МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Алгоритмы кодирования

Студент гр. 7383 _____ Власов Р.И

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Власов Р.А.

Группа 7383

Тема работы: Алгоритмы кодирования

Содержание пояснительной записки:

- Содержание
- Введение
- Описание функций для кодирования и декодирования текста
- Примеры работы программы
- Исследование реализованных алгоритмов
- Заключение
- Список использованных источников

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Власов Р.А.

Преподаватель _____ Размочаева Н.В.

АННОТАЦИЯ

В ходе работы была реализована программа на языке программирования C++ с использованием фреймворка Qt, обеспечивающая кодирование и декодирование текстовых файлов с использованием алгоритмов Хаффмана и Шеннона-Фано. Было проведено исследование эффективности алгоритмов в нескольких категориях. Был реализован графический пользовательский интерфейс для удобной работы с программой.

SUMMARY

During the work the program was implemented in C++ using the Qt framework. The program provides encoding and decoding text files using Huffman or Shennon-Fano algorithms. Research of algorithms was conducted in several categories. GUI was made for more convenient interaction with the program.

СОДЕРЖАНИЕ

ЗАДАНИЕ

Общие сведения

- Формат файла TXT

Программа должна реализовывать следующий функционал:

- Кодирование текста с использованием алгоритмов:
  - Хаффмана (статический);
  - Хаффмана (динамический);
  - Шеннона-Фано.
- Декодирование текста с использованием алгоритмов:
  - Хаффмана (статический);
  - Хаффмана (динамический);
  - Шеннона-Фано.
- Автоматическое исследование реализованных алгоритмов по следующим критериям.
  - Вес закодированного текста в зависимости от длины входного текста (текст генерируется автоматически);
  - Количество операций, выполняемых алгоритмом, в зависимости от длины входного текста (текст генерируется автоматически);
  - Вес закодированного текста в зависимости от размера алфавита во входном тексте фиксированной длины (текст генерируется автоматически).

ВВЕДЕНИЕ

Целью данной работы является создание GUI приложения на языке C++ с использованием фреймворка Qt для кодирования и декодирования текстовых файлов.

Для достижения поставленной цели требуется решить следующие задачи:

- Создание класса для построения кодового дерева;
- Реализация методов для кодирования и декодирования текста;
- Создание пользовательского GUI интерфейса для взаимодействия с программой;
- Реализация функции для автоматического исследования алгоритмов;
- Сборка и тестирование программы.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Алгоритмы используют коды переменной длины: часто встречающийся символ кодируется кодом меньшей длины, редко встречающийся — кодом большей длины.

Коды в статическом алгоритме Хаффмана и алгоритме Шеннона-Фано обладают свойством префиксности (то есть ни одно кодовое слово не является префиксом другого), что позволяет однозначно их декодировать.

Динамический алгоритм Хаффмана позволяет строить кодовую схему в поточном режиме (без предварительного сканирования данных), не имея никаких начальных знаний из исходного распределения, что позволяет за один проход сжать данные. Преимуществом этого способа является возможность кодировать на лету.

МЕТОДЫ И ФУНКЦИИ ДЛЯ КОДИРОВАНИЯ И ДЕКОДИРОВАНИЯ ТЕКСТА

1. **Класс CodeTree**

Класс CodeTree содержит набор методов для реализации удобного использования данного класса в качестве кодового дерева.

2. **Класс HuffmanStatic**

Класс HuffmanStatic содержит метод для построения кодового дерева с использованием предоставленных вероятностей символов или посчитанных автоматически, а также методы для кодирования и декодирования текста с помощью построенного кодового дерева.

3. **Класс ShennonFano**

Класс ShennonFano содержит метод для построения кодового дерева с использование предоставленных вероятностей символов или посчитанных автоматически, а также методы для кодирования и декодирования текста с использованием кодового дерева.

4. **Класс HuffmanDynamic**

Класс HuffmanDynamic содержит набор методов для построения кодового дерева и его изменения с учетом новой информации, а также методы для кодирования и декодирования текста с использованием имеющегося на данный момент дерева.

5. **Класс research_thread**

Класс research_thread содержит функцию автоматического исследования реализованных алгоритмов и построения графиков в отдельном потоке. Класс был реализован с целью предотвращения зависаний пользовательского интерфейса во время выполнения функции.

ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Программа собрана в операционной системе Ubuntu 18.04 с использованием компилятора g++ version 7.3.0 (Ubuntu 7.3.0-27ubuntu1~18.04). В других ОС и компиляторах тестирование не проводилось.

