Это означает, что для выполненной кодировки одному символу исходного алфавита в среднем соответствует 2,1 бита.

Д. Хаффман составил дерево для алфавита английского языка, которое является оптимальным (наилучшим из возможных, так как ему соответствует наименьшее значение *C*). В таблице представлены эти коды.

Бинарные коды английского алфавита в соответствии с деревом Хаффмана

Символ алфавита	Бинарный код	Символ алфавита	Бинарный код
E	100	M	00011
T	001	U	00010
A	1111	G	00001
O	1110	Y	00000
N	1100	P	110101
R	1011	W	011101
I	1010	B	011100
S	0110	V	1101001
H	0101	K	110100011
D	11011	X	110100001
L	01111	J	110100000
F	01001	Q	1101000101
C	01000	Z	1101000100

В текстах на английском языке наиболее часто встречается буква «e» (ей соответствует вероятность 0,13).

Арифметический метод сжатия. Рассмотренный выше метод Хаффмана эффективен, когда частоты появления символов пропорциональны $1/2^i$ (где i – натуральное положительное число). Действительно, код Хаффмана для каждого символа всегда состоит из *целого числа бит*. В случае, когда частота появления символа равна 0,2, оптимальный код, соответствующий этому символу, должен иметь длину $\log_2(0{,}2) = 2{,}3$ бита. Понятно, что префиксный код Хаффмана не может иметь такую длину, содержащую часть бита, т. е. в конечном итоге это приводит к ухудшению сжатия данных.

Арифметическое кодирование предназначено для того, чтобы решить эту проблему. *Основная идея арифметического метода*

сжатия заключается в том, чтобы присваивать коды не отдельным символам, а их последовательностям.

Таким образом, как и во всех энтропийных алгоритмах, исходной является информация о частоте встречаемости каждого символа алфавита. Алгоритмы прямого и обратного преобразований базируются на операциях с «*рабочим отрезком*».

Рабочим отрезком называется интервал [*a*; *b*] с расположенными на нем точками. Причем точки расположены таким образом, что длины образованных ими отрезков пропорциональны (или равны) частоте (вероятности) появления соответствующих символов.

Пример. Дана информационная последовательность «молоко». Рассчитываем статистику:

p(м) = 0,1666

p(л) = 0,1666

p(к) = 0,1666

p(о) = 0,5.

Строим основной рабочий отрезок (на нулевом шаге (рис. 7.6)).

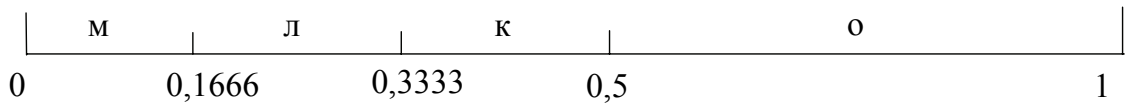

Рис. 7.6. Разметка и точки основного рабочего отрезка

Как показано на рис. 7.6, при инициализации алгоритма $a = 0$ $b = 1$ ($a_0 = 0$, $b_0 = 1$).

Прямое преобразование (сжатие). Один шаг сжатия (кодирования) заключается в простой операции: берется кодируемый символ, для него ищется соответствующий участок на рабочем отрезке. Найденный участок становится новым рабочим отрезком. Его тоже необходимо разбить с помощью точек.

Это и последующие разбиения отрезка (на шаге i) подразумевают определение новых значений верхней (H_i) и нижней (L_i) границ для всего участка и осуществляются по следующим правилам:

$$\left.\begin{aligned} H_i(\alpha_j) &= L_{i-1} + (H_{i-1} - L_{i-1}) \cdot H(\alpha_j)_0; \\ L_i(\alpha_j) &= L_{i-1} + (H_{i-1} - L_{i-1}) \cdot L(\alpha_j)_0, \end{aligned}\right\} \qquad (7.5)$$

где α_j – j-й символ сжимаемой последовательности; L_{i-1} и H_{i-1} – соответственно нижняя и верхняя границы рабочего отрезка на

$(i-1)$-м шаге; $L(\alpha_j)_0$ и $H(\alpha_j)_0$ – соответственно исходные нижняя и верхняя границы символа α_j.

