DOI: https://doi.org/10.15407/kvt202.04.005

UDC: 519.72

АНІСІМОВ А.В., д-р. фіз.-мат. наук, член.-кор. НАН України,
декан факультету комп'ютерних наук та кібернетики
e-mail: anatoly.v.anisimov@gmail.com
ЗАВАДСЬКИЙ І.О., д-р. фіз.-мат. наук, доцент,
доцент каф-ри математичної інформатики
e-mail: ihorza@gmail.com
ЧУДАКОВ Т.С., студент
e-mail: timofey.chudakov@gmail.com
Факультет комп'ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка,
4д, пр. Глушкова, 03022, Київ, Україна

ЗАСТОСУВАННЯ МУЛЬТИРОЗДІЛЬНИКОВИХ КОДІВ ДО АРХІВУВАННЯ ПРИРОДНОМОВНИХ ТЕКСТІВ

Вступ. *Рівень ефективності сучасних архіваторів наблизився до теоретичної межі. Покращення коефіцієнту стиснення певного виду даних навіть на частки відсотка, у разі збереження розумного часу стискання й розпакування, вважається істотним досягненням. Актуальність цього дослідження визначається тим, що запропоновано новий спосіб кодування даних, який, зокрема, дає можливість досягти значно суттєвішого підвищення коефіцієнту стиснення текстів англійською та німецькою мовами.*

Метою *статті є вирішення проблеми немонотонності словника мультироздільникових кодів та дослідження доцільності використання мультироздільникового кодування на стадії попереднього оброблення природномовних текстів у процесі їхнього архівування.*

Результати. *Введено поняття реверсного мультироздільникового коду. Побудовано монотонне кодувальне, а також декодувальне відображення з множини натуральних чисел на множину кодових слів реверсного мультироздільникового коду. Досліджено ефективність застосування реверсних мультироздільникових кодів до стискання природномовних текстів. Запропоновано механізм оптимізації використання словника у процесі архівування природномовних текстів. Проведено експерименти, результати яких свідчать, що попереднє кодування англійських та німецьких текстів реверсними мультироздільниковими кодами та застосування запропонованого методу оптимізації словника дає змогу на 1–3% покращити граничну ефективність стискання, яка досягається найкращими архіваторами в режимі максимального стиснення.*

Висновки. *Реверсні мультироздільникові коди є ефективним засобом стиснення природномовних текстів, що, у разі окремого застосування, характеризується стійкістю до помилок, дає можливість виконувати пошук інформації у стисненому файлі, а також надає можливість його швидкого декодування. У разі застосування як засобу попереднього оброблення природномовного тексту для подальшого архівування, разом зі спеціальними механізмами оптимізації словника, реверсні мультироздільникові коди дають можливість підвищити ефективність найкращих з відомих на сьогодні архіваторів.*

Ключові слова: *стискання, стиснення, архівування, стискальні коди, мультироздільниковий код, реверсний мультироздільниковий код, оптимізація словника, природномовний текст.*

ВСТУП

Методи стискання даних можна поділити на два великі класи: методи стискання з втратами інформації та без таких втрат. Основною перевагою стискання з втратами є, як правило, значно вищий ступінь стиснення, а недоліком — неможливість відновити вхідне повідомлення без викривлень.

Для багатьох типів даних, таких як відео, звук, зображення, невеликий рівень викривлень може бути цілком допустимим і навіть непомітним для користувача. Однак до таких даних не належать тексти, записані природною мовою або мовою програмування. Тому стискання текстів — це одна з найважливіших сфер застосування методів стискання без втрат даних. Іншою, не менш важливою, сферою є універсальні архіватори, які повинні відновлювати інформацію без викривлень. Це дослідження знаходиться на перетині цих двох сфер, адже його метою є пошук способів покращити ефективність архіваторів у разі їхнього застосування до стискання природномовних текстів.

ПОСТАНОВКА ПРОБЛЕМИ

Стискання без втрат даних є добре дослідженою галуззю кібернетики. Згідно з класифікацію, наведеною в [1], майже всі відомі методи у цій галузі можна поділити на 4 різновиди залежно від того, фіксованою чи змінною є довжина кодованого повідомлення та відповідного йому коду.

1. Коди типу «стала-стала» перетворюють частини вхідного повідомлення фіксованої довжини на блоки коду фіксованої (найчастіше — такої саме) довжини. Власне, йдеться не про стискальні коди, а про допоміжні перетворення, що переставляють або перенумеровують символи вхідного повідомлення так, що подальше застосування методів стискального кодування стає ефективнішим. Одним з найвідоміших таких перетворень є перетворення Барроуза-Віллера [2], яке вкупі з перетворенням «перемістити на початок» (MTF, Move-To-Front) [3] використовується як початковий етап стискання в багатьох архіваторах та форматах даних.

2. Коди типу «стала-змінна» є найпоширенішими. Як правило, це методи частотного кодування. Вони кожному символу вхідного алфавіту ставлять у відповідність послідовність символів коду (кодове слово) змінної довжини. Як правило, метод кодування побудовано таким чином, що символам з вищою частотою надаються коротші кодові слова. До цього класу кодів, зокрема, належать коди Гаффмана [4], патерн-коди [5], серед

яких найвідомішими є коди Фібоначчі [6], та нещодавно запропоновані мультироздільникові коди [7]. Зазначимо, що термін «символ вхідного алфавіту» є певною абстракцією, деякою атомарною з огляду на метод кодування одиницею, яка насправді може складатися зі змінної кількості літер, цифр тощо. Наприклад, коли ми застосовуємо мультироздільникові коди до стискання природномовних текстів, символом вхідного алфавіту вважаємо слово, тобто послідовність символів між двома пробілами — такий підхід є суттєво ефективнішим за кодування окремих літер.

3. Якщо у методі кодування враховується внутрішня структура символів алфавіту, отримуємо коди типу «змінна-змінна», тобто коди, у яких повідомленням змінної довжини відповідають кодові слова змінної довжини. Такими є всі коди, що кодують послідовності чисел, базуючись лише на їхніх значеннях, зокрема коди Елайеса [8] і Голомба [9]. Водночас фактичні частоти чисел у повідомленні, що передається, не враховуються, однак робиться певне припущення про загальний закон їхнього розподілу, зокрема в усіх відомих випадках припускається, що менші числа мають вищі частоти. Інколи код можна віднести до цього або попереднього класу, лише проаналізувавши метод кодування. Зокрема, метод кодування кодів Фібоначчі, викладений у [10], та методи кодування мультироздільникових кодів [7] ставлять кодові слова у відповідність натуральним числам, а отже, дають змогу розглядати ці коди, як коди типу «змінна-змінна».

4. Коди типу «змінна-стала» перетворюють послідовності символів вхідного повідомлення змінної довжини на кодові слова фіксованої довжини. Найвідомішим представником кодів цього типу є коди родини Лемпеля-Зіва [11] та їхні вдосконалення [12]. Їхня ефективність базується на використанні інформації не лише про частоти окремих символів, але й про частоти послідовностей символів.

Основним показником якості архіватора є коефіцієнт стиснення, тобто відношення обсягів початкових файлів до обсягів стиснутих на тому чи іншому наборі даних. Високих коефіцієнтів стиснення, характерних для сучасних архіваторів, як правило, вдається досягти лише завдяки поєднанню методів 1, 2 і 4 типів, а також застосуванню модифікацій арифметичного кодування [13], яке не можна віднести до жодного з перелічених класів кодів, але яке забезпечує максимально високий коефіцієнт стиснення. Водночас арифметичне кодування має численні недоліки:

- нестійкість до помилок у закодованому повідомленні;
- неможливість пошуку даних у закодованому файлі;
- повільність декодування (у десятки разів повільніше порівняно з кодами Гаффмана);
- відносна складність реалізації.

