Практическая работа 1
Изучение алгоритма сжатия Хаффмана

1.1. Цель работы

Изучить алгоритм оптимального префиксного кодирования Хаффмана и его использование для сжатия сообщений.

1.2. Теоретические сведения

Алгоритм Хаффмана — адаптивный жадный алгоритм оптимального префиксного кодирования алфавита с минимальной избыточностью. Был разработан в 1952 году аспирантом Массачусетского технологического института Дэвидом Хаффманом.

Сжатие данных по Хаффману применяется при сжатии фото- и видеоизображений (JPEG, стандарты сжатия MPEG), в архиваторах (PKZIP, LZH и др.), в протоколах передачи данных MNP5 и MNP7.

Жадный алгоритм (Greedy algorithm) — алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Идея алгоритма Хаффмана состоит в следующем: зная вероятности символов в сообщении, можно описать процедуру построения кодов переменной длины, состоящих из целого количества битов. Символам с большей вероятностью ставятся в соответствие более короткие коды. Коды Хаффмана обладают свойством префиксности (то есть ни одно кодовое слово не является префиксом другого), что позволяет однозначно их декодировать.

Этот метод кодирования состоит из двух основных этапов:

1. Построение оптимального кодового дерева.
2. Построение отображения код-символ на основе построенного дерева.

Классический алгоритм Хаффмана на входе получает таблицу частот встречаемости символов в сообщении. Далее на основании этой таблицы строится дерево кодирования Хаффмана (H-дерево).

1. Символы входного алфавита образуют список свободных узлов. Каждый лист имеет вес, который может быть равен либо вероятности, либо количеству вхождений символа в сжимаемое сообщение.
2. Выбираются два свободных узла дерева с наименьшими весами.
3. Создается их родитель с весом, равным их суммарному весу.
4. Родитель добавляется в список свободных узлов, а два его потомка удаляются из этого списка.

5. Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой — бит 0.

6. Шаги, начиная со второго, повторяются до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться корнем дерева.

Для примера рассмотрим кодирование фразы

```
пупкин василий кириллович
```

Можно видеть, что фраза состоит из $L = 25$ символов.

Первоначально необходимо сформировать алфавит фразы и построить таблицу количества вхождений и вероятности символов. Она представлена в табл. 1.1.

Таблица 1.1

Алфавит фразы и вероятности символов

Алфавит	п	у	к	и	н	« »	в
Кол. вх.	2	1	2	6	1	2	2
Вероятн.	0.08	0.04	0.08	0.24	0.04	0.08	0.08
Алфавит	а	с	л	й	р	о	ч
Кол. вх.	1	1	3	1	1	1	1
Вероятн.	0.04	0.04	0.12	0.04	0.04	0.04	0.04

Алфавит составляют $N_A = 14$ символов, включая знак пробела (« »).

Далее сортируем алфавит в порядке убывания вероятности появления символов (табл. 1.2).

Таблица 1.2

Алфавит фразы, отсортированный по вероятности (количеству вхождений)

Алфавит	и	л	п	к	« »	в	у
Кол. вх.	6	3	2	2	2	2	1
Вероятн.	0.24	0.12	0.08	0.08	0.08	0.08	0.04
Алфавит	н	а	с	й	р	о	ч
Кол. вх.	1	1	1	1	1	1	1
Вероятн.	0.04	0.04	0.04	0.04	0.04	0.04	0.04

Последовательно строим дерево кодирования, начиная с символов с наименьшими весами (вероятностями), пока не достигнем корня, который будет иметь вес равный 1 (по вероятности). Построение дерева показано на рис. 1.1.