В нашем примере на первом шаге берется первый символ последовательности: «м» (α_1 = «м») и ищется соответствующий участок на рабочем отрезке. Легко понять из рис. 7.6, что этот участок соответствует интервалу [0; 0,1666]. Он становится новым рабочим отрезком и опять разбивается согласно статистике и соотношениям (7.5):

$$H_1\,(\text{м}) = L_0 + (H_0 - L_0)\cdot H(\alpha_j = \text{м})_0 = 0 + (0{,}1666 - 0)\cdot 0{,}1666 = 0{,}0277;$$

$$L_1\,(\text{м}) = L_0 + (H_0 - L_0)\cdot L(\alpha_j = \text{м})_0 = 0 + (0{,}1666 - 0)\cdot 0 = 0;$$

$$H_1\,(\text{л}) = L_0 + (H_0 - L_0)\cdot H(\alpha_j = \text{л})_0 = 0 + (0{,}1666 - 0)\cdot 0{,}3333 = 0{,}055555;$$

$$L_1\,(\text{л}) = L_0 + (H_0 - L_0)\cdot L(\alpha_j = \text{л})_0 = 0 + (0{,}1666 - 0)\cdot 0{,}1666 = 0{,}0277$$

и т. д.

После выполнения всех вычислений на первом шаге получим разметку рабочего отрезка ($a_1 = L_1 = 0$, $b_1 = H_1 = 0{,}1666$) в соответствии с рис. 7.7.

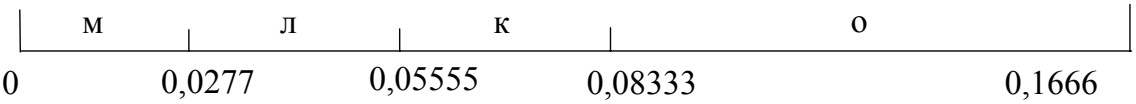

Рис. 7.7. Разметка и точки рабочего отрезка после анализа первого символа («м») последовательности (молоко)

Анализируем следующий символ (шаг 2). На новом рабочем отрезке очередному символу (о) соответствует интервал [$a_2 = L_2 =$ $= 0{,}083333$; $b_2 = H_2 = 0{,}1666$]. Для следующего шага он станет рабочим отрезком (рис. 7.8).

Строго говоря, на первом шаге не было необходимости рассчитывать новые границы для каждого символа. Как видим, это только увеличивает время анализа. Достаточно было определить новые верхнюю и нижнюю границы лишь для очередного (на шаге 2) символа: «о». Мы здесь привели расчеты только для того, чтобы помочь читателю понять процесс вычислений.

Поскольку очередным из анализируемых символов (на шаге 3) будет буква «л», достаточно в соответствии с (7.5) определить новые границы:

$$H_3\,(\text{л}) = L_2 + (H_2 - L_2)\cdot H(\alpha_j = \text{л})_0 =$$
$$= 0{,}08333 + (0{,}1666 - 0{,}08333)\cdot 0{,}3333 = 0{,}1111,$$

$$L_3\,(\text{л}) = L_2 + (H_2 - L_2) \cdot L(\alpha_j = \text{л})_0 =$$
$$= 0{,}08333 + (0{,}1666 - 0{,}08333) \cdot 0{,}1666 = 0{,}972.$$

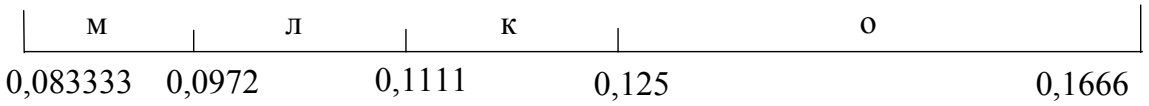

Рис. 7.8. Рабочий отрезок после анализа первых 2 символов

Новый рабочий отрезок определим как интервал [$a_3 = L_3 =$ $= 0{,}0972$; $b_3 = H_3 = 0{,}1111$], так как он соответствует символу «л». Вид разметки рабочего отрезка представлен на рис. 7.9.

