Сжатие данных. Методы сжатия без потерь

1. Сжатие информации
2. Статистические методы сжатия
3. Словарные алгоритмы сжатия

Избыточность информации

- **С НТ БРЬ**
- **Д Р ВО**
- **Н LLO**
- **М С OS FT**
- **WI DO S**

По рзелульаттам илссеовадний одонго анлигйсокго унвиертисета, не иеемт занчнеия, в кокам пряокде рсапожолены бкувы в солве.

Галвоне, чотбы преавя и пслоендяя бквуы блыи на мсете. Осатьлыне бкувы мгоут селдовтаь в плоонм бсепордяке, все-рвано ткест чтаитсея без побрелм.

Пичрионй эгото ялвятеся то, что мы не чиатем кдаужю бкуву по отдльенотси, а все солво цликеом

Сжатие информации

Сжатие информации - процесс преобразования информации, хранящейся в файле, с уменьшением избыточности в ее представлении

Сжатие

обратимое: тексты, компьютерные программы, документы, чертежи: GIF,PNG,ZIP,CAB,ARJ,RAR

необратимое: речь, музыка, изображения: JPEG,MPEG,MP3

Показатели эффективности сжатия

$$r = \frac{\textit{размер данных источника в битах}}{\textit{размер сжатых данных в битах}} = \frac{n \cdot \log(\dim A)}{k}$$

где ***r*** **–** коэффициент сжатия

dim A **–** размер алфавита данных **A**;

n **–** длина сообщения;

k **–** длина сжатого сообщения

$$R = k/n$$

где ***R*** **–** скорость сжатия, количество кодовых бит, приходящихся на отсчет данных источника

Упаковка целочисленных данных

25 36 -8 24 9 -3 4 -9 9 46 78 -1 0 112

Мощность алфавита = 12, код равномерный префиксный

Информационная емкость символа = 4 бита

BCD (Binary Coded Decimal) – формат для хранения целых чисел

0	**0000**
1	**0001**
2	**0010**
3	**0011**
…	**…**
8	**1000**
9	**1001**
-	**1010**
	1011

Вычислить коэффициент сжатия для последовательности

25 36 -8 24 9 -3 4 -9 9 46 78 -1 0 112

$$r = \frac{\textit{размер данных источника в битах}}{\textit{размер сжатых данных в битах}} = \frac{n \cdot \log(\dim A)}{k}$$

где A=256 (таблица ASCII);

n =38;

k=38*4

$$r = \frac{n \cdot \log A}{k} = \frac{38 \cdot 8}{38 \cdot 4} = 2$$

Дифференциальное кодирование

используется в тех случаях, когда **соседние значения незначительно отличаются друг от друга** (сами значения могут быть сколь угодно большими).

Для 8-битового растрового изображения

144, 147, 150, 146, 141, 142, 138, 143, 145, 142 **10*8 = 80 бит**

вычислим разности между соседними кодами:

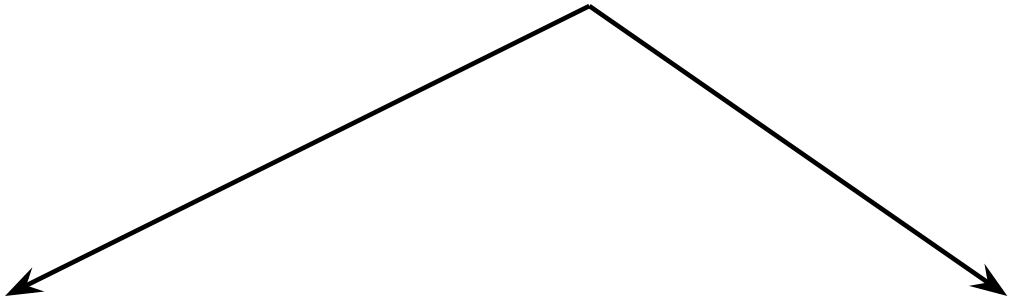
144	147	150	146	141	142	138	143	145	142
⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓
144	3	3	- 4	- 5	1	- 4	5	2	-3

Для первого числа 8 бит, все остальные по 4 бита (как BCD):

8 + 9*4 = 44 бит

$$r = \frac{n \cdot \log A}{k} = \frac{80}{44} \approx 1{,}82$$

АЛГОРИТМЫ ОБРАТИМОГО СЖАТИЯ



Статистические

- RLE;
- арифметическое кодирование
- Хаффмана;

Словарные

- Лемпела-Зива (LZ77, LZ78);
- Лемпела-Зива-Уэлча (LZW);
- Сторера-Шимански (LZSS)

Статистическое кодирование - кодирование неравномерными кодами, длительность кодовых комбинаций которых согласована **с вероятностью появления различных букв**

Более вероятные буквы кодируют более короткими комбинациями кода, менее вероятные - более длинными для уменьшения средней длины сообщения

Словарные методы кодирования основываются на замене **участка сообщения** (уже появлявшегося в тексте) на **указатель в словаре**

Метод «приспосабливается» к структуре текста; **функциональные** слова (часто появляющиеся) сохраняются как указатели. Новые слова и фразы могут формироваться из частей ранее встреченных слов.

