарна довжина кодів цих розподілів у відокремлених суміжних блоках *i* та *i+1* становить

$$H_i \times LZ_i + H_{i+1} \times LZ_{i+1}. \quad (4)$$

Внаслідок поєднання суміжних блоків рядків їх ентропія може зрости до максимальної і сумарна довжина коду становитиме

$$\max(H_i, H_{i+1}) \times (LZ_i + LZ_{i+1}). \quad (5)$$

Максимальне збільшення довжини коду обчислюється з різниці (5) та (4). Експериментально встановлено, що середнє збільшення довжини коду становить 75 % від максимального. Саме ця величина і порівнюється з 1500.

На практиці слід обчислити середнє збільшення довжини коду для всіх суміжних блоків та ітеративно поєднувати ті пари, де таке збільшення найменше, доки це збільшення не перевищує 1500. В

процесі поєднання суміжних блоків рядків слід сумувати також кількості елементів розподілу літералів/довжин, а ентропію поєднаного блоку слід усереднити для коректного розрахунку збільшень довжини коду з суміжними блоками:

$$H_{i,i+1} = \frac{H_i \times LZ_i + H_{i+1} \times LZ_{i+1}}{LZ_i + LZ_{i+1}}.$$

Для розрахунку можливості поєднань блоків, більших від мінімальних, до уваги слів брати максимум 32000 елементів розподілів літералів/довжин кожного з суміжних блоків, адже розмір блоку даних не повинен перевищувати 64 Кб. На цьому кроці завершується формування блоків однорідних рядків.

Розглянутий принцип поєднання між собою суміжних мінімальних блоків з близькими коефіцієнтами стиснення можна використати також для поєднання рядків з близькими коефіцієнтами стиснення в мінімальні блоки (замість кроків 2 – 4 цього алгоритму). Наприклад, поєднання рядків пікселів зображень набору KTCI у блоки за мінімальним КС після застосування коротких замін LZ77 забезпечує в середньому аналогічні результати поєднанню за ентропійними типами, відхиляючись по окремих зображеннях не більше, ніж на 1 Кб, хоча такі поєднання виконуються довше.

8. Визначити для кожного отриманого блоку рядків прогнозований КС алгоритму LZ77 як відношення кількості елементів розподілу літералів/довжин LZ_i до загальної кількості компонентів пікселів у блоці.

9. На основі визначених прогнозованих КС алгоритму LZ77 розбити блоки рядків на вісім типів LZ77, віднісши до першого типу блоки, що використовують короткі заміни, з КС в діапазоні (0; 0.125], до другого – в діапазоні (0.125; 0.25], до третього – в діапазоні (0.25; 0.5], до четвертого – в діапазоні (0.5; 1]. Типи з п'ятого по восьмий мають аналогічні відповідні діапазони, але для КС LZ77 без коротких замін. На практиці блоки рядків з КС LZ77 без коротких замін, що забезпечують стиснення у понад 4 рази зустрічаються рідко і мають близькі характеристики до аналогічних блоків з короткими замінами. Зсув діапазонів КС LZ77 стосовно ентропійних типів рядків пояснюється більшим можливим максимальним стисненням цього алгоритму. Отже, з першим діапазоном пов'язуються блоки, що стискуються алгоритмом

LZ77 у більше, ніж вісім разів, з другим – від восьми до чотирьох разів, з третім – від чотирьох до двох разів, з четвертим - менше, ніж у два рази.

