Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Сортировки»**

**Выполнил**:

студент группы 3821Б1ПМ2

Задиванов Алексей Андреевич

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2021

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 7](#_Toc26962566)

[Результаты экспериментов 8](#_Toc26962567)

[Заключение 9](#_Toc26962568)

[Приложение 10](#_Toc26962569)

# Постановка задачи

Реализовать быструю сортировку, сортировку пузырьком, сортировку слиянием и поразрядную сортировку массива с элементами типа данных float.

# Метод решения

Сортировка пузырьком

Самый маленький элемент переходит со своего места в начало массива. Мы начинаем сравнивать 2 элемента с конца (n-2)(n-1) если массив размера n , если элемент (n-1) оказался меньше, меняем местами их значения, дальше мы сравниваем уже элементы с индаксами (n-3)(n-2) , если элемент (n-2) оказался меньше, меняем местами их значения, продолжаем до начала списка. За один проход самый маленький элемент станет на 1 место, затем второй и т.д. всего может быть n-1 проход по массиву, и он отсортирован. Сложность работы сопоставимо с квадратом числа элементов. Худший случай, когда массив отсортирован в обратном направлении. Сортировка устойчивая т.к. не меняет порядок одинаковых элементов.

Быстрая сортировка

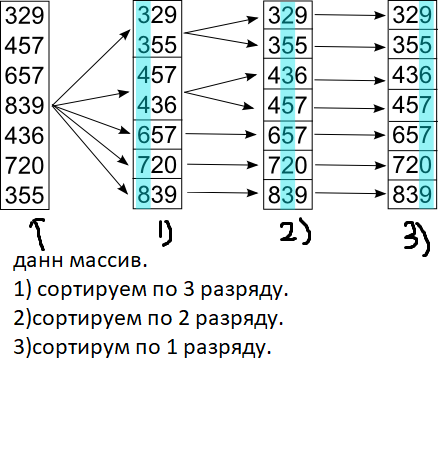
Из массива выбирается элемент, который разделяет массив на 2 части, удобно брать элемент со средним индексом, затем запускается цикл поиска элементов слева и справа, которые не на своем месте, и меняются местами, с изменением индексов. Затем снова применяется эта сортировка для каждой из двух частей. Наилучший случай, когда элемент который мы выбрали за разделитель будет стоять на своем месте. Тогда за 1 итерацию 1 элемент найдет свое место, затем еще 2, потом 4 и так до конца. В этом случае сложность сортировки сравнима с nlogn , но оно может быстро перейти к квадрату числа элементов, если элементы будут подбираться плохо, например если средний элемент, который мы выбрали в качестве разделителя будет минимальным или максимальным.

Сортировка слиянием

Разбиение массива на части по 1 элементу, затем начинаются слияния первых двух элементов затем двух вторых они объединяются в четверки, затем со следующей четверкой в восьмерки и т.д. Сложность алгоритма постоянна и равна nlogn это и + и - , т.к. работает одинаково долго на почти отсортированных массивах и совсем не отсортированных. Нет наилучших и наихудших случаев. Сортировка устойчива, т.к. не меняет порядок одинаковых элементов.

Поразрядная сортировка

Сортировка чисел по разрядам, числа делятся на разряды и сравниваются между собой. Если они равные то идут в одну группу, далее идет сортировка идет по следующему разряду и предыдущая группа разбивается еще на подгруппы, продолжаем так идти и получаем отсортированный массив. Например:



Элементы с одинаковыми числами в разрядах не меняют свое местоположение, значит сортировка устойчива. Сложность О(m\*n) m - количество бит на хранение 1 элемента n – число элементов

# Руководство пользователя

Ввести цифру нужной сортировки, подождать, увидеть результат.

# Описание программной реализации

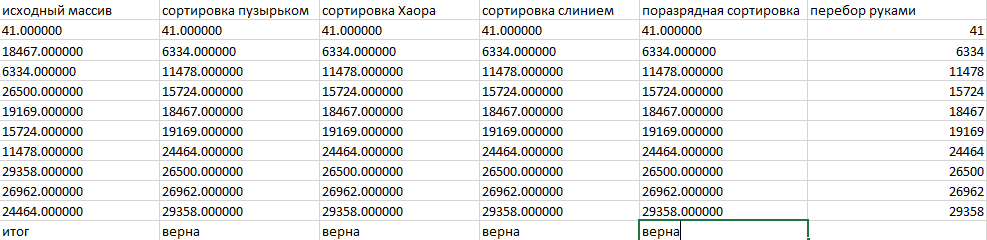
Главный файл main.

