Тема 10. Сжатие данных

Постановка задачи — получить набор данных меьшего объема, но с сохранением необходимой информации.

Сжатие с потерями. Это математика — приближения и аппроксимации.

Сжатие без потерь. Устранение/сокращение информационной избыточности.

Алгоритм группового кодирования (RLE).

Название алгоритма происходит от слов Run Length Encoding.

Байтовый вариант.

Идея: вместо последовательности подряд идущих одинаковых байтов передавать на выход только длину последовательности и один такой байт. Если же во входном потоке нет постоянных повторяющихся цепочек, то кодирование не производится. Более формальное описание алгоритма может выглядеть так:

1. Разобъем входной поток на цепочки байт по следующим критериям:

- длина каждой цепочки не превосходит 127 байт;

- цепочка целиком состоит из одинаковых байтов и имеет длину больше 1 либо не имеет в себе двух одинаковых соседних байтов.

2. Каждой цепочке сопоставим выходной код по следующим правилам:

— для постоянных цепочек длины выходным кодом являются два байта, где старший бит первого байта равен 0, а оставшиеся 7 бит кодируют число n, второй байт кода равен байту, составляющему данную цепочку;

— для непостоянных цепочек длины n выходным кодом являются $n + 1$ байт, где старший бит первого байта равен 1, а оставшиеся 7 бит кодируют число n, далее идут n байт исходной цепочки.

Нетрудно оценить наилучшую и наихудшую степень сжатия. В том случае, когда файл содержит повторяющиеся группы из 127 одинаковых байт, мы получаем максимальное сжатие с коэффициентом 127/2. Если в файле нет ни одной повторяющейся группы, то на каждые 127 байт мы получаем один дополнительный байт, т.е. длина "сжатого" набора данных увеличивается на 1/127 (это меньше 1%).

Битовый вариант.

На входе последовательность битов, на выходе байтовый код.

```
биты   7 6 5 4 3 2 1 0
       0 0 длина строки пикселов-нулей
       0 1 длина строки пикселов-единиц
       1 оставшиеся 7 бит непосредственно принадлежат изображению
```

Как видно из этого представления, длина цепочки здесь ограничена значением 64. Это ограничение позволяет получить сжатие до 8 раз.

Этот метод часто применяется для сжатия изображений (построчно — строка пикселов, 0 черный, 1 белый)

Алгоритм Хаффмена

Семейство кодов: посимвольное кодирование, т.е. таблица символ $\longleftrightarrow$ код

Алгоритм Хаффмена (иногда его называют CCITT кодирование) состоит в сопоставлении каждому символу входного потока некоторого кодовых цепочек переменной длины так, что наиболее часто встречающиеся символы кодируются наиболее короткими цепочками. Для этого строится бинарное дерево (дерево Хаффмена).

Иллюстрация: построения дерева Хаффмена и кодирования.

Алгоритм.

1. Создаем множество свободных вершин, содержащее все символы входного алфавита, задаем каждой вершине вес в соответствии с таблицей количеств (частота×длина сообщения).

2. В множестве свободных вершин находим две вершины (обозначим их A и B), имеющие наименьшие веса, создаем новую родительскую вершину C, присоединяем A и B к C справа и слева, устанавливает вес C равным сумме весов A и B, исключаем A и B из множества свободных вершин, добавляем C к множеству свободных вершин.

3. Если в множестве свободных вершин более одного элемента, то перейти к пункту 2, иначе единственный оставшийся элемент есть корень дерева Хаффмена.

Далее, левым ребрам построенного дерева сопоставляется число 0, а правым ребрам — 1. Теперь, спускаясь от корня к конкретной концевой вершине и последовательно выписывая нули и единицы, сопоставленные проходимым ребрам, мы получаем код символа в этой вершине.

Код Хаффмена является префиксным, т.е. код любого символа не содержит в качестве префикса никакой другой код. Это позволяет выдавать на выход непрерывную последовательность битов кодов, поскольку она будет однозначно интерпретироваться при декодировании (спуском по ветви дерева Хаффмена).