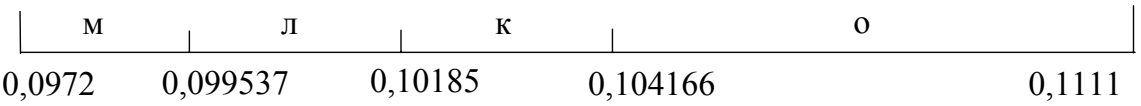

Рис. 7.9. Рабочий отрезок с разметками после анализа последовательности «мол»

Продолжаем анализировать символы сообщения. Находим на рабочем отрезке интервал [0,104166; 0,1111], который соответствует очередному символу: «о». Производим разметку нового рабочего отрезка (рис. 7.10).

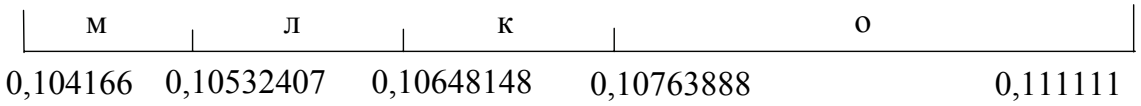

Рис. 7.10. Разметка рабочего отрезка на четвертом шаге

Следующий символ последовательности – «к». На рабочем отрезке данному символу принадлежит интервал [0,10648148; 0,10763888]. Новый рабочий отрезок имеет вид, как показано на рис. 7.11.

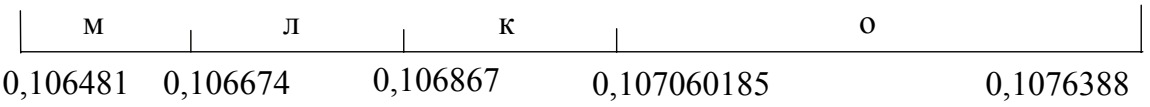

Рис. 7.11. Разметка рабочего отрезка на пятом шаге прямого преобразования

У нас остался последний символ: «о». Результатом кодирования цепочки символов является любое число с итогового рабочего отрезка [0,107060185; 0,10763888]. Обычно таким числом является

нижняя граница указанного отрезка. Поступим мы в соответствии с указанным принципом: возьмем число с меньшим количеством знаков после запятой.

Таким образом, итогом сжатия входной последовательности «молоко» будет число 0,107060185 (для упрощения дальнейших вычислительных операций округлим его до **0,1071**).

Декомпрессия. Для восстановления исходного сообщения необходима информация:

- о значении числа, являющегося итогом сжатия сообщения (в нашем случае 0,1071);
- о количестве символов в сжатом сообщении;
- о вероятностных параметрах всех символов исходного сообщения (таблица вероятностей).

Как и при сжатии, вначале необходимо начальный рабочий отрезок [0; 1) разбить на интервалы, длины которых равны вероятностям появления соответствующих символов.

Анализ будем проводить с использованием конкретных данных, взятых из последнего примера.

Итак, в качестве исходного участка для обратного преобразования принимается исходный участок для прямого преобразования с одинаковыми точками его разбиения (рис. 7.12). Ниже он повторяется.

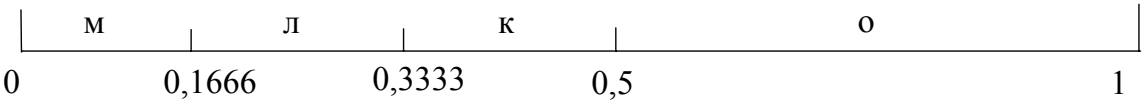

Рис. 7.12. Разметка рабочего отрезка на первом шаге обратного преобразования

На каждом шаге обратного преобразования выбираем отрезок, в который попадает текущее число (код). Символ, который соответствует данному отрезку, является очередным символом восстановленного (распакованного) сообщения.

