**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра** **ВТ**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: «Алгоритмы сжатия без потерь»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0322 |  | Руссу В.А. |
| Преподаватель |  | Пестерев Д. О. |

Санкт-Петербург

2023

**ЗАДАНИЕ**

**на курсовую работу**

|  |
| --- |
| Руссу Василий Андреевич |
| Группа 0322 |
| Тема работы: Алгоритмы сжатия без потерь. |
| Исходные данные: Реализовать следующие алгоритмы сжатия символьных данных:   1. Алгоритм Хаффмана (HA) 2. Кодирование длин серий (RLE) 3. Алгоритм Лемпеля-Зива (LZ78) 4. Преобразование Барроуза-Уиллера (BWT) 5. MTF 6. Арифметическое кодирование (AC) |

# Описание программы

Данная программа получает текст в виде символьных данных из файла (enwik8tm), далее кодирует данный текст при помощи алгоритмов и сохраняет их в файлы названные в честь данных алгоритмов.

Теоретические сведения

Алгоритм Хаффмана работает в несколько этапов:

Сначала алгоритм анализирует исходный текст и определяет, какие символы встречаются в нем и как часто они встречаются.

Затем он строит дерево Хаффмана, используя эти частоты. Для этого он создает листья для каждого символа и помещает их в приоритетную очередь, отсортированную по частотам символов. Затем он объединяет два узла с наименьшими частотами в новый узел и добавляет его обратно в очередь. Этот процесс продолжается до тех пор, пока все узлы не будут объединены в один корень дерева.

Далее каждому символу назначается уникальный двоичный код, который формируется путем прохождения от корня дерева к листьям. Каждый левый переход в дереве соответствует биту 0, а каждый правый переход - биту 1.

Затем исходный текст перекодируется с использованием назначенных кодов символов. Это позволяет кодировать часто встречающиеся символы короткими кодами, что приводит к сжатию данных.

Декодирование происходит обратным обходом дерева Хаффмана с использованием закодированных данных. Каждый бит считывается последовательно, и для каждого прочитанного бита происходит переход влево или вправо по дереву, пока не будет достигнут лист, который соответствует символу.

В результате применения алгоритма Хаффмана данные могут быть сжаты до значительно меньшего размера, чем в исходном виде, что позволяет экономить место при их хранении или передаче.

Кодирование длин серий (RLE)

Алгоритм сжатия RLE (Run-Length Encoding) - это простой метод сжатия данных, который основывается на повторяющихся последовательностях символов. Он применяется для сжатия текстовых и графических файлов.

Алгоритм RLE работает следующим образом:

Последовательные повторяющиеся символы заменяются на число повторов и сам символ. Например, последовательность "AAAA" может быть заменена на "4A".

Хотя алгоритм RLE прост в реализации и быстр в работе, он не всегда позволяет добиться высокой степени сжатия, особенно для файлов с небольшим количеством повторяющихся символов.

Преобразование Барроуза — Уилера (Burrows-Wheeler transform, BWT)

 Это алгоритм, используемый в техниках сжатия данных для преобразования исходных данных.

Преобразование выполняется в три этапа. На первом этапе составляется таблица всех циклических сдвигов входной строки. На втором этапе производится лексикографическая (в алфавитном порядке) сортировка строк таблицы. На третьем этапе в качестве выходной строки выбирается последний столбец таблицы преобразования.

MTF (Move-To-Front) - это алгоритм сжатия данных, который использует простую идею перемещения символов в начало списка на каждый встреченный символ. Каждый символ представляется целым числом от 0 до 255, соответствующим его ASCII коду.

Процесс кодирования MTF заключается в следующих шагах:

1. Создается список символов от 0 до 255.

2. Для каждого символа в исходном тексте ищется его индекс в списке.

3. Индекс символа записывается в выходной поток.

4. Символ перемещается в начало списка.

Описание алгоритма LZ78

LZ78 - это алгоритм сжатия данных, который использует словарь для замены повторяющихся последовательностей символов на ссылки на уже существующие подстроки. Процесс кодирования LZ78 заключается в том, что при каждом встреченном символе строится новая запись в словаре, содержащая ссылку на предыдущую запись и добавленный символ. При декодировании происходит обратное преобразование - по ссылкам на уже существующие записи восстанавливаются исходные последовательности символов.