Поєднати стискальну ефективність арифметичного кодування зі швидкістю декодування, порівняною зі швидкістю декодування кодів Гаффмана, вдалося завдяки кодуванню, яке базується на понятті асиметричних систем числення (ANS) [14], однак усі інші з перелічених недоліків залишаються властивими і для цього методу, а також для кодів Гаффмана і Лемпеля-Зіва.

Коди типу 3 («змінна-змінна») у стисканні природномовних текстів використовуються рідше, оскільки забезпечують коефіцієнт стиснення гірший,

ніж коди типу 2 або 4. Однак саме до цього класу належать коди, що не мають, як показано в [7], жодного з перелічених вище недоліків — мультироздільникові коди. Зниження коефіцієнта стиснення для цих кодів у багатьох випадках може розглядатися як цілком прийнятне — 2–4 % порівняно з арифметичним кодуванням.

Проте мультироздільникові коди мають інший істотний недолік — немонотонність словника. Цю проблему детально досліджено далі. Також запропоновано модифікацію мультироздільникових кодів — реверсні мультироздільникові коди, які усувають згадану проблему, зберігаючи всі переваги «прямих» мультироздільникових кодів. Однак, мабуть, найцікавішими результатами цього дослідження є експериментальні дані. Вони свідчать про те, що якість стискання природномовних текстів архіватором може бути покращено, якщо спочатку закодувати текст «недостатньо потужним» реверсним мультроздільниковим кодом, а вже потім заархівувати отриманий файл. Але цього вдається досягти, лише якщо виконати певне передоброблення словника, яке можна класифікувати як метод типу 1, «стала-стала».

МУЛЬТИРОЗДІЛЬНИКОВІ КОДИ І ПРОБЛЕМА НЕМОНОТОННОСТІ СЛОВНИКА

Припустимо, що $M=\{m_1, \ldots, m_t\}$ — послідовність цілих чисел, яка зростає: $0 < m_1 < \ldots < m_t$. Дамо означення мультироздільникового коду коду D_M, позначаючи через 1^x послідовність з x одиничних бітів.

Означення 1. Мультироздільниковим кодом називають код D_M, який містить усі слова вигляду $1^\alpha 0$, $\alpha \in M$, і всі слова, що задовольняють таким умовам:

- слово не починається з послідовності $1^\alpha 0$, $\alpha \in M$;
- слово закінчується послідовністю $1^\alpha 0$, $\alpha \in M$;
- слово не містить послідовності $1^\alpha 0$, $\alpha \in M$ ніде, крім закінчення.

Разом з патерн-кодами [5], мультироздільникові коди належать до класу кодів з роздільниками — спеціальними послідовностями бітів, якими закінчуються всі кодові слова, але які не можуть трапитися всередині кодових слів. Кожен патерн-код має один роздільник, а мультироздільникові коди можуть мати їх кілька — це і є характерною рисою, що вирізняє такі коди з-поміж усіх інших. Точніше, роздільниками у коді D_M є послідовності вигляду $01^\alpha 0$, $\alpha \in M$, однак код містить також слова вигляду $1^\alpha 0$, $\alpha \in M$, які утворюють роздільник разом з останнім нулем попереднього кодового слова у закодованому тексті. У [15] пояснено, які переваги має така структура кодових слів порівняно з іншими відомими патерн-кодами, такими як коди Фібоначчі, а у [7] доведено такі теоретичні властивості мультироздільникових кодів, як універсальність та повнота, обчислено їхню асимптотичну щільність, описано методи кодування та декодування, а також експериментально досліджено ефективність цих кодів у стисканні природномовних текстів.

Що стосується кодування/декодування мультироздільникових кодів, то ці процедури може бути описано схемою, характерною для всіх методів типу 2 і 3, згідно з наведеною вище класифікацією. А саме, під час кодування використовується ***словник*** (*слово тексту*, *кодове слово*), у

якому слова тексту відсортовано за спаданням частот, а кодові слова відсортовано за зростанням довжин. Під час декодування застосовується зворотна процедура: потрібно побудувати відображення з множини кодових слів на множину слів тексту. З метою часової оптимізації декодування слова вихідного тексту варто зберігати за допомогою структури даних, що забезпечує швидкий доступ до своїх елементів, а найефективнішою з таких структур є масив із числовими індексами. Тоді декодувальне відображення фактично діятиме з множини кодових слів на множину натуральних чисел. Його вважатимемо зворотним і позначимо як ψ^{-1}, а пряме відображення, яке позначимо через ψ, діятиме з множини натуральних чисел на множину кодових слів.

У [7] описано пряме ψ та зворотне відображення ψ^{-1} для мультироздільникових кодів. Однак відображення ψ має той недолік, що довжини кодових слів $\psi(1)$, $\psi(2)$, … , які відповідають послідовним натуральним числам, не впорядковано за зростанням. Тому слова тексту w_i у згаданому вище словнику мають бути розташовані не в порядку зниження їхніх частот $p(w_i)$. Це не є проблемою, оскільки можна сформулювати простий принцип розташування слів тексту в словнику: якщо першим i елементам впорядкованої за спаданням частот послідовності слів $\{w_i\}$ вже поставлено у відповідність певну множину кодових слів, то слову вхідного тексту w_{i+1} ставимо у відповідність будь-яке з найкоротших кодових слів, що не належать цій множині. Проте проблема полягає в тому, що кодові слова $\psi(1)$, … , $\psi(n)$ не утворюють множини n найкоротших кодових слів, а значить, масив слів тексту з індексами $\psi^{-1}(c_1)$, …, $\psi^{-1}(c_n)$, де c_1, …, c_n — n найкоротших кодових слів, буде розрідженим. Навіть більше, як показано у [16], розмір цього масиву зростатиме експоненційно залежно від n, а саме не менш ніж у $O\left((1+\varepsilon)^{(n-2)/(m_1+1)}\right)$ разів, де величина ε залежить від асимптотичної щільності коду і перебуває в діапазоні від 0,65 до 0,85.

Звичайно, таке суттєве зростання обсягу словника обмежує область застосування кодів лише порівняно невеликими текстами. У [17] розглянуто кілька обхідних шляхів вирішення цієї проблеми, однак найкращим варіантом було б конструювання інших алгоритмів кодування та декодування мультироздільникових кодів, згідно з якими зростаюча послідовність натуральних чисел відображалася б у послідовність кодових слів неспадної довжини. Цю можливість ми і дослідимо далі, навівши спочатку означення монотонного відображення множини натуральних чисел на множину кодових слів.

Означення 2. Відображення ψ множини натуральних чисел на множину слів коду C називатимемо ***монотонним***, якщо для будь-яких натуральних чисел a і b, таких, що $a < b$, виконується нерівність $|\psi(a)| \le |\psi(b)|$.

(Тут і далі через $|x|$ позначатимемо бітову довжину слова x).

Зауважимо, що для побудови монотонного відображення ψ слова коду можна відсортувати за зростанням їхніх довжин, а всередині кожної групи слів однакової довжини — в лексикографічному порядку. Пронумерувавши отриману послідовність кодових слів послідовними натуральними числами, отримаємо шукане відображення. Кодування

згідно з описаним відображенням виконується тривіально і полягає просто у видобуванні кодового слова з масиву за його індексом. Однак наївний спосіб декодування, тобто обчислення $\psi^{-1}(u)$, що полягає в послідовному пошуку кодового слова u в масиві, є неприпустимо довгим. Застосовувати напряму прискорені методи пошуку, такі як двійковий пошук, неможливо, оскільки в цілому масив кодових слів лексикографічно не відсортовано. Отже, найпростіший спосіб декодування потоку конкатенованих кодових слів полягав би в тому, щоб спочатку виявити межі чергового кодового слова й обчислити його довжину, а потім виконати двійковий пошук у лексикографічно відсортованому наборі кодових слів заданої довжини. Проте часова складність такої процедури була б суттєво (в рази) більшою за часову складність прискореного декодування, описаного в [7].