В общем случае код символа, восстанавливаемого на шаге i, вычисляется соотношением

$$\text{код } i = [\text{код } (i-1) - L(\alpha_{i-1})_0]/[H(\alpha_{i-1})_0 - L(\alpha_{i-1})_0], \qquad (7.6)$$

где код $(i-1)$ – число, анализ которого производится на предыдущем шаге – $(i-1)$-м; $H(\alpha_{i-1})_0$ и $L(\alpha_{i-1})_0$ – соответственно верхняя и нижняя исходные границы символа сообщения, восстановленного на предыдущем шаге.

Первый шаг: определяем интервал, в который попадает начальное число: 0,1071 (код 1 = 0,1071). Это интервал [0; 0,166666] и ему соответствует символ «м». Данный интервал становится новым рабочим отрезком, а первый символ восстановленного сообщения – «м».

Второй шаг: производим вычисление в соответствии с (7.6):

$$\text{код } 2 = [\text{код } 1 - L(\alpha_1 = \text{м})_0]/[H(\alpha_1 = \text{м})_0 - L(\alpha_1 = \text{м})_0] = \\ = (0{,}1071 - 0)/(0{,}1666 - 0) = 0{,}643.$$

Вычисленный код соответствует символу «о».

Третий шаг: выполняем известное вычисление:

$$\text{код } 3 = [\text{код } 2 - L(\alpha_2 = \text{о})_0]/[H(\alpha_2 = \text{о})_0 - L(\alpha_1 = \text{м})_0] = \\ = (0{,}643 - 0{,}5)/(1 - 0{,}5) = 0{,}286.$$

Полученное число соответствует символу «л».

Аналогичным образом производятся и последующие вычислительно-аналитические операции.

Другой вариант алгоритма распаковки «сжатого» сообщения основан на *свойстве рекуррентности прямого преобразования*. Производя на каждом шаге вычисление новых границ рабочего участка в соответствии с восстановленным на данном шаге символом (наподобие тех вычислений, которые мы выполняли при сжатии) и анализируя, на какой из отрезков этого участка попадает входное число (в нашем примере это 0,1071), восстанавливаем исходное сообщение.

Таким образом, на первом шаге это число соответствует букве «м», которая попадает в интервал [0; 0,1666]. Этот новый диапазон будет рабочим участком на шаге 2.

На втором шаге интервал [0; 0,1666] опять разбивается согласно статистике (рис. 7.13).

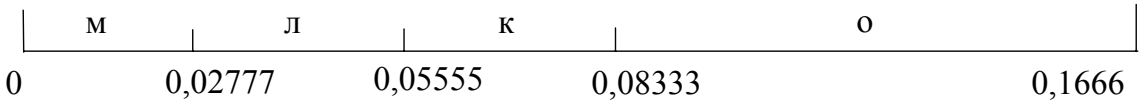

Рис. 7.13. Рабочий отрезок с разметкой
после первого шага восстановления

Снова ищем интервал, в который попадает наше число 0,1071. Выбираем последний интервал [0,083333; 0,16666] в качестве рабочего отрезка (рис. 7.14) для последующих шагов, а символ «о» добавляем к восстановленному сообщению.

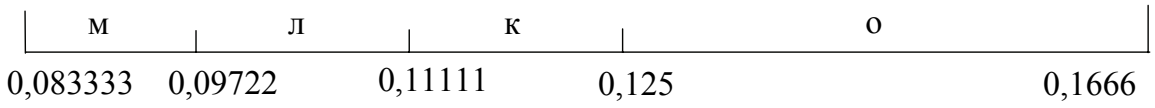

Рис. 7.14. Вид рабочего отрезка после 2 шагов

Определяем следующее соответствие. Число 0,1071 попадает в интервал [0,097222; 0,111111], который становится очередным рабочим отрезком (рис. 7.15). К восстановленной части сообщения «*мо*» присоединяем символ интервала «л».

