Информация и информационные процессы

§ 3. Сжатие данных

Что такое сжатие?

Алфавит: **A, B, C, ␣**

Сообщение: **ABA CABABA**

A → 00 **C** → 10

B → 01 **␣** → 11

ABA CABABA → 00 01 00 11 10 00 01 00 01 00

20 битов

80 битов в 8-битной кодировке!

Как раскодировать?

Словарь:

	00	01	10	11
00000100_2	01000001_2	01000010_2	01000011_2	00100000_2
4 символа	**A** (код 65)	**B** (код 66)	**C** (код 67)	**пробел** (код 32)

Коэффициент сжатия

Сообщение: **10240 символов**
Алфавит: **A, B, C, ␣**
Словарь: **5 байтов**
Длина кода:
10240×2 = 20480 битов = **2560 байтов**
Длина сжатого сообщения:
5 + 2560 = **2565 байтов**

Коэффициент сжатия – это отношение размеров исходного и сжатого файлов.

$$k = \frac{10240}{2565} \approx 4$$

Сжатие без потерь

Сжатие без потерь – это такое уменьшение объема закодированных данных, при котором можно восстановить их исходный вид из кода без искажений.

? За счёт чего сжимается сообщение?

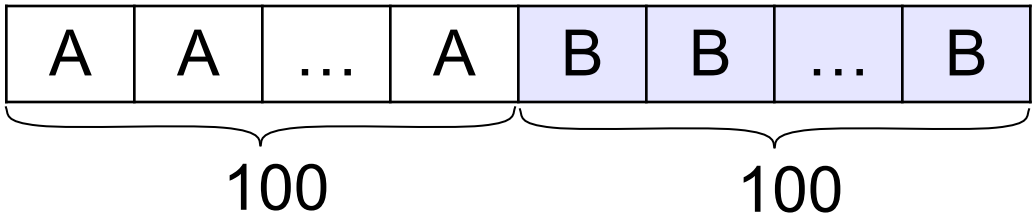
! В данных должна быть избыточность!

используются только 4 символа из 256

Алгоритм RLE

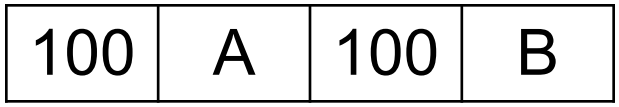
RLE (англ. *Run Length Encoding,* кодирование цепочек одинаковых символов)

Файл `qq.txt`



200 байтов

Файл `qq.rle` (сжатый)



4 байта

сжатие в 50 раз!

? В чем состоит избыточность?

? Сжатие с потерями или без?

? Что в худшем случае?

Алгоритм RLE

управляющие байты

8F C0 02 C1 C2$_{16}$

8F$_{16}$	**C0$_{16}$**	**02$_{16}$**	**C1$_{16}$**	**C2$_{16}$**
10001111_2	11000000_2	00000010_2	11000001_2	11000010_2
повтор 15	**A** (код 192)	2	**Б** (код 193)	**В** (код 194)

Распаковка:

15 2

AAAAAAAAAAAAAAAAA

Применение:

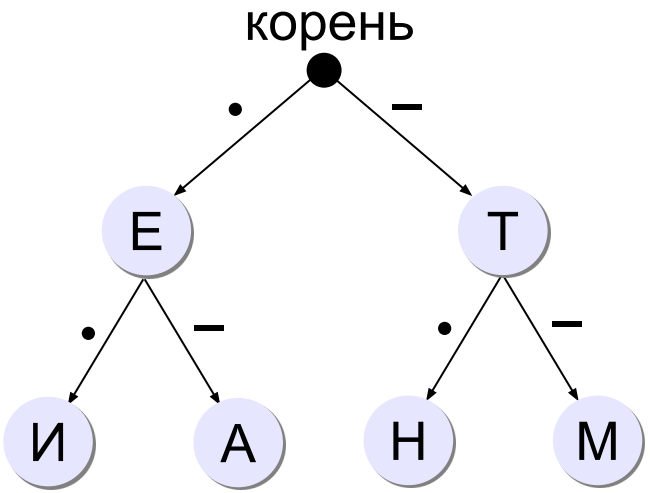
- сжатие рисунков `*.bmp` (с палитрой)
- один из этапов сжатия рисунков `*.jpg`

Неравномерные коды

Идея: кодировать часто встречающиеся символы более короткими кодовыми словами.

