Сжатие данных

Синонимы: упаковка, компрессия, сжимающее кодирование, кодирование источника

Антонимы: восстановление, распаковка, декомпрессия

# Определения

Сжатие данных – алгоритм преобразования данных с целью уменьшения занимаемого ими объема на носителе информации. Процесс сокращения числа бит, необходимых для хранения одного и того же числа информации. Сокращение избыточности.

Самый простой алгоритм сжатия: добавляем бит к повторяющейся последовательности, 0 - данные, 1 - число повторов.

Методы:

1. устранение повторений
2. изменение модели хранения данных
3. сжатие с потерями

**Избыточность** – превышение информации, используемой для передачи сообщения над его информационной энтропией.

Пример 1: повторение в тексте фрагментов. *Устраняется* заменой повторяющейся последовательности ссылкой на уже закодированный фрагмент с указанием его длины.

Пример 2:изменение модели хранения данных. *Устраняется* заменой часто встречающихся данных короткими кодовыми словами, а редких длинными кодовыми словами. [Энтропийное кодирование]

Сжатие данных с потерями – метод сжатия данных при использовании которого восстановление сжатых данных может быть неоднозначным.

Примеры: использовать для аудио файлов, для видео файлов.

Сжатие данных без потерь – метод сжатия данных, при использовании которого восстановление сжатых данных однозначно до бита, пикселя и так далее.

Примеры: использовать для текста, для программного кода.

Модель источника данных (модель избыточности) – формальное описание априорных сведений об источнике сжимаемых данных.

Модель источника данных может быть статической и динамической.

* Статическая – неизмененная для всего сжимаемого сообщения.
* Динамическая – строится и параметризируется на этапе сжатия.

Метод сжатия может быть адаптивным и неадаптивным.

* Адаптивный – методы, позволяющие на основе входных данных изменять модель избыточности информации.
* Неадаптивный – методы, применяемые для работы с данными, обладающими хорошо определенными и неизменными характеристиками.

Коэффициент сжатия:

Если k = 1, то сжатия нет. Если k < 1, то сжатие неэффективно. Если k > 1, то сжатие эффективно.

Алгоритм Шеннона-Фано, 1948

1. Строится таблица частот встречающихся символов.

2. Выбираются два подмножества символов, в которых сумма частот оных примерно равна (например: 10,9,5,1 частоты -> 10+1, 9+5), строится дерево. 0 - налево по дереву, 1 - направо. Буквы можно кодировать, например:

ABCD

0 BC

0 B

1 C

1 AD

0 A

1 D

Таким образом, A=10

Подмножества выделяются до тех пор, пока множество не будет содержать один символ.

Недостаток - нет четкой формализации, как будет выглядеть модель сжатия (сжимать можно разными методами). Четкого соответствия нет - множества выбираются по каким-то внутренним принципам, которые могут расходиться в разных случаях.

Алгоритм:

1. Символы первичного алфавита m1 выписывают по убыванию вероятностей.
2. Символы полученного алфавита делят на две части, суммарные вероятности символов которых максимально близки друг другу.
3. В префиксном коде для первой части алфавита присваивается двоичная цифра «0», второй части — «1».
4. Полученные части рекурсивно делятся и их частям назначаются соответствующие двоичные цифры в префиксном коде.

## Алгоритм Хаффмана

Основное отличие – построение дерева начинается с листьев в корень.

1. *Таблица частот* (сортировка)
2. Строится *дерево*: два символа с наименьшим весом (встречаемостью) (C,D). Они образуют узел с весом их суммарной встречаемости. Суммарный вес (6) заменяет их в таблице частот. Дальше берем 6 и B, образуют узел 15. Дальше берем 15 и A, образуем узел 25.
3. Аналогично Ш-Ф, *помечаются дуги* дерева как 0 налево и 1 направо.
4. ~~Бит 1 в файле всегда обозначает лист, а количество нулей перед ним - какой именно из листов.~~

//*Нихрена подобного!* [*http://habrahabr.ru/post/144200/*](http://habrahabr.ru/post/144200/)

Вместе с файлом необходимо хранить и исходную таблицу частот (либо само дерево).

