Алгоритмы сжатия

Сжатие — это кодирование с уменьшением объема данных и возможностью однозначного декодирования. Обратный процесс — декодирование — называется разжатие. Другие названия: *компрессия/декомпрессия, упаковка/распаковка.* Эффективность алгоритма сжатия зависит не только от *степени сжатия* (отношение длины несжатых данных к длине соответствующих им сжатых данных), но и скорости сжатия и разжатия, объема памяти, необходимого для работы алгоритмов и т.д. На практике выделяют следующие два вида компрессии данных.

Сжатие без потерь (lossless compression) – собственно сжатие в смысле приведенного выше определения.

Сжатие с потерями (lossy compression) – процесс, состоящих из двух этапов

1. выделение сохраняемой части информации в зависимости от цели сжатия и особенностей приемника и источника;
2. собственно сжатие без потерь.

Методы сжатия без потерь. В основе всех методов сжатия лежит простая идея: если представлять часто используемые элементы короткими кодами, а редко используемые – длинными кодами, то для хранения блока данных требуется меньший объем памяти, чем если бы все элементы представлялись кодами одинаковой длины. Точная связь между вероятностями и кодами установлена в теореме Шеннона о кодировании источника: элемент a_i, вероятность появления которого равняется p_i выгоднее всего представлять $-\log_2 p_i$ бит. Если при кодировании размер элементарных кодов в точности получается равным $-\log_2 p_i$, то длина кода будет минимальной из всех возможных способов алфавитного кодирования и равна энтропии $H = -\sum p_i \log p_i$. Сжатие всегда осуществляется за счет устранения статистической избыточности в представлении информации. Компрессор не может сжать любой файл. Размер некоторых файлов уменьшается, а остальных остается неизменным или увеличивается. Рассмотрим 4 метода сжатия без потерь.
Коды Хаффмана (Huffman Coding) или коды с минимальной избыточностью. Характеристики: степени сжатия – от 1 до 8, средняя -1,5, не увеличивает размер файла (не считая таблицы перекодировки).

Кодирование длин повторов, Run Length Encoding (RLE, групповое кодирование) Один из наиболее старых методов сжатия, идея метода состоит в замене идущих подряд одинаковых символов (бит или байт) парой (количество, символ). В основном используется для кодирования растровых изображений. Характеристика: степень сжатия от 0,5 до 32.

Пример. Рассмотрим бинарное изображение. Групповой код A задает количество нулевых и единичных значений в порядке их следования. Групповой код B задает индексы границ единичных участков.
исходное изображение: 0000 0000 1111 1000 0000 0000 0111 0000 0001 1111 1111 0000
A: 8(0)5(1)12(0)3(1)7(0)9(1)4(0) B: (8,12) (25,27) (35,43)
Задание. Построить коды A и B для изображения 011 110 000 111 011 111

Алгоритмы Зива-Лемпела (**LZ**-методы). Реализации LZ77, **LZ78**, LZW и LZH. Относятся к *словарным методам* сжатия, т.е. сообщение кодируется не побуквенно (алфавитное кодирование), а по словам. Характеристики LZ78: степень сжатия в зависимости от данных, обычно 2-3, алгоритмы универсальны, но лучше всего подходят для сжатия текстов, рисованных картинок или других однородных данных.

Идея словарного метода состоит в обнаружении во входном тексте повторяющихся цепочек байтов (слов), составлении таблицы таких слов (словаря). Все слова в исходном сообщении заменяются на код — номер слова в словаре. Алгоритм устроен так, что распаковщик, анализируя сжатые данные, может построить копию исходной таблицы, и следовательно, ее не надо хранить вместе с сжатыми данными.

Алгоритм LZ78. Кодируемый текст разбивается на слова, каждое из которых кодируется парой (номер, символ). За один проход по тексту динамически строится словарь и сжатый текст. Изначально словарь содержит одно слово с номером 0 - пустое Λ. На каждом шаге алгоритма считываются символы, начиная с текущего, и формируется слово наибольшей длины, совпадающее с каким-нибудь из уже имеющихся в словаре плюс еще один символ.

