Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования  
«Уральский федеральный университет имени первого Президента России Б.Н.Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Анализ сложности алгоритмов сортировки строк

Отчёт по лабораторной работе  
по дисциплине «Алгоритмы, структуры данных и анализ сложности» Вариант 2

Выполнил: студент группы, Аристов В.Е.

Преподаватель: доцент, к.ф.-м.н. Трофимов С.П.

2023

**Оглавление**

**Задание**

Реализовать один из алгоритмов сортировки строк:  
6. QSort  
Выбор алгоритма выбирается по согласованию с преподавателем.

Для алгоритма определить сложность относительно наиболее характерной операции (сравнение, перестановка и др.). Вид функции сложности F(n) подобрать в соответствии с теорией. Например, для оптимальных алгоритмов F(n) = C\*n\*log2(n). Найти также коэффициент пропорциональности C. Для аппроксимации можно использовать метод наименьших квадратов и сервис «Поиск решения».

План проведения эксперимента с алгоритмом называется массовой задачей. Представьте план в виде xml-файла.

**Теоретическая часть**

Быстрая сортировка, сортировка Хоара (англ. quicksort), часто называемая qsort (по имени в стандартной библиотеке языка Си) — алгоритм сортировки, разработанный английским информатиком Тони Хоаром в 1960 году. Алгоритм был придуман Хоаром во время его пребывания в Советском Союзе, где он обучался в Московском университете компьютерному переводу и занимался разработкой русско-английского разговорника.

Общая идея алгоритма состоит в следующем:

1. Выбрать опорное значение. Это может быть среднее между первым и последним элементами или среднее между минимальным и максимальным элементами или элемент из середины массива. Нагляднее, если опорное значение будет отличаться от элементов массива.

2. Массив надо разделить на два подмассива так, чтобы левый массив был меньше опорного элемента, а правый – больше опорного элемента.

3. Для этого перебираем элементы исходного массива двумя способами: слева направо и справа налево. Сравниваем эти два элемента с опорным числом.

При необходимости – переставляем.

4. Для массивов «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина массива больше единицы.

Особенности алгоритма:

1. Алгоритм O(logn) дополнительной памяти хранения опорных значений при рекурсивных вызовах,

2. Использует две операции: сравнение и перестановку.

3. Средняя сложность алгоритма O(nlogn).

4. В худшем случае сложность алгоритма O(n\*n).

5. Может привести к переполнению стека

6. Неустойчив, то есть может изменить взаимное расположение одинаковых элементов.

**Инструкция пользователя**

**Инструкция программиста**