Описание арифметического кодирования

Арифметическое кодирование - это метод сжатия данных, который основан на представлении исходной последовательности символов в виде дробного числа в интервале [0,1]. Каждому символу присваивается вероятность его появления в последовательности, и на основе этих вероятностей вычисляется дробное число, которое затем кодируется в битовую последовательность. При декодировании происходит обратное преобразование - битовая последовательность превращается в дробное число, которое разбивается на символы с помощью вероятностей исходной последовательности. Арифметическое кодирование обеспечивает более эффективное сжатие данных, чем многие другие методы сжатия.

# Пример работы программы

На рисунке 1 представлен пример работы программы.

Для начала программы выводит текст “Got the text” – это означает, что данные из файлы были считаны корректно.   
Далее выводится название каждого алгоритма, время и вывод о том, готово ли сжатие или нет.

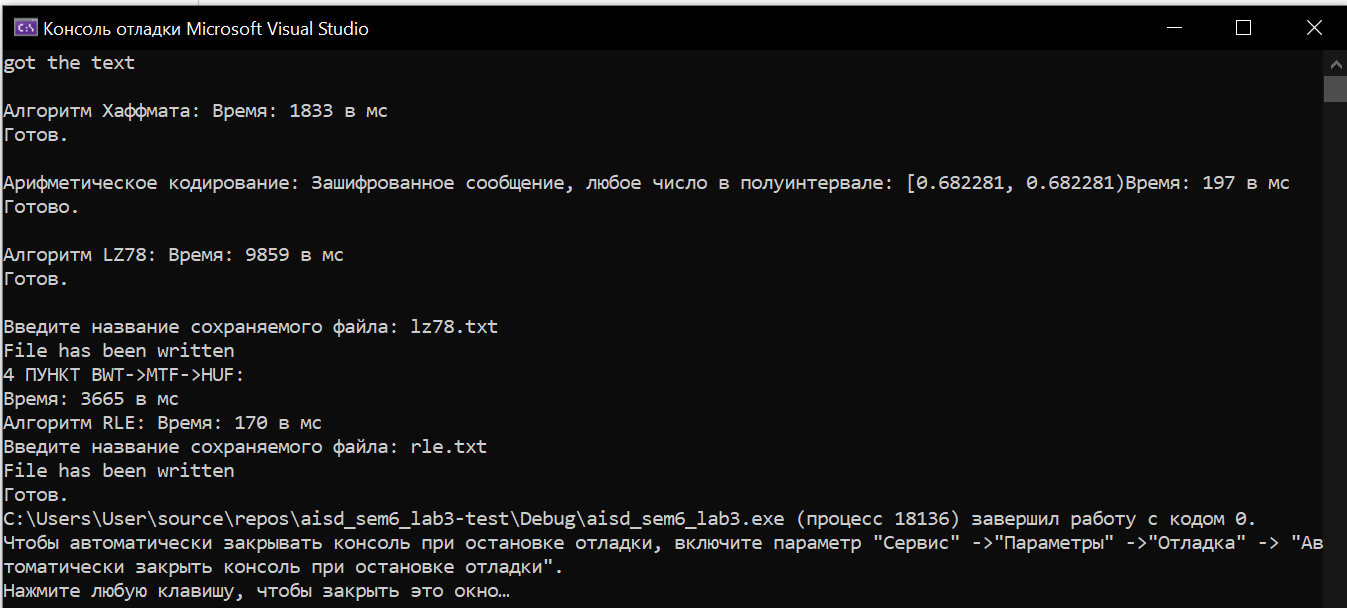


Рисунок 1

**Результаты работы программы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Изначальный объем | Итоговый объем | Процент сжатия | Время сжатия, мс |
| HA | 107 778 байт | 66 616 байт | 61 % | 1761 |
| LZ78 |  | 76 994 байт | 72 % | 9882 |
| BWT->MTF->HUF |  | 68 487 байт | 64 % | 2669 |
| RLE |  | 64 470 байт | 59.8 % | 164 |

# Заключение

В ходе выполнения курсовой работы были приобретены навыки кодирования информации при помощи следующих алгоритмов: Хаффмана, LZ78, Преобразование Барроуза (BWT), MTF и RLE.

Ссылка на репозиторий https://github.com/RVA-t/aisd\_sem6\_lab3-test.git