Найкращим варіантом було б віднайти алгоритм обчислення $\psi^{-1}(u)$, тобто декодування слова u, який би обчислював позицію u у впорядкованому за зростанням довжин наборі кодових слів без обчислення довжини u, на основі опрацювання частин слова u зліва направо. Однак це видається проблематичним, оскільки, не знаючи довжини u, ми не знаємо, як впливає на позицію u в наборі кодових слів опрацювання того чи іншого префіксу або підрядка u. Цієї проблеми, можливо, вдалося б уникнути, якби мультироздільникові коди не були б префіксними, тобто деякі префікси довших кодових слів були б коротшими кодовими словами (зауважимо, що префіксність коду є достатньою, але не необхідною умовою його однозначної декодовності). Тоді декодування слова uv, де u — коротке кодове слово, а v — суфікс довшого кодового слова, можна було б виконувати поетапно: спочатку обчислити позицію u у впорядкованому наборі кодових слів довжини не більше ніж $|u|$ (для цього не потрібно знати нічого про v), а потім до цієї позиції додати зміщення uv відносно u у наборі кодових слів довжини не більше ніж $|uv|$.

Зауважимо, що якщо слова мультироздільникового коду записувати справа наліво, то отримаємо якраз непрефіксний, але однозначно декодовний код, який назвемо ***реверсним***. Його однозначна декодовність випливає з однозначної декодовності прямого мультироздільникового коду, адже будь-який алгоритм декодування прямого коду є цілком застосовним і до реверсного, якщо опрацьовувати закодований файл справа наліво.

Однак найцікавішим є той факт, що у разі опрацювання реверсних мультироздільникових кодів зліва направо можна побудувати просте монотонне кодувальне відображення ψ і зворотне до нього декодувальне відображення ψ^{-1}, яке задовольнятиме всім розглянутим вище вимогам. Такі коди й відображення буде детально досліджено в наступному розділі.

РЕВЕРСНІ МУЛЬТИРОЗДІЛЬНИКОВІ КОДИ

Нехай, як і раніше, $M=\{m_1, \dots, m_t\}$ — зростаюча послідовність цілих чисел, $0 < m_1 < \dots < m_t$.

Означення 2. Реверсним мультироздільниковим кодом R_M називається код, що містить усі слова вигляду 01^{α}, $\alpha \in M$, і всі слова, які задовольняють таким умовам:

- слово починається з послідовності $01^{\alpha}0$, $\alpha \in M$;
- слово не закінчується послідовністю 01^{α}, $\alpha \in M$;
- слово не містить послідовності $01^{\alpha}0$, $\alpha \in M$, ніде, крім початку.

Роздільниками у такому коді є послідовності вигляду $01^{\alpha}0$, $\alpha \in M$, однак код містить також слова вигляду 01^{α}, $\alpha \in M$, які утворюють роздільник разом з першим нулем наступного кодового слова в закодованому тексті.

Спираючись на доведені в [7] властивості однозначної декодовності, повноти та універсальності у розумінні Еліаса [8] прямих мультироздільникових кодів, легко довести відповідні властивості і для реверсних кодів.

Теорема 1. Будь-який реверсний мультироздільниковий код R_M є однозначно декодовним, повним і універсальним.

*Доведення***.** Зауважимо, що будь-яка послідовність слів реверсного коду R_M у разі читання справа наліво збігається з деякою послідовністю слів коду D_M, а отже, код R_M є однозначно декодовним. Також однозначне декодування може бути виконано у разі оброблення закодованого файлу зліва направо за таким принципом:

- зчитуємо файл біт за бітом, поки не зчитаємо послідовності $01^{\alpha}0$, $\alpha \in M$;
- поточне кодове слово закінчується перед початком цієї послідовності, а наступне кодове слово починається з цієї послідовності.

Щоб довести повноту коду R_M, зауважимо, що кількість слів будь-якої заданої довжини L у цьому коді дорівнює кількості слів довжини L у коді D_M. А оскільки, як показано в [7], нерівність Крафта-Макмілана для коду D_M виконується як рівність:

$$\sum_{c \in C} 2^{-|c|} = 1,$$

де C — множина кодових слів, то вона виконуватиметься як рівність і для коду R_M, а отже, цей код є повним.

Щоб довести універсальність коду R_M, розглянемо довільний текст і закодуємо його спочатку кодом D_M, а потім запишемо закодований файл справа наліво, отримавши код R_M.

Застосування цього методу кодування даватиме змогу отримувати для довільного вхідного тексту коди R_M, довжини яких дорівнюватимуть довжинам кодів D_M для цього ж тексту. А оскільки, як доведено в [7], код D_M є універсальним, то і код R_M буде універсальним. ■

Тепер опишемо монотонне відображення множини натуральних чисел на множину слів реверсного мультироздільникового коду R_M, яке також можна назвати принципом формування довших кодів з коротших.

Нехай R_M — реверсний мультироздільниковий код, а $K=\{k_1, \ldots, k_q\}$ — впорядкована за зростанням послідовність усіх невід'ємних цілих чисел в діапазоні від 0 до m_t+1 включно, які не належать M. Наприклад, для коду $R_{2,3,5}$ $K=\{0, 1, 4, 6\}$. Насамперед зауважимо, що будь-яке слово коду R_M починається з послідовності 01^{α}, $\alpha \in M$. Щодо частини слова, яка розташована правіше, можливі 2 варіанти:

1) ця частина порожня, тобто все кодове слово має вигляд 01…1;

2) ця частина є конкатенацією груп вигляду 01…1, кількості одиниць у яких є числами з множини K або цілими числами, що більші за найбільший елемент K.

Отже, будь-яке слово коду R_M довжиною L бітів утворюється одним з таких способів:

1) це слово вигляду 01^{α}, $\alpha \in M$;

2) це слово утворюється зі слова довжини $L - k - 1$ бітів дописуванням справа послідовності 01^k, де де $k \in K$;

3) це слово утворюється зі слова довжини $L - 1$ бітів, що закінчується послідовністю вигляду 01…1 з не менше ніж $m_t + 1$ одиницями, дописуванням справа одиничного біта.

Достатньо очевидно також, що будь-яке кодове слово може бути утворено лише одним із цих трьох способів. Дійсно, якщо це слово вигляду 01^{α}, $\alpha \in M$, то воно утворено способом 1, якщо слово закінчується групою бітів вигляду 01…1 з не більш ніж $m_t + 1$ одиницями, то його можна утворити лише способом 2, а якщо воно закінчується групою бітів вигляду 01…1 з не менш ніж $m_t + 2$ одиницями, то його можна утворити лише способом 3.

Враховуючи ці факти, сформулюємо принцип побудови впорядкованого набору кодових слів довжини L, якщо набори слів меншої довжини вже побудовано:

1) спочатку запишемо всі кодові слова довжини $L–1$ у тому порядку, як їх було зазначено у наборі слів довжини $L–1$, з доданням справа символу «0», тобто послідовності вигляду 01…1 з $k_1=0$ одиницями;

2) потім запишемо всі кодові слова довжини $L–k_2–1$ у тому порядку, як їх було зазначено у наборі слів довжини $L–k_2–1$, з доданням справа послідовності вигляду 01…1 з k_2 одиницями тощо;

3) коли значення з набору K закінчаться, запишемо всі кодові слова довжини $L–1$, що закінчуються послідовністю вигляду 01…1 з не менше ніж $m_t + 1$ одиницями, у тому порядку, як їх було зазначено в наборі слів довжини $L–1$, із додаванням справа символу «1»;

4) нарешті, якщо значення L на одиницю більше за m_i — значення одного з параметрів коду m_1 , … , m_t , то додамо до набору слово вигляду 01…1 з m_i одиницями.Тепер опишемо алгоритм формування послідовності кодових слів довжини L формально.