На рисунке 1 представлен пользовательский интерфейс программы. Окно содержит настройки алгоритма и алфавита для кодирования или декодирования, два текстовых поля для ввода, вывода или редактирования декодированного и закодированного текстов и кнопку, запускающие работу программы. Также присутствует возможность загрузить текст из файла или сохранить результат в файл. Запуск быстрого исследования алгоритмов вынесен в отдельную кнопку.

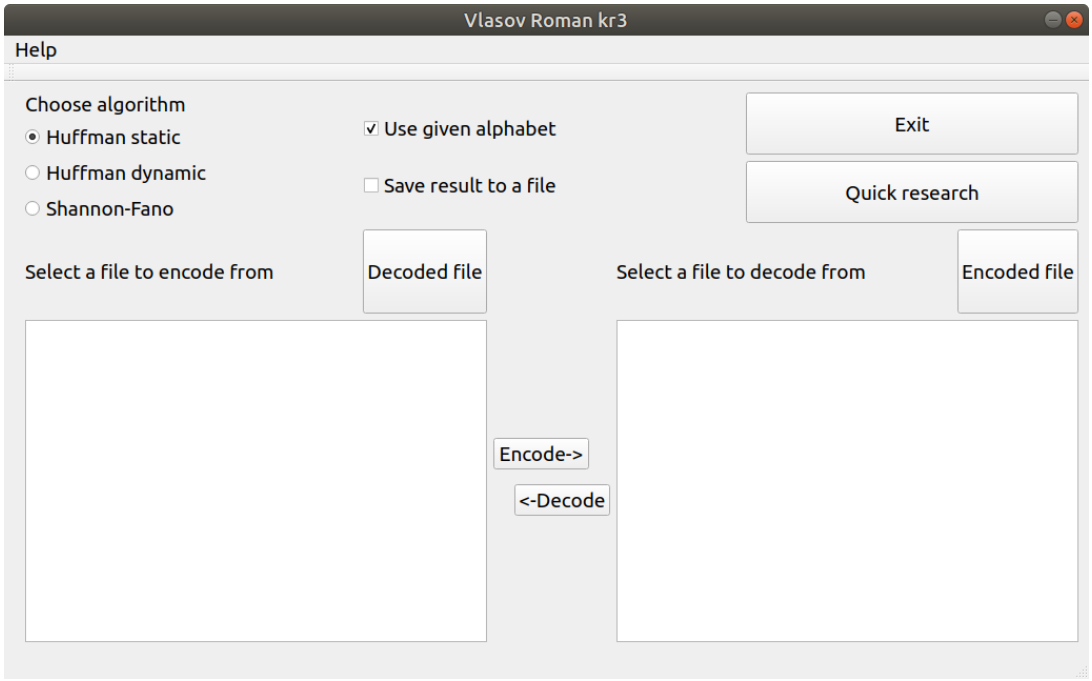

Рисунок 1 – Пользовательский интерфейс программы

Для удобства пользователя вывод в поле закодированного текста осуществляется посимвольно в десятичном виде (см. рис. 2).

