DOI 10.31891/2307-5732-2019-279-6-131-133
УДК 004.627

М.М. ЛЕБІГА, О.А. ПАСІЧНИК, Т.К. СКРИПНИК, В.Ю. МЕДВЕДЧУК
Хмельницький національний університет

КОМБІНОВАНИЙ АЛГОРИТМ СТИСНЕННЯ ДАНИХ, ПРЕДСТАВЛЕНИХ В ТЕКСТОВОМУ ФОРМАТІ

В роботі розроблено комбінований алгоритм стиснення текстових даних задля мінімізації об'єму контенту, який є основою відповідної інформаційної технології. В результаті аналізу отриманих експериментальних даних було обрано оптимальні шляхи мінімізації об'єму текстового контенту на основі алгоритму Хаффмана за допомогою основних і найбільш розповсюджених типів структур текстових даних.

Ключові слова: комбінований алгоритм, алгоритм Хаффмана, мінімізація, оптимізація, граф, дерево, текстові дані, словник.

M.M. LEBIGA, O.A. PASICHNYK, T.K. SKRYPNYK, V.Y. MEDVEDCHUK
Khmelnytskyi National University

COMBINED TEXT DATA COMPRESSION ALGORITHM

The purpose of the work is to develop a combined algorithm for text data compression. To achieve this goal, the following research objectives have been identified: reviewing existing universal data compression algorithms; analysis of data types and text formats; development of a specialized compression algorithm based on the Huffman algorithm, taking into account the data structure. The two main and most common types of text data structure are graph and tree. To achieve the greatest compression of texts in natural languages, it is decided to use a tree-like structure of the dictionary with a fixed number of branches at each level, as the size of branching decreases with increasing depth, which with a fixed size of branches will lead to a strong redundancy and high memory consumption. To achieve the highest compression of html and xml texts, it is decided to use a dictionary based on a weighted graph with an unspecified number of branches on the node. The graph provides a flexible system by which any structure can be emulated, but constructing such a structure is challenging. An analysis of the amount of savings by optimization is calculated based on the difference between the savings of recording branch information and the number of bits that need to be recorded to inform the decoder of branch usage. If the difference is greater than zero, the information is considered to be advantageous for recording. To record tree-based dictionary optimization information, you must specify all the dictionary branches used. This can be done using a recursive algorithm. The result of the work is the development of a combined algorithm for compressing text data whose structure can be represented as a tree or graph. The developed compression algorithm is based on the Huffman algorithm and is the basis for implementation of the relevant information system.

Keywords: combined algorithm, Huffman algorithm, minimization, optimization, graph, tree, text data, dictionary.

Вступ

В сучасному світі електронні засоби зберігання, обробки і передачі інформації є всюдисущими і буденними, ми вже насилу уявляємо життя без них. Інформація стала основою нашого життя і її стає дедалі більше, що вимагає все більших ресурсів для її зберігання. Обсяги інформації, одержувані і відправлені за допомогою мережі Інтернет, зростають, а в сукупності це означає зріст витрат, необхідних для зберігання і транспортування інформації з використанням існуючих архітектур і технологій. У зв'язку з цим алгоритми стиснення даних актуальні і активно розвиваються.

Стиснення даних – це процедура перекодування даних, яка проводиться з метою зменшення їхнього обсягу, розміру, об'єму.

Стиснення базується на усуненні надлишку інформації, яка міститься у вихідних даних. Наприклад, повторення в тексті фрагментів (наприклад, слів природної або машинної мови). Подібний надлишок зазвичай усувається заміною повторюваних послідовностей коротшим значенням (кодом). Інший вид надлишковості пов'язаний з тим, що деякі значення в даних, що стискаються, трапляються частіше інших, при цьому можна замінювати дані, що часто трапляються, коротшими кодами, а ті, що рідко, довшими (ймовірнісне стиснення). Стиснення даних, які не мають властивості надлишку (наприклад випадковий сигнал чи шум), неможливе. Також, зазвичай, неможливо стиснути зашифровану інформацію.

Існує багато практичних алгоритмів стиснення даних, але всі вони базуються на трьох теоретичних способах зменшення надлишковості даних. Перший спосіб полягає в зміні вмісту даних, другий – у зміні структури даних, а третій – в одночасній зміні як структури, так і вмісту даних.

Аналіз останніх досліджень та публікацій

Донині було проведено багато досліджень різноманітних алгоритмів стиснення даних, представлених у текстовому форматі. У роботах таких закордонних вчених, як A. Lempel, J. Ziv [1, 2], T.A. Welch [3], J.A. Storer [4], розглядається розробка та застосування універсальних та спеціалізованих алгоритмів стиску даних представлених у різних форматах, але найбільше уваги приділено саме текстовому. Ще детальніше проблеми застосування алгоритмів стиснення даних представлених у текстовому форматі розглянуто у опублікованій доповідній роботі німецьких дослідників D. Hucke, M. Lohrey, C.P. Reh [5]. Також зацікавленість у дослідженні алгоритмів стиснення даних, представлених у текстовому форматі, виявили вітчизняні дослідники та їх колеги з країн СНД. Однією з останніх публікацій є книга О.В. Аграновського та Р.А. Хаді [6], де розглядаються алгоритми стиснення текстових даних та метод криптографічного стиснення.