Алгоритм 1. Формування послідовності кодових слів довжини L реверсного мультироздільникового коду R_M.

Вхід: довжина кодового слова L, параметри коду $m_1,\ldots,m_t$.

Результат: Послідовність слів коду R_M довжини L.

1. $i = 1$.
2. Поки $L - k_i > m_1 + 1$ і $i \le |K|$, виконуємо кроки 2a–2b.

2a. Додаємо до набору усі коди довжини $L - k_i - 1$, дописуючи до них справа послідовність 01…1, що містить k_i одиниць. Коди додаються саме у тій послідовності, у якій вони були вказані у наборі кодів довжини $L - k_i - 1$.

2b. $i = i + 1$.

3. Якщо існують кодові слова довжини $L–1$, що закінчуються послідовністю вигляду 01…1 з не менше ніж $m_t + 1$ одиницями, додаємо їх

до набору, дописуючи справа одиничний біт. Коди додаються саме у тій послідовності, у якій вони були вказані у наборі кодів довжини $L - 1$.

4. Якщо $L = m_j + 1$ для деякого j, то додаємо до набору слово 01…1, що містить m_j одиниць.

Кінець алгоритму.

Увесь набір слів коду R_M формується послідовно: спочатку записуємо найкоротше слово 01…1 довжини m_1+1, потім за описаним вище принципом формуємо набір слів довжини m_1+2 і дописуємо їх у кінець послідовності слів тощо. Найважливішою властивістю цього впорядкованого набору кодових слів є те, що їхні бітові довжини утворюють неспадну послідовність.

Для прикладу, на Рис. 1 наведено список слів реверсного мультироздільникового коду $R_{2,3,5}$ довжини не більше ніж 8 і показано, як слова довжини 8 утворюються зі слів довжин 7, 6 та 3 дописуванням справа бітових послідовностей 0, 01 та 01111 відповідно.

Будь-який реверсний мультироздільниковий код є регулярною мовою і тому розпізнається скінченним автоматом, на основі якого можна побудувати декодувальне відображення. Структура такого автомату залежить від параметрів коду. Для прикладу, на Рис. 2 зображено декодувальний автомат для коду $R_{2,3,5}$. Над кожною стрілкою, що символізує перехід з одного стану автомату в інший, перед вертикальною рискою записано символ (0 або 1), у разі зчитування якого відбувається цей перехід, а після риски — операції, які виконуються під час переходу.

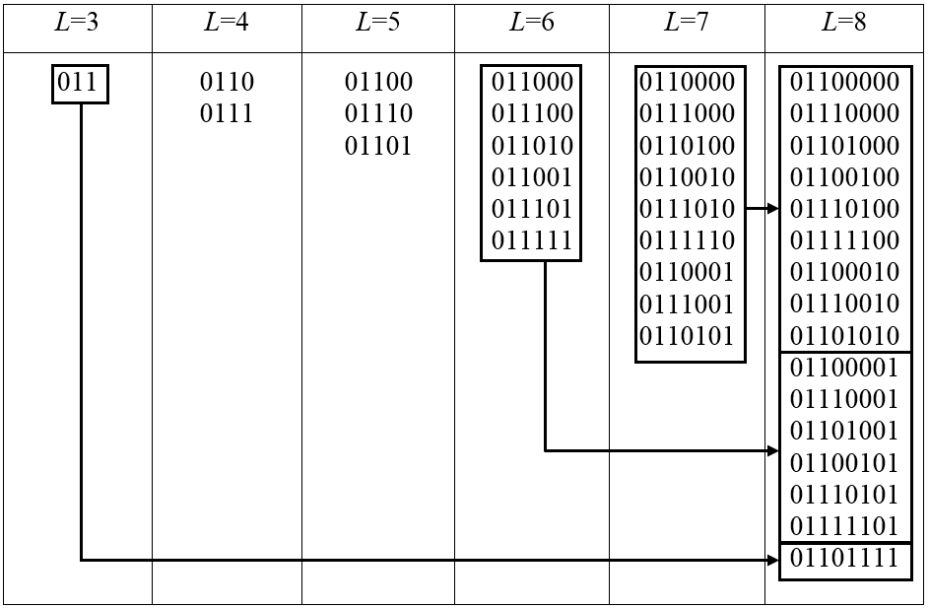

Рис. 1. Слова реверсного мультироздільникового коду $R_{2,3,5}$ довжини не більше за 8 символів.

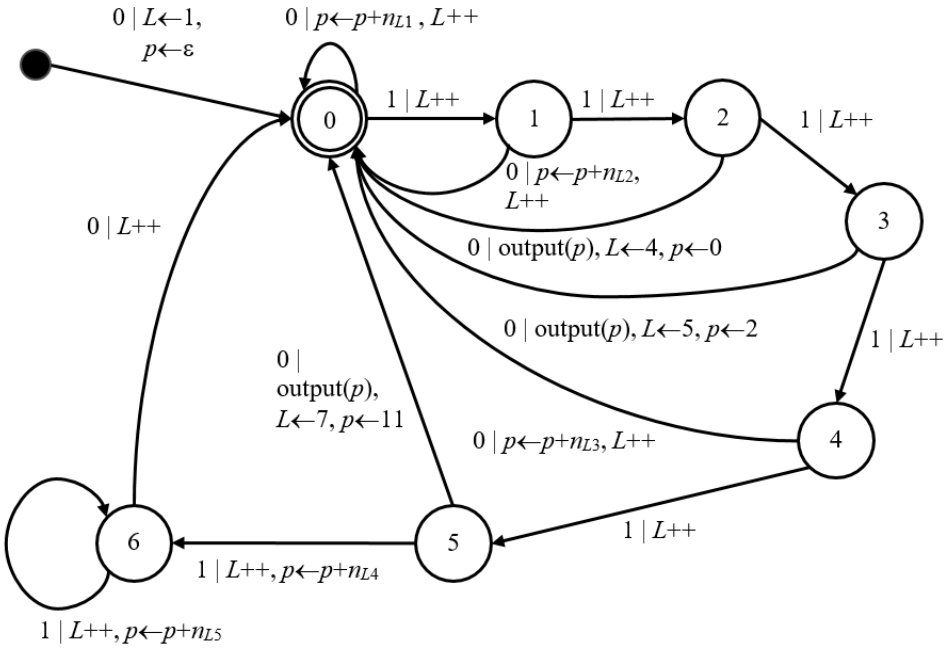

Рис. 2. Декодувальний автомат для коду $R_{2,3,5}$.

Довжину згенерованої частини кодового слова позначено як L. Під час опрацювання кожного біта коду вона збільшується на 1 (цей інкремент позначено як L++). А як p позначено величину, що наприкінці опрацювання автоматом кодового слова c дорівнює шуканому числу $\psi^{-1}(c)$. Порядок виконання операцій є важливим, він є саме таким, як зазначено біля стрілок переходів.

