Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и информатики

Высшая школа прикладной математики и вычислительной физики

Лабораторная работа №1 по дисциплине

Дискретная математика

Тема: «Кодирование»

Вариант 1 – Алгоритм Фано

Выполнил студент гр. 5030102/20202 Соколов А.Н.

Руководитель Новиков Ф. А.

«___» _____ 202__ г.

Санкт-Петербург

2024

1. Формулировка задачи и ее формализация

Формулировка задачи

1. Необходимо разработать консольное приложение, реализующее функции кодирования и декодирования файлов с помощью алгоритма Фано.
2. Поддержать возможность вывода кодов на экран при кодировании и декодировании информации для отслеживания правильности работы алгоритма.
3. Сравнить результаты кодирования с равномерным кодированием (ASCII) на текстах разной длины и сделать выводы об эффективности алгоритма.

2. Использованные технологии

Язык программирования

- C++ 23

Система сборки

- CMake 3.27
- Ninja 1.12.1

Исходные файлы программы:

https://github.com/azya0/dm2025/tree/master/lab1

3. Описание алгоритма кодировки

Алгоритм кодирования Фано — это один из методов сжатия данных, основанный на принципах кодирования с переменной длиной. Он позволяет эффективно представлять символы с использованием бинарных кодов, основываясь на частоте встречи определенного символа в строке.

Основные понятия

Энтропия информации: количество информации, необходимое для сжатия данных. Чем выше вероятность появления символа, тем меньше бит необходимо для его кодирования.
Вероятностное распределение: вероятности появления каждого символа в сообщении, которые мы собираемся закодировать.

Шаги алгоритма кодирования Фано

Алгоритм можно разбить на несколько основных этапов:

Шаг 1: Подсчет частот символов
Первый шаг заключается в анализе данных и подсчета количества вхождений каждого уникального символа. Это поможет нам определить вероятности их появления. Например, если в строке "АБААБ" символ "А" встречается 4 раза, а "Б" 2 раза, то вероятности будут следующими:

$P(A) = \frac{4}{6} = \frac{2}{3}$
$P(B) = \frac{2}{6} = \frac{1}{3}$

Шаг 2: Сортировка символов
После вычисления вероятностей все символы сортируются по убыванию их частоты. Это необходимо для дальнейшего распределения кодов.

Шаг 3: Распределение кодов
На основе подсчитанных вероятностей мы формируем и присваиваем код каждому символу. Вычисляя медиану относительно вероятности, мы делим массив символов на 2 подмассива, присваивая левой части подмассива код “0”, а правой код “1”, пока в каждом из подмассивов не останется по 1 символу. Сумма кодов на пути к “подмассиву-символу” и будет искомым бинарным кодом для символа.

Шаг 4: Запись полученных кодов
После завершения всех предыдущих шагов, у нас будет набор бинарных кодов для каждого символа, по которым мы и будем кодировать текст.

Пример:

Предположим, у нас есть строка "AAABBC":

Подсчет частот:
A: 3
B: 2
C: 1
Сортируем по частоте: A (3), B (2), C (1).
Разделяем группы. Сначала включим A в первую группу (группа 1) и B в другую (группа 2). Получится: группа 1 (A), группа 2 (B, C).
Присваиваем коды:
Группа 1: 0
Группа 2: 1
Рекурсивно работает с группой 2. В ней символ B получает 10, а C — 11.
Итоговая кодировка будет:

A: 0
B: 10
C: 11

Заключение

Алгоритм Фано — это мощный и эффективный метод кодирования, который позволяет сжимать данные, основываясь на вероятностях появления символов. Его реализация требует четкого понимания вероятностного анализа и умения работать с группами символов.

4. Практическая реализация

Кодирование

Отличием реализованного алгоритма, от базового алгоритма фано, станет возможность декодирования любого файла без информации о его кодировки. Для этого в начале каждого закодированного файла, записывается следующее:

- Число символов в тексте
- Число уникальных символов в тексте
- Далее для каждого уникального символа выделяется:
    - 8 бит для уникального символа
    - 8 бит для указания размера его кодировки
    - Кодировка уникального символа

После чего следуют только коды символов

5. Область применения

Запрос к программе

- Неверно переданный аргумент:
    - Пользователь ввел команду, не предусмотренную написанной программой
- Недостаточное количество аргументов:
    - Пользователь нарушил синтаксис написания программой, заполнив неверное количество аргументов
- Повторное указание опциональных аргументов
    - Пользователь ввел несколько необязательных аргументов

Кодирование

Программа выдаст уведомление об ошибке при:

- Отсутствии указанного файла для счета информации
- Ошибке при создании файла для вывода закодированной информации
- Количество определенного символа в тексте программы превышает $3,4 * 10^{38}$. Переполнение типа данных *float.*

Декодирование

Программа выдаст уведомление об ошибке при:

- Отсутствии указанного файла для счета информации
- Ошибке при создании файла для вывода расшифрованной
- Ошибке при счете информации из закодированного файла:
    - Файл кодировался не при помощи этой программы
    - Файл был поврежден до процесса расшифровки

Во всех остальных случаях программа будет работать корректно

6. Формат входных и выходных данных

Консольное приложение запрашивает названия файлов под считывание информации из них, а также название файла под запись. Есть возможность добавление флага “-s” для вывода кодов на экран при кодировании и декодировании информации для отслеживания правильности работы алгоритма.