Актуальність дослідження

Мета роботи полягає у розробці комбінованого алгоритму стиснення текстових даних. Для досягнення поставленої мети визначені такі задачі дослідження: проведення огляду існуючих універсальних алгоритмів стиснення даних; виконання аналізу типів даних та форматів побудови тексту; розробка спеціалізованого алгоритму стиснення на основі алгоритму Хафмана з урахуванням структури даних. Дослідження, що спрямовані на розробку комбінованого алгоритму стиснення текстових даних, є актуальною задачею

Виклад основного матеріалу

Двома основними і найбільш розповсюдженими типами структури текстових даних є граф і дерево.

Граф [1] – це складна нелінійна багатозв'язна динамічна структура, яка відображає властивості і зв'язки складного об'єкта.

Графічне представлення орієнтованого графа наведено на рис. 1.

Багатозв'язна структура має такі властивості:

– на кожний елемент (вузол, вершину) може бути довільна кількість посилань,

– кожний елемент може мати зв'язок з будь-якою кількістю інших елементів,

– кожна зв'язка (ребро, дуга) може мати напрямок і вагу.

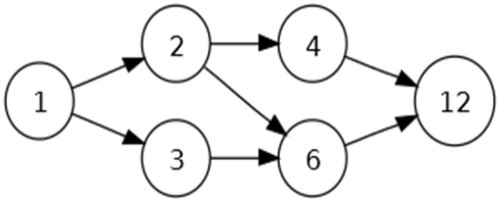

Рис. 1. Графічне представлення орієнтованого графа

У вузлах графа міститься інформація про елементи об'єкта. Зв'язки між вузлами задаються ребрами графа. Граф можна розбити по типу ребер, які він містить:

– орієнтований граф – всі ребра графа напрямлені,

– неорієнтований граф – всі ребра графа ненапрямлені,

– змішаний граф – граф з ребрами обох типів.

Для орієнтованого графа число ребер, що входять у вузол, називається напівстепенем заходу вузла, що виходять з вузла – напівстепенем результату. Кількість вхідних і вихідних ребер може бути будь-якою, в тому числі і нульовою. Граф без ребер є нуль-графом.

Якщо ребрам графа відповідають деякі значення, то граф і ребра називаються виваженими. Мультиграфом називається граф, що має паралельні (що з'єднують одні і ті ж вершини) ребра, в іншому випадку граф називається простим.

Шлях в графі – це послідовність вузлів, пов'язаних ребрами; елементарним називається шлях, у якому всі ребра різні, простим називається шлях, у якому всі вершини різні. Шлях від вузла до самого себе називається циклом, а граф, що містить такі шляхи – циклічним.

Два вузла графа суміжні, якщо існує шлях від одного з них до іншого. Вузол називається інцидентним до ребра, якщо він є його вершиною, тобто ребро направлено до цього вузла.

Дерево [2] – це структура даних, що представляє собою сукупність елементів і відношень, що утворюють ієрархічну структуру цих елементів. Кожний елемент дерева називається вершиною (вузлом) дерева. Вершини дерева з'єднані напрямленими дугами, які називають гілками дерева. Початковий вузол дерева називають коренем дерева, йому відповідає нульовий рівень. Листям дерева називають вершини, в які входить одна гілка і не виходить жодної гілки.

Дерева особливо часто використовують на практиці при зображенні різних ієрархій. Приклад графічного представлення дерева представлений на рис. 2.

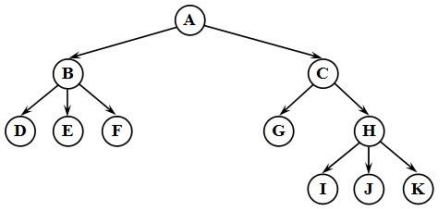

Рис. 2. Графічне представлення дерева

Для досягнення найбільшого стиснення текстів на природних мовах вирішено використовувати деревоподібну структуру словника з нефіксованою кількістю гілок на кожному рівні, так як зі збільшенням глибини розмір розгалуження знижується, що при фіксованому розмірі гілок призведе до сильної надмірності і великому споживанню пам'яті.

Для досягнення найбільшого стиснення текстів що мають формат html і xml вирішено використовувати словник заснований на орієнтованому зваженому графі з нефіксованою кількістю гілок на вузлі. Граф надає гнучку систему, за допомогою якої можна емулювати будь-яку структуру, однак побудова такої структури є складним завданням.