Азбука Морзе:

E	•	**И**	• •
T	–	**A**	• –
		H	– •
		M	– –



! Проблема: разделить последовательность на кодовые слова!

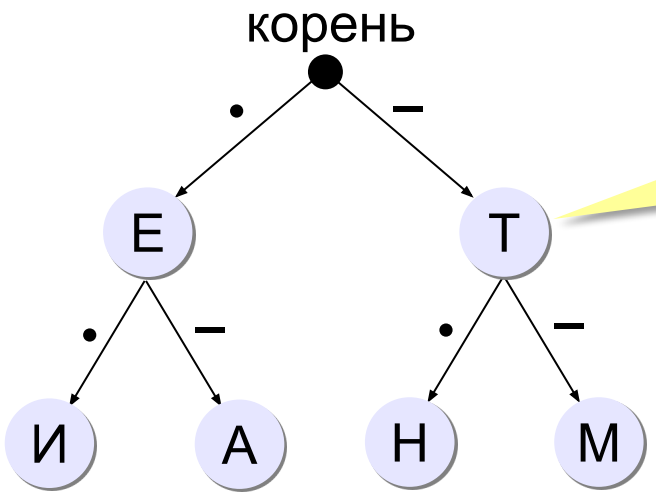
• • → **И**, **ЕЕ**

? Можно ли обойтись без разделителя?

Префиксные коды

Префиксный код – это код, в котором ни одно кодовое слово не является началом другого кодового слова (условие Фано).

Е	•	**И**	• •
Т	–	**А**	• –
		Н	– •
		М	– –



не все символы в листьях!

! Это не префиксный код!

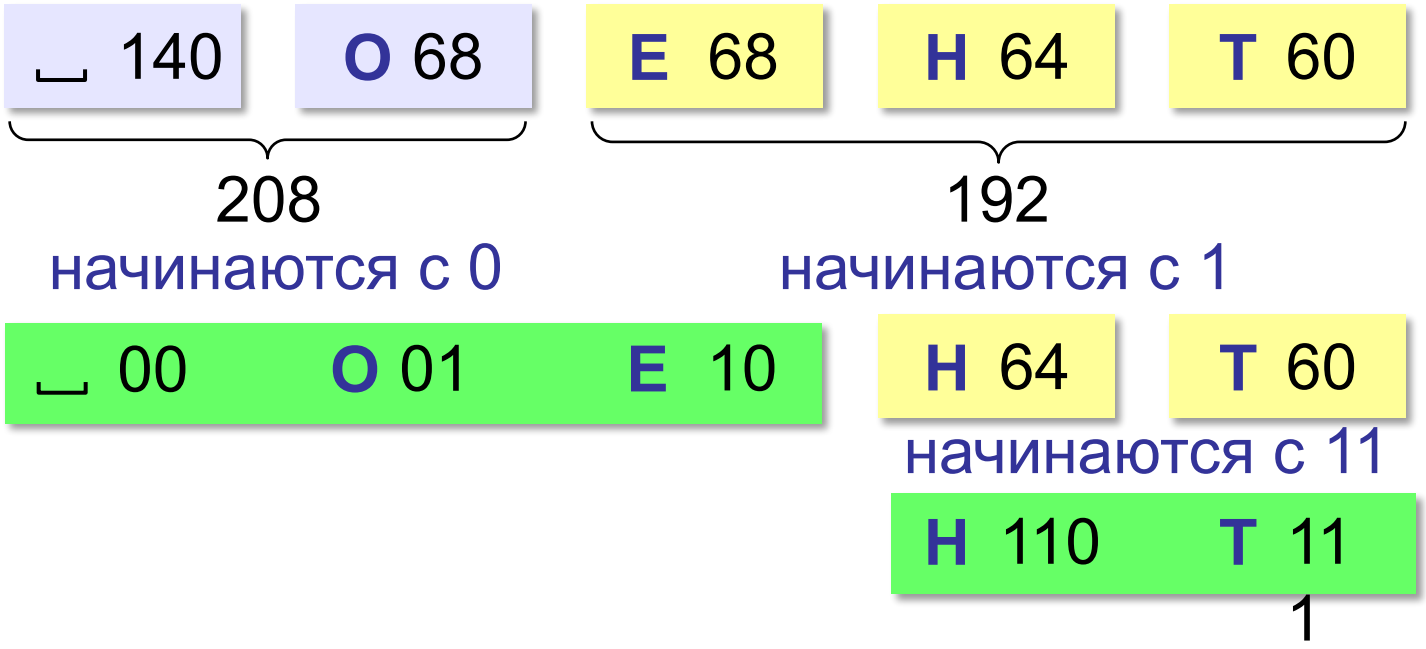
! Проблема: как построить префиксный код?