Рассмотрим, например, входную последовательность 010 001 011 001 010 001 101 011 00. На первом шаге рассматривается слово из одного символа 0, оно добавляется в словарь под номером 1, и кодируется в выходной последовательности (0,0), что означает слово из словаря под номером 0 (пустое слово)+ символ 0. На втором шаге в словарь добавляется слово 1 под номером 2, имеющее код (0,1). На третьем шаге имеем совпадение с непустым словом в словаре 0, в словарь добавляется слово 00, имеющее код (1,0). И т.д. На шестом шаге мы уже закодировали последовательность 010001011, словарь имеет вид {Λ,0, 1, 00, 01, 011}, остался текст 0010100011. Максимальное совпадение с третьим словом словаря '00'. Данная последовательность вместе со следующим символом (здесь 1) кодируется парой чисел (номер совпадающего слова в словаре, следующий символ) (здесь (3,1)) и добавляется в словарь (теперь словарь имеет вид {Λ,0, 1, 00, 01, 011, 001}). На следующем шаге считывается следующий за уже закодированными символ. И т.д. до конца входного текста.

В результате последовательности 010 001 011 001 010 001 101 011 00
соответствует словарь {Λ,0, 1, 00, 01, 011, 001, 010, 0011, 0101, 10},
разбиение последовательности на слова и код

0	1	00	01	011	001	010	0011	0101	10	0,
$(0,0)$,	$(0,1)$,	$(1,0)$,	$(1,1)$,	$(4,1)$,	$(3,1)$,	$(4,0)$,	$(6,1)$,	$(7,1)$,	$(2,0)$,	$(0,0)$

При декодировании одновременно строится словарь и сжатое сообщение.

Задание.

Закодируйте текст

1). 010 010 001 2). aba adb abc ecd ebc ea 3). 001 101 110 010 100 110 100 010 111 001 010 011 010 110 100

Раскодируйте текст

1). (0,0), (0,1), (2,0), (3,1), (2,1), (1,1), (4,1), (7,1), (6,0), (1,0), (9,1), (2,0)
2). (0,0), (0,1), (2,1), (2,0), (1,0), (3,0), (6,0), (7,1), (1,1), (9,1), (5,0),(0,1)
3). (0,γ), (0, α), (0,β), (1,γ), (2,β), (2,δ), (0,δ), (0,γ)

Применение.
LZ-методы: архиваторы (форматы `rar`, `zip`, `arj`, `cab`, `ace` и т.д.); графические файлы `gif`, `tiff`.
RLE: графические файлы `jpeg`, `tiff`.
Коды Хаффмана: `tiff`, `jpeg`.

Арифметическое сжатие. (ARIC, Arithmetic Coding) Характеристики: один из самых эффективных методов, степень сжатия от 1 до 8, т.е. не увеличивает размер данных в худшем случае, обеспечивает лучшую степень сжатия, чем алгоритм Хаффмана (в среднем 1-10%). Не является алфавитным кодированием. Весь кодируемый текст представляется в виде дроби из $[0, 1)$.

Дробь строится таким образом, чтобы текст был представлен как можно компактнее. Возьмем начальный интервал $[0, 1)$ и разобьем на подынтервалы с длинами, равными вероятностям появления символов в потоке. В дальнейшем будем называть их диапазонами соответствующих символов. Пусть $x =$ математика. Получим таблицу

символ	частота	вероятность	диапазон
а	3	0,3	[0; 0,3)
м	2	0,2	[0,3; 0,5)
т	2	0,2	[0,5; 0,7)
е	1	0,1	[0,7; 0,8)
и	1	0,1	[0,8; 0,9)
к	1	0,1	[0,9; 1)