Для удобства перерисуем построенное дерево, упорядочив его слева-направо, начиная с корня. Каждой ветви дерева присвоим свой код. При

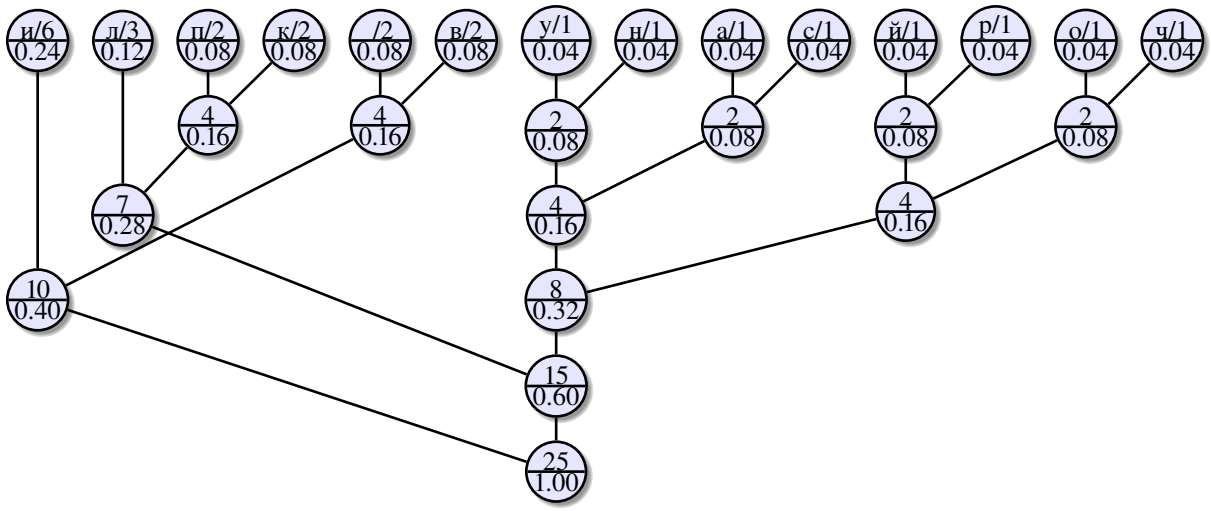

Рис. 1.1. Дерево кодирования Хаффмана

этом, ветвь, идущая выше, будет иметь код «0», а ветвь, идущая ниже — «1». Упорядоченное дерево показано на рис. 1.2.

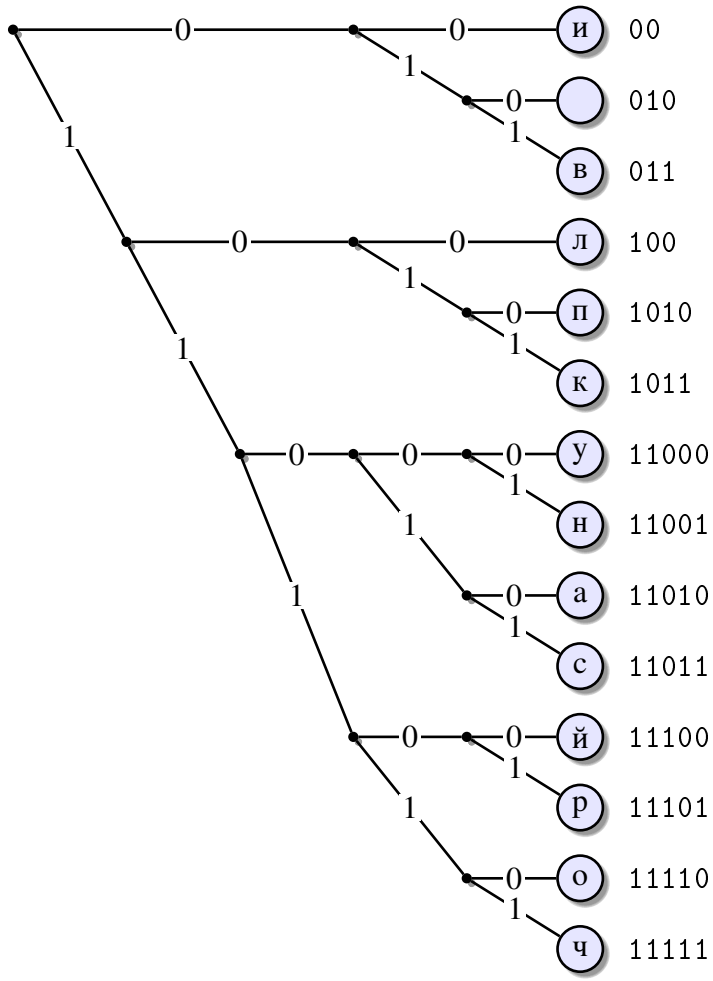

Рис. 1.2. Упорядоченное дерево кодирования Хаффмана

В итоге получается, что символам с большей вероятностью появления соответствуют более короткие коды.

