Санкт-Петербургский политехнический университет Петра Великого

Физико-механический институт
Кафедра прикладной математики и информатики

Отчёт по лабораторной работе №1
По дисциплине "Дискретная математика"
Тема : «Алгоритм Фано»

Выполнил студент гр. 5030102/30401
Кулиев Х. Т.

Санкт-Петербург

2025 г.

1 Цель и задачи работы

Целью данной лабораторной работы является реализация алгоритма Фано — метода неравномерного префиксного кодирования символов на основе их вероятностей появления в тексте.

В рамках работы требуется:

реализовать процедуры кодирования и декодирования текста с использованием алгоритма Фано;

обеспечить вывод промежуточных кодов на экран как при кодировании, так и при декодировании для контроля корректности работы алгоритма;

провести сравнительный анализ эффективности полученного кодирования с равномерным кодированием (ASCII) на текстах разной длины;

сделать выводы об эффективности алгоритма Фано в зависимости от статистических свойств входного текста.

2 Описание алгоритма Фано

Алгоритм Фано — это метод построения префиксного кода, основанный на рекурсивном разбиении множества символов на две подгруппы с максимально близкими суммарными вероятностями. Цель разбиения — приблизить среднюю длину кода к энтропии источника, что обеспечивает высокую эффективность сжатия.

2.1 Основные шаги алгоритма

1. **Подсчёт частот и вероятностей**: для каждого символа во входном тексте вычисляется частота появления и вероятность $p_i = \frac{n_i}{N}$, где n_i — количество вхождений символа, N — общая длина текста.

2. **Сортировка**: все символы сортируются по убыванию вероятности.

3. **Рекурсивное разбиение**: отсортированный список символов разбивается на две части так, чтобы разность между суммами вероятностей в левой и правой частях была минимальной. Это делается функцией `_med`.

4. **Присвоение битов**: символам первой (левой) части добавляется бит «0» к текущему коду, символам второй (правой) — бит «1».

5. **Рекурсия**: процедура повторяется для каждой из двух частей до тех пор, пока в подсписке не останется один символ. В этот момент формирование кода для этого символа завершено.

2.2 Особенности реализации

В данной реализации:

Используется класс `FanoEncoder`, инкапсулирующий логику кодирования и декодирования.

Построение кодов выполняется в методе `_build_codes`, который вызывает рекурсивную функцию `_fano_recursive`.

Декодирование осуществляется с помощью обратного словаря «код → символ» и последовательного чтения битов до совпадения с одним из кодов.

Все этапы сопровождаются подробным выводом на экран (если включён режим `verbose`), что позволяет отслеживать корректность работы.

Алгоритм гарантирует получение **префиксного кода**, то есть ни один код не является началом другого, что обеспечивает однозначное декодирование без разделителей.

3 Демонстрация работы алгоритма

Алгоритм был протестирован на трёх текстах разной длины: коротком (13 символов), среднем (245 символов) и длинном (5270 символов). Ниже приведены результаты.

3.1 Короткий текст: `"Hello, World!"` (13 символов)

Исходный текст: `"Hello, World!"`

Вероятности символов (отсортированы по убыванию):

Символ	Вероятность	Частота
l	0.230769	3
o	0.153846	2
H, e, „ ’ ’, W, r, d, !	0.076923	1

Построенные коды Фано:

Символ	Код	Длина	$p \cdot l$
l	00	2	0.4615
o	010	3	0.4615
H	011	3	0.2308
r	110	3	0.2308
e	1000	4	0.3077
,	1001	4	0.3077
’ ’	1010	4	0.3077
W	1011	4	0.3077
d	1110	4	0.3077
!	1111	4	0.3077

Средняя длина кода: **3.2308 бит/символ**.

Кодирование первых символов:

H → 011, e → 1000, l → 00, l → 00, o → 010, , → 1001, ’ ’ → 1010, …

Закодированная строка (42 бита): `011100000000101001101010110101100011101111`

Декодирование полностью восстановил исходный текст.

Сравнение с ASCII:

ASCII: 104 бита (13 байт)

Фано: 42 бита (5.25 байт)

Теоретический минимум: 41 бит

Коэффициент сжатия: **59.62%**

Эффективность относительно энтропии: **98.45%**

3.2 Средний текст (245 символов)

Текст содержит повторяющиеся фразы на русском языке. Алгоритм корректно построил префиксные коды для 22 уникальных символов и успешно выполнил кодирование и декодирование.

Результаты:

Длина текста: 245 символов

Закодировано: **1045 бит** (вместо 1960 бит в ASCII)

Энтропия: 4.2262 бит/символ

Средняя длина кода: 4.2653 бит/символ

Коэффициент сжатия: **46.68%**

Эффективность относительно энтропии: **99.08%**

3.3 Длинный текст (5270 символов)

Текст представляет собой многократно повторённый технический фрагмент с использованием кириллицы, знаков препинания и пробелов (43 уникальных символа). Кодирование и декодирование прошли без ошибок.