Код Шеннона-Фано

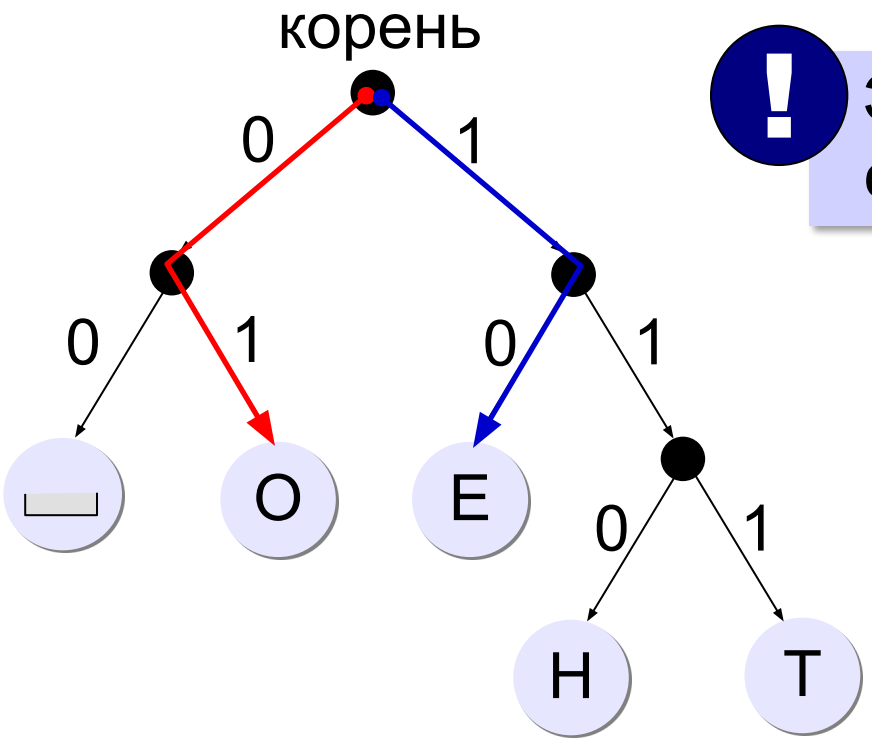
Алфавит: О, Е, Н, Т, ␣

Количество символов в сообщении:

На 2 группы с примерно равным числом символов:



Код Шеннона-Фано



Это префиксный код (все символы в листьях дерева)!