Щоб пояснити принцип роботи автомату, зауважимо, що будь-яке слово коду R_M є конкатенацією груп вигляду 01^x, де $x \geq 0$. Після оброблення кожної такої групи та наступного за нею нуля автомат опиняється у стані 0. Групи 011, 0111 та 011111 можуть бути розташовані лише на початку кодового слова. Після їхнього оброблення автомат переходить до стану 0 зі станів 2, 3 або 5 і величина p ініціалізується значеннями 0, 2 та 11, які відповідають положенню слів 011, 0111 та 011111 у послідовності всіх слів коду $R_{2,3,5}$ (Рис. 1), а величина L ініціалізується значеннями 4, 5 або 7 відповідно. Автомат завершує оброблення кодового слова, коли зчитує роздільник на початку наступного кодового слова, і при цьому він також переходить до стану 0 зі станів 2, 3 або 5. Тоді у змінній p міститься отримане в результаті значення $\psi^{-1}(c)$, яке виводиться командою output(p). Отже, зображений на Рис. 2 автомат опрацьовує потоковий код $R_{2,3,5}$, що є конкатенацією кодових слів. Автомат розпочинає роботу в стані, позначеному чорним кружком, і завершує роботу в стані 0 після зчитування останнього символу файлу з кодом (зауважимо, що наприкінці файлу має бути дописано якийсь роздільник, наприклад 0110). Під час переходу з початкового стану до стану 0

величина p ініціалізується порожнім рядком і, оскільки файл з кодом починається з рядка 0110, 01110 або 0111110, то другий перехід до стану 0 буде виконано зі станів 2, 3 або 5, а отже, жодна команда додання до p значення n_{Lj} до ініціалізації змінної p не виконається.

Під час роботи автомату використовуються допоміжний двовимірний масив, елементи якого позначено як n_{Lj}. Якщо $j \le |K|$ (для коду $R_{2,3,5}$ $|K| = 4$), то значення n_{Lj} дорівнює величині, на яку збільшується позиція кодового слова у побудованій за алгоритмом 1 послідовності кодових слів у разі дописування до цього слова справа такої групи вигляду 01…1 з k_j одиницями, що бітова довжина отриманого в результаті слова дорівнюватиме L. Зміст величини $n_{L,|K|+1}$ той самий, тільки до кодового слова дописується справа один символ «1». Так, у разі переходу зі стану 0 до стану 0 до кодового слова довжини L–1 справа дописується нульовий біт, що зміщує отримане кодове слово на n_{L1} позицій уперед (оскільки $k_1 = 0$). Перехід зі стану 1 до стану 0 означає дописування до кодового слова справа групи 01 (після якої йде символ «0») і, оскільки $k_2 = 1$, то до величини p додається значення n_{L2}.

Аналогічно, у разі переходу зі стану 4 до стану 0 до кодового слова додається рядок 01111 (після якого йде символ «0»), а до величини p — значення n_{L3} (оскільки $k_3 = 4$). У разі переходу зі стану 5 до стану 6 до кодового слова додається рядок 0111111, а до величини p — значення n_{L4} (оскільки $k_3 = 6$). Якщо ж після додавання до кодового слова рядка вигляду 01…1 із 6 або більше одиницями автомат знову зчитує символ «1», то до величини p додається значення n_{L5}. Зауважимо, що, згідно з алгоритмом 1, зсув усіх кодових слів довжини L у разі додавання до них тієї саме бітової послідовності вигляду 01…1 буде однаковим, так само, як і зсув усіх слів довжини L не менш ніж із 6 одиничними бітами наприкінці у разі додавання до них одиничного біта.

Реверсні мультироздільникові коди мають такий саме коефіцієнт стискання, як і прямі мультироздільникові коди зі словником достатньо великого розміру, тобто словником у вигляді масиву, що містить слова вихідного тексту в таких позиціях i_1, …, i_n, що образи $\psi(i_1)$, …, $\psi(i_n)$ є n найкоротшими кодовими словами, де n — кількість різних слів у тексті. Однак, як зазначено вище, словник за такого підходу збільшується експоненційно відносно n і тому навіть для текстів середнього розміру, таких як Біблія, використання такого словника є практично неможливим.

Для порівняння практичної стискальної ефективності прямих та реверсних мультироздільникових кодів було проведено обчислювальний експеримент, результати якого наведено в табл. 1.

Коди було апробовано на трьох текстах: відносно невеликому українському, а також на англомовних текстах середнього та великого розміру. Середній текст — це англійський переклад Біблії, а великий текст, який ми позначатимемо EN, — вибірка випадкових статей з Вікіпедії. З текстів було видалено знаки пунктуації, а також скасовано розрізнення великих і малих літер. Символами вхідного алфавіту вважалися слова, тобто послідовності літер між двома пробілами.

Табл. 1. ***Порівняння стискальної ефективності мультироздільникових та реверсних мультироздільникових кодів (наведено середню довжину кодового слова в бітах і відсоток перевищення значення ентропії)***

Текст	Кількість слів у тексті	Розмір словника	Значення ентропії	$D_{2,3,5}$	$D^{L}_{2,3,5}$	$R_{2,3,5}$	$R_{2-\infty}$	$R_{2,4-\infty}$
Хіба ревуть воли як ясла повні, П. Мирний (українською)	104 832	22 512	11,519	12,476 8,3%	12,826 11,3%	12,305 6,8%	12,418 7,8%	12,192 5,8%
Біблія (англійською)	790 018	14 018	8,879	9,236 4%	9,315 4,9%	9,211 3,7%	9,082 2,3%	9,2 3,6%
EN — статті з Вікіпедії (англійською)	19 507 783	288 179	11,0783	11,488 3,7%	11,544 4,2%	11,464 3,5%	11,6 4,7%	11,393 2,8%

У стовпцях табл. 1, що відповідають кодам, наведено середні довжини слів у бітах, а також перевищення цими значеннями інформаційної ентропії — теоретичної нижньої межі для такої величини, як середня довжина кодового слова у разі застосування частотного методу кодування. Поряд з реверсним кодом $R_{2,3,5}$ використано прямий мультироздільниковий код з незбільшеним словником (тобто до складу цього коду включено слова, які є образами перших n натуральних чисел) — $D^{L}_{2,3,5}$ — та код $D_{2,3,5}$, словник якого збільшено у 4 рази порівняно з мінімальним. Як свідчать наведені в табл. 1 результати, реверсний мультироздільниковий код забезпечує коефіцієнт стискання тексту на 0,2–0,3 % вищий, ніж відповідний прямий мультироздільниковий код з у 4 рази більшим словником, та на 0,7–1,2 % вищий, ніж прямий мультироздільниковий з незбільшеним словником. Враховуючи, що загалом перевищення ентропійної межі для цих кодів становить 3,5–5 %, отримуємо для реверсних мультироздільникових кодів на 20–30 % краще наближення до ентропійної межі, ніж для прямих мультироздільникових кодів з незбільшеним словником, що є достатньо суттєвою перевагою.

Підвищити ступінь стиснення можна ще більше завдяки використанню кодів з більш ніж трьома роздільниками. Підбір оптимального набору роздільників реверсного коду для певного тексту легко автоматизувати, що й було зроблено для двох текстів з табл. 1. Для Біблії це виявився код з усіма роздільниками, що містять від 2 або більше одиниць; у табл. 1 його позначено як $R_{2-\infty}$, а для статей з Вікіпедії — код, що має роздільник з двома одиницями, а також усі роздільники, що містять не менше 4 одиниць; у табл. 1 його позначено як $R_{2,4-\infty}$.

Зауважимо, що насправді у проведеному обчислювальному експерименті найдовший роздільник у кодах $R_{2-\infty}$ та $R_{2,4-\infty}$ містив 15 одиниць, а додавання наступних роздільників підвищувало ступінь стискання незначно. Так, коди R_{2-16} порівняно з R_{2-15} для Біблії та $R_{2,4-16}$ порівняно з $R_{2,4-15}$ для статей з Вікіпедії були коротшими менш ніж на 0,01 %. Тому можна вважати, що $R_{2-15} \approx R_{2-\infty}$ та $R_{2,4-15} \approx R_{2,4-\infty}$.

