### 

Запускаем программу и делаем различные тесты.

**1. Выгодно ли сжатие без учёта контекста?**

* **Для текстовых файлов** (например, test\_text.txt):  
  Частотности распределены неравномерно (чаще встречаются пробелы, буквы «о», «е» и т. п.). Энтропия **значительно меньше 8 бит/байт**, поэтому сжатие без учёта контекста (типа Huffman/Arithmetic без контекста) будет давать выигрыш.
* **Для равномерных бинарных файлов** (test\_bin.bin):  
  Байты распределены почти равномерно → энтропия близка к 8 бит/байт. Выигрыша нет, а размер архива даже **увеличится** из-за накладных расходов (таблицы частот).
* **Для монотонных файлов** (test\_repeated.bin):  
  Все байты одинаковые → энтропия = 0. Теоретически файл можно сжать «в ноль», но на практике архив будет **больше** за счёт служебной информации (минимум таблица частот).
* **Для маленьких файлов** (test\_small.bin):  
  Сжатие невыгодно: накладные расходы (256 байт или 2048 байт таблицы частот) многократно превышают сам файл.

Итог: сжатие без учёта контекста выгодно **только для «естественных» данных** с неравномерным распределением символов (тексты, логи, JSON и т. п.), но **бесполезно или вредно** для равномерных и маленьких файлов.

**2. Нужно ли нормировать частоты?**

* Для **оценки энтропии** обязательно нормировать:
* Для **архиватора** можно хранить:
  + **ненормированные частоты** (count) — они точнее, но занимают больше места (например, таблица из 256 чисел по 64 бита = 2048 байт),
  + **нормированные** (например, 8-битные вероятности) — меньше накладные расходы (256 байт), но модель менее точная.

Выбор зависит от формата архива: в теории лучше хранить ненормированные частоты, но на практике ради экономии места часто используют нормированные.

### 

Какой будет длина в октетах этих данных для исследуемых файлов? В нашей программе archive\_lower\_bound\_bytes – это оценка **размера данных в октетах**, учитывая энтропию текста и метаданные.

|  |  |
| --- | --- |
| Вводимый файл | Размер данных в октетах (байт) |
| mixed.txt --encoding UTF-32 | 274 |
| mixed.txt --encoding UTF-16 | 232 |
| mixed.txt --encoding UTF-8 | 224 |
| binary.bin --encoding UTF-8 | 301 |
| binary.bin --encoding UTF-16 | 330 |
| binary.bin --encoding UTF-32 | 390 |
| text.txt --encoding UTF-32 | 163 |
| text.txt --encoding UTF-16 | 139 |
| text.txt --encoding UTF-8 | 127 |
|  |  |

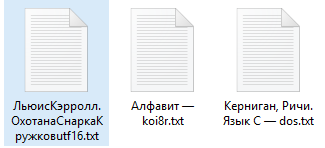
Для одинаковых длинных utf-8 файлов заметим: чем длиннее файл, тем больше разница между хранением только реально встречающихся символов и полным Unicode.

**Влияние выбора алфавита:**

* Если хранить только реально встречающиеся символы → размер метаданных маленький → архив значительно меньше.
* Если хранить весь Unicode → размер метаданных огромный → размер архива вырастает до ≈12–13 МБ, независимо от текста. Почти весь размер архива занимает таблица частот всего Unicode (≈12.75 МБ), независимо от длины файла.

### 

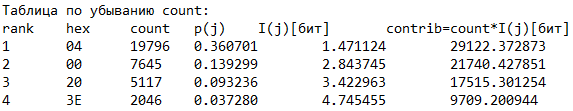
Возьмем 3 рандомных файла в разной кодировке:



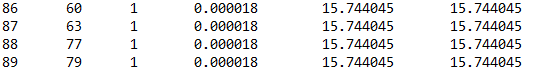
Введем команды

**py byte\_entropy.py ЛьюисКэрролл.ОхотанаСнаркаКружковutf16.txt -o report\_utf16.txt**

4 самых частых октета:

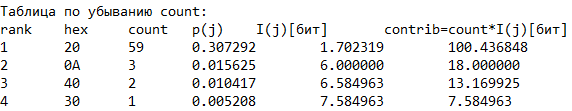


4 частых октета не ascii:

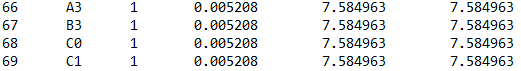


**py byte\_entropy.py Алфавитkoi8r.txt -o report\_koi8r.txt**

4 самых частых октета:

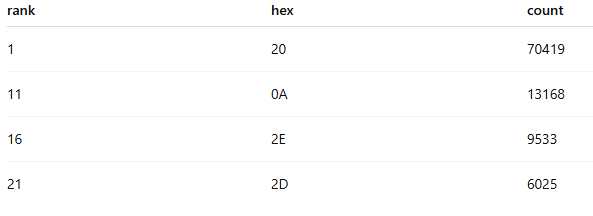


4 частых октета не ascii:



**py byte\_entropy.py Керниган,Ричи.ЯзыкCdos.txt -o report\_cdos.txt**

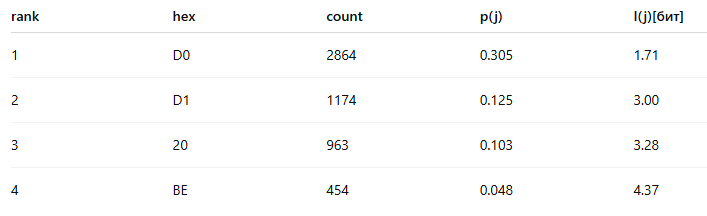
4 самых частых октета:



4 частых октета не ascii:



Теперь для текста по варианту 1 (1.txt):



Это наиболее частые октеты ascii.

* **0xD0 и 0xD1** — это характерные байты для кириллицы в кодировках **UTF-8** и **Windows-1251**, но чаще для UTF-8 много байт начинается с D0/D1 для русских букв.
* **0x20** — это пробел, нормальное значение для текста.

Делаем вывод, что у нас русскоязычный текст в кодировке utf-8.