МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
КАФЕДРА 25

ПРЕПОДАВАТЕЛЬ

Доцент к.т.н.	27.12.2024	Е. М. Линский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ ПО КУРСОВОЙ РАБОТЕ

”АЛГОРИТМ ФАНО”

по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

Студент гр. №	2352	27.12.2024	Т. А. Потапов
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

Содержание

1 ПОСТАНОВКА ЗАДАЧИ

Задачей данной курсовой работы является разработка программы для кодирования и декодирования файлов с использованием алгоритма Шеннона-Фано. Программа должна предоставлять возможность архивировать одиночные текстовые файлы, сжимая их размер, а затем восстанавливать их до исходного состояния.

Программа реализует алгоритм Фано, который основывается на разбиении символов исходного текста на группы, исходя из их частот, и присвоении им уникальных префиксных кодов.

Свойства алгоритма Фано:

- Кодирование должно быть префиксным, чтобы обеспечить однозначность декодирования.
- Часто встречающиеся символы получают более короткие коды, реже встречающиеся — более длинные.
- Программа должна эффективно обрабатывать текст с произвольным количеством символов.

Пример задачи: Дан текстовый файл с содержимым hello world. Программа должна сжать его, создав закодированный файл и сопутствующий словарь кодов. После разархивации из сжатого файла должен быть восстановлен исходный текст.

Литература: М.Н. Аршинов, Л.Е. Садовский, Коды и математика, Издательство: «Наука», 1996.

2 ОПИСАНИЕ АЛГОРИТМА

2.1 Основные идеи алгоритма

Алгоритм Шеннона-Фано основан на следующих концепциях:

1. Символы сортируются по убыванию частоты их появления.
2. Множество символов разбивается на две группы с примерно равной суммарной частотой.
3. К первой группе добавляется бит 0, ко второй — 1.
4. Процесс повторяется рекурсивно для каждой группы, пока все символы не будут закодированы.

2.2 Подробное описание алгоритма

1. Сбор статистики: анализируется входной текст, и подсчитывается частота каждого символа.
2. Сортировка символов: символы сортируются по частоте в порядке убывания.
3. Построение кодов: применяется рекурсивный метод, описанный выше.

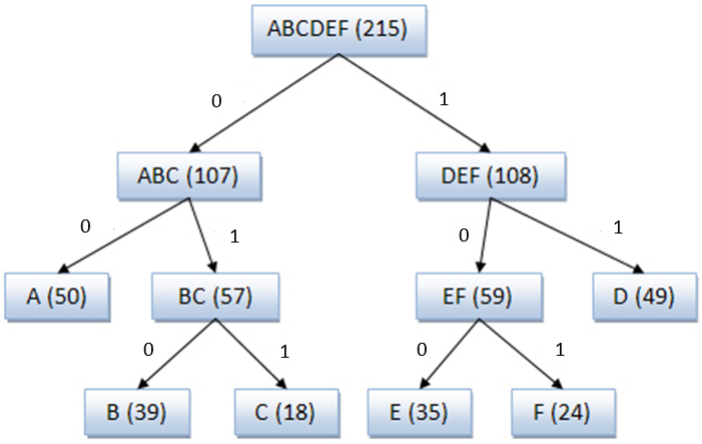

Рис. 1 – Пример построения кодов

4. Создание словаря: генерируется отображение символов в их коды.

5. Кодирование текста: каждый символ текста заменяется соответствующим кодом.

6. Декодирование: используя словарь, сжатый текст преобразуется обратно в исходный.

2.3 Пример выполнения алгоритма

Шаги алгоритма для текста abacabad:

1. Частоты символов: $a : 4, b : 2, c : 1, d : 1$.

2. Сортировка: $a(4), b(2), c(1), d(1)$.

3. Разбиение:

   - Первая группа: $a(4) \rightarrow$ код 0.
   - Вторая группа: $b(2), c(1), d(1) \rightarrow$ код 1.

4. Рекурсивное разбиение второй группы:

   - $b(2) \rightarrow$ код 10.
   - $c(1), d(1) \rightarrow$ код 11.