ОПТИМІЗАЦІЯ ОБРОБЛЕННЯ СЛОВНИКА

У попередніх розділах ми досліджували теоретичні властивості кодів та порівнювали їхній потенціал у стисканні природномовних текстів. Для цього було достатньо знати, який обсяг мав той чи інший закодований текст. Однак на практиці, якщо елементами вхідного алфавіту є слова природної мови, то крім власне закодованого тексту, у файлі архіву потрібно зберігати ще й словник *Dict* — це необхідна умова для побудови на основі коду архівувальної програми. Словник збільшує обсяг закодованого файлу і тому його теж потрібно зберігати в більш чи менш компактній формі. Зауважимо, що для мультироздільникових кодів зберігати словник у вигляді послідовності пар (слово вхідного тексту, кодове слово) недоцільно. Адже достатньо зберегти лише послідовність слів вхідного тексту, впорядкованих у порядку спадання їхніх частот. Тоді декодування виглядатиме так:

1) знаходимо для чергового кодового слова *code* натуральне число, що є його прообразом $\psi^{-1}(code)$;

2) додаємо до декодованого тексту елемент словника $Dict[\psi^{-1}(code)]$.

Опишемо деякі прості прийоми, що дають змогу досягти певного покращення коефіцієнту стискання текстів, як без, так і із застосуванням архіватора, якщо разом із текстом зберігається словник. Легко зауважити надлишковість у кодуванні слів, що трапляються в тексті один раз: у закодованому файлі вони трапляються і в словнику, і у вигляді кодового слова в самому тексті. Можна було б уникнути подання таких слів у вигляді кодового слова взагалі, записуючи їх у тексті в незакодованому вигляді. Однак тоді постає завдання виявлення початків та кінців таких слів, що вимагає використання додаткових спеціальних символів. Тому ефективнішим видається описаний нижче підхід до подолання цієї проблеми.

1. Відсортувати фрагмент словника, що містить слова із частотою 1, у порядку появи цих слів у тексті. Позначимо цей відсортований фрагмент як $Dict_1$.

2. Кодувати слова, що мають частоту 1, не кодами їхніх номерів у словнику, а певним спеціальним коротким кодовим словом $code_1$ (однаковим для всіх таких слів).

3. Під час декодування, коли трапляється кодове слово $code_1$, замінювати його черговим словом з фрагменту словника $Dict_1$.

Спосіб вибору кодового слова $code_1$ є достатньо очевидним. У щойно описаному методі під час кодування та декодування всі слова із частотою 1 розглядаються як одне те саме слово, яке має певну (достатньо високу) частоту. А отже, у впорядкованому за спаданням частот списку слів тексту це слово має певний номер, якому і відповідає кодове слово $code_1$.

У результаті для запису кожного слова *word* із частотою 1 замість $|word| + |\psi(word)|$ бітів використовується $|word| + |code_1|$ бітів. Економія відбувається завдяки тому, що $|code_1| < |\psi(word)|$. Так, наприклад, у тексті EN слово $word_1$ має 15-ту позицію за частотою, якщо рахувати від найбільшої, а $|code_1|=7$, тоді як середнє значення $|\psi(word))|$, $word \in Dict_1$, становить 25,68.

Звичайно, потрібно також зберігати позицію першого слова у фрагменті словника $Dict_1$. Під це число можна відвести фіксовану кількість бітів на початку файлу архіву або закодувати його одним з універсальних кодів.

Для прикладу розглянемо принцип кодування фрази «Мій великий стіл – це мій стіл». Для спрощення переведемо всі слова у нижній регістр: «мій великий стіл це мій стіл». Множиною унікальних слів разом з їхніми частотами є [мій → 2; великий → 1; стіл → 2; це → 1]. Будуємо відображення з множини унікальних слів у множину натуральних чисел таким чином, щоб словам з більшою частотою відповідали менші числа. Побудоване відображення буде мати вигляд [мій → 1; стіл → 2; великий → 3; це → 4]. Словником у цьому випадку буде [мій стіл великий це]. Далі можемо записати закодовану послідовність: [мій стіл великий це][1 3 2 4 1 2]. Для спрощення прикладу натуральні числа не замінювалися кодовими словами. У реальних експериментах така заміна здійснюється, а також між кодовими словами не записують пробілів. Як видно, слова «великий» та «це» трапляються у закодованій послідовності двічі. Кількість слів із частотою 1 дорівнює 2. Нам потрібно модифікувати відображення з множини унікальних слів на множину натуральних чисел, видаливши всі слова частоти 1 та додавши одне нове слово, частотою якого буде кількість видалених слів. Модифіковане відображення буде мати вигляд [$word_1$ → 1; мій → 2; стіл → 3]. У результаті кодування буде мати вигляд [2][мій стіл великий це][2 1 3 1 2 3]. Префікс [2] дає інформацію про кількість слів із частотою 1.

Описаний спосіб оптимізації словника можна також поширити на слова із частотою більше ніж 1. Нехай $Dict_k$ — це фрагмент словника, що містить слова із частотою k, впорядковані за місцем першої появи в тексті. Під час кодування тексту, у разі якщо слово $word \in Dict_k$ трапляється вперше, кодуємо його спеціальним словом $code_k$, а якщо не вперше — то кодовим словом, яке відповідає номеру слова *word* у загальному словнику. Під час декодування, коли трапляється кодове слово $code_k$, замінюємо його черговим словом з фрагменту словника $Dict_k$.

Так, у попередньому прикладі було два слова із частотою 2: «мій» та «стіл». Якщо замінювати також слова із частотою 2, то модифіковане відображення словника на натуральні числа матиме вигляд [$word_2$ → 1; $word_1$ → 2; мій → 3; стіл → 4]. У результаті кодування буде мати вигляд [2 2][мій стіл великий це][1 2 1 2 3 4]. У префіксі [2 2] записана кількість слів із частотою 2 та із частотою 1.

Звичайно, чим більшим буде значення k, тим менший ефект даватиме такий прийом, оскільки, по-перше, у природномовному тексті кількість різних слів із частотою k зі збільшенням k зменшується (а з k екземплярів кожного такого слова скорочуватиметься код лише одного); по-друге, чим більшою є частота k слова *word*, тим коротшим є його оригінальний код ψ(*word*) і, нарешті, використання деяких кодових слів як зарезервованих дещо збільшує довжину інших кодових слів. На випробуваних природномовних текстах для $k \geq 3$ застосування описаного методу не дає практично ніякого покращення коефіцієнту стискання.

ПОКРАЩЕННЯ ЕФЕКТИВНОСТІ ВІДОМИХ АРХІВАТОРІВ

Аналіз ефективності стисканя, крім вищеназваного тексту англійською мовою (EN), було проведено на текстах німецькою (GE), російською (RU) та українською (UA) мовами. Файл EN, як уже згадувалося, є вибіркою статей з Вікіпедії. Файли іншими природними мовами було створено за допомогою автоматичної системи збирання новин з Інтернету. Для текстів застосовували передоброблення у вигляді вилучення всіх розділових знаків з тексту та запису слів великими літерами через один пробіл. Елементами алфавіту були послідовності символів між двома пробілами. Характеристики всіх отриманих файлів наведено в таблиці 2.

Надалі будемо використовувати наступні позначення: Enc — застосування кодування до файлу, Dict — додання словника, Collapsed_1 — оптимізація словника для слів із частотою 1, Collapsed_1_2 оптимізація словника для слів із частотами 1 та 2, Ar — застосування архіватора. Порівняння також проводилося з такими потужними ентропійними кодуваннями, як коди Гаффмана (Huff) [4] та ентропійне кодування зі скінченними станами (FSE, Finite State Entropy) на основі асиметричних систем числення [14].

Різні архіватори дають різні результати стиснення. Тому важливим є порівняння їхніх характеристик. У контексті цієї роботи використовувалися архіватори 7z, zstd, bzip2 та gzip. У таблиці 3 наведено результати архівування файлу EN. Як видно, найкращі показники має архіватор 7z, причому його перевага була стабільною в усіх проведених експериментах. Тому в подальшому наведено результати саме для цього архіватора.