Словарь содержится в обрабатываемых данных в неявном виде.

Сжатие способом кодирования серий
(**RLE** Run Length Encoding: .bmp, .pcx)

Исходный код

11111111 11111111 11111111 11111111
11111111 11110000 00001111 11000011
10101010 10101010 10101010 10101010

Управляющий байт: при повторениях первый бит **1**; при неповторяющихся группах первый бит **0** и затем идет **счетчик**, показывающий, сколько за ним следует неповторяющихся данных

Сжатый код

10000101 11111111
00000011 11110000 00001111 11000011
10000100 10101010

r=12*log(256)/8*log(256)=1.5

Упаковать методом RLE двоичное сообщение, вычислить коэффициент сжатия

11100011 11100011 10011101 10011101 10011101
10011101 00111100 11000011 00111100 11000011
00111100 11000011 11000011 00001111 00001111
10000001 10000010 10000011 10000100 10000101
10000110

10000010 11100011 10000100 10011101 00000101
00111100 11000011 00111100 11000011 00111100
10000010 11000011 10000010 00001111 00000110
10000001 10000010 10000011 10000100 10000101
10000110

r=18*log(256)/18*log(256)=1

алгоритм Хаффмана

1. Символы алфавита отсортированы по вероятности их появления в тексте
2. Последовательно объединяют два символа с min вероятностями появления в **новый составной символ**, суммируя их вероятности
3. Строится дерево, каждый узел которого имеет **суммарную вероятность всех узлов, находящихся ниже него**
4. Выполняется **новая сортировка**
5. Задаются коды к вершинам, с учетом направления к узлам (например, направо - 1, налево – 0)

Алгоритм Хаффмана

A B C D E
10 5 8 13 10

B C A E D
5 8 10 10 13

A E BC D
10 10 13 13

BC D AE
13 13 20

AE BCD
20 26

AEBCD
46

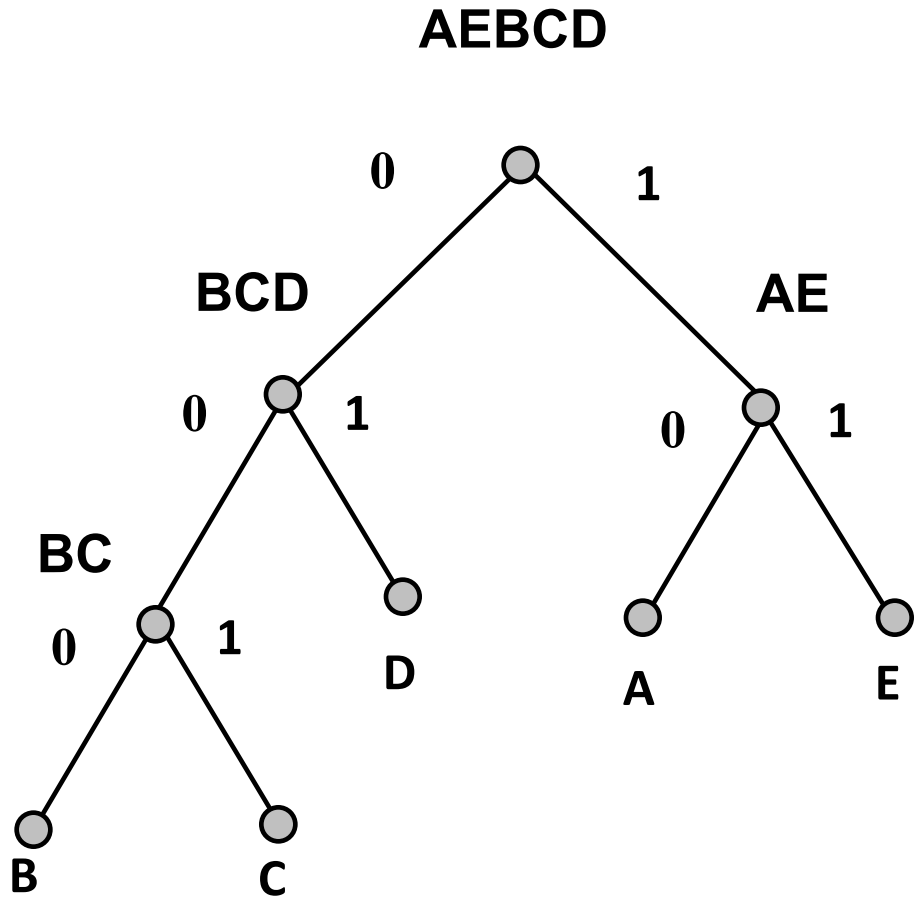


Таблица кодирования				
A	**B**	**C**	**D**	**E**
10	**000**	**001**	**01**	**11**