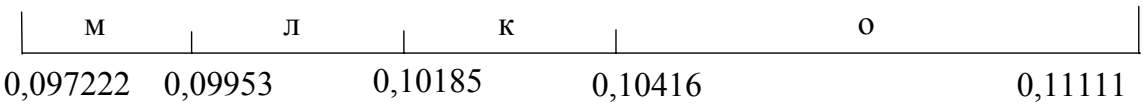

Рис. 7.15. Вид рабочего отрезка после 3 шагов
обратного преобразования

Следующий интервал – [0,104166; 0,111111] – соответствует символу «о». Его деление показано на рис. 7.16.

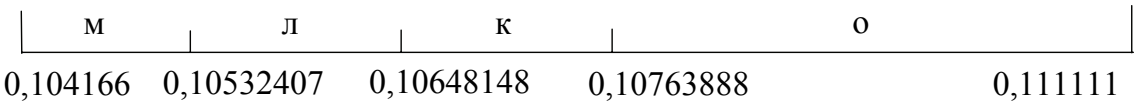

Рис. 7.16. Рабочий отрезок на 4 шаге

На пятом шаге наше число попадает в интервал символа «к». Данный символ присоединяем к тексту сообщения, а интервал [0,104166; 0,1076388] становится рабочим отрезком (рис. 7.17).

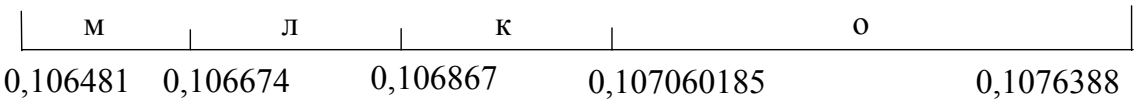

Рис. 7.17. Вид рабочего отрезка на последнем шаге обратного
преобразования сообщения «молоко»

Так как нам известно, что символов в сообщении было 6, то на последнем шаге необходимо только определить недостающий символ. Таким символом становится символ «о». Полученное сообщение «молоко» и есть восстановленная («распакованная») последовательность.

Пример. Дана информационная последовательность: «*переговоры*».

Подсчитываем частоту встречаемости каждого символа в сообщении. Разбиваем интервал [0; 1) на 7 отрезков. Длина каждого отрезка – вероятность встречаемости соответствующего символа в информационной последовательности. Далее необходимо осуществить ряд уже известных читателю операций.

Прямое преобразование. Первый шаг: первым выбираем интервал из рабочего отрезка, который соответствует первому символу в сообщении. Это символ «п». Границы интервалов, соответствующие отдельным символам, здесь и далее для компактности сведены в таблицы. Границы анализируемого на данном шаге символа выделены жирным шрифтом.

символ	**п**	е	р	г	о	в	ы
0,00000	**0,10000**	0,30000	0,50000	0,60000	0,80000	0,90000	1,00000

Таким образом, рабочий участок после первого шага составляет диапазон [0; 0,1].

Второй шаг: разбиваем полученный отрезок [0; 0,1] на интервалы. Определяем отрезок, который соответствует второму символу: «е». Это отрезок [0,1; 0,3].

символ	п	**е**	р	г	о	в	ы
0,00000	**0,01000**	**0,03000**	0,05000	0,06000	0,08000	0,09000	0,10000

Третий шаг: разделяем текущий рабочий интервал [0,01; 0,03] на отрезки согласно статистике появления символов. Находим отрезок, который соответствует символу «р». Этот отрезок принимаем за новый рабочий и делим его на 7 частей.

символ	п	е	**р**	г	о	в	ы
0,01000	0,01200	**0,01600**	**0,02000**	0,02200	0,02600	0,02800	0,03000

Четвертый шаг: из текущего интервала выбираем отрезок [0,0164; 0,0172], который соответствует символу «е». Актуальная таблица представлена ниже.

символ	п	**е**	р	г	о	в	ы
0,01600	**0,01640**	**0,01720**	0,01800	0,01840	0,01920	0,01960	0,02000

Пятый шаг: продолжаем вычисление для следующего символа: «г». На рабочем отрезке выбираем диапазон [0,0168; 0,01688].

