DOI: 10.24412/2413-2527-2023-436-42-46

# Разработка алгоритма сжатия данных на основе метода Хаффмана: практический аспект

Р. М. Слесарев, М. А. Добровольская, к.и.н. А. В. Забродин
Петербургский государственный университет путей сообщения Императора Александра I
Санкт-Петербург, Россия
romsyt2002@gmail.com, dobro-7878@mail.ru, teach-case@yandex.ru

***Аннотация.*** **Статья посвящена разработке алгоритма сжатия данных различных типов, представленных в виде файлов во внешней памяти компьютера, на основе метода Хаффмана. Подробно рассмотрен алгоритм Хаффмана минимального префиксного кодирования. Целью работы является изучение и сравнение между собой различных алгоритмов сжатия данных, а также практическое применение алгоритма на основе метода Хаффмана для написания программы-компрессора файлов.**

***Ключевые слова:*** **сжатие без потерь, сжатие с потерями, алгоритмы сжатия, кодирование, кодовое дерево.**

## Введение

С увеличением объема данных в современном мире возникает настоятельная потребность в эффективных методах хранения и передачи информации. Для оптимизации использования памяти было разработано множество алгоритмов сжатия данных. Существуют две основные категории таких алгоритмов: сжатие данных с потерями и без потерь. Первая используется для уменьшения размера фотографий и видео, в то время как вторая включает в себя методы и алгоритмы, направленные на сжатие файлов без утраты информации. Среди методов без потерь выделяются алгоритмы кодирования Хаффмана, которые являются одними из самых старых, но, несмотря на свою долгую историю, остаются востребованными и эффективными. Основная цель работы заключается в исследовании и практическом применении алгоритма на основе метода Хаффмана в контексте сжатия текстовых данных.

## Основные термины и определения

Сжатие данных представляет собой процесс преобразования данных с целью уменьшения их объема путем удаления избыточности. Главная цель сжатия данных заключается в уменьшении занимаемого ими пространства в памяти и повышении скорости их обработки [1].

Избыточность является ключевым понятием в теории сжатия информации, поскольку сжатие возможно только для данных, содержащих избыточную информацию. Данные, не содержащие избыточности (например, файлы, состоящие из случайных равновероятных байтов), невозможно сжать [2].

Методы сжатия данных представляют собой специфический случай методов кодирования информации. Код представляет собой взаимно-однозначное отображение конечного упорядоченного множества символов из определенного алфавита на другое множество символов.

С точки зрения возможности полного восстановления данных методы сжатия данных можно разделить на два основных класса [3]: сжатие без потерь (обратимое) и сжатие с потерями (необратимое).

Сжатие без потерь позволяет снизить объем данных без потери информации, то есть позволяет восстановить закодированную порцию данных без изменений.

Сжатие с потерями не позволяет путем декодирования восстановить исходные данные в полном объеме, но при этом обеспечивается максимальная степень сжатия данных [4].

Методы сжатия данных с точки зрения возможности их использования для сжатия информации различных типов (текста, изображений, звука или видео) можно разделить на два типа [5]: специальные и универсальные.

Универсальные методы представляют информацию как простую последовательность битов, поэтому могут использоваться для сжатия любой цифровой информации. Такие методы являются обратимыми, так как позволяют добиться полного восстановления сжатых данных.

Специальные методы предназначены для сжатия информации только определенного типа. Такие методы учитывают специфику восприятия изображений, звука или видео человеком. Благодаря удалению малозначимой для восприятия информации удается достичь высокой степени сжатия. Таким образом, специальные методы являются необратимыми.

Для оценки производительности алгоритмов сжатия применяют следующие величины [6]:

1. Коэффициент сжатия:

$$\text{Коэффициент сжатия} = \frac{\text{размер выходного файла}}{\text{размер входного файла}}.$$

2. Величина, обратная коэффициенту сжатия, называется фактором сжатия. Если эта величина меньше 1, то идет сжатие, если больше 1 — расширение.

$$\text{Фактор сжатия} = \frac{\text{размер входного файла}}{\text{размер выходного файла}}.$$

3. Качество сжатия: $100 * (1 — k)$, где $k$ - коэффициент сжатия, отражает качество сжатия.

$$\text{Качество сжатия} = 100 \times (1 - k),$$

где k — коэффициент сжатия.