Декодирование:

1 1 0 1 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 1 1 1

1 Т О 1 Т О ␣ Е Н О 1 Т

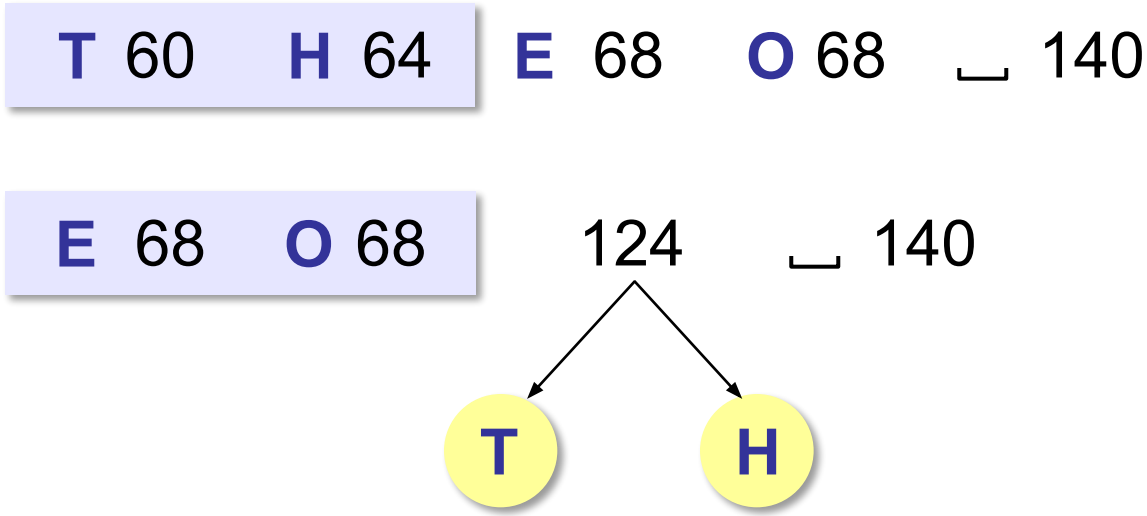
Код Шеннона-Фано

- учитывается частота символов
- не нужен символ-разделитель
- код префиксный – можно декодировать по мере поступления данных

- нужно заранее знать частоты символов
- код неоптимален
- при ошибке в передаче сложно восстановить «хвост»
- не учитывает повторяющиеся последовательности символов

