Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **«Информатика и системы управления»**

КАФЕДРА **«Программное обеспечение ЭВМ и информационные технологии»**

Отчет по лабораторной работе №3 по курсу «Анализ алгоритмов»

Тема Сравнение алгоритмов сортировок

Студент Ковалев Д.А.

Группа ИУ7-53Б

Преподаватели Волкова Л.Л., Строганов Ю.В.

Москва — 2020 г.

Оглавление

Введение

Сортировкой, или упорядочением массива называется расположение его элементов по возрастанию (или убыванию). [1] Для решения задачи сортировки было разработано множество алгоритмов, различающихся по трудоемкости и эффективности в связи с затрачиваемыми ими ресурсами ЭВМ. К таким алгоритмам относятся сортировка пузырьком, сортировка вставками, сортировка выбором, сортировка Шелла, сортировка расческой, быстрая сортировка и многие другие.

В рамках этой лабораторной работы будут рассмотрены следующие алгоритмы сортировки: сортировка пузырьком, сортировка вставками и быстрая сортировка.

Целью данной лабораторной работы является изучение, реализация и сравнительный анализ алгоритмов сортировки массивов.

В данной лабораторной работе требуется решить пять задач.

1. Изучить и программно реализовать алгоритм сортировки пузырьком;
2. Изучить и программно реализовать алгоритм сортировки вставками;
3. Изучить и программно реализовать алгоритм быстрой сортировки;
4. Оценить трудоемкость вышеперечисленных алгоритмов;
5. Сделать сравнительный анализ по затрачиваемым ресурсам (времени) компьютера на реализацию каждого рассмотренного алгоритма.

Критерии оценки эффективности алгоритма.

1. Количество шагов алгоритма, необходимых для упорядочения.
2. Количество сравнений элементов.
3. Количество перестановок, выполняемых при сортировке.

1. Аналитическая часть

В данном разделе будут показаны теоретические сведения о каждом алгоритме, их преимущества и недостатки.

1.1 Сортировка пузырьком

Сортировка пузырьком - это сортировка, в которой обходим массив от начала до конца, попутно меняя местами неотсортированные соседние элементы. В результате первого прохода на последнее место «всплывёт» максимальный элемент. Теперь снова обходим неотсортированную часть массива (от первого элемента до предпоследнего) и меняем по пути неотсортированных соседей. Данный алгоритм не применяется на практике из-за низкой эффективности. [2]

1.2 Сортировка вставками

Сортировка вставками - алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов. [3]

1.3 Быстрая сортировка

Быстрая сортировка - представляет собой усовершенствованный метод сортировки, основанный на принципе обмена. [4]

Основные шаги алгоритма:

1. Выбрать из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность. [5]

2. Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньшие опорного», «равные» и «большие».

3. Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

1.4 Вывод

В данном разделе приведены основные принципы каждого алгоритма сортировок. Указано, что необходимо сделать для сортировки.

2. Конструкторская часть

В данной части будут приведены схемы алгоритма сортировки пузырьком, вставкой и схема быстрой сортировки.

2.1 Сортировка пузырьком

На рисунке 2.1 представлена схема алгоритма сортировки пузырьком.

