Федеральное государственное автономное образовательное учреждение высшего образования «**Уральский федеральный университет** имени первого Президента России Б. Н. Ельцина»

Отчет по лабораторной работе №2

дисциплины «Алгоритмы и анализ сложности»

Студент: Волков Илья Евгеньевич  
Группа: РИ-200004 (АТ-08)  
Преподаватель: Рыбкина Мария Николаевна

2022 год.

# Оглавление

[Оглавление 2](#_Toc98153488)

[Задание 3](#_Toc98153489)

[Теоретическая часть 3](#_Toc98153490)

[Инструкция пользователя 4](#_Toc98153491)

[Инструкция программиста 4](#_Toc98153492)

[Тестирование 4](#_Toc98153493)

[Выводы 7](#_Toc98153494)

# Задание

Лабораторная работа №2

Алгоритмы сортировки строк.  
  
Реализовать алгоритмы сортировки. Выбрать наиболее трудоемкую операцию алгоритма. Создать план эксперимента с алгоритмом и получить статистические данные о работе алгоритма. Определить сложность алгоритма для этой операции: найти порядок сложности и коэффициент пропорциональности.

Сортировки:  
1. Bubble  
2. QSort  
3. SortTree  
4. Сортировка вставками  
5. Сортировка слиянием  
6. Сортировка с помощью красно-черного дерева

# Теоретическая часть

Есть два способа сделать данное задание:

1. Все с нуля. Все сортировки писать с нуля. В этом случае на задание уйдет много времени, однако легко будет выполнить письменную часть задания.

2. Готовый код сортировок. Найти код сортировок, изучить, понять алгоритм работы. В этом случае на задание уйдет меньше времени, но и выполнить письменную часть задания сложнее. Я выбрал этот способ.

# Инструкция пользователя

Для запуска приложения необходимо запустить Lab2.exe. Далее выбрать тестируемую сортировку. Можно увидеть результат работы.

# Инструкция программиста

Для изменения сортируемых значений в тестах необходимо изменить массив/лист arr.   
  
Для использования сортировок писать код по типу: BubbleSort.Sorting(<Данные>);

# Тестирование

Суть задания по своей сути заключается в тестировании каждого метода сортировки. В данном пункте я укажу сложность каждого из методов в худшем и лучшем случае. Реализацию самих алгоритмов можно взглянуть в .cs файлах

1. BubbleSort.

Суть работы:   
  
Будем идти по массиву слева направо. Если текущий элемент больше следующего, меняем их местами. Делаем так, пока массив не будет отсортирован. Заметим, что после первой итерации самый большой элемент будет находиться в конце массива, на правильном месте. После двух итераций на правильном месте будут стоять два наибольших элемента, и так далее.

Асимптотика: в худшем и среднем случае – O(n2), в лучшем случае – O(n).  
  
Эксперименты подтвердили асимптотику.

1. QSort.

Суть работы:  
  
Выберем некоторый опорный элемент. После этого перекинем все элементы, меньшие его, налево, а большие – направо. Рекурсивно вызовемся от каждой из частей. В итоге получим отсортированный массив, так как каждый элемент меньше опорного стоял раньше каждого большего опорного. Моя реализация этого алгоритма совершенно стандартна, идем одновременно слева и справа, находим пару элементов, таких, что левый элемент больше опорного, а правый меньше, и меняем их местами. Помимо чистой быстрой сортировки, участвовала в сравнении и сортировка, переходящая при малом количестве элементов на сортировку вставками.

Асимптотика: O() в среднем и лучшем случае, в худшем O(). Наихудшая оценка достигается при неудачном выборе опорного элемента.

Эксперименты подтвердили асимптотику.

1. SortTree.

Суть работы:  
  
Будем вставлять элементы в двоичное дерево поиска. После того, как все элементы вставлены достаточно обойти дерево в глубину и получить отсортированный массив.

Асимптотика: O() в худшем, среднем и лучшем случае.

Эксперименты подтвердили асимптотику.

1. Сортировка вставками.

Суть работы:  
  
Создадим массив, в котором после завершения алгоритма будет лежать ответ. Будем поочередно вставлять элементы из исходного массива так, чтобы элементы в массиве-ответе всегда были отсортированы.

Асимптотика: в среднем и худшем случае – O(), в лучшем – O(n).

Эксперименты подтвердили асимптотику.

1. Сортировка слиянием.

Суть работы:  
  
Сортировка, основанная на парадигме «разделяй и властвуй». Разделим массив пополам, рекурсивно отсортируем части, после чего выполним процедуру слияния: поддерживаем два указателя, один на текущий элемент первой части, второй – на текущий элемент второй части. Из этих двух элементов выбираем минимальный, вставляем в ответ и сдвигаем указатель, соответствующий минимуму. Эта сортировка рекурсивна, как и быстрая, а потому возможен переход на квадратичную при небольшом числе элементов.  
  
Асимптотика: O().

Эксперименты подтвердили асимптотику, однако не некоторых данных O().

Сортировка с помощью красно-черного дерева  
Суть работы:  
  
Очень похоже на [дерево AVL](https://simpledevcode.wordpress.com/2014/09/16/avl-tree-in-c/) , которое также является самобалансирующимся и имеет одинаковую временную сложность O() для лучшего, среднего и худшего случая. Красно-черные деревья немного более эффективны при вставке и удалении, поскольку они требуют меньше работы, поскольку нам не нужно просматривать дерево на предмет дисбаланса, мы просто смотрим на цвет родительского узла и выполняем исправление.  
  
Асимптотика: O().  
  
Эксперименты подтвердили асимптотику.

# Выводы

Итого в данной выборке методов сортировок самой быстрой на одинаковых данных оказалась быстрая сортировка. Однако не стоит забывать, что на скорость может влиять множество факторов. Эта лабораторная работа позволила еще раз окунуться в сортировки и анализ их сложности.