Молдамурат Х. - кандидат технических наук, доцент
Абикенова Т. З., магистрант
Казахского университета технологии и бизнеса,
г. Астана, Казахстан

ПРИНЦИП ЭФФЕКТИВНОГО КОДИРОВАНИЯ ПО ХАФФМАНУ НА ПРАКТИКЕ

Использование одного из методов эффективного кодирования информации (принцип Хаффмана) на примере реализации алгоритма кодирования.

Use of one of methods of effective coding of information (Huffman's principle) on the example of realization of algorithm of coding.

Алгоритм основан на том факте, что некоторые символы из стандартного 256-символьного набора в произвольном тексте могут встречаться чаще среднего периода повтора, а другие, соответственно, – реже. Следовательно, если для записи распространенных символов использовать короткие последовательности бит, длиной меньше 8, а для записи редких символов – длинные, то суммарный объем файла уменьшится.

Хаффман предложил очень простой алгоритм определения того, какой символ необходимо кодировать каким кодом для получения файла с длиной, очень близкой к его энтропии (то есть информационной насыщенности) [1].

Сжатие данных производится в два этапа. На первом этапе считываются данные, которые должны подвергнуться сжатию; на втором определяется частота встречаемости отдельных байтов данных, которые могут принимать значения от 0 до 255.

Рассмотрим пример. Пусть требуется сжать словосочетание:

ПРОГРАММИРОВАНИЕ КОМПРЕССИИ
БЕЗ ПОТЕРЬ

До сжатия данных каждая буква представляется числом, которое лежит между 0 и 255. Если применяется так называемая альтернативная кодовая таблица (содержащая знаки русского алфавита), то буквы имеют значения между 128 и 159 (а-я), между 160 и 175 (а-п) и между 224 и 239 (Р-я), а пробел - значение 32. Запись приведенного словосочетания потребовала бы 38 байт - по одному байту на букву или пробел. Чтобы снизить размеры файла при записи, определим вначале частоту встречаемости отдельных букв:

Буква	Частота
П	3
Р	5
О	4
Г	1
А	2
М	3
И	4
В	1
Н	1
Е	4
К	1
С	2

№ п/п	Символ	Повтор	Частота
1	Р	5	0.132
	О	4	0.105
3	И	4	0.10
4	Е	4	0.105
5	П	3	0.079
6	М	3	0.079
7	ПРОБЕЛ	3	0.079
8	А	2	0.0526
9	С	2	0.0526
10	Г	1	0.0263
11	В	1	0.0263
12	К	1	0.0263

Б	1
З	1
Т	1
Ь	1
Пробел	3

13	Б	1	0.0263
14	З	1	0.0263
15	Т	1	0.0263
16	Ь	1	0.0263
17	Н	1	0.0263

Очевидно, что наиболее часто встречается буква **Р,** а другие буквы встречаются реже, остальные вообще не встречаются. Конечно, весьма расточительно кодировать каждую букву одинаковым числом бит. Было бы гораздо экономичнее кодировать наиболее часто встречающиеся буквы короткими кодами, например, 0 или 1. При этом на кодирование затрачивался бы всего один бит, т.е. в 8 раз меньше, чем согласно использованной кодовой таблице. Соответственно, по мере снижения частоты букв для их кодирования можно было бы использовать все более длинные численные значения. В этом заключается принцип кодирования по Хаффману [2].

После определения частоты встречаемости знаков строится схема кодирования, в которой наиболее часто встречаемые символы кодируются меньшим числом бит, и, наоборот, символы, частота встречаемости которых меньше, кодируются большим числом бит. Задача состоит в том, чтобы найти эффективную схему кодирования /декодирования. В сжатый файл необходимо записать поток битов и информацию о том, как этот поток следует интерпретировать. В этом потоке битов отдельные знаки представляются уже не фиксированным, а переменным числом битов, причем количество кодирующих битов уменьшается с ростом частоты встречаемости символа.

Рассмотрим пример реализации алгоритма кодирования.