Теперь можно закодировать исходную строку. Она будет иметь вид:

```
  п     у     п     к    и    н    « »   в     а     с    и   л   и    й
1010 11000 1010 1011 00 11001 010 011 11010 11011 00 100 00 11100
« »   к    и    р    и   л   л    о    в   и    ч
010 1011 00 11101 00 100 100 11110 011 00 11111
```

Для декодирования кодовой строки достаточно идти по упорядоченному дереву на рис. 1.2, до получения символа. Затем производится возврат к корню и определяется следующий символ.

Рассчитаем коэффициент сжатия относительно использования кодировки ASCII (8 бит/символ).

$L_{\text{ASCII}} = 8 \cdot 25 = 200$ бит.

$L_{\text{Huff}} = 6 \cdot 2 + 3 \cdot 3 + 2 \cdot 2 \cdot 3 + 2 \cdot 2 \cdot 4 + 8 \cdot 5 = 89$ бит.

Следовательно, коэффициент сжатия будет равен

$$K_{\text{сж}} = \frac{L_{\text{ASCII}}}{L_{\text{Huff}}} \approx 2.247$$

Коэффициент сжатия относительно равномерного кода (5 бит/символ, т. к. у нас всего 25 символов) будет равен

$$K_{\text{сж}} = \frac{5 \cdot 25}{L_{\text{Huff}}} = \frac{125}{89} \approx 1.404$$

Рассчитаем *среднюю длину полученного кода* по формуле

$$l_{\text{ср}} = \sum_s p_s \cdot l_s,$$

где s — множество символов алфавита; p_s — вероятность появления символа; l_s — количество бит в коде символа.

Для полученного кода средняя длина будет равна

$$l_{\text{ср}} = 0.24 \cdot 2 + 0.12 \cdot 3 + 2 \cdot 0.08 \cdot 3 + 2 \cdot 0.08 \cdot 4 + 8 \cdot 0.01 \cdot 5 = 2.36 \text{ бит/символ.}$$

Поскольку при построении дерева кода Хаффмана может возникнуть некоторый произвол, для выбора оптимального варианта кода используют *дисперсию*. *Дисперсия* показывает насколько сильно отклоняются длины индивидуальных кодов от их средней величины. Лучшим будет код с наименьшей дисперсией.

Дисперсия рассчитывается по формуле

$$\delta = \sum_s p_s (l_s - l_{\text{ср}})^2.$$

Для полученного кода дисперсия будет равна

$$\begin{aligned}\delta = \; & 0.24 \cdot (2-2.36)^2 + 0.12 \cdot (3-2.36)^2 + 2 \cdot 0.08 \cdot (3-2.36)^2 + \\ & + 2 \cdot 0.08 \cdot (4-2.36)^2 + 8 \cdot 0.01 \cdot (5-2.36)^2 = 1.1337\end{aligned}$$

Для уменьшения дисперсии кода существует правило: *когда на дереве имеется более двух узлов с наименьшей вероятностью, следует объединять символы с наибольшей и наименьшей вероятностью; это сокращает общую дисперсию кода*.

1.3. Порядок выполнения задания

1. В качестве исходной строки текста выбрать «Фамилия Имя Отчество» студента.
2. Сформировать алфавит фразы, посчитать количество вхождений символов и их вероятности появления (см. табл. 1.1).
3. Отсортировать алфавит в порядке убывания вероятности появления символов (табл. 1.2).
4. Построить дерево кодирования (см. рис. 1.1).
5. Упорядочить построенное дерево слева-направо (при необходимости). Присвоить ветвям коды. Определить коды символов (см. рис. 1.2).
6. Закодировать исходную строку.
7. Рассчитать коэффициенты сжатия относительно кодировки ASCII и относительно равномерного кода.
8. Рассчитать среднюю длину полученного кода и его дисперсию.

1.4. Контрольные вопросы

1. Порядок работы алгоритма Хаффмана.
2. Построение оптимального кодового дерева.
3. Средняя длина кода и ее расчет.
4. Дисперсия кода и ее расчет.