

# Алгоритм Шеннона — Фано

Сирота Александр  
[github.com/nukefluke](https://github.com/nukefluke)

ГУАП  
5 факультет  
Группа 5511

Санкт-Петербург, 2016

# Содержание

1 Основные сведения

2 Построение кодов

3 Бинарное дерево

4 Реализация

5 Эффективность

6 Итоги

# Создатели и идея алгоритма

**Алгоритм Шеннона — Фано** — один из первых алгоритмов сжатия, который сформулировали американские учёные Клод Шеннон и Роберт Фано.

Главная идея алгоритма — заменить часто встречающиеся символы более короткими кодами, а редко встречающиеся символы более длинными кодами.

Алгоритм:

- Относится к вероятностным методам сжатия
- Использует коды переменной длины
- Использует префиксный код

# Методы алгоритма

**Вероятностные методы сжатия** опираются на частоту встречаемости символа в тексте.

При использовании **кодов переменной длины** символы кодируются набором бит различной длины. Часто встречающийся символ кодируется кодом меньшей длины, редко встречающийся — кодом большей длины.

**Префиксный код** — код, в котором никакое кодовое слово не является префиксом какого-то другого кодового слова.

## Префиксное кодирование

Префиксное кодирование необходимо для однозначного разбиения закодированной последовательности на слова. Например, код, состоящий из слов

0 10 11

является префиксным, и сообщение

01001101110

можно разбить на слова единственным образом:

0 10 0 11 0 11 10

# Префиксное кодирование

Код, состоящий из слов

0 10 11 100

префиксным не является (слово 10 является префиксом слова 100),  
и то же сообщение можно трактовать уже несколькими способами.

0 10 0 11 0 11 10

0 100 11 0 11 10

## Алгоритм построения кодов

Символ	Вероятность
a	0.36
b	0.18
c	0.18
d	0.12
e	0.09
f	0.07

Для начала мы должны определиться, какие символы встречаются чаще, а какие реже. Необходимо проанализировать кодируемое сообщение и создать таблицу вероятностей.

## Алгоритм построения кодов

Символ	Вероятность
a	0.36
b	0.18
c	0.18
d	0.12
e	0.09
f	0.07

Далее, таблица символов делится на две группы таким образом, чтобы каждая из групп имела приблизительно одинаковую частоту по сумме символов.

Это отличие алгоритма Фано от других подобных алгоритмов.



## Алгоритм построения кодов

Символ	Вероятность	1
a	0.36	0
b	0.18	
c	0.18	1
d	0.12	
e	0.09	
f	0.07	

Первой группе устанавливается начало кода в '0', второй в '1'.

## Алгоритм построения кодов

Символ	Вероятность	1	2	3	4	Итог
a	0.36	0	00			00
b	0.18		01			01
c	0.18	1	10			10
d	0.12		11	110		110
e	0.09			111	1110	1110
f	0.07				1111	1111

Процедура рекурсивно повторяется, пока в группе не останется только один символ.

# Итого

Алгоритм Фано:

- 1 Выписать символы по убыванию вероятностей.
- 2 Разделить список на две части с равными долями вероятности.
- 3 Для первой части добавить к коду «0», для второй — «1».
- 4 Повторить шаги (1–3) для каждой части.

# Бинарное дерево

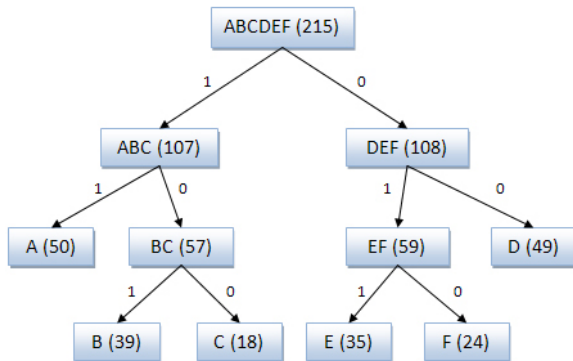
Для построения кодов удобно использовать бинарное дерево, так как после построения коды элементов-листьев получаются префиксными.

- Корень дерева — весь алфавит
- После разбиения группы на две подгруппы, каждая подгруппа образует узел
- Ветви до этих узлов обозначаются '0' и '1'
- Если в группе один элемент, то группа образует лист дерева
- Код элемента - это последовательность цифр на ветвях, которые ведут от корня к этому элементу

# Пример кодового дерева

Исходные символы:

Символ	Частота Встречаемости
A	0.23
B	0.18
C	0.08
D	0.23
E	0.16
F	0.11



Полученные коды:

A	B	C	D	E	F
11	101	100	00	011	010

## Проблемы реализации

- Иногда коды строятся неоптимально.  
Например, коды для последовательности  
«aaaaaaaaaaaaabbbbbbbccccddddddeeee»:

Символ	Вероятность	Код Хаффмана	Код Шеннона-Фано
a	0.4	0	00
b	0.20	111	01
c	0.14	101	10
d	0.14	110	110
e	0.11	100	111

Метод Хаффмана сжимает её до 77 бит, а Шеннона-Фано до 79 бит.

- Необходимо дописывать шапку в файл. Шапка содержит информацию о том, какие символы кодируются той или иной последовательностью.

# Интерфейс

## Запуск программы

```
$ ./fano <mode> <input> [ -o <output> ]
```

### <mode>

'-e' — кодирование. Расширение входного файла — любое.  
'-d' — раскодирование. Расширение входного файла **.fano**

### <input>

Входной файл.