Построим таблицу частоты повторяемости символов, деля число повторений символа на длину сообщения (38):

Произведем ряд последовательных преобразований исходной таблицы, складывая строки с наименьшей частотой и проводя их сортировку по убыванию

Шаг 1	
1	0.132
2	0.105
3	0.105
4	0.105
5	0.079
6	0.079
7	0.079
8	0.0526
9	0.0526
10,11	0.0526
12,13	0.0526
14,15	0.0526
16,17	0.0526

Шаг 2	
1	0.132
2	0.105
3	0.105
4	0.105
8,9	0.105
10,11,12,13	0.105
14,15,16,17	0.105
5	0.079
6	0.079
7	0.079

Шаг 3	
6,7	0.158
1	0.132
2	0.105
3	0.10
4	0.105
8,9	0.105
10,11,12,13	0.105
14,15,16,17	0.105
5	0.079

Шаг 4	
5,14,15,16,17	0.184
6,7	0.158
1	0.132
2	0.105
3	0.105
4	0.105
8,9	0.105
10,11,12,13	0.105

Шаг 5	
8,9,10,11,12,13	0.210
3,4	0.210
5,14,15,16,17	0.184
6,7	0.158
1	0.132
2	0.105

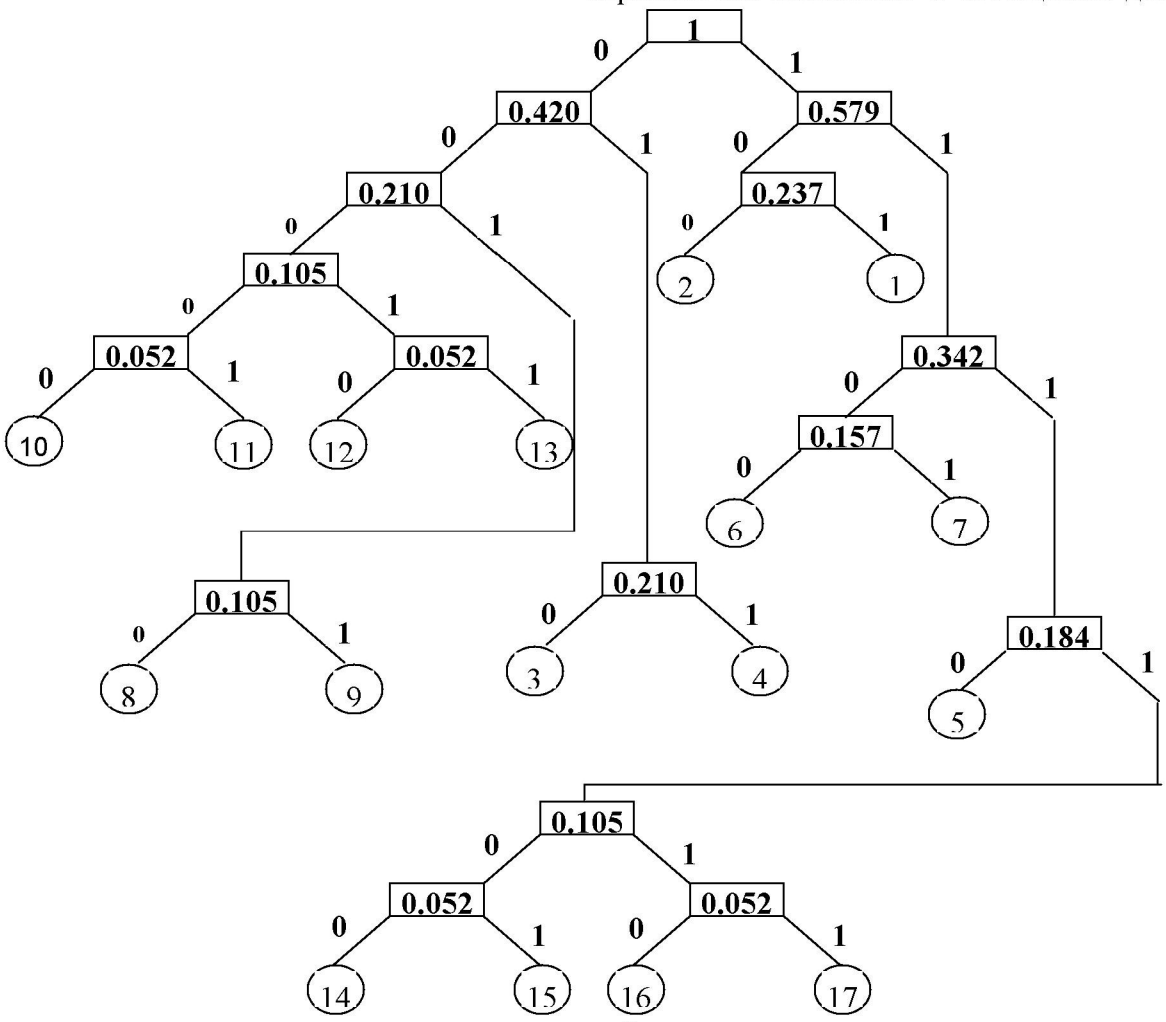
Шаг 6	
1,2	0.237
8,9,10,11,12,13	0.210
3,4	0.210
5,14,15,16,17	0.184
6,7	0.158