Теорема. *Для входной последовательности с заданным фиксированным частотным распределением символов метод Хаффена дает кодовую последовательнсть наименьшей длины среди всех префиксных кодов.*

Пусть нам дана некоторая входная последовательность символов (байтов) длины n и каждый символ со значением k имеет вес (количество появлений) w_k. Построим дерево Хаффмена. Все символы будут концевыми вершинами, и длины их кодов есть длины соответствующих ветвей от корня к этим вершинам. Если обозначить l_k — длина ветви от корня до вершины символа k, то длина кода для исходного набора данных есть

$$L_{Huf} = \sum_k w_k l_k \,.$$

Т.е. L_{Huf} не больше длины любой другой последовательности, полученной префиксным кодированием.

Лемма. *Пусть некоторый способ префиксного кодирования на указанном наборе дает выходную кодовую последовательноь наименьшей длины. Тогда существует способ кодирования, дающий ту же самую длину кодовой последовательности, и такой, что две минимальные по весу вершины лежат в самых длинных ветвях и имеют одного родителя.*

Доказательство.

Рассмотрим кодовое дерево исходного метода. Пусть a и b — два минимальных по весу символа с весами w_a, w_b, лежащих в ветвях с длинами l_a и l_b соответственно. Пусть также x и y — два символа, имеющие одного родителя и лежащие в самых длинных ветвях. Заметим, что выполнены неравенства

$$w_a \le w_x, \quad w_b \le w_y, \quad l_a \le l_x, \quad l_b \le l_y.$$

Переставим местами символы a и x а также b и y и посмотрим какое влияние окажет эта перестановка на длину выходной кодовой последовательности. До перестановки вклад символов a, b, x, y в выходную кодовую последовательность был равен $L_1 = w_a l_a + w_b l_b + w_x l_x + w_y l_y$, а после перестановки стал равен $L_2 = w_a l_x + w_b l_y + w_x l_a + w_y l_b$. Рассмотрим разность

$$L2 - L1 = w_a l_x + w_b l_y + w_x l_a + w_y l_b - w_a l_a - w_b l_b - w_x l_x - w_y l_y =$$

$$= (w_a - w_x)(l_x - l_a) + (w_b - w_y)(l_y - l_b) \le 0.$$

Последнее неравенство означает, что $L2 \le L1$, или, в силу минимальности $L1$, что $L2 = L1$. Вклад всех остальных символов остался без изменения. Таким образом,

перестроенное дерево удовлетворяет условиям утверждения и определяет искомый метод кодирования.

Докажем теперь теорему.

Пусть T_h — кодирующее дерево Хаффмена, а T_o — кодирующее дерево некоторого другого оптимального метода. Докажем, что кодовая последовательнось, получаемая с помощью дерева T_o не короче кодовой последовательности Хаффмена. В силу только что доказанного утверждения можно считать, что в дереве T_o две вершины, отвечающие символам a и b с минимальными весами w_a, w_b, имеют общего родителя и расположены в самых длинных ветвях. Напомним, что точно таким же свойством обладает по построению и дерево Хаффмена T_h. Отождествим символы a и b и удалим соответствующие им вершины из деревьев T_h и T_o, оставив только родительскую вершину с весом $w_a + w_b$. Рассмотрим теперь новые полученные деревья T_h и T_o как кодирующие деревья для множества символов, содержащих на один символ меньше (поскольку символы a и b отождествились в один новый "суммарный" символ). Длины кодирующих последовательностей, получаемых по обоим этим деревьям, уменьшились на $w_a + w_b$ бит по сравнению длиной исходных кодовых последовательностей. При этом дерево T_h остается деревом Хаффмена для нового множества символов, а дерево T_o остается деревом оптимального кодирования. Действительно, если бы для текущего набора символов существовало другое кодирующее дерево, дающее меньшую длину, то "разделив" обратно объединенный символ и добавив в это дерево вершины a и b, мы получили бы меньшую длину кодовой последовательности и для исходного входного потока, что невозможно в силу предположения об оптимальности начального состояния дерева T_o.