У таблиці 4 наведено результати кодування англійського тексту за допомогою кодування $R_{2,4-\infty}$, $R_{2-\infty}$, FSE та Гаффмана, із застосуванням оптимізації словника та комбінуванням з архіватором 7z. Зауважимо, що застосування оптимізації словника завжди покращує кінцеві результати стискання, незалежно від того, який саме реверсний мультироздільниковий код застосовувався, а також від того, застосовувався потім архіватор чи ні. Крім того, оптимізація за словами із частотами 1 і 2 (Collapsed_1_2) завжди дає більший ефект, ніж лише за словами із частотою 1 (Collapsed_1). У найкращому випадку вдається досягти показника, що на 0,58 % перевищує значення ентропії. Зауважимо, що цей результат є найкращим серед усіх досліджених методів стиснення для файлу EN в рамках цієї роботи. Покращення відносно застосування одного лише архіватора 7z складає 3.02 %.

Табл 2. ***Характеристики вихідних текстів***

Файл	Кількість слів	З них унікальних слів	Розмір, Мб	Ентропія, Мб	Словник, Мб
EN	19507783	288179	116,92	27,01	2,59
GE	22040994	432386	150,5	32,48	5,47
RU	9934417	298343	132,46	16,09	5,62
UK	9859970	291218	126,85	15,87	5,36

Табл 3. ***Результати стиснення англійського тексту архіватором 7z***

Файл	Архіватор	Розмір, Мб	% перевищення ентропії
EN	7z	27,98	3,6 %
EN	Zstd	28,15	4,2 %
EN	Bzip	31,66	17,2 %
EN	Gzip	37,35	38,3 %

Табл. 4. ***Результати стиснення англійського тексту за допомогою реверсних мультироздільникових кодів, кодів Гаффмана та кодування FSE із та без подальшого архівування архіватором 7z.***

Файл	Enc	Метод	Арх.	Розмір, Мб	% перевищення ентропії
EN	$R_{2,4-\infty}$	Enc		30,37	12,4 %
EN	$R_{2,4-\infty}$	Collapsed_1 Enc		30,11	11,47 %
EN	$R_{2,4-\infty}$	Collapsed_1_2 Enc		30,05	11,22 %
EN	$R_{2,4-\infty}$	Enc	7z	27,52	1,88 %
EN	$R_{2,4-\infty}$	Collapsed_1 Enc	7z	27,34	1,21 %
EN	$R_{2,4-\infty}$	Collapsed_1_2 Enc	7z	27,29	1,01 %
EN	$R_{2-\infty}$	Enc		30,87	14,3 %
EN	$R_{2-\infty}$	Collapsed_1 Enc		30,58	13,21 %
EN	$R_{2-\infty}$	Collapsed_1_2 Enc		30,50	12,9 %
EN	$R_{2-\infty}$	Enc	7z	27,64	2,32 %
EN	$R_{2-\infty}$	Collapsed_1 Enc	7z	27,23	0,80 %
EN	$R_{2-\infty}$	Collapsed_1_2 Enc	7z	27,17	0,58 %
EN	FSE	Enc		29,91	10,7 %
EN	FSE	Enc	7z	28,30	4,77 %
EN	Huff	Enc		29,97	10,95 %
EN	Huff	Enc	7z	28,00	3,64 %

Природним є питання про те, чи буде розроблений метод давати покращення для інших мов, крім англійської. У таблиці 5 наведено результати стиснення тексті німецькою, російською та українською мовами. Для файлу GE покращення становить 2,99 %. Для російського та українського текстів запропонований нами метод покращення не дає. Мабуть, це пояснюється високофлективним характером слов'янських мов, що приводить до значно більшої кількості різних слів відносно обсягу тексту.

Також варто зауважити, що в наведених результатах розмір стиснених даних буває меншим за рівень ентропії. Це пояснюється тим, що під час визначення ентропії елементами вихідного алфавіту послідовності символів між двома пробілами, а насправді текст можна поділяти на слова багатьма способами, що і практикується в універсальних архіваторах. Інакше кажучи, значення ентропії, обчислене за «традиційного» поділу тексту на слова не є мінімально можливим, якщо порівнювати з іншими способами обчислення цієї величини.

Табл. 5. ***Результати стиснення файлів GE, RU та UK окремо архіватором 7z та у комбінації з розробленими методами передобробки.***

Файл	Enc	Метод	Ar	Розмір, Мб	% перевищення ентропії
GE			7z	32,78	0,96 %
GE	FSE	Enc	7z	33,96	4,56 %
GE	Huff	Enc	7z	32,85	1,14 %
GE	$R_{2,4-\infty}$	Collapsed_1_2 Enc	7z	31,97	-1,57 %
GE	$R_{2-\infty}$	Collapsed_1_2 Enc	7z	31,82	-2,04 %
RU			7z	14,77	-8,17 %
RU	FSE	Enc	7z	17,52	8,89 %
RU	Huff	Enc	7z	16,29	1,24 %
RU	$R_{2,4-\infty}$	Collapsed_1_2 Enc	7z	15,84	-1,55 %
RU	$R_{2-\infty}$	Collapsed_1_2 Enc	7z	15,74	-2,20 %
UK			7z	15,33	-3,41 %
UK	FSE	Enc	7z	17,31	9,05 %
UK	Huff	Enc	7z	16,55	4,28 %
UK	$R_{2,4-\infty}$	Collapsed_1_2 Enc	7z	16,14	1,65 %
UK	$R_{2-\infty}$	Collapsed_1_2 Enc	7z	16,05	1,15 %

Загалом можливість покращення коефіцієнту стиснення архіваторів завдяки мультироздільниковим кодам можна пояснити тим, що ці коди не тільки стискають текст, але і вносять в нього певну регулярність, яка потім використовується архіваторами.

ВИСНОВКИ

У статті було досліджено новий клас кодів із роздільниками — реверсні мультироздільникові коди, які є вдосконаленою версією нещодавно винайдених «прямих» мультироздільникових кодів. Реверсні коди вирішують проблему немонотонності словника, властиву прямим мультироздільниковим кодам, зберігаючи всі їхні позитивні властивості, такі як швидкість декодування, стійкість до помилок та можливість пошуку інформації в закодованому тексті без його розкодування.

Також було досліджено можливість підвищення ефективності архівування природномовних текстів завдяки їх попередньому кодуванню реверсними кодами. Зазначимо, що в сучасних архіваторах застосовуються найрізноманітніші методи стиснення даних і можливість покращення коефіцієнту стиснення, що досягається цими програмами, видається вельми проблематичною. Однак завдяки використанню реверсних мультироздільникових кодів та описаних у статті способів оптимізації словника у цьому напрямі вдалося досягти позитивних результатів для текстів нефлективними мовами (англійською та німецькою), покращивши коефіцієнт стиснення текстів на 1–3% при тому, що використовувався найкращий із відомих архіваторів в режимі максимального стиснення.

