Если же вероятности не являются отрицательными степенями двойки, то все-равно можно вычислить числа $l'_i = \log \frac{1}{p_i}$. Они не будут целыми, поэтому их нельзя использовать в качестве длин кодовых слов. Тем не менее, они удовлетворяют неравенство Крафта:

$$\sum_{i=1}^{m} \frac{1}{2^{l'_i}} = \sum_{i=1}^{m} 2^{\log p_i} = \sum_{i=1}^{m} p_i = 1.$$

Выберем теперь l_i как наименьшее целое, которое равно или больше, чем l'_i. Поскольку $l_i \geq l'_i$, то неравенство Крафта выполнено и для этого набора натуральных чисел.

Кроме того, $l_i < -\log p_i + 1$, поэтому

$$L = \sum_{i=1}^{m} p_i l_i < \sum_{i=1}^{m} p_i(-\log p_i + 1) = H + 1.$$

□

3. Сжатие информации

Сжатие информации — это любая процедура, которая уменьшает объем информации без нарушения целостности данных. Сжатие (*компрессия*) информации — это важная часть современных информационных технологий, которая повсеместно используется для уменьшения размеров музыкальных, графических и видео файлов. Широко известны форматы MP3 и JPG, которые являются скомпресованными аналогами исходных звуковых или графических файлов. В настоящее время методы компрессии рассматриваются как часть теории кодирования.

3.1. Метод Хаффмана. В 1951 году студент МТИ Дэвид Хаффман в курсовой работе по курсу теории информа-

ции предложил метод нахождения префиксных кодов минимальной длины, который впоследствии назвали *кодированием по Хаффману*. Оказалось, что метод Хаффмана приводит к коду, длина которого очень близка к энтропии, то есть является оптимальной ввиду неравенства (5).

Хаффман рассуждал следующим образом. Рассмотрим случай бинарного алфавита. Длину кода для двух информационных символов можно оценить следующим образом

$$L = pl_1 + (1-p)l_2 \geq p + (1-p) = 1,$$

где $l_1 \geq 1$ и $l_2 \geq 1$ — длины кодовых слов. В этом случае код $\begin{vmatrix} s_1 \to 0 \\ s_2 \to 1 \end{vmatrix}$ как-раз и обладает свойством минимальности.

Ситуация усложняется в случае трех информационных символов. Идея Хаффмана состоит в том, чтобы временно объединить два символа с минимальными вероятностями в один. Тогда мы получим два символа (один их них — составной). Для этих символов применяем простое кодирование 0 и 1. Если составной символ получил код 0, то кодом Хаффмана является 00, 01, 1, где коды 00 и 01 относятся к символам наименьшей вероятности.

При возрастании количества символов применяются аналогичные рассуждения. Работу алгоритм Хаффмана продемонстрируем на простом примере.

Пример 5. Рассмотрим источник информации, состоящий из пяти символов s_1, s_2, s_3, s_4, s_5, вероятности которых равны 0.3, 0.2, 0.2, 0.2 и 0.1. Энтропия этого источника равна

$$H = -[0.3 \log 0.3 + (3 \times 0.2) \log 0.2 + 0.1 \log 0.1] \approx 2.246.$$

Первый шаг алгоритма Хаффмана, который мы называем *сворачиванием*, состоит из последовательности двух действий:

(a) объединить два символа с минимальными вероятностями в один,

(b) упорядочить вероятности по возрастанию.

Работа алгоритма для нашего примера представлена ниже:

$$\left|\begin{matrix} s_1 & 0.3 \\ s_2 & 0.2 \\ s_3 & 0.2 \\ s_4 & 0.2 \\ s_5 & 0.1 \end{matrix}\right| \overset{(a)}{\to} \left|\begin{matrix} s_1 & 0.3 \\ s_2 & 0.2 \\ s_3 & 0.2 \\ s_{4,5} & 0.3 \end{matrix}\right| \overset{(b)}{\to} \left|\begin{matrix} s_1 & 0.3 \\ s_{4,5} & 0.3 \\ s_2 & 0.2 \\ s_3 & 0.2 \end{matrix}\right| \overset{(a)}{\to} \left|\begin{matrix} s_1 & 0.3 \\ s_{4,5} & 0.3 \\ s_{2,3} & 0.4 \end{matrix}\right| \overset{(b)}{\to} \left|\begin{matrix} s_{2,3} & 0.4 \\ s_1 & 0.3 \\ s_{4,5} & 0.3 \end{matrix}\right|.$$

Следующая комбинация действий (a) и (b) приводит к символам $\left|\begin{matrix} s_{1,\{4,5\}} & 0.6 \\ s_{2,3} & 0.4 \end{matrix}\right|$ и это заканчивает сворачивание.

Второй шаг алгоритма Хаффмана, который мы называем *разворачиванием*, заключается в последовательном кодировании составных символов:

$$\left|\begin{matrix} s_{1,\{4,5\}} \to 0 \\ s_{2,3} \to 1 \end{matrix}\right| \Rightarrow \left|\begin{matrix} s_1 \to 00 \\ s_{4,5} \to 01 \\ s_{2,3} \to 1 \end{matrix}\right| \Rightarrow \left|\begin{matrix} s_1 \to 00 \\ s_4 \to 010 \\ s_5 \to 011 \\ s_{2,3} \to 1 \end{matrix}\right| \Rightarrow \left|\begin{matrix} s_1 \to 00 \\ s_4 \to 010 \\ s_5 \to 011 \\ s_2 \to 10 \\ s_3 \to 11 \end{matrix}\right|.$$

Длина полученного кода Хаффмана

$$L = 2 \times 0.3 + 2 \times 0.2 + 3 \times 0.3 + 3 \times 0.2 + 3 \times 0.1 = 2.3$$

близка к энтропии $H = 2.246$.