### Описание алгоритма Хаффмана

Простой алгоритм минимального префиксного кодирования был предложен в 1952 году Дэвидом Хаффманом (David Huffman). Он позволяет сжимать данные до их энтропии, что является оптимальным методом для сжатия без потерь. Алгоритм работает путем построения кодового дерева снизу вверх [7]:

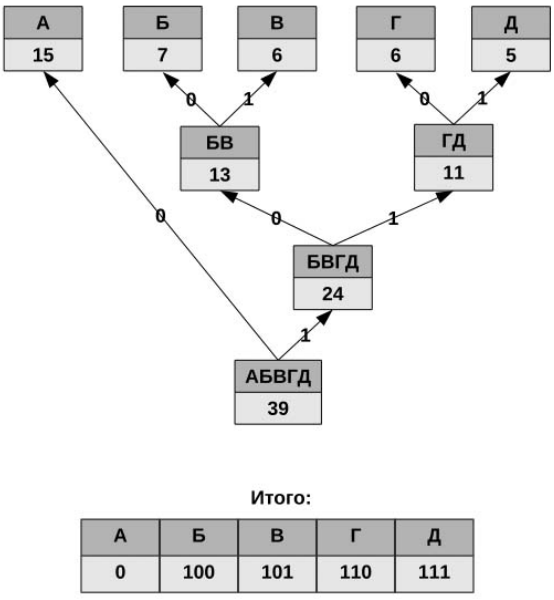
1. Каждому символу сопоставляется дерево, состоящее из одной вершины, содержащей код символа и его частоту.

2. Составляется список всех вершин в порядке убывания их вероятностей.

3. На каждом шаге выбираются две вершины с наименьшими вероятностями, удаляются из списка и заменяются вспомогательной вершиной, представляющей эти два символа. Вспомогательной вершине приписывается вероятность, равная сумме вероятностей выбранных на этом шаге символов.

4. Когда список сокращается до одного вспомогательного символа, представляющего весь алфавит, дерево объявляется построенным. Алгоритм завершается.

Построенное дерево кодов Хаффмана показано на рисунке 1.




Итого:

| А | Б | В | Г | Д |
|---|---|---|---|---|
| 0 | 100 | 101 | 110 | 111 |

Рис. 1. Дерево кодов Хаффмана

Математическая модель данного алгоритма будет выглядеть таким образом [8]:

*Ввод.*

Алфавит $A = \{a_1, a_2, \ldots, a_n\}$, представляющий собой алфавит символов размера $n$.

Кортеж $W = \{w_1, w_2, \ldots, w_n\}$, представляющий собой кортеж положительных весов символов (обычно пропорциональных вероятностям).*Вывод.*

Код $C(W) = \{c_1, c_2, \ldots, c_n\}$, представляющий собой набор двоичных кодовых слов, где $c_i$ — кодовое слово для $a_i$, $i = \{1, 2, \ldots, n\}$.

На практике наиболее часто применяют два алгоритма, основанных на методе Хаффмана: полуадаптивный и адаптивный (динамический).

При использовании полуадаптивного кодирования Хаффмана, кодировщик считывает сжимаемый файл дважды. Во время первого чтения вычисляются частоты встречаемости символов, а во время второго чтения происходит непосредственное сжатие файла. Для обеспечения возможности декодирования кодировщик добавляет в начало сжатого файла таблицу частот, занимающую несколько сотен байт, которая позволяет восстановить дерево кодирования и воссоздать исходный сжатый файл.

Основная идея адаптивного кодирования заключается в том, что компрессор и декомпрессор начинают работать с «пустого» дерева Хаффмана, а потом модифицируют его по мере чтения и обработки символов. Соответственно, кодер и декодер должны модифицировать дерево одинаково, чтобы все время использовать один и тот же код, который может меняться по ходу процесса. При таком кодировании нет необходимости сохранять таблицу частот в сжатый файл, а для кодирования требуется всего одно прочтение исходного файла (вместо двух), но при этом сжатие получается менее эффективным, чем при использовании полуадаптивного алгоритма [9, 10].

В данной работе будет реализован полуадаптивный вариант метода Хаффмана, который позволяет достичь баланса между эффективностью сжатия и вычислительной сложностью алгоритма.

### Описание проблемы

Практической задачей для оценки разрабатываемого алгоритма является обработка информации, передаваемой студентами в систему дистанционного образования. Группа из 30 студентов бакалавриата за один семестр загружает в систему не менее 120 мегабайт информации (примерно 4 мегабайта на каждого студента). Учитывая, что каждый студент в течение учебы может загрузить до 1 гигабайта данных, важно иметь эффективный алгоритм сжатия. Он должен обеспечить оптимальное использование ресурсов университета для хранения и обработки этой информации. Помимо сжатия, также важно учесть долгосрочное хранение данных после выпуска студентов, что подчеркивает значимость эффективности и надежности разработанного алгоритма.

В одном университете в среднем насчитывается 6 факультетов, каждый из них может иметь 5 и более кафедр, ответственных за группы студентов (причем на каждой кафедре более 3 групп). Если подсчитать объем информации, проходящей через систему дистанционного образования за один год со всех групп и факультетов, то количество информации составляет более 21 гигабайта, не считая значительного объема других данных, циркулирующих в системе.

Этот аспект может вызвать сложности в эффективном управлении большим объемом информации, передаваемой через систему дистанционного образования за год, что в свою очередь создает трудности в хранении, обработке и мониторинге всех работ, отправленных различными группами и факультетами.

Решением данной проблемы является кодирование информации, что поможет уменьшить объем, занимаемый лабораторными и практическими работами студентов.

### Описание программы-компрессора

Программа включает в себя классы и функции, которые можно увидеть на диаграмме классов (рис. 2).

Описание классов:

1. Классы побитового чтения/побитовой записи файлов (BitFileWriter, BitFileReader). Классы, обеспечивающие побитовый ввод данных в файл и вывод данных из файла.

2. Класс кода Хаффмана (HuffmanCode). Класс, предназначенный для хранения сколь угодно длинных последовательностей нулей и единиц.

3. Класс кодового дерева (HuffmanTree). Класс, предназначенный для кодирования информации с помощью алгоритма Хаффмана.

4. Функция сжатия файла (compressFile). Функция, предназначенная для создания двух файлов:

– файла, содержащего последовательность кодов Хаффмана, полученного из исходного файла;

– текстового файла, содержащего расширение исходного файла и массив частот байтов, необходимый для построения дерева Хаффмана при декомпрессии.

5. Функция восстановления исходного файла (decompressFile). Функция предназначена для восстановления исходного файла.

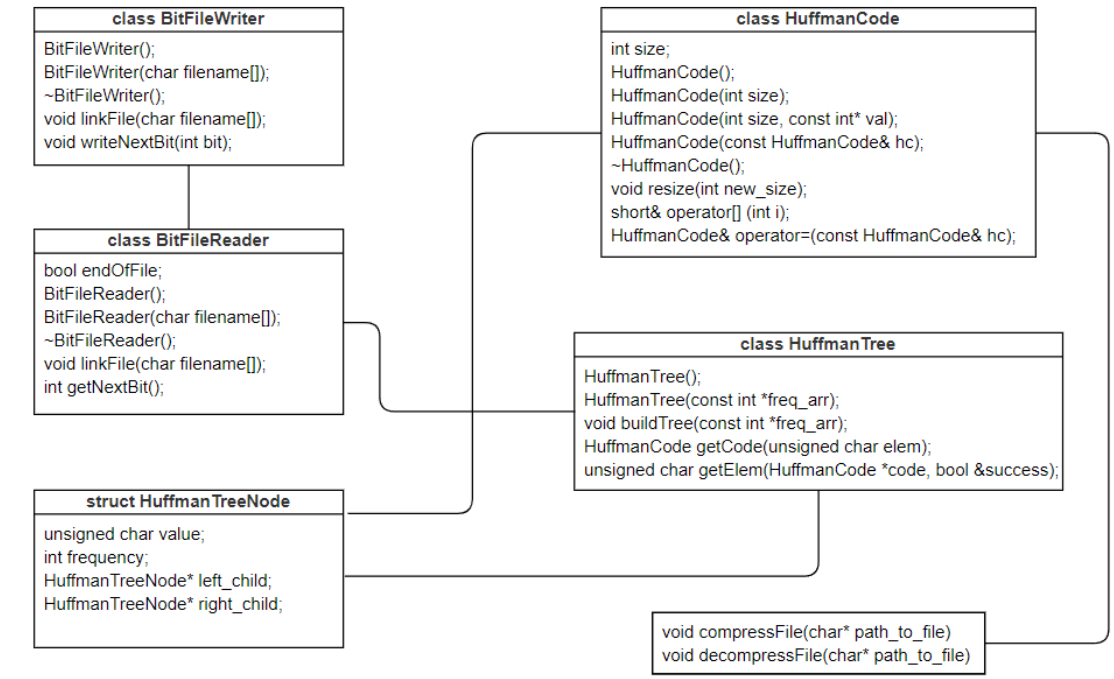



Рис. 2. Диаграмма классов программы

### Оценка эффективности алгоритма

Дать асимптотическую оценку эффективности работы созданной программы в зависимости от размера входных данных — достаточно сложная задача. Скорость работы разработанной реализации алгоритма Хаффмана зависит от размера исходного файла, а также его содержания (частот байтов).

Для изучения эффективности была проведена серия тестов, включающая использование различных файлов с лабораторными и практическими работами по различным дисциплинам. Результаты исследования представлены в таблице 1.

Таблица 1

Эффективность сжатия текстовых файлов в зависимости от их размера

| **Объем исходного файла, КБ** | **Объем сжатого файла, КБ** | **Объем файла с частотами, КБ** | **Суммарный объем после сжатия, КБ** | **Фактор сжатия** |
|---|---|---|---|---|
| 2 | 1,17 | 0,81 | 1,98 | 0,99 |
| 4 | 2,37 | 0,83 | 3,20 | 0,80 |
| 8 | 4,57 | 0,84 | 5,41 | 0,67 |
| 16 | 9,14 | 0,87 | 10,01 | 0,62 |
| 32 | 17,85 | 0,90 | 18,75 | 0,58 |
| 64 | 35,67 | 0,93 | 36,60 | 0,57 |
| 128 | 70,30 | 0,97 | 71,27 | 0,56 |
| 256 | 138,80 | 1,00 | 139,80 | 0,55 |
| 512 | 281,70 | 1,00 | 282,70 | 0,55 |
| 1 024 | 553,40 | 1,10 | 554,50 | 0,54 |

Из данных в таблице 1 следует, что эффективность сжатия текстовых файлов растет с увеличением объема сжимаемого файла. При достаточно больших объемах достигается сжатие примерно в два раза.

Измерения временных затрат на компрессию/декомпрессию текстовых файлов разного объема приведены в таблице 2.

Таблица 2

Время компрессии/декомпрессии текстовых файлов

| **Объем исходного файла, КБ** | **Время компрессии, мс** | **Время декомпрессии, мс** |
|---|---|---|
| 2 | 6 | 1 |
| 4 | 6 | 3 |
| 8 | 10 | 3 |
| 16 | 20 | 5 |
| 32 | 33 | 8 |
| 64 | 65 | 22 |
| 128 | 134 | 29 |
| 256 | 260 | 66 |
| 512 | 523 | 120 |
| 1 024 | 1 051 | 233 |

Из данных таблицы 2 можно выявить линейную зависимость времени сжатия и восстановления файлов программой от объема исходного файла.

## Заключение

Разработанная программа оперирует байтами в качестве кодируемых символов и, следовательно, эффективна лишь для сжатия текстовой информации. Аудио-, графические и видеоданные содержат байты с приблизительно одинаковой частотой, поэтому для сжатия нетекстовой информации предпочтительнее применять специализированные методы.

При кодировании текстовых файлов, содержащих осмысленный текст на русском языке, при достаточно большом объеме сжимаемых файлов достигается сжатие примерно в два раза, что поможет уменьшить объем, занимаемый работами студентов.

Одним из недостатков реализованной программы является необходимость использования отдельного файла для хранения расширения исходного файла и таблицы частот символов. Этот недостаток можно исправить, включив указанную информацию в сжатый файл, однако это усложнит процесс чтения файла. При обработке больших объемов данных (более 2-3 МБ) скорость работы программы становится низкой из-за неэффективного метода получения кода символа из дерева кодов.

## Литература

1. Garg, N. Computer Sc — Data Structures and Algorithms: Lecture Series on Data Structures and Algorithms. Lecture 19. Data Compression. URL: http://www.youtube.com/watch?v=5wRPin4oxCo&list=PLBF3763AF2E1C572F&index=19&t=251s (дата обращения 18.11.2023).

2. Виноградова, М. С. Сжатие данных. Алгоритм Хаффмана / М. С. Виноградова, О. С. Ткачева // Modern European Researches. 2022. Is. 3-1. Pp. 60–69.

3. Сэломон, Д. Сжатие данных, изображений и звука = A Guide to Data Compression Methods / Пер. с англ. В. В. Чепыжова. — Москва: Техносфера, 2004. — 368 с. — (Мир программирования. Цифровая обработка сигналов).

4. Петрянин, Д. Л. Сжатие текстовых данных / Д. Л. Петрянин, Е. А. Сидорова, А. В. Зорькин // Новые информационные технологии в автоматизированных системах: Материалы семнадцатого научно-практического семинара. — Москва: ИПМ им. М. В. Келдыша, 2014. — С. 505–507.