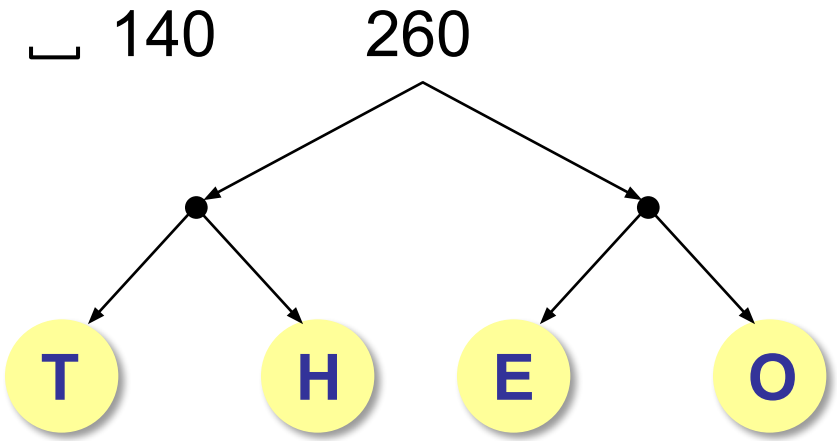
Алгоритм Хаффмана

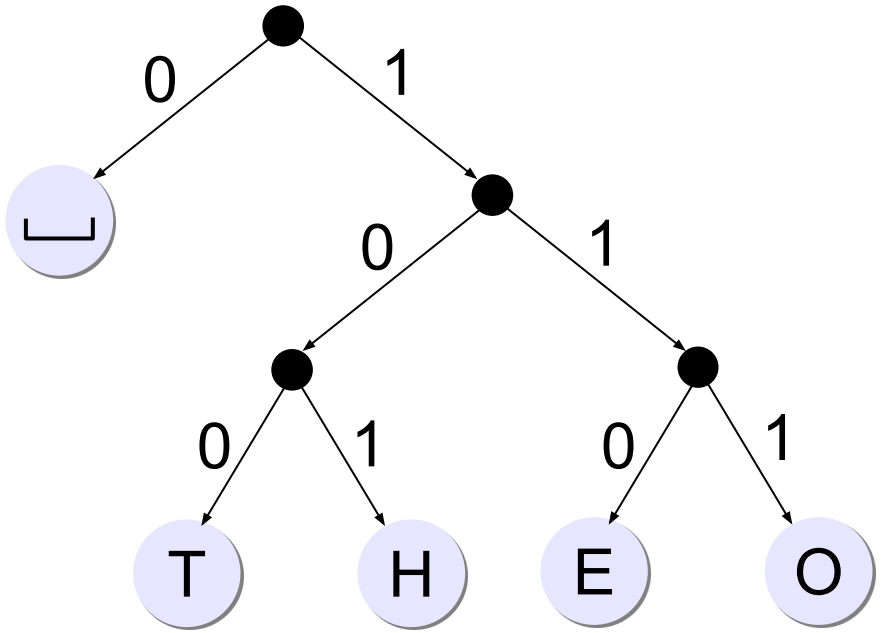
По увеличению частоты:



Дэвид Хаффман

Алгоритм Хаффмана





Код Хаффмана:

␣ 0 **T** 100 **H** 101

E 110 **O** 111

Сравнение алгоритмов

Количество символов в сообщении:

␣ 140 **О** 68 **Е** 68 **Н** 64 **Т** 60

Равномерное кодирование (8-битный код):

(140 + 68 + 68 + 64 + 60) · 8 = 3200 битов

Равномерное кодирование (3-битный код):

(140 + 68 + 68 + 64 + 60) · 3 = 1200 битов

+ словарь!

? В чём избыточность?

Сравнение алгоритмов

Количество символов в сообщении:

␣ 140 **О** 68 **Е** 68 **Н** 64 **Т** 60

Код Шеннона-Фано:

␣ 00 **О** 01 **Е** 10 **Н** 110 **Т** 111

$(140 + 68 + 68) \cdot 2 + (64 + 60) \cdot 3 = 924$ бита

$$k = \frac{1200}{924} \approx 1{,}299$$

Код Хаффмана:

␣ 0 **О** 11 **Е** 110 **Н** 101 **Т** 100

$140 + (68 + 68 + 64 + 60) \cdot 3 = 920$ бит

$$k = \frac{1200}{920} \approx 1{,}304$$

! Оптимален!

Алгоритм Хаффмана

- код оптимальный среди алфавитных кодов

- нужно заранее знать частоты символов
- при ошибке в передаче сложно восстановить «хвост»
- не учитывает повторяющиеся последовательности символов

Алгоритм LZW

1977: А. Лемпел и Я. Зив, 1984: Т. Велч

Идеи:

- кодировать не отдельные символы, а блоки
- последовательностям символов присваиваются числовые коды
- новая цепочка ⇒ занесение в словарь с новым кодом

- словарь строится по мере получения данных
- не нужны частоты символов ⇒ за один проход!

Применение:

- сжатие рисунков **`*.gif`**, **`*.tif`**
- сжатие документов **`*.pdf`**

Сжатие с потерями

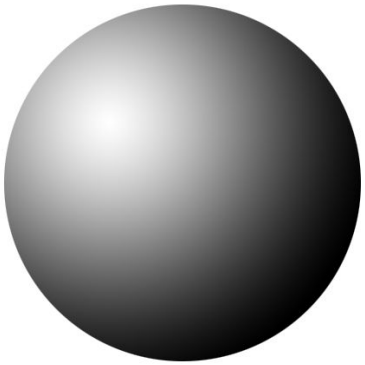
Сжатие с потерями – это такое уменьшение объема закодированных данных, при которых распакованный файл может отличаться от оригинала.