3.2. Код Зива–Лемпеля. Процедуры сжатия данных стали намного более эффективными после опубликования в 1977 году остроумного метода Дж. Зива и А. Лемпеля (метод впоследствии был назван LZ77). Их метод основан на эмпирическом наблюдении, что буквы в английских текстах чаще встречаются в определенных комбинациях, то есть появляются не совсем случайно. Метод Зива–Лемпеля использует список таких комбинаций для достижения лучшей компрессии данных.

При подготовке к отсылке очередного фрагмента (будем говорить "*нового*" фрагмента) алгоритм LZ77 находит идентичный фрагмент в старых данных и вместо отсылки нового фрагмента отсылает лишь указание на позицию старого. Для максимального сжатия следовало бы просматривать всю старую информацию, но для больших файлов это нереалистично. Размер просматриваемого старого текста является одним из параметров алгоритма.

Указание на повторяемый фрагмент имеет вид (x, y, z), где x — это начало повторяемого фрагмента в исходном тексте, y — длина этого фрагмента, z — буква в новом тексте, идущая после найденного фрагмента.

Пример 6. Рассмотрим работу упрощенного алгоритма LZ77 на примере кодирования следующего предложения на английском языке. Считаем, что для поиска идентичного фрагмента используется весь старый текст.

T	H	I	S	$-$	T	H	E	S	I	S	$-$	I	S	$-$	T	H	E	$-$	T	H	E	S	I	S
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Вторая строчка нужна лишь для объяснения работы алгоритма: под каждой буквой указана ее позиция в предложении. Символ "$-$" выполняет функцию пробела и также называется буквой.

Схема работы алгоритма представлена ниже:

Очередная буква	Совпадающая фраза	Посылаемая комбинация
T		$(0, 0, T)$
H		$(0, 0, H)$
I		$(0, 0, I)$
S		$(0, 0, S)$
$-$		$(0, 0, -)$
T	TH	$(1, 2, E)$
S		$(0, 0, S)$
I	$IS-$	$(3, 3, I)$
S	$S-THE$	$(4, 5, -)$
T	$THESIS$	$(6, 6, \varnothing)$

Первой в алгоритм “попадает” буква “T”. Поскольку буфер старых данных еще пуст, то никаких совпадений нет; на приемник пересылается комбинация $(0, 0, T)$. Первый ноль в этой комбинации означает, что совпадений не обнаружено и пересылается новая буква. Алгоритм работает аналогично и для следующих букв “H”, “I”, “S” и “$-$”. Отличие последующих шагов от первого заключается в том, что в буфер последовательно записываются фрагменты “TH”, “THI”, “$THIS$” и “$THIS-$”.

Схема работы меняется для следующего “T”: алгоритм находит одинаковые фрагменты “TH” в старых и новых данных и пересылает приемнику $(1, 2, E)$, что означает, что повторяемый фрагмент начинается в старом тексте с позиции 1, имеет длину 2 и в новых данных продолжается буквой “E”. После этого снова идет новая буква “S”. Однако затем идут повторяющиеся фрагменты “$IS-$”, “$IS-THE$” и, наконец, “$-THESIS$”.

Пример 7. Наибольшее сжатие алгоритм LZ77 показывает на графических файлах, в которых часто встречаются

повторяющиеся фрагменты. Рассмотрим, например, такое "изображение":

$$\star\star\star\star\star\star\star\star$$
$$\star\star\star\star\star\star\star\star\star\star\star$$
$$\star\star\star\star\star\star$$

Каждый символ $\star$ соответствует одному пикселу, то есть "изображение" — это три параллельные линии разной длины. Это "изображение" можно представить таким образом:

$$\begin{array}{cccccccccccccccccccccccccc} \star & \star & \star & \star & \star & \star & \star & \star & \square & \star & \star & \star & \star & \star & \star & \star & \star & \star & \star & \square & \star & \star & \star & \star & \star & \star \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 \end{array}$$

где $\square$ обозначает операцию перехода на новую строку. Следовательно все изображение состоит из 27 символов. Алгоритм LZ77, примененный к такой последовательности, закодирует ее в последовательность семи троек:

$$(0,0,\star)\quad (1,1,\star)\quad (1,3,\star)\quad (1,1,\square)\quad (1,8,\star)\quad (7,3,\star)\quad (1,5,\varnothing)$$

Таким образом, сжатие "изображения" алгоритмом LZ77 составило более 22%.

КОНТРОЛЬНЫЕ ВОПРОСЫ

Вопрос 1. Объяснить аналитически, почему целесообразно считать, что $H(0) = H(1) = 0$? Найти пределы $H(p)$ при $\downarrow 0$ и $p \uparrow 1$.

УПРАЖНЕНИЯ

Упражнение 1. Верно ли, что информация $I_{A\cap B}$ имеет максимальное значение, если события A и B независимы? Иными словами, верно ли, что

$$I_{A\cap B} \le I_A + I_B$$

(см. теорему 1.3)?