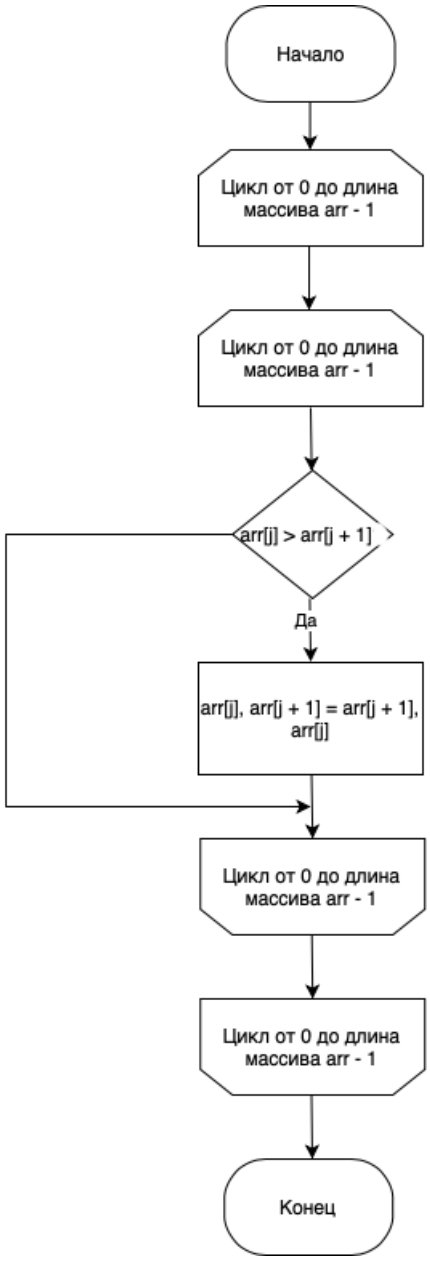

Рис 2.1: Схема алгоритма сортировки пузырьком

2.2 Сортировка вставками

На рисунке 2.2 представлена схема алгоритма сортировки вставками.

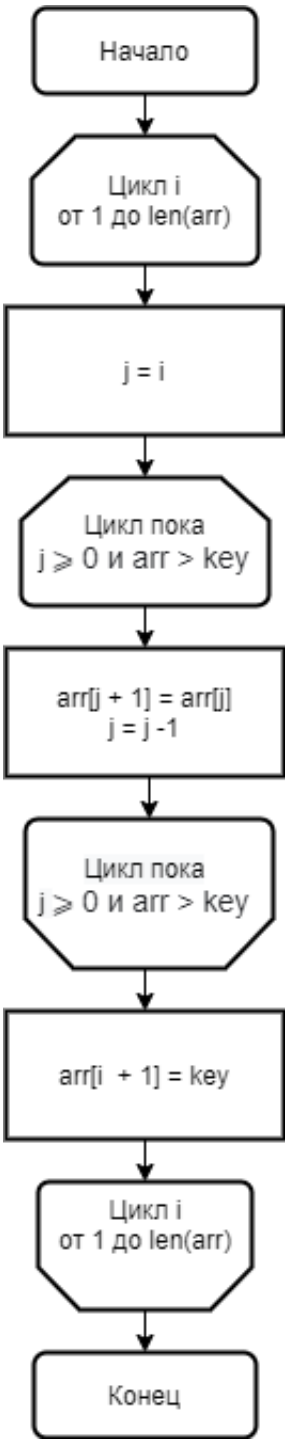

Рис 2.2: Схема алгоритма сортировки вставками

2.3 Быстрая сортировка

На рисунках 2.3 - 2.4 - 2.5 представлена схема алгоритма быстрой сортировки.

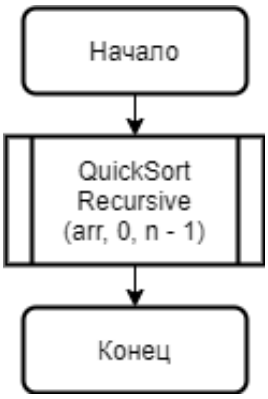

Рис 2.3: Схема алгоритма быстрой сортировки (часть 1)