10. За допомогою частот яскравостей компонентів та визначених типів стиснення LZ77 спрогнозувати для кожного блоку рядків довжини кодів Хафмана розподілів стиснутих блоків. Як показано вище, визначити наперед, які заміни виявляться ефективними, неможливо, тому довжину кожного елемента розподілу зміщень спрогнозуємо рівною 5 (середня довжина рівномірного коду цього розподілу). Прогнозування ж довжин елементів розподілів літералів/довжин замін будемо здійснювати з наступних міркувань: зрозуміло, що чим частіше окремий елемент зустрічається у блоці рядків, тим меншим буде його довжина коду. Якщо заміни LZ77 виконуються рідко, то довжини коду окремих елементів будемо прогнозувати рівними ентропійним, а довжини кодів замін – середнім довжинам рівномірного коду цього розподілу, тобто 8. Зі збільшенням кількості замін LZ77 довжини кодів окремих елементів збільшуються, а довжини кодів замін зменшуються. Заміни в основному виконуються завдяки елементам, які зустрічаються найчастіше. Тому збільшення максимального КС LZ77 між діапазонами удвічі призводить до зменшення прогнозованих довжин кодів замін і до збільшення прогнозованих довжин літералів на 1 біт. Максимальне збільшення прогнозованої довжини кодів літералів стосовно ентропійної дорівнює 3 (у 8 разів). Крім цього, для розподілів з короткими замінами і стисненням до 4 разів довжини кодів таких замін слід стосовно середніх довжин кодів замін зменшити, а для розподілів без коротких замін – збільшити. Прогнозовані довжини кодів коротких замін, кратних кількості компонент пікселів, для блоків з найбільшим КС LZ77 теж зменшуються (а не кратних – збільшуються), оскільки такі заміни виявляються ефективними найчастіше. Фрагмент програми мовою С для прогнозування довжин кодів розподілів пікселів з трьох компонент записується, наприклад, так:

```
// plusBitLenLiteral – додаткові біти до ентропійної
// довжини кодів літералів у кожному блоці
// lenZamina – прогнозована довжина кодів замін
k=0; // індекс першого елемента першого блоку
```

```
for (i=0; i<countBlockRow; i++) // цикл по блоках рядків
 {// визначення параметрів залежно від типу LZ-блоку
  switch (tupBlockRow[i])
   {case 1, 5: plusBitLenLiteral=3; lenZamina=5; break;
    case 2, 6: plusBitLenLiteral=3; lenZamina=6; break;
    case 3: plusBitLenLiteral=3; lenZamina=7; break;
    case 4: plusBitLenLiteral=2; lenZamina=8; break;
    case 7: plusBitLenLiteral=2; lenZamina=7; break;
    case 8: plusBitLenLiteral=1; lenZamina=8; break; }
  // визначення частот елементів у черговому блоці
  memset(freq, 0, sizeof(freq));
  l=nextPozStartBlockRow[i]-k; // к-ть елементів блоку
  k=nextPozStartBlockRow[i]; // початок наступного блоку
  while (pozImage++<k)
   freq[imageData[pozImage]]++; // накопмчення частот
  // визначення прогнозованих довжин кодів літералів
  // як суми ентропійної довжини і додаткових біт
  for (j=0; j<256; j++)
   if (freq[j])
    lenCodeLL[i][j]=-log2(freq[j]/l)+plusBitLenLiteral;
   else lenCodeLL[i][j]=0;
  // прогнозування довжин кодів баз довжин замін
  for (j=257; j<=285; j++)
   lenCodeLL[i][j]=lenZamina;
  // при максимальних КС орієнтуємося на цілі піксели
  if (tupBlockRow[i]==1 || tupBlockRow[i]==5)
   {// зменшуємо довжини кодів замін довжиною 3 і 6
    lenCodeLL[i][257]--; lenCodeLL[i][260]--;
    // збільшуємо довжини кодів решти коротких замін
    lenCodeLL[i][258]=lenCodeLL[i][259]=
    lenCodeLL[i][261]=lenCodeLL[i][262]=
    lenCodeLL[i][264]=lenCodeLL[i][266]=lenZamina+3; }
  // зменшуємо довжини кодів коротких замін з КС
  // до 4 разів для розподілів з короткими замінами
  if (tupBlockRow[i]==3 || tupBlockRow[i]==4)
   {lenCodeLL[i][257]=3; lenCodeLL[i][258]=3;
    lenCodeLL[i][259]=3; lenCodeLL[i][260]=4;
```

```
   lenCodeLL[i][261]=6; lenCodeLL[i][262]=6; }
  // збільшуємо довжини кодів коротких замін з КС
  // до 4 разів для розподілів без коротких замін
  if (tupBlockRow[i]==7 || tupBlockRow[i]==8)
   lenCodeLL[i][257]=lenCodeLL[i][258]=
   lenCodeLL[i][259]=lenZamina+3;
  // прогнозування довжин кодів розподілу зміщень
  for (j=0; j<=29; j++)
   lenCodeDistance[i][j]=5; }
```

Уточнити для кожного блоку рядків прогнозовані довжини кодів Хафмана розподілів літералів/довжин замін та зміщень після їх розрахунку (у випадку відсутності обмежень по часу) можна з розподілів, що утворюються за результатами додаткового проходу по даних цього блоку з використанням лінивого розкладу LZ77, кодування Хафмана та алгоритму мінімізації розміру стиснутих блоків.

Після виконання такого попереднього аналізу зображення здійснюється, власне, стиснення у форматі PNG кожного блоку рядків у відокремлені стиснуті блоки з використанням розрахованих довжин кодів елементів розподілів для лінивого чи майже оптимального розкладу LZ77.

Результати експериментів. На завершення розглянемо результати застосування описаного способу попереднього аналізу зображень для стиснення у форматі PNG файлів вже згаданого набору KTCI програмою з CD до [6] з описаними вище модифікаціями. Відповідні коефіцієнти стиснення наведено в табл. 3, а час компресії – в табл. 4. Результати аналізу застосовувалися для лінивого та майже оптимального розкладу LZ77 безпосередньо (відповідно, четвертий та шостий стовпці таблиць) та після додаткового проходу для уточнення прогнозованих довжин кодів (відповідно, п'ятий та сьомий стовпці). Крім цього, для порівняння ефективності стиснення у другому та третьому стовпцях таблиць наведено 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.