символ	п	Е	р	**г**	о	в	ы
0,01640	0,01648	0,01664	**0,01680**	**0,01688**	0,01704	0,01712	0,01720

Шестой шаг: разбиваем полученный интервал [0,0168; 0,01688] на отрезки. Выбираем отрезок, который соответствует символу «о».

символ	п	е	р	г	**о**	в	ы
0,016800	0,016808	0,016824	0,016840	**0,016848**	**0,016864**	0,016872	0,016880

Седьмой шаг: определяем интервал для символа «в». В качестве рабочего берем отрезок [0,016861; 0,016862].

символ	п	е	р	г	о	**в**	ы
0,016848	0,016850	0,016853	0,016856	0,016858	**0,016861**	**0,016862**	0,016864

Восьмой шаг: выбранный интервал [0,0168618; 0,0168621], который соответствует символу «о», используем на следующем этапе.

символ	п	е	р	г	**о**	в	ы
0,0168608	0,0168610	0,0168613	0,0168616	**0,0168618**	**0,0168621**	0,0168622	0,0168624

Девятый шаг: для анализа символа «р» используем отрезок [0,01686186; 0,01686192] из рабочего интервала.

символ	п	е	**р**	г	о	в	ы
0,01686176	0,01686179	**0,01686186**	**0,01686192**	0,01686195	0,01686202	0,01686205	0,01686208

Десятый шаг: Определяем последний интервал. Последний символ – «ы». Разбиваем полученный интервал [0,01686186; 0,01686192] на отрезки. Выбираем отрезок, который соответствует данному символу.

символ	п	е	р	г	о	в	**ы**
0,01686186	0,01686186	0,01686188	0,01686189	0,01686189	0,01686191	**0,01686191**	**0,01686192**

Процесс преобразования (сжатия) закончен. Предположим, что конечным результатом будет верхняя граница последнего диапазона: 0,01686192. Это число и есть сжатое сообщение.

Декомпрессия. Для восстановления исходного сообщения необходимо:

- вещественное число – 0,01686192;
- количество символов в сообщении – 10;
- таблица вероятностей встречаемости символов в этом сообщении.

Как и при сжатии, необходимо исходный интервал [0; 1) разбить на отрезки, длины которых равны вероятностям встречаемости символов. Далее на каждом шаге выбираем отрезок, в который попадает наше число (0,01686192). Символ, который соответствует данному отрезку, и есть очередной символ восстанавливаемого сообщения.

Первый шаг: первым выбираем отрезок исходного рабочего участка, в который попадает число 0,01686192. Это интервал, которому соответствует символ «п», что видно из нижеследующей таблицы.

символ	**п**	е	р	г	о	в	ы
0,00000	**0,10000**	0,30000	0,50000	0,60000	0,80000	0,90000	1,00000

Второй шаг: разбиваем текущий интервал [0; 0,1] на отрезки. Определяем отрезок, в который попадает число 0,01686192. Символ «е» будет следующим символом на выходе распаковщика, а соответствующий интервал [0,1; 0,3] – рабочим отрезком.

символ	п	**е**	р	г	о	в	ы
0,00000	**0,01000**	**0,03000**	0,05000	0,06000	0,08000	0,09000	0,10000

Третий шаг: число 0,01686192 попадает в интервал [0,016; 0,02]. На выходе – символ «р».

символ	п	е	**р**	г	о	в	ы
0,01000	0,01200	**0,01600**	**0,02000**	0,02200	0,02600	0,02800	0,03000

Процесс продолжается до полного восстановления информации.

Вопросы для контроля и самоконтроля

1. В чем заключается суть арифметического метода сжатия?
2. Поясните принцип определения границ отрезков на каждом шаге (на каждом рабочем интервале) прямого и обратного преобразования.
3. Осуществить сжатие и распаковку следующих сообщений арифметическим методом:

а) «мама мыла раму»;
б) «насос»;
в) «университет»;
г) «шалаш»;
д) «информатика»;
е) «246824328»;
ж) «101101001011010»;
з) ваша фамилия.