Необходимо также учитывать остающиеся в хвосте пустые биты - они могут раскодироваться в совершенно лишнюю информацию.

## Адаптивный алгоритм Хаффмана

Позволяет избавиться от необходимости хранить дерево. При считывании очередного символа - перестраивается модель данных. Считали второй символ - получили два символа с равной частотой, четвертый, пятый; каждый раз перекодируется дерево. Алгоритм будет катастрофически медленным. Узел должен встраиваться в дерево сразу же на правильную позицию.

Алгоритм LZW (Лемпэл, Зив 1978; Уэлч 1984)

Текст «Going, going, gone!».

Строится таблица: код символа | префикс | суффикс. Заполняется символами из АСКИ-таблицы с -1 во 2-м и темже символом в 3-м столбце.

|  |  |  |
| --- | --- | --- |
| Код | префикс | суффикс |
| 0 | -1 | 0 |
| 1 | -1 | 1 |
| . | . | . |
| 255 | -1 | 255 |

Считывается первый символ из файла (G=71) и второй (o=111):

|  |  |  |
| --- | --- | --- |
| 256 | 71 | 111 |

Третий (i=105):

|  |  |  |
| --- | --- | --- |
| 257 | 111 | 105 |

Далее:

|  |  |  |
| --- | --- | --- |
| 258 | 105 | 110 |
| 259 | 110 | 103 |
| 260 | 103 | 44 |
| . | . | . |
| 264 | 111 | 105 | //соответствует *oi* в *Going* |

Это позволяет нам заменить комбинацию 111 105 на 257 110

Закодированным файлом будет последовательность префиксов начиная с 256 строки. Когда считываются элементы, если символ выходит за пределы 256 - он уже где-то есть в таблице выше.

## Алгоритм арифметического кодирования

Опять таблица частот. BILL GATES ->

|  |  |  |
| --- | --- | --- |
| символ | вероятность | интервал |
| \_ | 0.1 | 0.0-0.1 |
| a | 0.1 | 0.1-0.2 |
| b | 0.1 | 0.2-0.3 |
| e | 0.1 | 0.3-0.4 |
| g | 0.1 | 0.4-0.5 |
| i | 0.1 | 0.5-0.6 |
| l | 0.2 | 0.6-0.8 |
| s | 0.1 | 0.8-0.9 |
| t | 0.1 | 0.9-1.0 |

Кодируется дробным числом. Интервал 0..1 разбивается на вероятности, получаем дерево интервалов. По нему можно построить другую таблицу:

|  |  |  |  |
| --- | --- | --- | --- |
| символ | низ | верх | интервал |
| B | .2 | .3 | .1 |
| I | .25 | .26 | .01 |
| L | .256 | .258 | .002 |

В конечном итоге, строка будет закодирована дробью: 0.257167752

# Хаффман

1. **Какими могут быть коды значений по размеру?** Если смотреть по дереву то от 1 до 255 байт (смотря от реализации мб сразу биты у вас конечно)
2. **Как хранится алфавит?** Либо деревом, либо таблицей частот
3. **И где хранится?** (ну тут у каждого свое)
4. **Как обрабатывается отступ до 8 бит, где хранится эта информация и как при расшифровке это работает?** Если максимальная длина 255, а была 8, то как файл может весить в итоге меньше - мы учитываем частоты, поэтому символы с коротким кодом компенсируют эту разницу.

# LZW

1. Что алгоритму надо знать для расшифровки?
2. Описать процесс расшифровки
3. **Что происходит, когда расшифровщик натыкается на символ из нижней части таблицы (>255)?** У него есть уже коды одинарных символов, так как он идёт по порядку, если код больше то он просто возьмёт сумму текущего и след. Вот [тут](https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_LZW) есть снизу пояснение. Там словарь заполняется одинарными каждый раз перед началом прогона алгоритма, если такого символа нет, значит это несколько символов то он берет предыдущий + текущий и заносит в словарь.