Примеры работы программы:

Пусть скомпилированный файл будет называться “lab1.exe”

Пример команды для кодирования файла “orwell.txt”, находящегося в верхней директории. Запись результата будет произведена в файл “coded-orwell.bin”, находящегося в той же директории, что и исходный

```
.\lab1.exe code -i ..\orwell.txt -o ..\coded-orwell.bin
```

Пример команды для расшифровки файла “..\coded-orwell.bin”, находящегося в верхней директории. Запись результата будет произведена в файл “decoded-orwell.txt”, находящегося в той же директории, что и исходный

```
.\lab1.exe decode -i ..\coded-orwell.bin -o ..\decoded-orwell.txt
```

Для вывода кодов в процессе можно добавить флаг “-s” сразу после указания команды, либо в самом конце

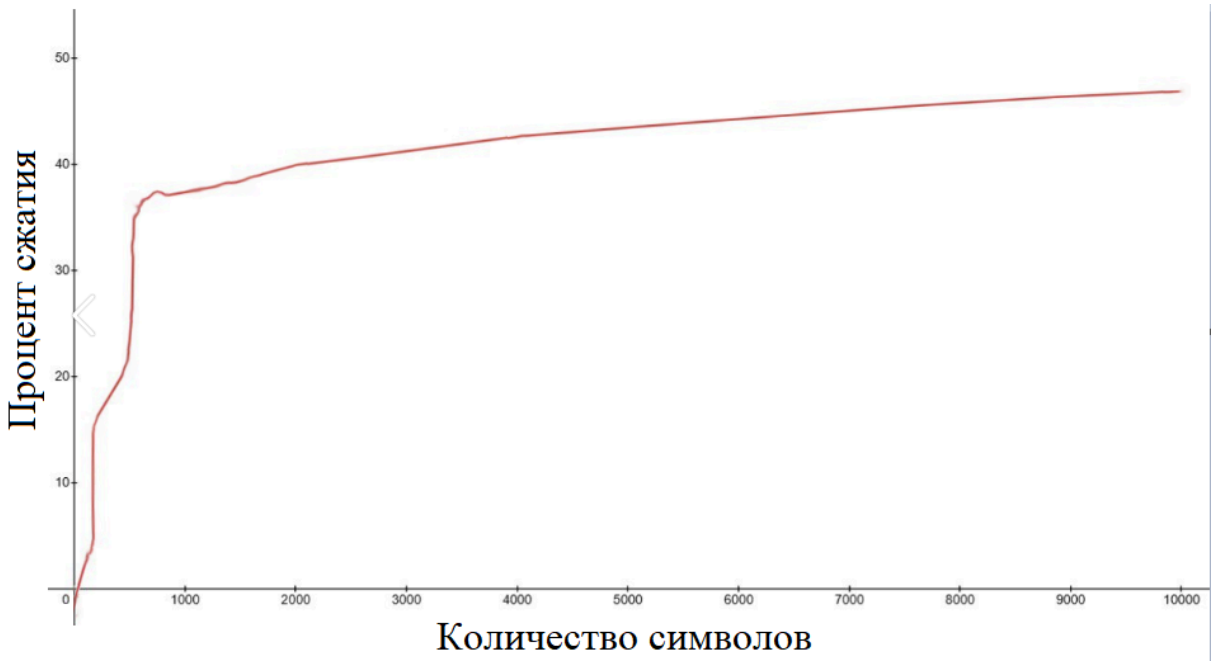
```
.\lab1.exe decode -s -i ..\coded-orwell.bin -o ..\decoded-orwell.txt
.\lab1.exe decode -i ..\coded-orwell.bin -o ..\decoded-orwell.txt -s
```

Для полного списка команд можно воспользоваться флагами “-h” или “--help”

```
.\lab1.exe -h
.\lab1.exe --help
```

7. Сравнение результатов кодирования с равномерным кодированием

Сравним результаты сжатия на произвольно выбранном файле. Построим график, где ось абсцисс - количество символов, считанных из этого файла, а ось ординат - процент сжатия. Выбранный шаг в диапазоне [0; 100] будет 100, а на (100; 10000] 500.



Из графика видно, что степень сжатия резко возрастает в диапазоне [400;600], после чего монотонно возрастает.
Результаты сжатия при малом количестве символов легко объяснить дополнительной информации, которая добавляется для однозначного декодирования файла без сохраненных данных при его кодировке

8. Вывод

В рамках данной лабораторной работы был реализован алгоритм неравномерного кодирования Фано, позволяющий в среднем сжимать огромные файлы почти вдвое.