Повторяя эти рассуждения и отождествляя следующие две вершины с минимальными весами, мы и далее будем одинаково уменьшать длины выходной кодирующей последовательности, пока в кодирующих деревьях не останется по две концевых вершины (отождествляющих в себе в сумме все исходные символы). При этом дерево дерево Хаффмена будет давать кодирующую последовательность в n бит так как два оставшихся кода есть 0 и 1. Оставшееся дерево T_o является оптимальным и также дает кодовую последовательность длины n, поскольку меньше просто быть не может (единственный выбор для оставшихся двух символов также коды 0 и 1). Так как каждый раз кодовые последовательности уменьшались на одинаковую величину, то их исходные длины также были одинаковы. Следовательно, дерево Хаффмена также давало кодирующую последовательность наименьшей длины. Теорема доказана.

Очевидным недостатком метода Хаффмена является необходимость передавать декодировщику таблицу частот символов вместе с кодирующей последовательностью, что уменшает суммарный коэффициент сжатия. Существует адаптивный вариант алгоритма Хаффмена, который не требует передачи дерева (или частотной таблицы) вместе с закодированными данными. Идея этого алгоритма заключается в адаптивном перестроении дерева с учетом поступивших символов при кодировании и соответствующем преобразовании дерева при декодировании. Пусть мы имеем некоторое дерево Хаффмена. Пронумеруем вершины этого дерева "слева-направо, снизу-вверх", т.е. нумерация начинается с элементов самого нижнего уровня дерева, потом переходит на предыдущий уровень, и т.д. Корень дерева получает максимальный номер.

Будем называть дерево упорядоченным, если веса вершин не убывают в порядке построенной нумерации.

Адаптивное перестроение дерева. Построим начальное дерево Хаффмена, считая, что все символы имеют вес 1. Теперь при появлении нового символа мы должны перестроить дерево так, чтобы сохранить свойство упорядоченности. Во-первых, надо увеличить вес соответствующей концевой вершины и всех родителей по ветви, ведущей к корню. Если после этого дерево остается упорядоченным, то ничего больше делать не надо. Если же вес какой-либо вершины A становится больше веса правого соседа (по нумерации), то среди правых соседей (по нумерации) ищется последний, вес которого меньше веса A, и эта вершина (вместе со своими потомками) обменива-

ется с A. Затем корректируются веса по ветвям, ведущим к корню от переставленных вершин. Указанный процесс продолжается, пока дерево не станет упорядоченным.

Кодирование и декодирование по адаптивному варианту алгоритма происходит единообразно:

1. Дерево инициализируется равномерным распределение символов (все символы имеют количество повторений 1).

2. Получить очередной символ (или код) и закодировать (раскодировать) его в соответствии с текущим состоянием дерева Хаффмена.

3. Добавить единицу к весу вершины, соответствующей только что полученному символу, и перестроить дерево на основе для сохранения свойства упорядоченности.

4. Если есть еще символы (коды) во входном потоке, то перейти к пункту 2, иначе закончить работу.

Естественно, адаптивный алгоритм не дает оптимальной кодовой последовательности. Однако помимо отказа от передачи таблицы частот от он обладает свойством настройки на текущее распределение частот. Рассмотрим следующий искуственный, но показательный пример. Пусть входная последовательность содержит достаточно длинные цепочки одинаковых байт, при этом в целом распределение частот является равномерным. Для такой последовательности обычный алгоритм Хаффмена не предложит ничего лучшего чем обычные восьмибитные коды байтов. Однако в адаптивном варианте он настроится на локальные частотные характеристики и назначит повторяющимся символам более короткие коды, что может дать выигрыш для файла в целом.