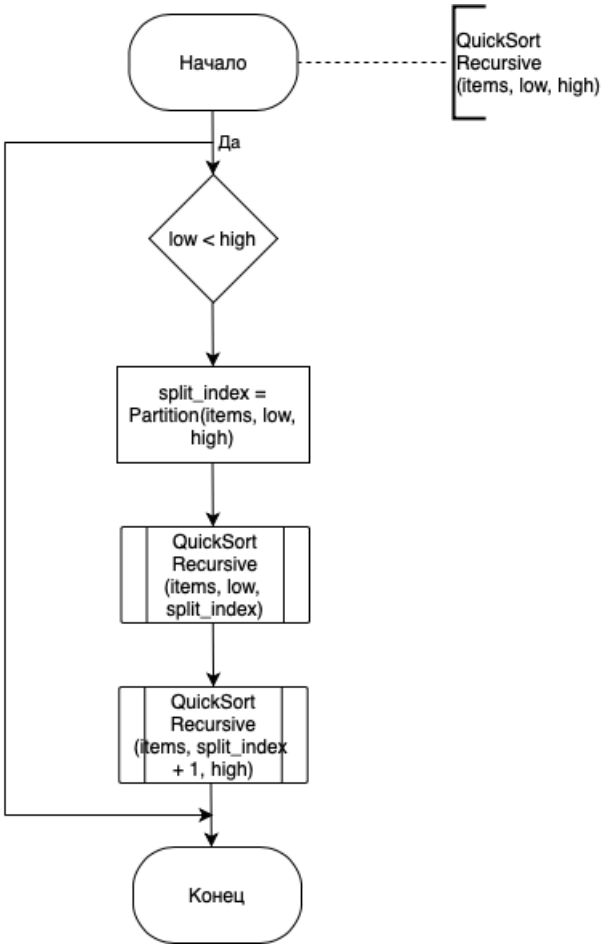

Рис 2.4: Схема алгоритма быстрой сортировки (часть 2)

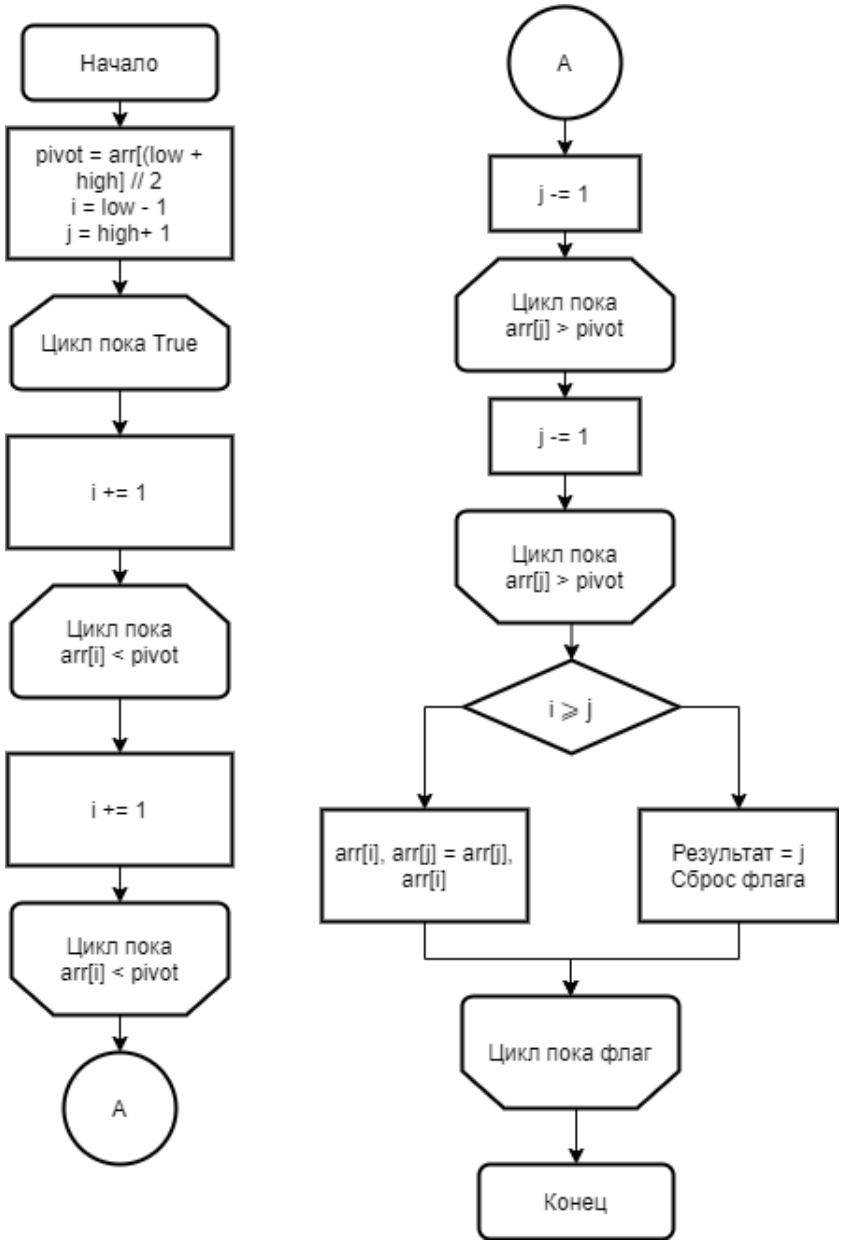

Рис 2.5: Схема алгоритма быстрой сортировки (часть 3)

2.4 Сравнительный анализ трудоемкости алгоритмов

Введем модель вычислений трудоемкости алгоритма. Пусть трудоемкость 1 у следующих базовых операций: +, -, *, /, =, ==, !=, <, <=, >, >=. Трудоемкость цикла: f цикла = f иниц + f сравн + N итер(f тела + fинкрем + fсравн).Трудоемкость условного перехода 1.