Будем считать, что таблица известна в компрессоре и декомпрессоре. Кодирование заключается в уменьшении рабочего интервала. Для первого символа в качестве рабочего интервала берется $[0; 1)$. Он разбивается на диапазоны в соответствии с заданными частотами символов. В качестве следующего рабочего интервала берется диапазон, соответствующий текущему кодируемому символу. Его длина пропорциональна вероятности появления этого символа в потоке. Далее считывается следующий символ. В качестве исходного берем рабочий интервал, полученный на предыдущем шаге, и опять разбиваем его в соответствии с таблицей диапазонов. Длина рабочего интервала уменьшается пропорционально вероятности текущего символа, а точка начала сдвигается вправо пропорционально началу диапазона для этого символа. Новый построенный диапазон берется в качестве рабочего и т.д.

текущий символ	рабочий интервал	длина интервала	1/10 длины
-	[0;1)	1	0,1
м [0,3; 0,5)	[0,3; 0,5)	0,2	0,02
а [0; 0,3)	[0,3; 0,36)	0,06	0,006
т [0,5; 0,7)	[0,33; 0,342)	0,012	0,0012
е [0,7; 0,8)	[0,3384; 0,3396)	0,0012	0,00012

Таким образом, окончательная длина интервала равна произведению вероятностей всех встретившихся символов, а его начало зависит от порядка следования символов в потоке. Если обозначить диапазон символ c как $[a(c), b(c))$, а интервал для i-го кодируемого символа $[l_i, r_i)$, то алгоритм имеет вид
$l_0 = 0$, $r_0 = 1$, $i = 0$;
пока не конец слова
$c = x[i]$; $i++$;
$l_i = l_{i-1} + a(c)(r_{i-1} - l_{i-1})$;
$r_i = l_{i-1} + b(c)(r_{i-1} - l_{i-1})$.

В качестве кода берется произвольное число, лежащее в полученном интервале $[l_i, r_i)$. Для последовательности $x =$ "мате", состоящей из 4 символов, можно взять $y = 0,339$. Этого числа достаточно для восстановления исходной цепочки, если известна исходная таблица диапазонов и длина цепочки. Дальше это число можно перевести в двоичную дробь и закодировать цифрами после запятой.

Рассмотрим работу алгоритма декомпрессии. Каждый следующий интервал вложен в предыдущий. Это означает, что если есть число $0,339$, то первым символом может быть только М, так как только его диапазон включает это число. В качестве интервала берется диапазон М $[0,3; 0,5)$, разбивается на подинтервалы, среди которых находится содержащий $0,339$. Это $[0,3; 0,36)$, соответствующий символу "А". Дальше разбивается в соответствии с таблицей этот интервал и т.д.

Задание.

1. Закодировать первые четыре символа сообщения x="ков.корова":
   - составить таблицу частот и диапазонов всех символов сообщения,
   - найти рабочий интервал для "ков." и выбрать число y – код слова,
   - найти рабочий интервал и код для слова "кова" (использовать таблицу диапазонов из предыдущего задания),
   - рассмотреть процесс декомпрессии (восстановления слова "ков." по числу y).
2. Выполнить декомпрессию кода $y = 0.75$, используя построенную в первом задании таблицу диапазонов, если известно, что длина сообщения 10 символов.
3. Выполнить декомпрессию сообщения $y = 0.5$ для любой длины n (таблица диапазонов из первого задания).
4. Для алфавита $A = \{0, 1\}$ с распределением вероятности $p = \{2/3, 1/3\}$ построить коды по методу арифметического сжатия для всех слов длины 3. Код слова y выбирать как число, представимое в виде целого числа, деленного на минимально возможную положительную степень двойки. Найти для y представление в виде двоичной дроби. Найти вероятности сообщений и среднее количество бит на символ (кодом y являются цифры двоичной дроби после точки), сравнить с энтропией.