ЛІТЕРАТУРА

1. D. Salomon. Variable-Length Codes for Data Compression. London. U.K.: Springer-Verlag, 2007.
2. Burrows M., Wheeler D. J. A Block-Sorting Lossless Data Compression Algorithm. Digital Systems Research Center, Palo Alto, CA, USA. 1994, Research Report 124.
3. Б. Я. Рябко, “Сжатие данных с помощью стопки книг”, Проблемы передачи информации, т. 16, вып. 2. с. 16-21, 1980.
4. Huffman D. A method for the construction of minimum redundancy codes. *Proc. IRE*. 1952, Vol. 40, pp. 1098–1101.
5. Lakshmanan K.B. On universal codeword sets. *IEEE Transactions on Information Theory*. 1981, Vol. 27, pp. 659–662.
6. Apostolico A., Fraenkel A. Robust transmission of unbounded strings using Fibonacci representations. *IEEE Transactions on Information Theory*. 1987, Vol. IT-33, no. 2, pp. 238–245.
7. Anisimov A.V., Zavadskyi I.O. Variable-Length Prefix Codes With Multiple Delimiters. *IEEE Transactions on Information Theory*. 2017, vol. 63, no. 5, pp. 2885–2895.
8. Elias P. Universal codeword sets and representation of the integers. *IEEE Transactions on Information Theory*. 1975, Vol. 21, no. 2, pp. 194–203.
9. Golomb S.W. Run-length encodings. *IEEE Transactions on Information Theory*. 1966, Vol. IT-12, no. 3, pp. 399–401.
10. Klein S.T., Ben-Nissan M.K. On the usefulness of Fibonacci compression codes. *Computer Journal*. 2010, Vol. 53, no. 6, pp. 701–716.
11. Ziv J., Lempel A. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*. 1977, Vol. 23, no. 3, pp. 337–343.
12. Welch T. A Technique for High-Performance Data Compression. *IEEE Computer*. 1984, Vol. 17, no. 6, pp. 8–19.
13. Witten I.H., Neal R.M., Cleary J.G. Arithmetic Coding for Data Compression. *Communications of the ACM*. 1987, Vol. 30, no. 6, pp. 520–540.
14. Duda J., Tahboub K., Gadgil N. J., Delp E. J. The use of asymmetric numeral systems as an accurate replacement for Huffman coding. 2015 Picture Coding Symposium (Cairns, QLD, Australia, 31th of May—3rd of Jun, 2015). Cairns, Australia, pp. 65–69.
15. Anisimov A.V., Zavadskyi I.O. Splittable Data Compression Codes. 2019 IEEE International Conference on Advanced Trends in Information Theory (Kyiv, Ukraine, 18–20th of Dec, 2019). Kyiv, Ukraine, pp. 71–74.
16. Завадський І.О. Подільні коди та їх застосування. Рукопис дисертації на здобуття наукового ступеня доктора фіз.-мат. наук. Київ, 2020, 335 с.
17. Zavadskyi I.O., Anisimov A.V. A Family of Data Compression Codes with Multiple Delimiters. Prague Stringology Conference (Prague, Czech Republic, 29–31th of Aug, 2016). Prague, Czech Republic, 2016, pp. 71–84.

Отримано 10.09.2020

REFERENCES

1. D. Salomon. Variable-Length Codes for Data Compression. London. U.K.: Springer-Verlag, 2007.
2. Burrows M., Wheeler D. J. A Block-Sorting Lossless Data Compression Algorithm. Digital Systems Research Center, Palo Alto, CA, USA. 1994, Research Report 124.
3. B. Ya. Ryabko, Data compression using the book stack transformation. Information transmission problems. 1980, Vol. 16, no. 2, pp. 16-21. (in Russian).
4. Huffman D. A method for the construction of minimum redundancy codes. *Proc. IRE*. 1952, Vol. 40, pp. 1098–1101.
5. Lakshmanan K.B. On universal codeword sets. *IEEE Transactions on Information Theory*. 1981, Vol. 27, pp. 659–662.
6. Apostolico A., Fraenkel A. Robust transmission of unbounded strings using Fibonacci representations. *IEEE Transactions on Information Theory*. 1987, Vol. IT-33, no. 2, pp. 238–245.

7. Anisimov A.V., Zavadskyi I.O. Variable-Length Prefix Codes With Multiple Delimiters. *IEEE Transactions on Information Theory*. 2017, vol. 63, no. 5, pp. 2885–2895.
8. Elias P. Universal codeword sets and representation of the integers. *IEEE Transactions on Information Theory*. 1975, Vol. 21, no. 2, pp. 194–203.
9. Golomb S.W. Run-length encodings. *IEEE Transactions on Information Theory*. 1966, Vol. IT-12, no. 3, pp. 399–401.
10. Klein S.T., Ben-Nissan M.K. On the usefulness of Fibonacci compression codes. *Computer Journal*. 2010, Vol. 53, no. 6, pp. 701–716.
11. Ziv J., Lempel A. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*. 1977, Vol. 23, no. 3, pp. 337–343.
12. Welch T. A Technique for High-Performance Data Compression. *IEEE Computer.* 1984, Vol. 17, no. 6, pp. 8–19.
13. Witten I.H., Neal R.M., Cleary J.G. Arithmetic Coding for Data Compression. *Communications of the ACM*. 1987, Vol. 30, no. 6, pp. 520–540.
14. Duda J., Tahboub K., Gadgil N. J., Delp E. J. The use of asymmetric numeral systems as an accurate replacement for Huffman coding. 2015 Picture Coding Symposium (Cairns, QLD, Australia, 31th of May—3rd of Jun, 2015). Cairns, Australia, pp. 65–69.
15. Anisimov A.V., Zavadskyi I.O. Splittable Data Compression Codes. 2019 IEEE International Conference on Advanced Trends in Information Theory (Kyiv, Ukraine, 18–20th of Dec, 2019). Kyiv, Ukraine, pp. 71–74.
16. Zavadskyi I.O. Splittable codes and their applications. Doctor of science thesis. Kyiv, 2020, 335 P. (in Ukranian)
17. Zavadskyi I.O., Anisimov A.V. A Family of Data Compression Codes with Multiple Delimiters. Prague Stringology Conference (Prague, Czech Republic, 29–31th of Aug, 2016). Prague, Czech Republic, 2016, pp. 71–84.

Received 10.09.2020

Anisimov A.V., DSc (Phys & Math), corresponding member
of National Academy of Sciences of Ukraine,
Dean of the Faculty of Computer Science and Cybernetics
e-mail: anatoly.v.anisimov@gmail.com
Zavadskyi I.O., DSc (Phys & Math),
associate professor of the Mathematical Informatics Department
e-mail: ihorza@gmail.com
Chudakov T.S., student
e-mail: timofey.chudakov@gmail.com
Faculty of Computer Science and Cybernetics
of Taras Shevchenko National University of Kyiv ,
4d, Glushkov av., 03022, Kyiv, Ukraine

APPLICATION OF MULTI-DELIMITER CODES
TO NATURAL LANGUAGE TEXT ARCHIVING

Introduction. *The efficiency of modern archivers is approaching to the theoretical limit. Even small compression ratio improvements for some specific data types, by less than 1%, is assumed to be essential when the reasonable time complexity is maintained. This research is actual since a new data encoding method is developed, which gives the possibility to achieve rather more significant improvement of the compression ratio when it comes to English or German texts archiving.*

The purpose of the paper *is to solve the problem of non-monotonicity of a multi-delimiter code dictionary and investigate the possibility of use the multi-delimiter encoding on the preprocessing stage of natural language texts archiving.*

Results. *The concept of the reverse multi-delimiter code is introduced. The monotonic encoding as well as the decoding mapping from the set of natural numbers to the set of reverse multi-delimiter code codewords is built. The efficiency of applying the reverse multi-delimiter codes to natural language text compression is investigated together with the method of dictionary optimization. The provided experiments show that the reverse multi-delimiter encoding of English and German texts on the preprocessing stage and applying the proposed dictionary optimization method allows us to improve the marginal compression efficiency of the most powerful archivers in the maximal compression mode by about 1–3%.*

Conclusions. *The reverse multi-delimiter codes can be considered as an efficient tool when it comes to compression of natural language texts. As a standalone solution, these codes are robust, provide the possibility to fast decode and search the data in a compressed file. As a tool for natural language text preprocessing for subsequent archiving, the reverse multi-delimiter codes together with the method of dictionary optimization allow us to improve the compression rate of the best up-to-date known archivers.*

Keywords: *compression, archiving, archiver, compression code, multi-delimiter code, reverse multi-delimiter code, dictionary optimization, natural language text.*