Аналіз розміру економії за рахунок оптимізації [3] розраховується виходячи з різниці між економією від запису інформації про гілку і кількістю біт, які необхідно записати, щоб повідомити декодеру [4] про використання гілки.

Якщо різниця більше нуля, то інформація вважається вигідною для запису. Для запису інформації про оптимізацію словника на основі дерева нам необхідно вказати всі використовувані гілки словника. Це можна зробити за допомогою рекурсивного алгоритму [5], що складається з наступних кроків:

– якщо вузол пустий, то завершуємо роботу. В іншому випадку:

– якщо поточний вузол має економію від запису використовуваних гілок, то пишемо 1;

– знаючи номери граничних гілок (тобто коли гілка і використовується, а гілка і + 1 або і-1 не використовується), пишемо всі граничні гілки (тобто пишемо кордон діапазонів використовуваних гілок, поточного вузла). Кількість біт, необхідних для запису номера гілки, розраховується за формулою (1);

– для кожної використовуваної гілки повторюємо алгоритм.

– інакше пишемо 0 і завершуємо роботу.

Для запису інформації про оптимізацію словника на основі графа нам необхідно вказати гілки, які використовуються для кодування, для кожного вузла. Це можна зробити за допомогою алгоритму, що складається з наступних кроків для кожного вузла:

– якщо вузол має економію від запису використовуваних гілок, то пишемо 1.

– знаючи номери граничних гілок (тобто коли гілка і використовується, а гілка і + 1 або і-1 не використовується), пишемо всі граничні гілки (тобто пишемо кордон діапазонів використовуваних гілок, поточного вузла). Кількість біт, необхідних для запису номера гілки, розраховується за формулою (1)

$$K = \frac{\log(N)}{\log(2)} + 1, \tag{1}$$

де N – кількість гілок в поточному вузлі;

– інакше пишемо 0.

Висновки

Результатом роботи є розробка комбінованого алгоритму стиснення даних представлених в текстовому форматі структура яких може бути представлена як дерево або граф. Розроблений алгоритм стиснення базується на алгоритмі Хаффмана і є основою для реалізації відповідної інформаційної системи.

Література

1. Lempel A., Ziv J. Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory. 1978. Т. 24, № 5. P. 530–536.

2. Ziv J. A constrained-dictionary version of LZ78 asymptotically achieves the finite-state compressibility with a distortion measure. IEEE Information Theory Workshop. 2015. P. 1–4.

3. Welch T. A. A technique for high-performance data compression. Computer. 1984. Т. 6, № 17. P. 8–19.

4. Storer J. A. Data Compression: Methods and Theory. New York, USA: Computer Science Press, 1988. 413 p.

5. Hucke D., Lohrey M., Reh C. P. The smallest grammar problem revisited. String Processing and Information Retrieval (SPIRE). 2016. Т. 9954. P. 35–49.

6. Граф [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Граф_(математика).

7. Дерево [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Дерево_(теорія_графів).

8. Оптимізація [Електронний ресурс]. – Режим доступу : https://uk.wikipedia.org/wiki/Оптимізація_(інформатика).

9. Декодер [Електронний ресурс]. – Режим доступу : https://en.wikipedia.org/wiki/Decoder.

10. Рекурсивний алгоритм: [Електронний ресурс]. – Режим доступу : http://mmsa.kpi.ua/sancho/ASD_HTM/Recurs01.html.

References

1. Lempel A., Ziv J. Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory. 1978. T. 24, № 5. P. 530–536.

2. Ziv J. A constrained-dictionary version of LZ78 asymptotically achieves the finite-state compressibility with a distortion measure. IEEE Information Theory Workshop. 2015. P. 1–4.

3. Welch T. A. A technique for high-performance data compression. Computer. 1984. T. 6, № 17. P. 8–19.

4. Storer J. A. Data Compression: Methods and Theory. New York, USA: Computer Science Press, 1988. 413 p.

5. Hucke D., Lohrey M., Reh C. P. The smallest grammar problem revisited. String Processing and Information Retrieval (SPIRE). 2016. T. 9954. P. 35–49.

6. Graph [Electronic resource]. – Access mode: https://uk.wikipedia.org/wiki/Graph_(mathematics).

7. Tree [Electronic resource]. – Access mode: https://uk.wikipedia.org/wiki/Tree_(graph theory).

8. Optimization [Electronic resource]. – Access mode: https://uk.wikipedia.org/wiki/ Optimization_(Computer Science).

9. Decoder [Electronic resource]. – Access mode: https://uk.wikipedia.org/wiki/ Decoder.

10. Recursive algorithm [Electronic resource]. – Access mode: https://uk.wikipedia.org/wiki/http://mmsa.kpi.ua/sancho/ASD_HTM/Recurs01.html.

Рецензія/Peer review : 15.12.2019 р. Надрукована/Printed : 02.01.2020
Рецензент: д.т.н., проф. Сорокатий Р.В.