Шаг 7	
5,6,7,14,15,16,17	0.342
1,2	0.237
8,9,10,11,12,13	0.210
3,4	0.210

Шаг 8	
3,4,8,9,10,11,12,13	0.420
5,6,7,14,15,16,17	0.342
1,2	0.237

Шаг 9	
1,2,5,6,7,14,15,16,17	0.579
3,4,8,9,10,11,12,13	0.420

вероятности появления в сообщении дан-



ной группы символов.

Теперь, пользуясь таблицами в обратной последовательности, построим дерево кодирования. Из вершины начинаются две ветви, соответствующие значениям

Ветви помечаются значениями 0 (левая, с меньшим значением слагаемого вероятности) и 1 (правая, с большим). Дере-

во закончится каждым символом алфавита сообщения, а значения, приписанные ветвям дерева на пути от вершины к данному символу, образуют двоичный код данного символа

По результатам данных операций можно составить таблицу кодирования

№ п/п	Символ	Двоичный код
1	Р	101
2	О	100
3	И	010
4	Е	011
5	П	1110
6	М	1100
7	ПРОБЕЛ	1101
8	А	0010
9	С	0011
10	Г	00000
11	В	0000
12	К	00010
13	Б	00011
14	З	111100
15	Т	111101
16	Ь	111110
17	Н	111111

Подсчитаем полученную степень сжатия, умножая длину кода на сумму символов, кодируемых кодом данной длины:

3(5+4+4+4) + 4(3+3+3+2+2) +5(1+1+1+1) + 6(1+1+1+1) = 147 бит (18.4 байт)

Таким образом, имеем коэффициент сжатия К=38/19=2

Проверим декодирование сообщения. Например, кодируется слово ПРОГРАММА:

11101011000000010100101100110000010

При декодировании просмотр начинается слева на выявление возможных 3-битовых комбинаций, затем 4-, 5-, и 6-. В нашем случае комбинация 111 отсутствует в кодовой таблице, а в списке 4-битовых имеется комбинация 1110, которая декодируется как буква П. Просмотр следующих битов в сообщении производится аналогичным образом.

Граница применимости этой схемы сжатия очевидна: данные можно сжимать, только если отдельные элементы данных различаются по частоте встречаемости. Если же элементы данных распределены статистически равномерно, то сжатие невозможно.

Применительно к растровой графике: если изображение имеет равномерно окрашенные области – выигрыш будет значительным.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шаврин Ю.А. Информационные технологии: Учебное пособие. Том 1: Основы информатики и информационных технологии. Том 2: Офисная технология и информационные системы. Серия: Информатика. – Москва, 2007.

2. Цымбал В.П. Задачник по теории информации и кодированию. - Москва: Высшая школа, 1994. – 76 с.