Табл. 3. Коефіцієнти стиснення файлів зображень набору KTCI у форматі PNG після застосування різних варіантів програм, bpp

№ файла	OptiPng стандарт	OptiPng макс.	lazy parsing	lazy parsing з проходом	Almost optimum parsing	Almost optimum parsing з проходом
01	14.70	14.65	14.05	14.05	13.72	13.61
02	12.57	12.55	12.22	12.20	12.01	11.91
03	11.01	10.22	10.20	10.20	9.95	9.89
04	12.97	12.84	12.59	12.59	12.45	12.34
05	15.76	15.74	15.65	15.63	15.49	15.42
06	12.93	12.59	12.53	12.51	12.26	12.20
07	11.51	11.34	11.28	11.26	11.07	10.99
08	15.44	15.42	15.36	15.36	15.15	15.13
09	11.86	11.80	11.20	11.18	10.93	10.78
10	12.07	12.05	11.72	11.70	11.51	11.43
11	12.68	12.61	12.18	12.16	11.91	11.82
12	11.57	10.80	10.68	10.66	10.41	10.26
13	16.86	16.71	16.55	16.51	16.32	16.24
14	14.55	14.07	14.03	13.99	13.78	13.70
15	12.47	12.20	12.03	12.01	11.84	11.72
16	11.30	10.87	10.82	10.82	10.53	10.39
17	12.59	12.24	12.07	12.05	11.80	11.72
18	15.61	15.59	15.53	15.49	15.38	15.34
19	13.43	13.41	13.05	13.01	12.82	12.72
20	10.18	10.01	9.95	9.93	9.78	9.68
21	13.45	12.95	12.70	12.68	12.45	12.30
22	14.28	14.26	14.05	14.03	13.93	13.84
23	11.34	11.32	11.20	11.18	11.09	10.95
24	14.11	13.93	13.82	13.78	13.61	13.53
Average	**13.14**	**12.92**	**12.73**	**12.71**	**12.51**	**12.41**

В середньому, для визначення предикторів рядків зображень набору KTCI витрачається 4 с, кожен жадібний розклад LZ77 займає 5 с, лінивий розклад LZ77 та додатковий прохід для уточнення прогнозованих довжин кодів – 6.5 с, майже оптимальний розклад LZ77 – 15 с. Аналізуючи дані табл. 1, 3 та 4, бачимо, що в середньому по набору KTCI:

- ентропійний вибір предикторів рядків після застосування коротких замін LZ77 (стовпець 6 з табл. 1) дозволяє досягнути кращих КС, ніж програма OptiPNG з параметрами стандартного перебору (стовпець 2 з табл. 3), витрачаючи при цьому в 2.91 рази менше часу;

Табл. 4. Час стиснення файлів зображень набору KTCI у форматі PNG різними варіантами програм, с

№ файла	OptiPng стандарт	OptiPng макс.	lazy parsing	lazy parsing з проходом	Almost optimum parsing	Almost optimum parsing з проходом
01	23	810	31	37	36	41
02	33	1090	36	45	45	52
03	45	1281	39	48	57	65
04	28	1030	35	44	44	52
05	21	791	32	38	36	43
06	33	1018	34	40	42	49
07	41	1487	38	47	53	60
08	24	482	31	37	35	40
09	36	706	35	42	42	49
10	36	679	36	45	48	53
11	36	596	33	39	43	49
12	38	686	36	43	46	52
13	20	433	30	35	32	37
14	24	484	31	36	35	41
15	35	618	37	45	47	56
16	31	636	35	40	43	49
17	31	650	36	43	45	51
18	21	484	32	38	35	41
19	28	526	36	43	43	49
20	57	813	41	49	58	66
21	29	830	35	41	41	46
22	24	940	34	41	39	46
23	37	822	42	52	54	64
24	29	626	34	40	42	48
Average	**32**	**772**	**35**	**42**	**44**	**50**

- використання описаного способу попереднього аналізу зображень перед лінивим розкладом LZ77 дає змогу отримати кращі КС від програми OptiPNG з параметрами максимального перебору на 0.19 bpp, а перед майже оптимальним розкладом LZ77 – на 0.41 bpp, витрачаючи для цього відповідно у 22 та 17 разів менше часу;
- застосування додаткового проходу для уточнення прогнозованих довжин кодів розподілів по обраних даних блоків у 5 разів ефективніше для майже оптимального розкладу LZ77 стосовно лінивого розкладу LZ77, оскільки ці довжини використовуються першим зі згаданих розкладів для аналізу ефективності стиснення з кожного літерала, а другим – лише для аналізу ефективності стиснення з позицій можливих замін;

- попередній аналіз зображень дозволив, ефективно використовуючи прогнозування довжин кодів розподілів та майже оптимальний розклад LZ77, зменшити КС файлів набору KTCI в середньому до 12.41 bpp (на 733 Кб стосовно загального розміру оригінальних файлів набору у форматі PNG).