Функции:

1. main() главная работает с пользователем, вызывает все оставшиеся функции.
2. bubble\_sort() функция сортировки пузырьком.
3. quick\_sort() функция быстрой сортировки Хаора
4. merge\_sort() функция сортировки слиянием.
5. RadixSort() функция поразрядной сортировки.
6. radixPass() функция сортировки в рамках 1 разряда
7. signedradixPass() функция ставит отрицательные числа в начало
8. createCounters() функция сортировки подсчетом на 256 элементов.

# Подтверждение корректности

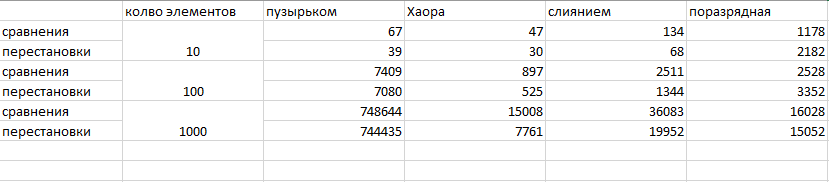
Для подтверждения корректности в программе можно все сортировки сравнить с заведомо рабочей сортировкой пузырька , при своих введенных данных, или самому отсортировать массив.



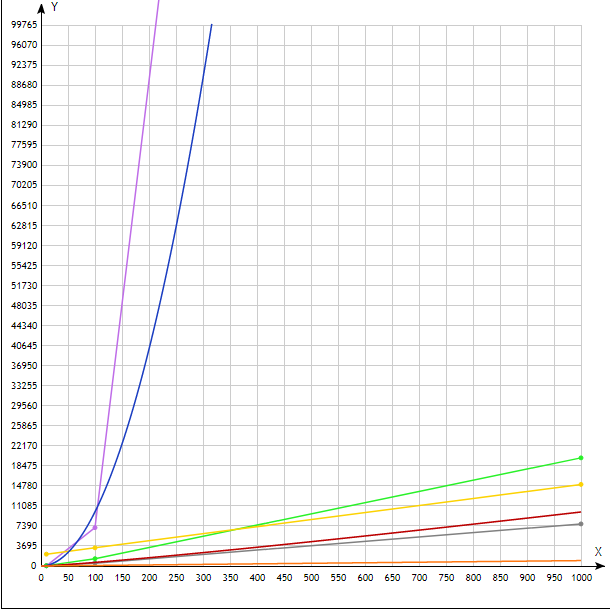
Как вижу все сортировки вывели один и тот же результат значит работают корректно.

# Результаты экспериментов

По данным экспериментов видно, что при малых значениях количества элементов сортировки отрабатывают за примерно одинаковое количество перестановок и сравнений, кроме поразрядной, при больших значениях количества элементов сортировка пузырьком не имеет смысла, т.к. затрачивает больше ресурсов чем другие. Оптимальной сортировкой является быстрая сортировка Хаора.



По данным таблицы можно построить примерные графики и сравнить с графиками теоретической сложности.



Сортировка пузырьком (фиолетовый график) очень похожа на график n^2 (синий график), то есть теоретическая сложность подтверждена.

Быстрая сортировка (серый график) и сортировка слиянием (зеленый график) похожи на график функции nlog2n (красный график), примерная теоретическая сложность.

Поразрядная сортировка (желтый график) и график функции y = x (оранжевый график)

Схожи ,т.к. теоретическая сложность примерна равна О (х\*n) если график y = x умножить на константу будет еще больше схожести, теоретическая сложность подтверждена.

# Заключение

При малом размере массива нет разницы каким алгоритмом будешь сортировать, самая удобная сортировка как по мне это быстрая сортировка, она работает за оптимальное количество времени и действий.

# Приложение

Часть кода, который вы считаете основным