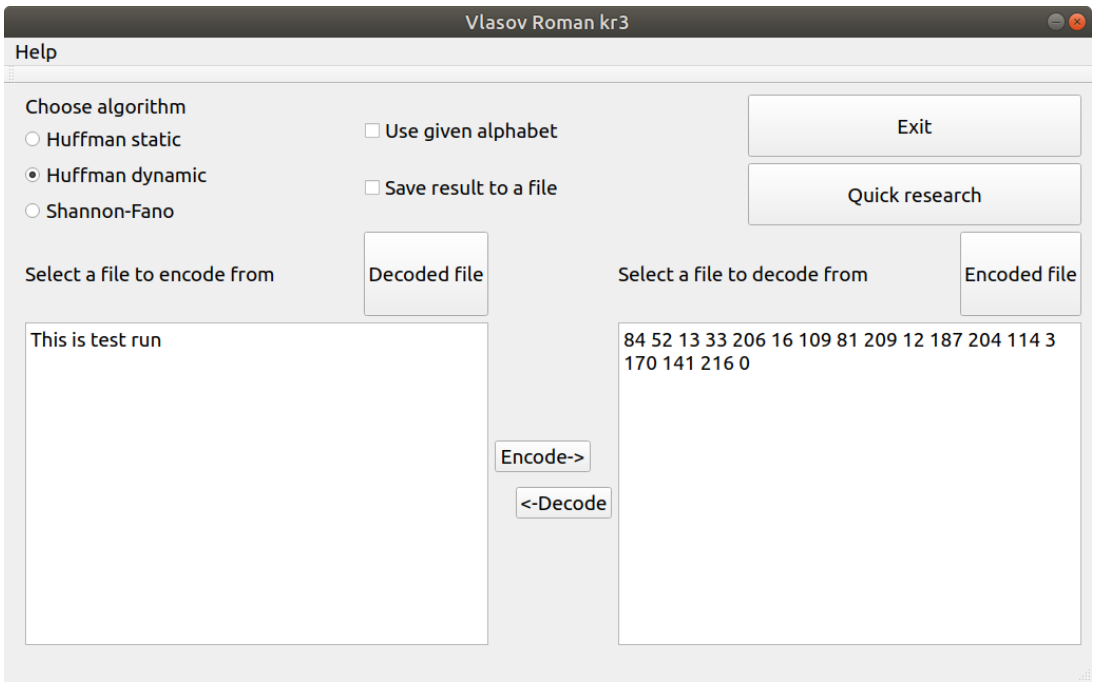

Рисунок 2 – Пример работы текстовых полей

ИССЛЕДОВАНИЕ

Для проведения исследования реализованных алгоритмов было принято сравнивать следующие критерии:

- Вес закодированного текста в зависимости от длины входного текста (текст генерируется автоматически);
- Количество операций, выполняемых алгоритмом, в зависимости от длины входного текста (текст генерируется автоматически);
- Вес закодированного текста в зависимости от размера алфавита во входном тексте фиксированной длины (текст генерируется автоматически).

В ходе исследования зависимости веса закодированного текста от длины входного текста и размера алфавита входного текста было выявлено, что динамический алгоритм Хаффмана проявляет себя наилучшим образом, а алгоритм Шеннона-Фано – наихудшим при любой длине входного текста и размере его алфавита. Однако динамический алгоритм Хаффмана производит наибольшее количество операций при аналогичной длине входного текста.

Результаты исследования зависимости веса закодированного текста от длины входного текста приведены в приложении А, зависимости колиечства операций от длины входного текста приведены в приложении Б, а зависимости веса закодированного текста размера алфавита – в приложении В.

ЗАКЛЮЧЕНИЕ

В ходе работы была реализована программа на языке C++ с использованием фреймворка Qt для кодирования и декодирования текста с использованием алгоритмов Хаффмана и Шеннона-Фано. Также было реализовано автоматическое исследование представленных алгоритмов. Для взаимодействия с программой был реализован графический интерфейс.

Исследование алгоритмов показало, что самым эффективным с точки зрения сжатия из представленных алгоритмов является динамический алгоритм Хаффмана. Также было отмечено, что при сопоставимых размерах входного алфавита и текста все алгоритмы являются неэффективными. Однако динамический алгоритм Хаффмана оказался самым неэффективным с точки зрения количества выполняемых операций.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

http://doc.qt.io/

Статический алгоритм Хаффмана

Алгоритм Шеннона-Фано

Динамический алгоритм Хаффмана

ПРИЛОЖЕНИЕ А: ИССЛЕДОВАНИЕ ЗАВИСИМОСТИ ВЕСА ЗАКОДИРОВАННОГО ТЕКСТА ОТ ДЛИНЫ ВХОДНОГО ТЕКСТА

В таблице 1 приведены размеры закодированного текста для алгоритмов Хаффмана и Шеннона-Фано в зависимости от размера входного текста, график зависимости представлен на рисунке 3.

Таблица 1 – Вес закодированного текста

Длина входного текста	Статический алгоритм Хаффмана	Алгоритм Шеннона-Фано	Динамический алгоритм Хаффмана
1000	1408	1425	1015
3003	3259	3323	2784
9009	8518	8697	8047
27027	24406	24953	23813
83333	73665	75405	73087
250000	219605	224693	218901
1000000	875846	896394	875169

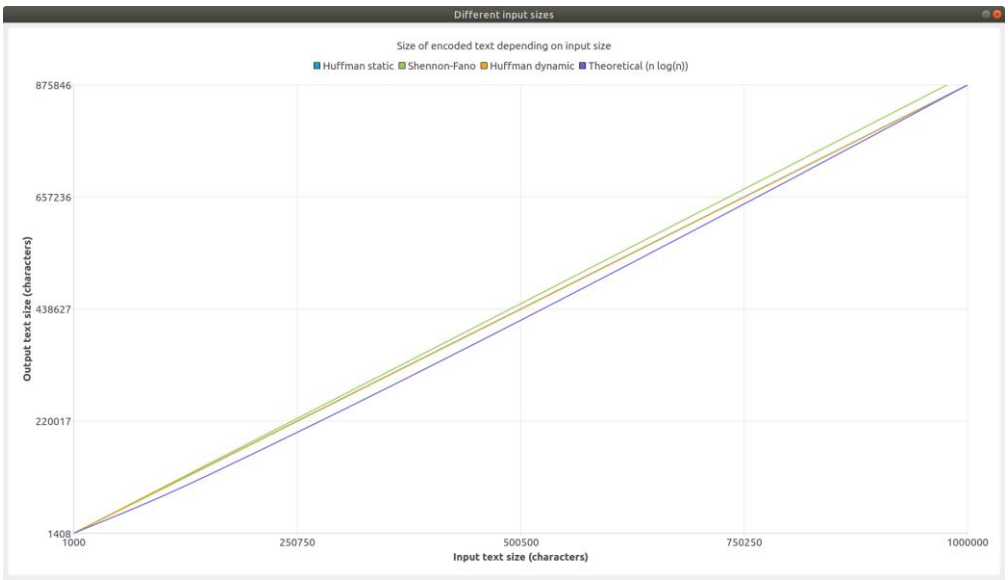

Рисунок 3 – Зависимость веса закодированного текста от размера входного текста

ПРИЛОЖЕНИЕ Б: ИССЛЕДОВАНИЕ ЗАВИСИМОСТИ КОЛИЧЕСТВА ВЫПОЛНЯЕМЫХ ОПЕРАЦИЙ ОТ ДЛИНЫ ВХОДНОГО ТЕКСТА

В результате исследования количества выполняемых операций для алгоритмов Хаффмана и Шеннона-Фано в зависимости от размера входного текста было выявлено, что статический алгоритм Хаффмана выполняет наименьшее количество операций, а динамический алгоритм Хаффмана – наибольшее. График зависимости представлен на рисунке 4.

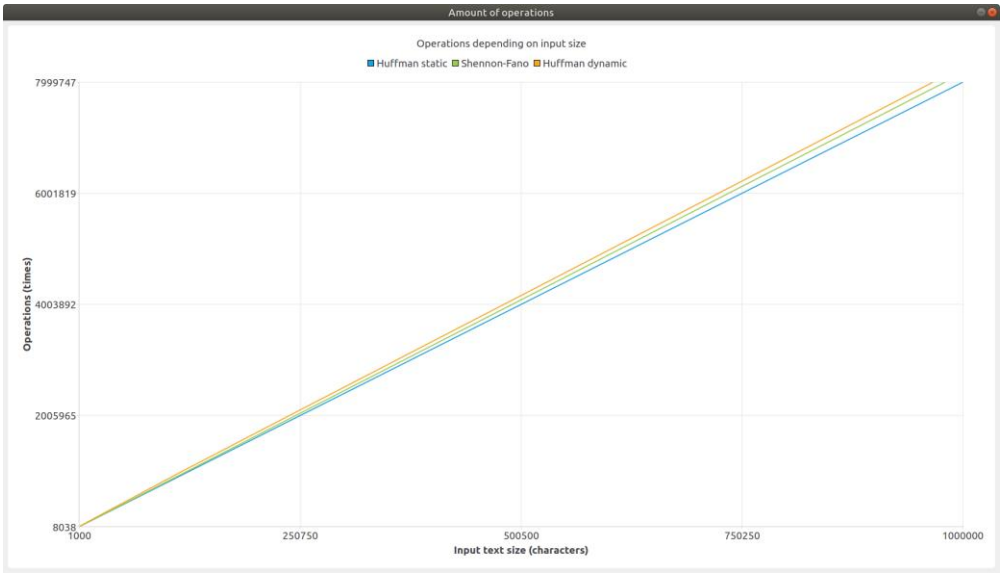

Рисунок 4 – Зависимость количества выполняемых операций от длины входного текста

ПРИЛОЖЕНИЕ В: ИССЛЕДОВАНИЕ ЗАВИСИМОСТИ ВЕСА ЗАКОДИРОВАННОГО ТЕКСТА ОТ РАЗМЕРА АЛФАВИТА ВХОДНОГО ТЕКСТА

Для исследования зависимости от размера алфавита входного текста было принято использовать текст, длинной в 500 символов. В таблице 2 приведены размеры закодированного текста для алгоритмов Хаффмана и Шеннона-Фано в зависимости от алфавита входного текста.

Таблица 2 – Вес закодированного текста

Размер алфавита входного текста	Статический алгоритм Хаффмана	Алгоритм Шеннона-Фано	Динамический алгоритм Хаффмана
2	107	107	97
4	166	178	144
8	236	241	206
16	336	345	275
32	475	484	353
64	651	660	449
128	938	948	571

Исследования было проведено на текстах, длиной 500 и 1500 символов. Графики зависимостей представлены на рисунках 5 и 6. Из результатов исследования видно, что при сопоставимых размерах алфавита и размера входного текста все алгоритмы являются неэффективными.

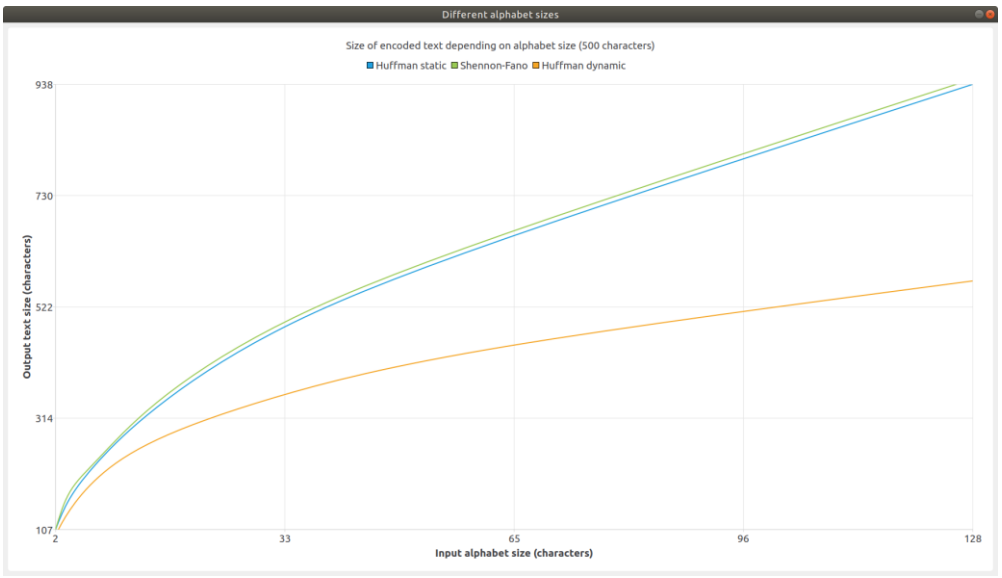

Рисунок 5 – Зависимость веса закодированного текста от размера алфавита входного текста при длине входного текста 500 символов

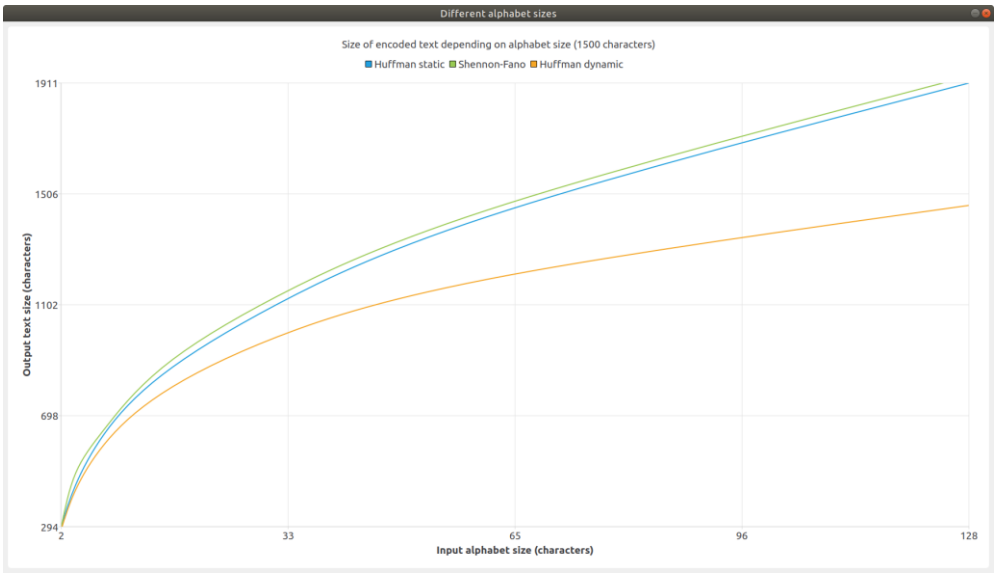

Рисунок 6 – Зависимость веса закодированного текста от размера алфавита входного текста при длине входного текста 1500 символов