Висновки з наведених результатів дослідження та тестування:

1. Для підвищення ефективності стиснення даних у форматах, що послідовно використовують алгоритми декількох методів, слід враховувати взаємний вплив цих алгоритмів. Під час попередньої обробки даних перед стисненням у таких форматах для розрахунку параметрів окремих алгоритмів стиснення доцільно використовувати не прогнозований загальний КС, а прогнозовані КС окремих алгоритмів;

2. Розглянутий спосіб попередньої обробки дозволяє отримати найменші КС зображень у форматі PNG серед відомих авторам програм за рахунок: поділу зображень на блоки рядків пікселів з близькими значеннями ентропії, вибору для кожного блоку рядків найкоротшого з альтернативних стиснутих блоків динамічних кодів Хафмана та ітеративного зменшення його розміру, визначення прогнозованих довжин кодів розподілів блоків для наступного LZ-стиснення;

3. Описаний спосіб попереднього аналізу зображень не вимагає модифікації декодера чи програм перегляду зображень і за рахунок зменшення розмірів файлів лише прискорює їх роботу. Саме тому розглянутий алгоритм мінімізації розмірів стиснутих блоків може бути ефективно використаний для компактного збереження даних у стандартах, що застосовують формат словникового стиснення DEFLATE, а алгоритм розбиття зображень на блоки рядків пікселів з близькими значеннями ентропії можна з користю застосовувати для покращення КС і в інших форматах, що використовують різні предиктори для окремих рядків пікселів.

1. *Boutell T.* PNG Specification. Version 1.0 / Boutell T., et. all // RFC 2083, Boutell. Com, inc. – March 1997. – 102 p.
2. *Deutsch P.* DEFLATE Compressed Data Format Specification version 1.3 / P. Deutsch // RFC 1951, Alladin enterprises. – May 1996. – 15 p.
3. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М. : ДИАЛОГ-МИФИ, 2003. – С. 17–106.

4. *Ziv J.* A universal algorithm for sequential data compression / Ziv J., Lempel A. // IEEE Transactions on Information Theory. – May 1977. – Vol. 23(3). – P. 337-343.
5. *Huffman D.* A Method for the Construction of Minimum Redundancy Codes / D. Huffman // Proceedings of the IRE. – Vol. 40(9). – P. 1098–1101.
6. *Миано Дж.* Форматы и алгоритмы сжатия изображений в действии: Учеб. пособ. / Дж. Миано. – М.: Триумф, 2003. – 336 с., ил. – (Серія: Практика программирования).
7. *Шпортько О. В.* Вибір найкоротшого з альтернативних стиснутих блоків динамічних кодів Хафмана у форматі PNG / О. В. Шпортько // Комп'ютинг. – 2009. – Т. 8, вип. 2. – С. 58-67.

Рівненський державний гуманітарний університет, м. Рівне
E-mail: chportko@yandex.ru
chportko@ukr.net
ДВНЗ "Рівненський коледж економіки та бізнесу", м. Рівне
E-mail: lchportko@yandex.ru

Надійшла 24.04.2010

Шпортько А. В., Шпортько Л. В. ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ СЖАТИЯ ИЗОБРАЖЕНИЙ В ФОРМАТЕ PNG ПРИ ПОМОЩИ ИХ РАЗБИЕНИЯ НА БЛОКИ ОДНОРОДНЫХ СТРОК // *Предлагается способ разбиения изображений на блоки однородных строк пикселей с близкими значениями энтропии и установки параметров их сжатия перед кодированием в формате PNG. Приведены фрагменты программ на языке С для оценки энтропии строк пикселей непосредственно и после применения контекстно-зависимого алгоритма. Как показывают эксперименты, приведенный анализ позволяет, например, уменьшить коэффициенты сжатия естественных изображений набора Kodak True Color Image в формате PNG в среднем до 12,41 bpp.*

Shport'ko A. V., Shport'ko L. V. INCREASE OF EFFICIENCY OF COMPRESSION OF IMAGES IN FORMAT OF PNG BY LAYING THEM OUT IN BLOCKS OF HOMOGENEOUS LINES // *The method of laying out of images in the blocks of homogeneous lines of pixels with the alike values of entropi and establishment of parameters of their compression before coding in the format of PNG is offered. The fragments of the programs in C the language for the estimation of entropi of lines of pixels directly during and after application of context-dependent algorithm are presented. As experiments show, the analysis described, allows, for example, to decrease the aspects of compression of natural images of Kodak True Color Image set in the format of PNG on the average to 12,41 bpp.*