Идея: «отбросить» часть данных, которые не влияют на восприятие информации человеком (доп. размытие фотографий, частоты выше 20 кГц, …)

Применение:

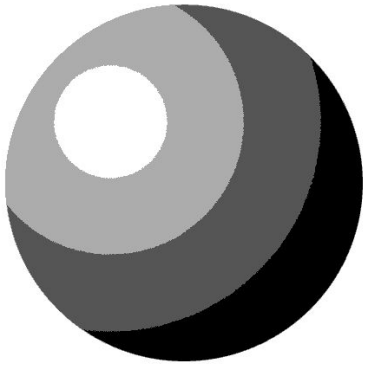
- сжатие рисунков **`*.jpg`**, **`*.jpeg`**
- сжатие звука **`*.mp3`**, **`*.aac`**, **`*.ogg`**, …
- сжатие видео **`*.mpg`**, **`*.wmv`**, **`*.mov`**, …

Снижение глубины цвета



8 битов на пиксель
(256 цветов)

4 бита на пиксель
(16 цветов)



2 бита на пиксель
(4 цвета)

размер ↓

качество ↓

Сжатие JPEG

яркость

«синева»

RGB → Y Cb Cr

«краснота»

$Y = 0{,}299 \cdot R +$ глаз чувствительнее к зелёному!

$Cb = 128 - 0{,}1687 \cdot R - 0{,}3313 \cdot G + 0{,}5 \cdot B$

$Cr = 128 + 0{,}5 \cdot R - 0{,}4187 \cdot G - 0{,}0813$

? Что для чёрно-белого (серого)?

Cb = Cr = 128

Сжатие JPEG

Идея: глаз наиболее чувствителен к яркости

Y_1, Cb_1, Cr_1	Y_2, Cb_2, Cr_2
Y_3, Cb_3, Cr_3	Y_4, Cb_4, Cr_4

12 чисел

6 чисел

$$\Rightarrow Y_1, Y_2, Y_3, Y_4, Cb, Cr$$

например:

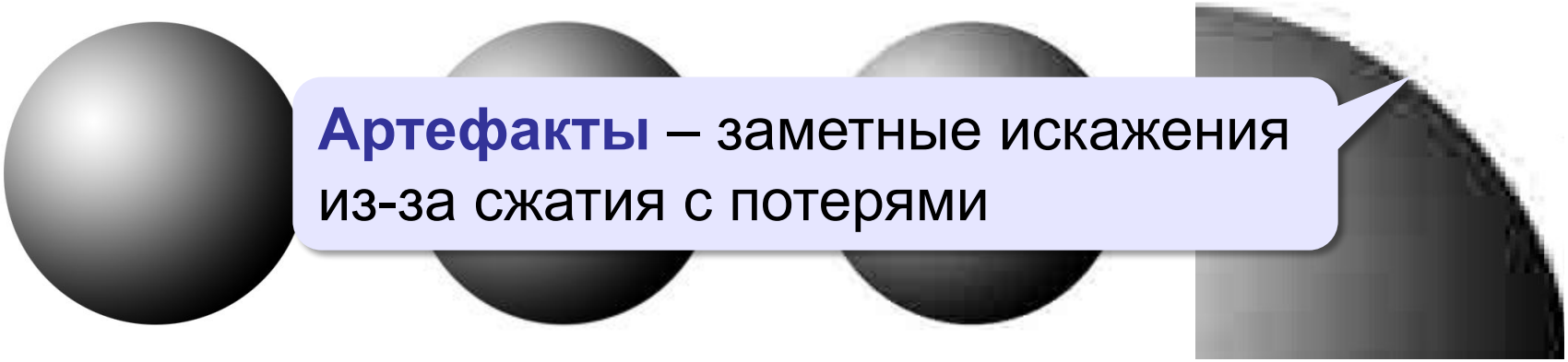
$$Cb = \frac{Cb_1 + Cb_2 + Cb_3 + Cb_4}{4}$$

$$Cr = \frac{Cr_1 + Cr_2 + Cr_3 + Cr_4}{4}$$

потери!