5. Итоговые коды: $a : 0, b : 10, c : 110, d : 111$.

6. Кодирование текста: abacabad $\rightarrow$ 010011010111.

2.4 Псевдокод алгоритма

Algorithm 1 Анализ текста для подсчета частот символов

```
function analyzeText(текст)
    Создать словарь частот символов
    for каждый символ в тексте do
        if символ пробел then
            Увеличить счётчик частоты для "\x20"
        else if символ новая строка then
            Увеличить счётчик частоты для "\n"
        else
            Увеличить счётчик частоты для строки, содержащей этот символ
        end if
    end for
    for каждая пара символ-частота в словаре do
        Добавить символ с его частотой в список символов
    end for
    Отсортировать список символов по убыванию частот
    Вызвать buildCodes(0, размер списка символов - 1)
    Построить словари для кодировки и декодировки символов
end function
```

Algorithm 2 Рекурсивное построение кодов для символов

```
1: function buildCodes(начало, конец)
2:     if начало >= конец then
3:         Завершить выполнение функции
4:     end if
5:     totalFrequency = 0
6:     for i от начало до конец do
7:         totalFrequency += частота символа[i]
8:     end for
9:     halfFrequency = 0
10:    splitIndex = начало
11:    for i от начало до конец do
12:        halfFrequency += частота символа[i]
13:        if halfFrequency >= totalFrequency / 2 then
14:            splitIndex = i
15:            Прервать цикл
16:        end if
17:    end for
18:    for i от начало до splitIndex do
19:        Добавить "0"в код символа[i]
20:    end for
21:    for i от splitIndex + 1 до конец do
22:        Добавить "1"в код символа[i]
23:    end for
24:    buildCodes(начало, splitIndex)
25:    buildCodes(splitIndex + 1, конец)
26: end function
```

2.5 Анализ сложности алгоритма

1. Подсчет частот символов: Сложность: $O(n)$
2. Перенос частот в список: Сложность: $O(k)$
3. Сортировка списка: Сложность: $O(k \log k)$
4. Построение кодов (рекурсивно): Сложность: $O(k \log k)$

   Итоговая сложность: $O(n + k \log k)$

- Если $n \gg k$, то сложность $O(n)$, так как обработка текста доминирует.
- Если $k \gg \log k$, то сложность $O(k \log k)$.

3 ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

1. Запуск программы:

./ShanonFano <режим> <входной файл> <выходной файл>

- <режим>: encode для сжатия или decode для разархивации.
- <входной файл>: имя файла для обработки.
- <выходной файл>: имя файла для сохранения результата.

2. Формат входного файла:

- Режим encode: Текстовый файл с любым содержимым. Поддерживаются пробелы и переносы строк.
- Режим decode: Бинарный файл с закодированными данными.

3. Формат выходного файла:

- Режим encode: Бинарный файл с закодированными данными, текстовый файл со словарем символов.
- Режим decode: Текстовый файл с восстановленным содержимым.

4 ТЕСТОВЫЕ ПРИМЕРЫ

1. Тест 1:
    - Ссылка на текст: https://www.gutenberg.org/cache/epub/12299/pg12299.txt
    - Вес текста до архивации: 385 KB
    - Вес текста после архивации: 221 KB
2. Тест 2:
    - Ссылка на текст: https://www.gutenberg.org/cache/epub/12299/pg12299.txt
    - Вес текста до архивации: 280 KB
    - Вес текста после архивации: 169 KB
3. Тест 3:
    - Ссылка на текст: https://www.gutenberg.org/cache/epub/80/pg80.txt
    - Вес текста до архивации: 620 KB
    - Вес текста после архивации: 371 KB

5 СПИСОК ЛИТЕРАТУРЫ

1. М. Н. Аршинов, Л. Е. Садовский, Коды и математика (рассказы о кодировании).- М.: Наука, Главная редакция физико-математической литературы, 1983. - 144 с.

2. Алгоритм Шеннона-Фано. // Алгоритмы: электронные текстовые данные, URL: https://habr.com/ru/articles/137766/, Год публикации: 2012. Режим доступа: открытый. Дата обращения к источнику: 18.12.2024