Алгоритм сортировки пузырьком обладает трудоемкостью 2.1.

$$N^2 * \begin{bmatrix} 4 \text{ л.с.} \\ 8.5 \text{ х.с} \end{bmatrix} + N * \begin{bmatrix} 3 \text{ л.с.} \\ \text{-}1.5 \text{ х.с} \end{bmatrix} - 4 \tag{2.1}$$

Трудоемкость квадратичная от размера массива.

Сортировка вставками в лучшем случае, если уже отсортированный массив: O(N). В худшем случае, если обратно отсортированный массив: $O(N^2)$.

Быстрая сортировка в лучшем случае: O(Nlog(N)).

В худшем случае: $O(N^2)$.

2.5 Вывод

В данном разделе приведены схемы трех алгоритмов: сортировка пузырьком, сортировка вставками и быстрая сортировка, а так же приведена трудоемкость каждого алгоритма.

3. Технологическая часть

В данном разделе будет описана программная реализация трех алгоритмов

3.1 Выбор языка программирования

Для реализации алгоритмов методом сортировки был выбран язык программирования Golang как наиболее удобный и понятный для реализации алгоритмов сортировки. Для замера времени использовалась система тестирования Benchmark.

В листинге 3.1 представлена функция сортировки "пузырьком".

Листинг 3.1: Подпрограмма алгоритма сортировки "пузырьком"

```
func BubbleSort(numbers []int) []int{
  for i:= len(numbers); i > 0; i--{
    for j := 1; j < i; j++{
      if numbers[j-1] > numbers[j]{
        t := numbers[j]
        numbers[j] = numbers[j-1]
        numbers[j-1] = t
      }
    }
  }
  return numbers
}
```

В листинге 3.2 представлена подпрограмма сортировки вставками.

Листинг 3.2: Подпрограмма алгоритма сортировки вставками

```
func InsertionSort(arr []int) []int {
    len := len(arr)
    for i := 1; i < len; i++ {
        for j := 0; j < i; j++ {
            if arr[j] > arr[i] {
                arr[j], arr[i] = arr[i], arr[j]
```

```
            }
        }
    }
    return arr
}
```

В листинге 3.3 представлена подпрограмма быстрой сортировки.

Листинг 3.3: Подпрограмма алгоритма быстрой сортировки

```
func QuickSort(a []int) []int {
    if len(a) < 2 {
        return a
    }

    left, right := 0, len(a)-1

    pivot := rand.Int() % len(a)

    a[pivot], a[right] = a[right], a[pivot]

    for i, _ := range a {
        if a[i] < a[right] {
            a[left], a[i] = a[i], a[left]
            left++
        }
    }

    a[left], a[right] = a[right], a[left]

    QuickSort(a[:left])
    QuickSort(a[left+1:])

    return a
}
```

3.2 Тесты

В данном разделе будут представлены тесты.

1. При массиве arr = [1, 3, 5, 2, 4] Сортировка пузырьком: 1 2 3 4 5 Сортировка вставками: 1 2 3 4 5 Быстрая сортировка: 1 2 3 4 5

2. При массиве arr = [10, 9, 8, 7, 6, 5, 1, 2, 3, 4] Сортировка "пузырьком": 1 2 3 4 5 6 7 8 9 10 Сортировка вставками: 1 2 3 4 5 6 7 8 9 10 Быстрая сортировка: 1 2 3 4 5 6 7 8 9 10

3. При массиве arr = [1] Сортировка "пузырьком": 1 Сортировка вставками: 1 Быстрая сортировка: 1

4. Исследовательская часть

В данном разделе будет сделано сравнение трех алгоритмов сортировки в разных случаях

На таблице 4.1 показана сравнительная характеристика времени выполнения всех алгоритмов при заполнении массива разной длины.