\+ *дискретное косинусное преобразование*, алгоритмы RLE и Хаффмана

Сжатие JPEG



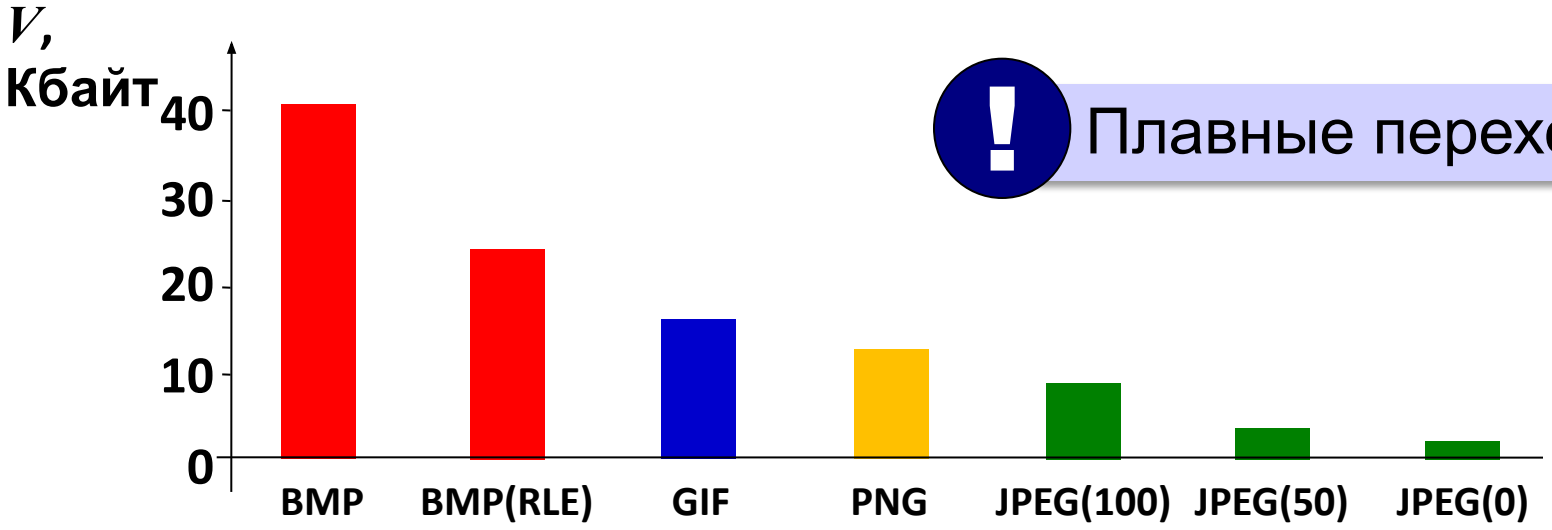
Артефакты – заметные искажения из-за сжатия с потерями

качество 100 (8400 байтов)

качество 50 (3165 байтов)

качество 0 (1757 байтов)

качество 0 (фрагмент)

