# Архиватор

Разработайте приложение, которое бы сжимало и распаковывало произвольный файл. Имена входного и выходного файла и режим работы должны считываться из командной строки. Выходной текстовый файл должен содержать таблицу или дерево Хаффмана в компактном виде, а также выходную битовую последовательность (в текстовом виде).

Для сжатия данных используем кодирование Хаффмана - алгоритм сжатия данных, который используется в том числе и как часть алгоритма сжатия изображений JPEG. Кодирование Хаффмана основано на анализе входного потока и присваивании каждому символу кода (последовательности бит) так, чтобы наиболее частые символы имели короткие коды, а те, что встречаются реже — более длинные. В выходном потоке каждый символ заменяется своим кодом, что обычно приводит к сокращению объема потока. Например, если необходимо сжать слово «вата», коды можно назначить следующим образом:

|  |  |
| --- | --- |
| а   в   т | 0   10  11 |

Сжатый текст будет следующим:100110, т.е. 6 бит вместо 32 при обычном хранении. (На практике необходимо так же хранить таблицу кодов, поэтому кодирование Хаффмана более эффективно на больших объемах данных).

Коды находятся построением дерева, в котором располагаются все символы текста:

0/\1

/ \

а 0/\1

/ \

в т

Код каждого символа вычисляется как путь от корня дерева до этого символа. Все переходы влево маркируются 0, а вправо — 1. более частые символы располагаются выше в дереве и получают более короткие коды.

Для построения дерева сначала построим тривиальное дерево (содержащее только один символ) для каждого символа входного потока, присвоив ему вес, равный количеству вхождений данного символа во входной поток. Например, строим

* дерево, содержащее только «в» с весом 1,
* дерево, содержащее только «а» с весом 2,
* дерево, содержащее только «т» с весом 1.

Затем соединяем два дерева с *минимальными* весами в одно, присваивая ему суммарный вес исходных деревьев. Например, объединяя деревья «в» и «т» получаем дерево с весом 2:

0/\1

/ \

в т

Данный шаг повторяется до тех пор, пока не останется одно дерево с весом, равным общему количеству символов в файле. Коды Хаффмана можно считать из полученного дерева.

Определите функцию encode, которая кодирует текст, заменяя символы их кодами. Например: encode exampleCodeTable "вата" == "100110".

Определите функцию extractCodes, которая преобразует дерево Хаффмана в соответствующую таблицу кодов.

Определите функцию buildTree, которая по списку частичных деревьев получает полное дерево, объединяя по два дерева с наименьшими весами.

Определите функцию huffmanCode, которая строит таблицу кодов по данной строке.

Определите функцию compress, которая сжимает строку по методу Хаффмана. На выходе функции должна быть закодированная битовая последовательность и дерево или таблица кодов.

Определите функцию expand, которая по корректной закодированной битовой последовательности и дереву или таблице кодов разжимает строку по методу Хаффмана.