4.1 Случайное заполнение массива

Таблица 4.1: Сравнительная таблица времени выполнения (сек.) всех алгоритмов при заполнении массива случайными элементами за 50 раз

Длина массива	Пузырек	Вставки	Быстрая сортировка
20	0.0034351	0.0010774	0.0012343
50	0.0244715	0.0066717	0.0050528
80	0.0444493	0.0133443	0.0057573
100	0.0678084	0.0193913	0.0075033

График зависимости времени работы каждого алгоритма от количества элементов в последовательности(результаты приведены в (c), каждый алгоритм был выполнен 50 раз) - рисунок 4.2

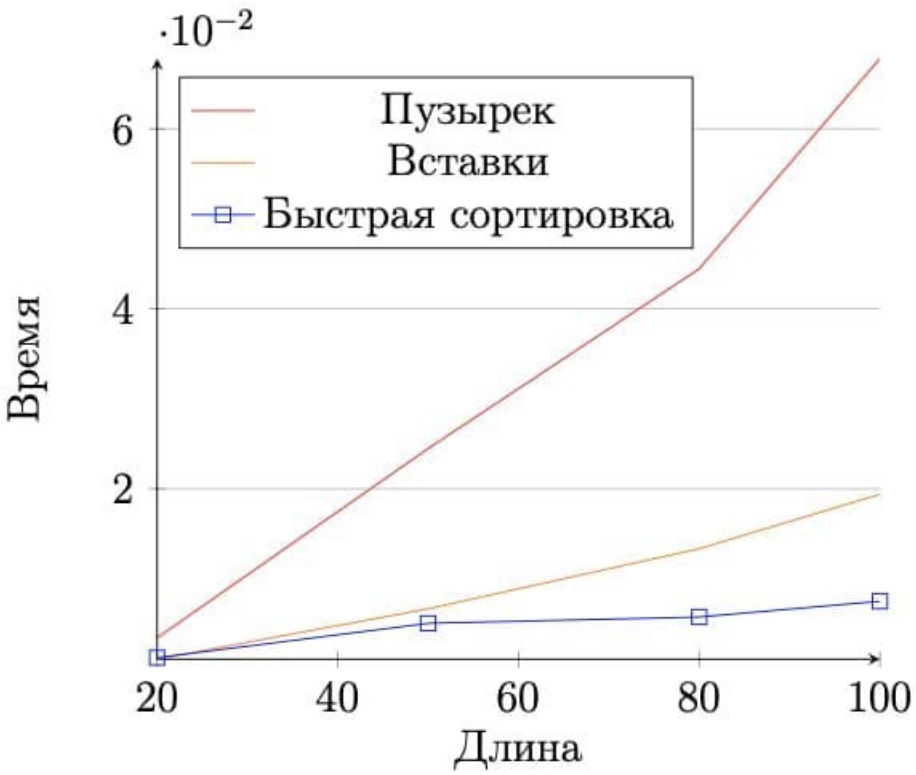

Рис 4.2: График зависимости времени работы каждого алгоритма от количества элементов в последовательности.

4.2 Лучшая ситуация

Рассмотрим случай, когда массив длиной 50 уже изначально отсортирован(замеры произведены в (c) за 50 раз). Лучший случай для быстрой сортировки: Если в каждой итерации каждый из подмассивов будет делиться на два равных по величине массива

На рисунке 4.3 показана работа алгоритмов сортировки массивов при лучшем случае.

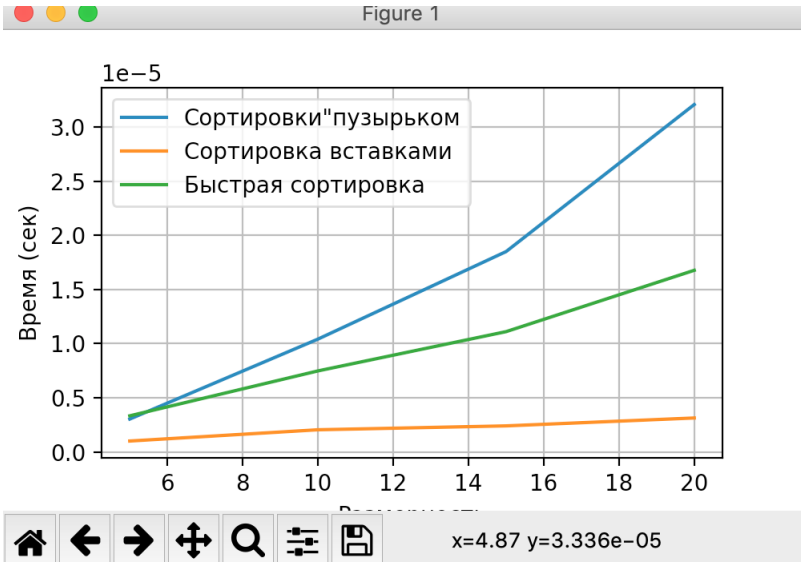

Рис 4.3: Сравнение времени выполнения алгоритмов (лучший случай)