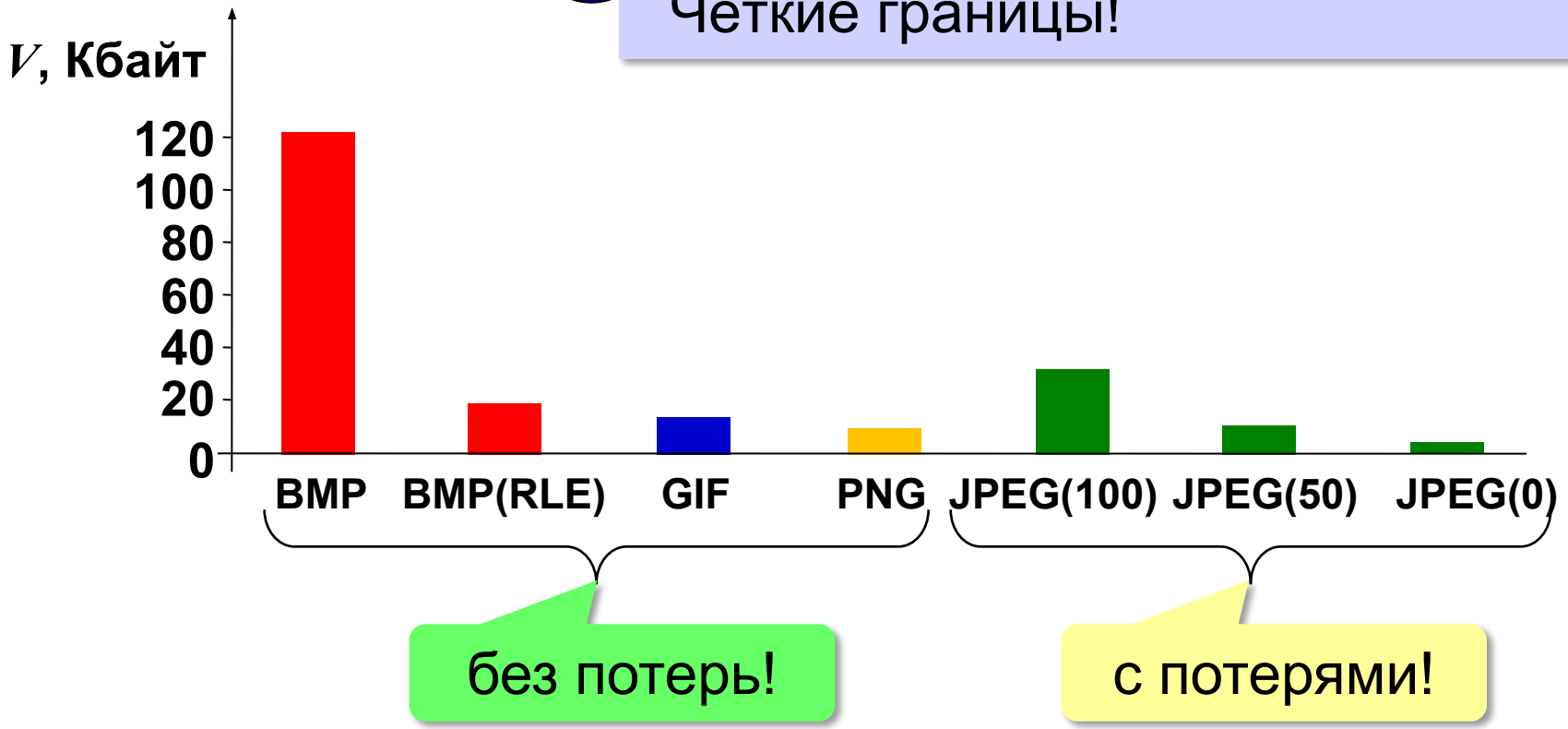


! Плавные переходы!

Сжатие рисунков с потерями и без

? Что особенного?

! Большие области одного цвета!
Чёткие границы!



Сжатие звука (MP3)

MP3 = MPEG-1 Layer 3, *кодирование восприятия*

Битрейт – это число бит, используемых для кодирования 1 секунды звука.

MP3: от 8 до 320 кбит/с

Без сжатия на CD (1 сек, 44 кГц, 16 бит, стерео):
2×88000 = 176 000 байт = 1 408000 бит = **1408 кбит**

Сжатие MP3 (**256 кбит**/с):

$$k = \frac{1408}{256} \approx 5{,}5$$

Сжатие видео

видео = изображения + звук

Кодек (кодировщик/декодировщик) – это программа для сжатия данных и восстановления сжатых данных.

MJPEG, MPEG-4, DivX, Xvid, H.264, …

Артефакты – заметные искажения из-за сжатия с потерями

Сжатие: итоги

Сжатие уменьшает избыточность данных!

Хорошо сжимаются:

- тексты (`*.txt`)
- документы (`*.doc`)
- несжатые рисунки (`*.bmp`)
- несжатый звук (`*.wav`)
- несжатое видео (`*.avi`)

Нужно ли стремиться к полному удалению избыточности?

Плохо сжимаются:

- случайные данные
- сжатые данные в архивах (`*.zip`, `*.rar`, `*.7z`)
- сжатые рисунки (`*.jpg`, `*.gif`, `*.png`)
- сжатый звук (`*.mp3`, `*.aac`)
- сжатое видео (`*.mpg`, `*.mp4`, `*.mov`)