Результаты:

Длина текста: 5270 символов

Закодировано: **23080 бит** (вместо 42160 бит в ASCII)

Энтропия: 4.3543 бит/символ

Средняя длина кода: 4.3795 бит/символ

Коэффициент сжатия: **45.26%**

Эффективность относительно энтропии: **99.42%**

4 Анализ устойчивости и возможных сбоев

Реализованная программа демонстрирует высокую устойчивость при выполнении в рамках поставленной задачи. Рассмотрим условия, при которых возможны сбои, и случаи гарантированно корректной работы.

4.1 Случаи корректной работы

Программа корректно обрабатывает:

тексты любой длины (включая односимвольные);

тексты с произвольным набором символов: латиница, кириллица, цифры, знаки препинания, пробелы, символы перевода строки (`\n`) и табуляции (`\t`);

тексты с высокой и низкой неравномерностью распределения символов.

Во всех проведённых тестах (включая интерактивный ввод) проверка `decoded == original` завершилась успешно, что подтверждает корректность как кодирования, так и декодирования.

4.2 Потенциальные сбои

Сбои могут возникнуть только при нарушении условий эксплуатации:

Декодирование «чужой» битовой строки. Если на вход методу `decode()` подать битовую последовательность, не полученную с помощью того же экземпляра `FanoEncoder` (например, случайную строку или результат другого кодировщика), алгоритм может:

- зависнуть (если ни один префикс не совпадает с известными кодами);
- выдать неполный или искажённый текст (если частичные совпадения приведут к ложному распознаванию). Однако в рамках лабораторной работы декодирование всегда применяется к корректно закодированной строке, поэтому такие ситуации исключены.
- **Пустой входной текст**. В текущей реализации пустая строка обрабатывается без ошибок: список вероятностей оказывается пустым, кодирование возвращает пустую битовую строку, декодирование — пустой текст. Таким образом, даже этот крайний случай обработан корректно.

5 Формат входных и выходных данных

В рамках реализации алгоритма Фано входные и выходные данные представлены в следующем виде:

5.1 Входные данные

Программа поддерживает два способа задания входного текста:

1. **Автоматическое тестирование**: три заранее определённых текста (короткий, средний, длинный) передаются программно.
2. **Интерактивный ввод**: после завершения автоматических тестов пользователю выводится приглашение:

   ```
   Введите свой текст для кодирования (или оставьте пустым для выхода):
   ```

   и ожидается ввод строки через стандартный поток ввода (`stdin`).

В обоих случаях входными данными является **строка текста произвольной длины**, содержащая:

буквы латинского и кириллического алфавитов (в любом регистре);

цифры, знаки препинания, пробелы;

специальные символы: символ перевода строки (`\n`), табуляции (`\t`) и другие допустимые символы кодировки UTF-8. Данные передаются в оперативной памяти как объект типа `str` в языке Python.

5.2 Выходные данные

Программа формирует следующие выходные данные:

Словарь кодов: соответствие «символ → битовая строка», хранится в атрибуте `codes` как `Dict[str, str]`.

Закодированная строка: последовательность битов, представленная строкой из символов `'0'` и `'1'`.

Декодированный текст: строка, идентичная входной (при корректном кодировании).

Статистическая информация: энтропия источника, средняя длина кода, коэффициент сжатия, эффективность относительно энтропии — выводится на экран в человекочитаемом виде. Все выходные данные выводятся через стандартный поток вывода (`stdout`). Файловый ввод/вывод не используется.

6 Выполнение дополнительных требований по заданию

В соответствии с вариантом задания была реализована возможность вывода кодов на экран как при кодировании, так и при декодировании, что позволило наглядно отслеживать корректность работы алгоритма на каждом этапе.

Кроме того, был проведён сравнительный анализ эффективности алгоритма Фано с равномерным кодированием (ASCII) на трёх текстах разной длины. Результаты представлены в таблице 1.

Таблица 1: Сравнение эффективности кодирования Фано и ASCII

Текст	Длина	ASCII (бит)	Фано (бит)	Сжатие
Короткий	13	104	42	59.62%
Средний	245	1960	1045	46.68%
Длинный	5270	42160	23080	45.26%

Выводы об эффективности:

Алгоритм Фано обеспечивает значительное сжатие по сравнению с ASCII: от **45% до 60%** в зависимости от статистики текста.

Наибольший коэффициент сжатия достигается на коротких текстах с высокой неравномерностью распределения символов (например, повторяющаяся буква «l» в «Hello, World!»).

На длинных текстах с естественным языковым распределением сжатие стабилизируется около **45%**, что соответствует теоретическим ожиданиям для русскоязычных текстов.

Средняя длина кода Фано отличается от энтропии менее чем на **1%**, что свидетельствует о высокой близости к оптимальному кодированию.

Во всех случаях декодирование полностью восстанавливало исходный текст, что подтверждает корректность реализации.