4.3 Худшая ситуация

Рассмотрим случай, когда массив длиной 50 отсортирован по убыванию (замеры произведены в (с) за 50 раз).

Для метода вставок наихудшим случаем является массив, отсортированный в порядке, обратном нужному, аналогично и для метода пузырька. Для метода быстрой сортировки худший случай наступает в 3-х ситуациях.

1. Массив уже отсортирован в том же порядке.
2. Массив уже отсортирован в обратном порядке.
3. Все элементы одинаковы (особый случай случаев 1 и 2).

На рисунке 4.4 представлено сравнение времени выполнения алгоритмов в худшем случае.

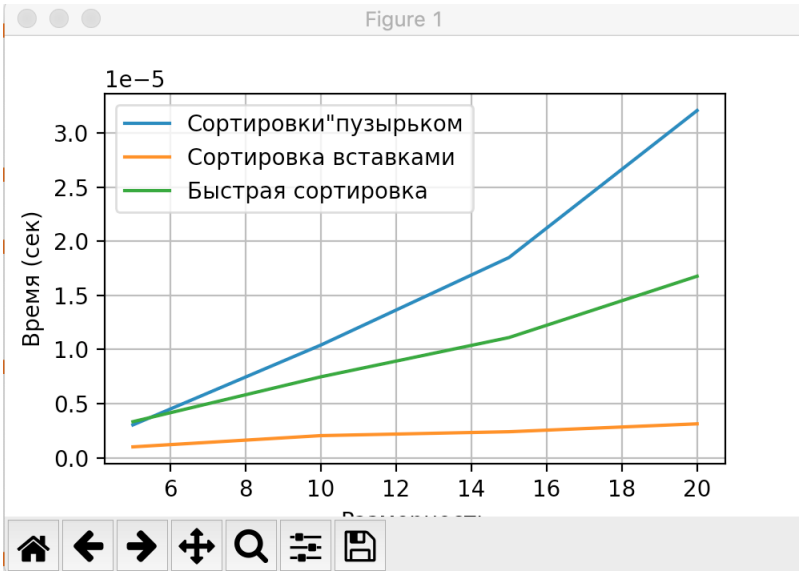

Рис 4.4: Сравнение времени выполнения алгоритмов (худший случай)

4.4 Вывод

В данном разделе представлены сравнительные характеристики трех алгоритмов. Можно утверждать, что в любом случае наихудшим алгоритмом является пузырек, он работает дольше всего. Если массив изначально отсортирован, то лучшее время показал алгоритм, использующий метод вставок. В случайном случае наилучшую работоспособность показал алгоритм быстрой сортировки, так как данная ситуация встречаются чаще всего, то можно утверждать, что лучшим алгоритмом для сортировки является алгоритм быстрой сортировки.

Заключение

В ходе данной лабораторной работе были изучены и программно реализованы алгоритмы сортировок(пузырек, вставка, метод быстрой сортировки). Сделан вывод, что в худшем случае все эти алгоритмы имеют квадратичную сложность. В остальных случаях наилучший результат показал алгоритм быстрой сортировки. Так же хотелось бы отметить, что алгоритм пузырька прост для написания, в отличие от алгоритма быстрой сортировки. Алгоритм пузырек проигрывает по времени остальным алгоритмам, следовательно, является самым медленным.

В заключении достигнута поставленная цель и решены следующие задачи.

1. Изучен и программно реализован алгоритм сортировки "пузырьком".
2. Изучен и программно реализован алгоритм сортировки вставками.
3. Изучен и программно реализован алгоритм быстрой сортировки.
4. Оценена трудоемкость вышеперечисленных алгоритмов.
5. Сделан сравнительный анализ по затрачиваемым ресурсам (времени) компьютера на реализацию каждого рассмотренного алгоритма.

Литература

[1] Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн - