МИНОБРНАУКИ РФ

ФГБОУ ВПО Тверской государственный технический университет

Кафедра “Программное обеспечение”

Курсовая работа

дисциплина “Алгоритмы и структуры данных”

Тема: “ Разделяй и властвуй ”

Выполнил: студент группы

ПИН 17.05

Завгороднев Егор

Проверил:

Мальков А.А

Тверь 2019

Оглавление

[Введение 3](#_Toc532936049)

[Аналитическая часть 3](#_Toc532936050)

[Краткая история 3](#_Toc532936051)

[Алгоритмы и их O(N) 5](#_Toc532936052)

[Сравнения алгоритмов 12](#_Toc532936053)

[Проектная часть 13](#_Toc532936054)

[Функционал 13](#_Toc532936055)

[Архитектура приложения 13](#_Toc532936056)

[Классы 14](#_Toc532936057)

[Реализация проекта 16](#_Toc532936058)

[Тестирование 18](#_Toc532936059)

[Заключение 19](#_Toc532936060)

[Литература 20](#_Toc532936061)

# Введение

Цель: Ознакомится с парадигмой разработки алгоритмов “Разделяй и властвуй” и реализовать алгоритмы этого типа.

Задача: Windows Form проект на c#.

1. Реализовать алгоритмы:

* Нисходящая сортировка слиянием
* Восходящая сортировка слиянием

1. Сравнить время выполнения
2. Реализовать алгоритмы:

* Рекурсивный алгоритм нахождения ближайших пар точек
* Алгоритм нахождения ближайших пар точек простым перебором

1. Сравнить время выполнения

# Аналитическая часть

## Краткая история

Одним из самых важных и наиболее широко применимых методов проектирования эффективных алгоритмов является метод, называемый методом декомпозиции (или метод разделяй и властвуй, или метод разбиения). Этот метод предполагает такую декомпозицию (разбиение) задачи размера N на более мелкие задачи, что на основе решений этих более мелких задач можно легко получить решение исходной задачи.

Примеры задач, которые допускают решение таким способом:

* Сортировка слиянием
* Алгоритм нахождения ближайших пар точек

Любой алгоритм, решающий задачу методом «Разделяй и властвуй» состоит из нескольких шагов:

* Разбиение задачи на несколько подзадач меньшей размерности, которые имеют такую же структуру, а следовательно каждая из них разрешима методом «Разделяй и властвуй»
* Преобразование решений подзадач в одно решение

Ранние примеры этих алгоритмов - это прежде всего алгоритм «уменьшай и властвуй» - исходная проблема последовательно разбивается на отдельные подзадачи и действительно может быть решена итеративно.

Бинарный поиск, алгоритм «уменьшай и властвуй», где подзадачи имеют примерно половину исходного размера, имеет длинную историю. Хотя четкое описание алгоритма на компьютерах появилось в 1946 году в статье Джона Мочли, идея использования отсортированного списка предметов для облегчения поиска восходит по крайней мере к Вавилонии в 200 году до нашей эры. Другим древним алгоритмом уменьшения и завоевания является евклидов алгоритм, позволяющий вычислить наибольший общий делитель двух чисел путем сокращения чисел до меньших и меньших эквивалентных подзадач, которые датируются несколькими столетиями до нашей эры.

Ранним примером алгоритма «разделяй и властвуй» с несколькими подзадачами является описание Гауссом в 1805 году того, что сейчас называется алгоритмом быстрого преобразования Фурье (FFT).

Ранним алгоритмом «разделяй и властвуй» с двумя подзадачами, который был специально разработан для компьютеров и должным образом проанализирован, является алгоритм сортировки слиянием, изобретенный Джоном фон Нейманом в 1945 году.

## Алгоритмы и их O(N)

**Сортировка слиянием. Сложность:** *O(n log2 n)*

Сортировка слиянием (англ. merge sort) — алгоритм сортировки, который упорядочивает списки (или другие структуры данных, доступ к элементам которых можно получать только последовательно, например — потоки) в определённом порядке. Эта сортировка — хороший пример использования принципа «разделяй и властвуй». Сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи.

**Пример:**

Сортировку слиянием можно проиллюстрировать примером: 44 55 12 42 94 18 06 67.

Разобьем на первом шаге файл на два файла: 44 55 12 42 и 94 18 06 67.

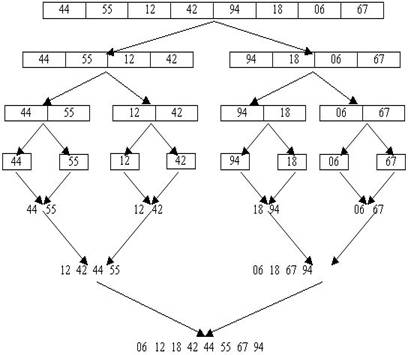
Объединим эти два файла снова в один файл: 44 94 18 55 06 12 42 67.

Снова разобьем файл и объединим пары в четверки: 44 94 18 55 и 06 12 42 67.

Слияние: 06 12 44 94 18 42 55 67.

Разобьем файл и объединим четверки: 06 12 44 94 и 18 42 55 67.

Слияние: 06 12 18 42 44 55 67 94.



**Восходящая сортировка слиянием. Сложность:**  *O(n log2 n)*

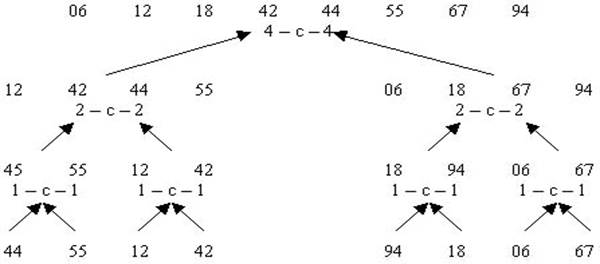
От­ка­жем­ся от ре­кур­сив­ной реа­ли­за­ции, и сде­ла­ем сор­ти­ров­ку сле­дую­щим об­ра­зом:

Сначала выполняются все слияния 1 элемент с 1, что соответствует нижнему уровню дерева, затем все слияния 2 с 2, затем слияния 4 с 4 и т. д. То есть дерево обходится снизу, с полной обработкой каждого уровня. Такую схему сортировки можно выполнить с помощью итеративного алгоритма.

Он полностью исключает первоначальное разделение массива на части, и сразу начинает выполнять слияние частей из 1-ого элемента, 2х, 3х и т.д.

Та­кой ал­го­ритм на­зы­ва­ет­ся вос­хо­дя­щей сор­ти­ров­кой слия­ни­ем.

**Пример:**

****

**Тривиальный алгоритм нахождения ближайших пар точек. Сложность:** O(n^2).

Даны n точек p_i на плоскости, заданные своими координатами (x_i,y_i). Требуется найти среди них такие две точки, расстояние между которыми минимально:

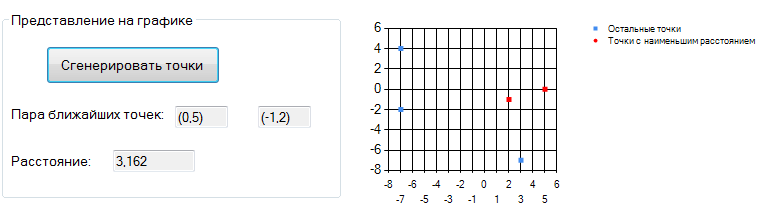
 \min_{\scriptstyle i,j=0 \ldots n-1, \atop \scrip[...]

Расстояния мы берём обычные евклидовы:

 \rho(p_i,p_j) = \sqrt{ (x_i-x_j)^2 + (y_i-y_j)^2 [...]

Тривиальный алгоритм — перебор всех пар и вычисление расстояния для каждой — работает за O(n^2).

**Пример:**



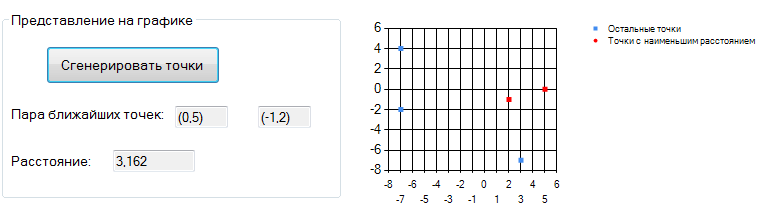
**Алгоритм нахождения ближайших пар точек. Сложность:** O(n log2 n)

Построим алгоритм по общей схеме алгоритмов **"разделяй-и-властвуй"**: алгоритм оформляем в виде рекурсивной функции, которой передаётся множество точек; эта рекурсивная функция разбивает это множество пополам, вызывает себя рекурсивно от каждой половины, а затем выполняет какие-то операции по объединению ответов. Операция объединения заключается в обнаружении случаев, когда одна точка оптимального решения попала в одну половину, а другая точка — в другую (в этом случае рекурсивные вызовы от каждой из половинок отдельно обнаружить эту пару, конечно, не смогут). Основная сложность, как всегда, заключается в эффективной реализации этой стадии объединения. Если рекурсивной функции передаётся множество из n точек, то стадия объединения должна работать не более, чем O(n), тогда асимптотика всего алгоритма T(n) будет находиться из уравнения:

 T(n) = 2 T(n/2) + O(n). 

Решением этого уравнения, как известно, является T(n) = O (n \log n).

**Пример:**



## Сравнения алгоритмов

|  |  |  |
| --- | --- | --- |
|  | Тривиальный алгоритм нахождения ближайших пар точек O(n^2). | Алгоритм нахождения ближайших пар точек  “Разделяй и властвуй”  O(n log2 n) |
| Достоинства | Простой в реализации | Быстрый |
| Недостатки | Очень медленный | - |

# Проектная часть

## Функционал

Программа:

* Выполняет два алгоритма сортировки слиянием для 10000, 20000,40000,80000 элементов.
* Подсчитывает время выполнения
* Выводит график выполнения рекурсивной сортировки слиянием
* Выполняет рекурсивную сортировку слиянием введеных пользователем чисел
* Выполняет алгоритм нахождения ближайших пар точек для пяти случайных точек, выводя результат на графике
* Выполняет два алгоритма нахождения ближайших пар точек с файла, загружаемого пользователем
* Подсчитывает время выполнения для каждого алгоритма

## Архитектура приложения

Программа состоит из четырех частей: библиотеки классов, Windows Form проекта, консольного приложения для тестов, Юнит-тестов.

В библиотеке представлены классы и методы для реализации алгоритмов.

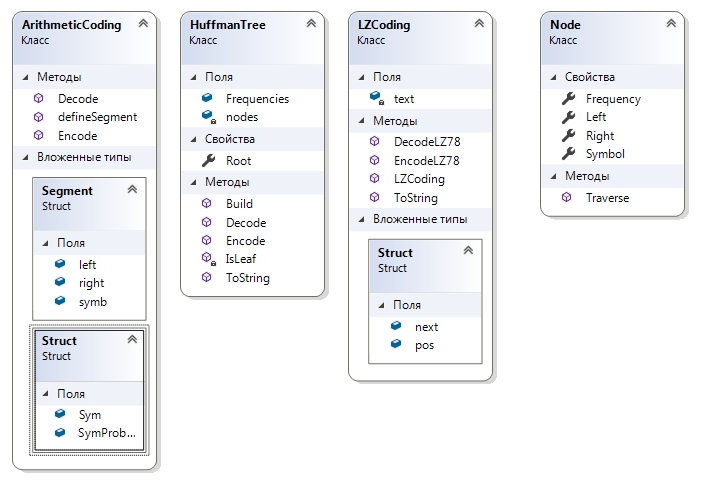
В WinForm проекте – интерфейс и его связь с классами из библиотеки.

В консольном приложении и юнит тестах выполняется тестирования приложения.

## Классы

ClassLibraryTextCoding:

* **SourceProp** - свойстватекста для арифметического кодирования
* **ArithmeticCoding** - методы для арифметического кодирования
* **LZCoding** – методы для алгоритма LZ78
* **Node** – структура узла дерева Хаффмана и метод для нахождения кода символа
* **HuffmanTree** – построение дерева Хаффмана + методы для кодирования и декодирования

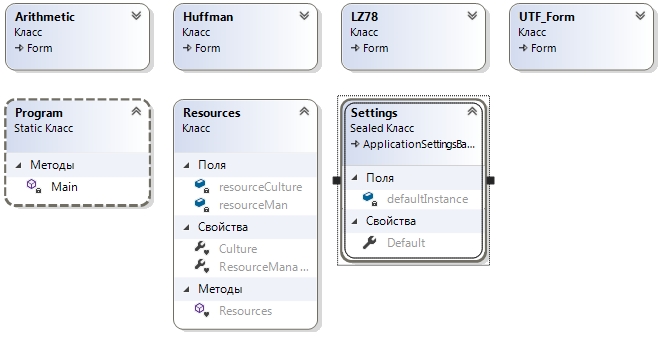


WinForm:

Классы интерфейса для алгоритмов:

* **Arithmetic**

* **Huffman**
* **LZ78**
* **UTF-8**

****

## Реализация проекта

Хронология работы над проектом:

24.11 – Начало разработки , интерфейс программы

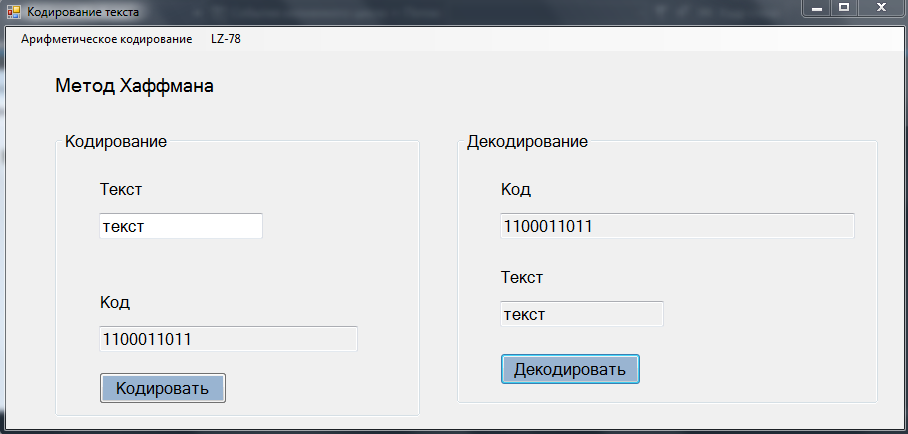
25.11 – Был написан алгоритм арифметического кодирования

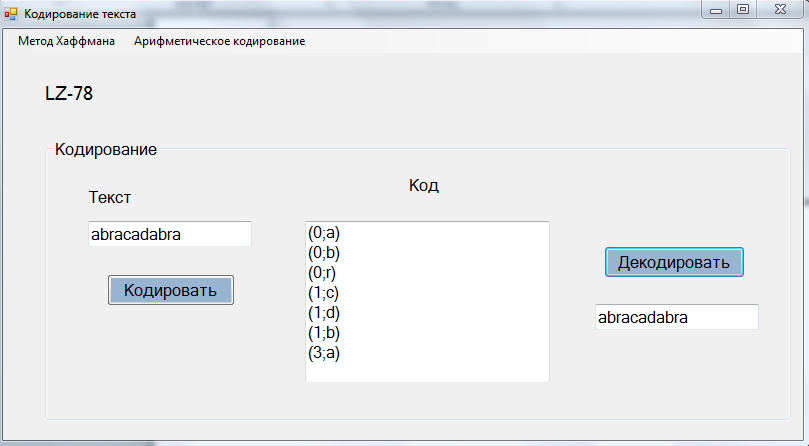
26.11 – Полностью рабочий алгоритм LZ78

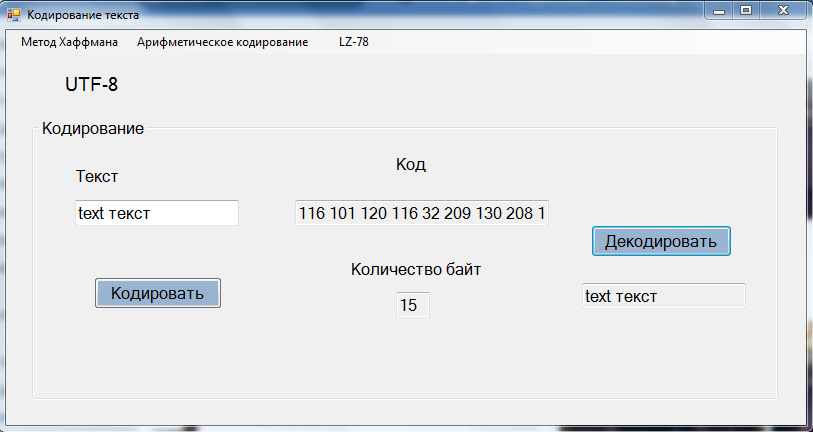
27.11 – Добавлен код Хаффмана, интерфейс привязан к библиотеке классов

8.12 - Кодирование текста в UTF-8 , исправление ошибок программы , Unit-тестирование

Интерфейс программы:







## 

Среда разработки:

Microsoft Visual Studio

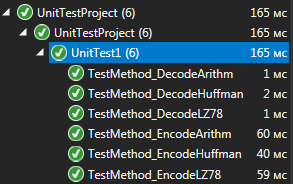
Инструменты:

C# + .NET

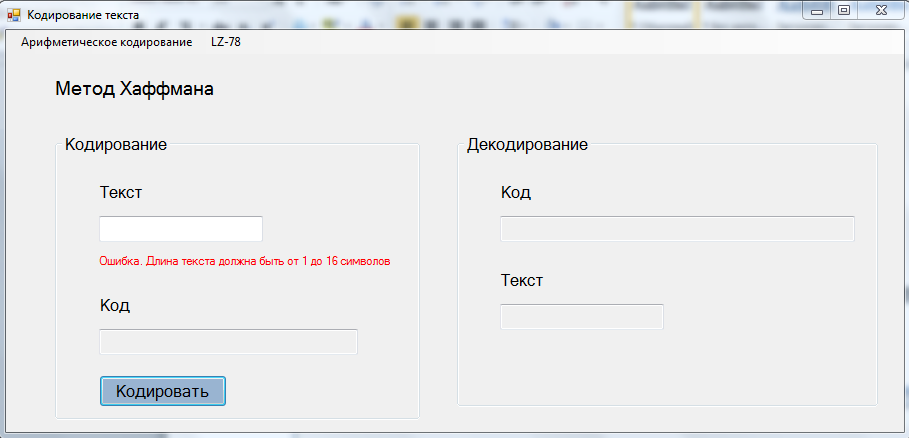
## Тестирование

Для автоматического тестирования я добавил в проект UnitTestProject.

Результаты тестов:



Также программа обрабатывает неправильные действия пользователя, которые приводят к исключениям.



# Заключение

В данной курсовой работе я реализовал 3 алгоритма кодирования и декодирования текста.

Проект был успешно протестирован.

# Литература

<https://ru.wikipedia.org> - Информация о алгоритмах

<http://gubsky.ru/study/5/soad/sh/6.htm>- Алгоритм сортировки слиянием

<https://www.intuit.ru/studies/courses/12181/1174/info> – Курс “Алгоритмы на С++”