УДК 519.6

Особенности кодирования текста с помощью алгоритма Хаффмана

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Кузьмина Богдана Сергеевна
Приамурский государственный университет имени Шолом-Алейхема
К.т.н., доцент кафедры информационных систем, математики и методик обучения

Аннотация
В этой статье изложены базовые концепции кода Хаффмана. Описаны особенности использования алгоритма Хаффмана для кодирования текста. Представлен практический пример преобразования текста в код Хаффмана. Охарактеризована полезность метода Хаффмана при сжатии данных.
Ключевые слова: Сжатие данные, код Хаффмана, бинарное дерево, алгоритм.

Features of text encoding using the Huffman algorithm

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
Student

Kuzmina Bogdana Sergeevna
Sholom-Aleichem Priamursky State University
candidate of technical sciences, associate professor of the Department of Information Systems, Mathematics and training methodic

Abstract
This article outlines the basic concepts of the Huffman code. Features of using the Huffman algorithm for text encoding are described. A practical example of converting text into Huffman code is presented. The usefulness of the Huffman method for data compression is characterized.
Keywords: Compression data, Huffman code, binary tree, algorithm.

В современном мире постоянно увеличивается количество информации, передаваемой по сети интернет. И проблема сжатия данных каждый год становится более острой. Одним из множества решений этой проблемы это применение алгоритма Хаффмана.

Метод кодирования Хаффмана широко известен. Так, К.А.Шмалева в работе «Код Хаффмана» [5] рассматривает метод кодирования информации с

помощью метода Дэвида Хаффмана на примере передачи цифрового сообщения. М.О.Смирнова и А.П.Смирнов в своей публикации «Программный продукт для демонстрации применения оптимального кодирования на примере алгоритмов Шеннона-Фано и Хаффмана» [4] подробно описывают программный продукт, с помощью которого можно продемонстрировать применение оптимальных кодов. В.Э.Родионов, Т.В.Чекулаева в статье «Сжатие данных с использованием кодирования Хаффмана» [3] показывают возможности практического применения алгоритма Хаффмана. Разработанная А.О.Кизяновым программа для автоматического преобразования текста в код Хаффмана была представлена в работе [2].

Рассмотрим особенностей работы кодов Хаффмана.

Есть много путей отображения набора символов, в компьютере это байты и поэтому текст на компьютере представляет из себя текст в виде бит. Например, ANSI является таблицей, которая содержит латинский алфавит и отображает каждый символ в виде одного байта. Когда вы получаете файл в ANSI кодировке, он содержит байты, которые при сканировании переводятся в символы. Одинаковое количество бит необходимо для представления каждого символа, но это не является оптимальным, потому что часто используемые символы получают такое же количество памяти, как и редко используемые символы.

Некоторые методы кодировок кодируют символы в битах, так что различные символы отображаются с одинаковым размером битов. Это очень неэффективно из-за использования часто встречаемых символов такой же длиной битов, как и редко встречаемых символов. Одно из решений этой проблемы является использование минимальных по длине битов для представления различий между битов, представляющих различные символы, и другое решение — это использовать бинарные деревья, как дерево Хаффмана.

Префиксный код представляет собой тип кода, который позволит вам расшифровать закодированный текст без специальных маркеров. Например, отображение {a = 0, b = 10, c = 11} является префиксом кодом. Если у вас есть «000101011», то ясно, когда один список битов для символа заканчивается и начинается другой, вы переводите его непосредственно в «aaabbc». Другой пример, отображение {a = 0, b = 1, c = 11} не является префиксом кодом в строке «111» может быть переведено в «BBB» или «BC», то есть, нам нужен разделитель маркера между списком битов в этом коде.

Пусть кодирование представляет таблицу символов c со списком битов. Определим функцию « длина (c)» как число битов, представляющих символ c, а функцию «частота (c)» как частоту, с которой символ c употребляется в тексте. Тогда эффективность кода может быть вычислена по формуле:

$$\sum_{c \epsilon \text{Алфавит}} \text{длина(c)} \times \text{частота}(c) = \text{Код Хаффмана}$$

Таким образом, средний вес кодирования длин зависит от их частот.

Каждый префиксный код также может быть представлен в виде бинарного дерева, где каждое ребро помечаются как «0» или «1». На рис. 1, показан пример такого бинарного дерева.

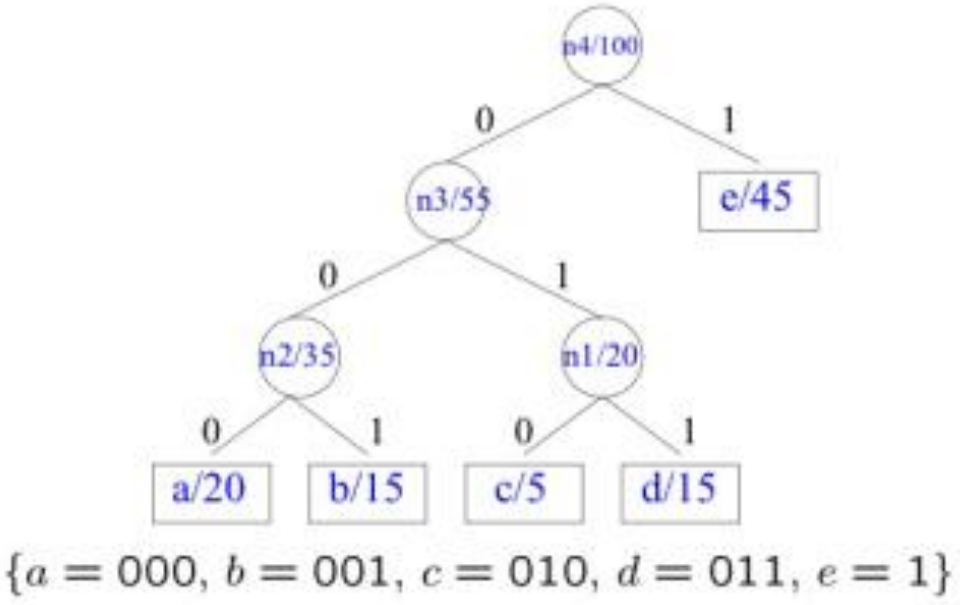

Рис. 1 Разложение по бинарному дереву

Глубина представляет собой размер кода и, следовательно, обеспечивает альтернативу эффективного кода на языке деревьев:

$$B(T) = \sum_{c \epsilon \text{Алфавит}} \text{глубина}(c) \times \text{частота}(c)$$

где глубина(с), глубина с в дереве.

Алгоритм построения дерева Хаффмана включает в себя 4 основных шага [1].

1 шаг. Создание узла для каждого символа с его частотой и добавление его в список.

2 шаг. Удаление из списка двух узлов с минимальными частотами. Затем создание дерева, где два узла являются «братья и сестры» и их родителей помечены как сумма их частот.

3 шаг. Добавление корневого узла в список (список теперь короче на один элемент).

4 шаг. Повторение шага 3 до тех пор, пока 1 узел не останется в списке. Этот узел будет корень дерева Хаффмана.

Рассмотрим данный алгоритм на примере, в котором сначала указана частота буквы, а потом сама буква {20 «a», 15 «b», 5 «c», 15 «d», 45 «e»}.

Шаг 1. Преобразуем данный список в список, удобный для обработки:

{a / 20, b / 15, c / 5, d / 15, e / 45}.

Итерация 1. Шаг 2. Выбираем два минимальных узла, например, «c» и «d». Можно также взять пару «c» и «b» в качестве узлов минимальных частот. Создаем дерево (см. рис. 2).

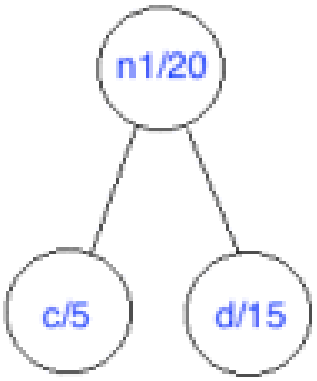

Рис. 2. Создание узла из двух других узлов

Назовем этот узел как «n1» для последующей маркировки. Заметим, что его частота составляет 20 = 5 + 15.

Шаг 3. Добавляем «n1» в список. Получается следующий список:

{a / 20, n1 / 20, b / 15, e / 45}.

Итерация 2

Шаг 2. Аналогично выбираем «a» и «b» и делаем новый «n2» узел и дерево (см. рис. 3).

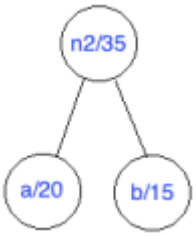

Рис. 3. Создание узла из двух других узлов

Шаг 3. Получаем список: {n2 / 35, n1 / 20, e / 45}.

Итерация 3. Шаг 2.

Соединяем в одно дерево n2 / n1 и 35/20 в соответствии с новым узлом n3 / 55 (см. рис. 4).

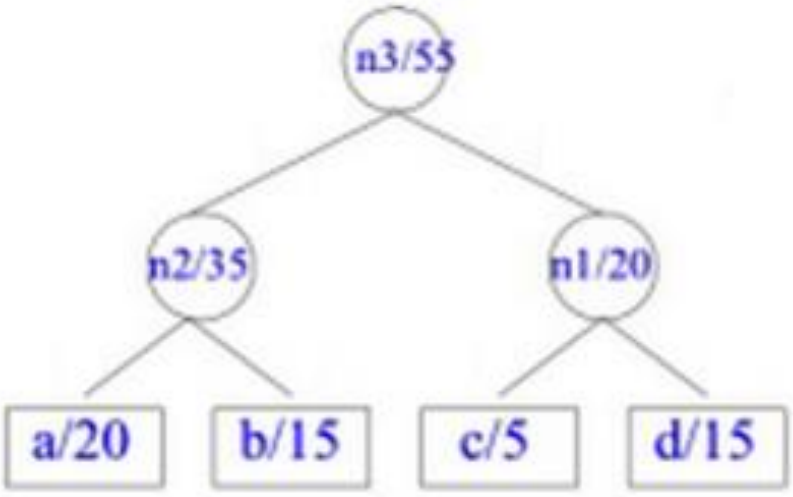

Рис. 4. Полное дерево из новых узлов

Шаг 3. Получаем итоговый список: {n3 / 55, e / 45}.

Понятно, что после 4-й итерации дерево Хаффмана будет закончено.

Заметим, что в рассмотренном примере оптимальное кодирование означает следующее. Если взять фиксированный алфавит с известными частотами, то код Хаффмана будет иметь минимальное значение эффективности кода, рассчитанные выше по сравнению со всеми возможными кодами, доступными для этого алфавита.

Алгоритм Хаффмана широко применяется в сжатии данных, в том числе и сжатии фото и видеоизображении (JPEG, MPEG), в популярных архиваторах (ZIP) и протоколе передачи данных(HTTP, HTTPS).

Библиографический список

1. Алгоритм Хаффмана URL: http://metod.vt.tpu.ru/lab/huffman/theory.html (Дата обращения: 6.06.2017)
2. Кизянов А.О. Реализация алгоритма Хаффмана на языке программирования Python // Постулат. 2017. №5. URL: http://e-postulat.ru/index.php/Postulat/article/view/617/638 (Дата обращения: 6.06.2017)
3. Родионов В.Э., Чекулаева Т.В. Сжатие данных с использованием кодирования Хаффмана // Новая наука: теоретический и практический взгляд. 2016. №9. С. 141-143. URL: https://elibrary.ru/item.asp?id=26563912 (Дата обращения: 28.05.2017)
4. Смирнова М. О., Смирнов А. П. программный продукт для демонстрации применения оптимального кодирования на примере алгоритмов Шеннона-Фано и Хаффмана // Прикаспийский журнал: управление и высокие технологии. 2010. №2. С. 33-40. URL: https://elibrary.ru/item.asp?id=15220697 (Дата обращения: 28.05.2017)
5. Шмалева К.А. Код Хаффмана // Современные проблемы управления и регулирования: инновационные технологии и техника. 2016. С. 7-11. URL: https://elibrary.ru/item.asp?id=26199791 (Дата обращения: 28.05.2017)