5. Марков, А. А. Введение в теорию кодирования. — Москва: Наука. Главная редакция физико-математической литературы, 1982. — 192 с.

6. Variable-Length Code // Wikipedia. — Обновлено 11.11.2023. URL: http://en.wikipedia.org/wiki/Variable-length_code (дата обращения 19.11.2023).

7. Huffman Coding // Wikipedia. — Обновлено 20.11.2023. URL: http://en.wikipedia.org/wiki/Huffman_coding (дата обращения 22.11.2023).

8. Охотин, А. Математические основы алгоритмов. Курс лекций. URL: http://users.math-cs.spbu.ru/~okhotin/teaching/algorithms_2019 (дата обращения 22.11.2023).

9. Алгоритм Хаффмана // Викиконспекты. — Обновлено 04.09.2022. URL: http://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Хаффмана (дата обращения 22.11.2023).

10. Левитин, А. В. Алгоритмы. Введение в разработку и анализ = Introduction to The Design and Analysis of Algorithms / пер. с англ. и ред. С. Г. Тригуб, И. В. Красикова. — Москва: Вильямс, 2006. — 576 с.

DOI: 10.24412/2413-2527-2023-436-42-46

# Development of a Data Compression Algorithm Based on the Huffman Method

R. M. Slesarev, M. A. Dobrovolskaya, PhD A. V. Zabrodin
Emperor Alexander I St. Petersburg State Transport University
Saint Petersburg, Russia
romsyt2002@gmail.com, dobro-7878@mail.ru, teach-case@yandex.ru

***Abstract.*** **The article is devoted to the development of an algorithm for compressing data of various types, represented as files in the external memory of a computer, based on the Huffman method. The Huffman algorithm of minimal prefix coding is considered in detail. The purpose of writing this paper is to study and compare various data compression algorithms, as well as the practical application of the algorithm based on the Huffman method for writing a file compressor program.**

***Keywords:*** **lossless compression, lossy compression, compression algorithms, coding, code tree.**

REFERENCES

1. Garg, N. Computer Sc — Data Structures and Algorithms: Lecture Series on Data Structures and Algorithms. Lecture 19. Data Compression. URL: http://www.youtube.com/watch?v=5wRPin4oxCo&list=PLBF3763AF2E1C572F&index=19&t=251s (accessed 18 Nov 2023).

2. Vinogradova M. S., Tkacheva O. S. Data Compression. Huffman Algorithm [Szhatie dannykh. Algoritm Khaffmana], *Modern European Researches*, 2022, Is. 3-1, Pp. 60–69.

3. Salomon D. A Guide to Data Compression Methods [Szhatie dannykh, izobrazheniy i zvuka], Moscow, Tekhnosfera Publishing House, 2004, 368 p.

4. Petryanin D. L., Sidorova E. A., Zorkin A. V. Text Data Compression [Szhatie tekstovykh dannykh], *New Information Technologies in Automated Systems: Materials of the 17th Scientific and Practical Seminar [Novye informatsionnye tekhnologii v avtomatizirovannykh sistemakh: Materialy semnadtsatogo nauchno-prakticheskogo seminara]*. Moscow, Keldysh Institute of Applied Mathematics of RAS, 2014, Pp. 505–507.

5. Markov A. A. Introduction to coding theory [Vvedenie v teoriyu kodirovaniya]. Moscow, Nauka Pulishers, 1982, 192 p.

6. Variable-Length Code, *Wikipedia*. Last update at November 11, 2023. Available at: http://en.wikipedia.org/wiki/Variable-length_code (accessed 19 Nov 2023).

7. Huffman Coding, *Wikipedia*. Last update at November 20, 2023. Available at: http://en.wikipedia.org/wiki/Huffman_coding (accessed 22 Nov 2023).

8. Okhotin A. Mathematical foundations of algorithms: A course of lectures [Matematicheskie osnovy algoritmov. Kurs lektsiy] Available at: http://users.math-cs.spbu.ru/~okhotin/teaching/algorithms_2019 (accessed 22 Nov 2023).

9. Huffman Algorithm [Algoritm Khaffmana], *Wikinotes [Vikikonspekty]*. Last update at September 04, 2022. Available at: http://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Хаффмана (accessed 22 Nov 2023).

10. Levitin A. V. Introduction to The Design and Analysis of Algorithms [Algoritmy. Vvedenie v razrabotku i analiz]. Moscow, Williams Publishing House, 2006, 576 p.