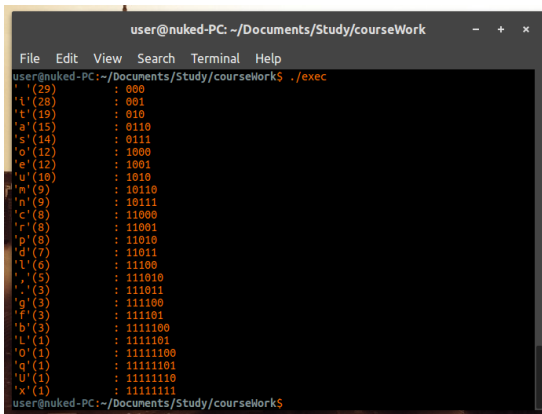
### <output>

Выходной файл.

# Пример сгенерированного словаря

Для фразы:

*«Lorem ipsum dolor sit amet, consectetur adipisicing elit. Odit sint cupiditate magni, illo officia facere magnam, ad pariatur ipsum explicabo sit nostrum aliquid nisi necessitatibus natus temporibus. Ut, optio, odit.»*



```
user@nuked-PC: ~/Documents/Study/courseWork
File Edit View Search Terminal Help
user@nuked-PC:~/Documents/Study/courseWork$ ./exec
' '(29) : 000
' '(28) : 001
' '(19) : 010
' a'(15) : 0110
' s'(14) : 0111
' o'(12) : 1000
' e'(12) : 1001
' u'(10) : 1010
' m'(9) : 10110
' n'(9) : 10111
' c'(8) : 11000
' r'(8) : 11001
' p'(8) : 11010
' d'(7) : 11011
' l'(6) : 11100
' ,'(5) : 111010
' .'(3) : 111011
' g'(3) : 111100
' f'(3) : 111101
' b'(3) : 1111100
' l'(1) : 1111101
' o'(1) : 11111100
' q'(1) : 11111101
' U'(1) : 11111110
' x'(1) : 11111111
user@nuked-PC:~/Documents/Study/courseWork$
```



# Оценка сложности

Кодирование файла:

- Подсчёт символов:  $O(n)$
- Построение дерева:  $O(c)$
- Создание шапки для закодированного файла:  $O(c)$
- Запись закодированных данных в файл:  $O(n)$

Раскодирование файла:

- Чтение шапки:  $O(c)$
- Создание таблицы кодировки:  $O(c)$
- Раскодирование:  $O(n)$
- Запись раскодированных данных в файл:  $O(n)$

Итоговая сложность — линейная  $O(n)$

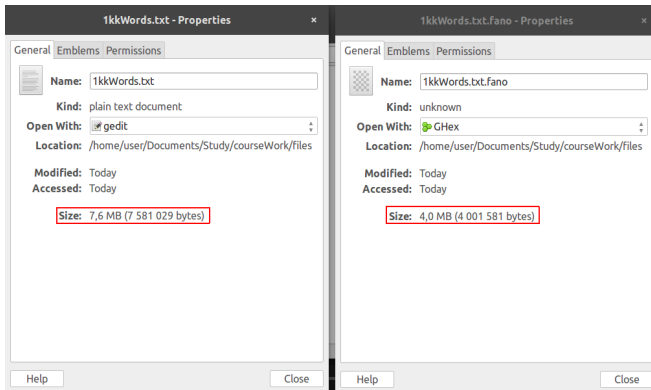
# Исходный код

Репозиторий на github

<https://github.com/nukefluke/fano-algorithm>

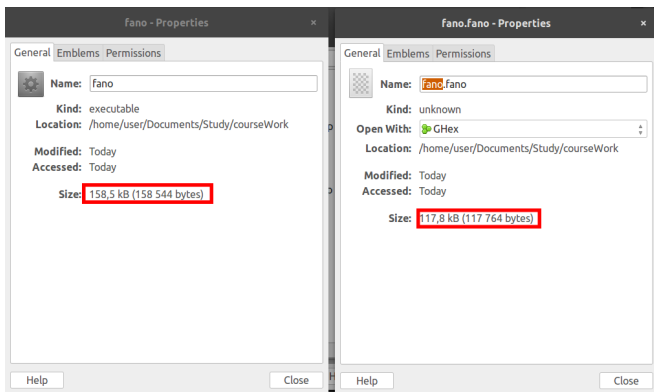
# Текстовый файл

При сжатии текстового файла, содержащего один миллион слов (7,6 Мб), получился файл размером 4 Мб. Эффективность сжатия - 47%.



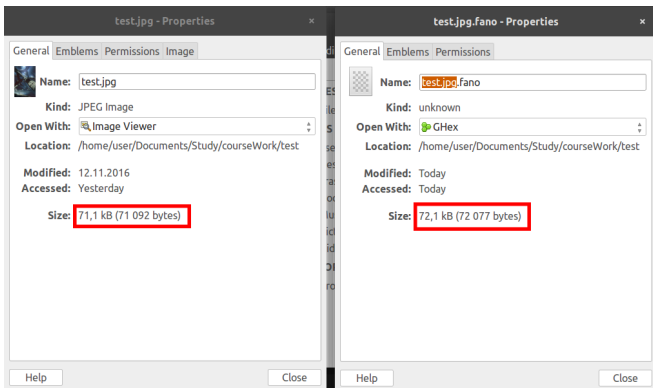
# Программа

При сжатии программой самой себя (158.5 Кб), получился файл размером 117.8 Кб. Эффективность сжатия - 25%.



# Пиксельный рисунок

При сжатии рисунка (71,7 Кб), получился файл размером 72,1 Кб.  
Здесь сжатие показало отрицательную эффективность.



## Итоги

- Алгоритм Шеннона — Фано вероятностный, использует коды переменной длины
- Для построения кодов используется бинарное дерево
- Программа работает быстро — линейная сложность
- Эффективность сжатия может быть отрицательной

Спасибо за внимание!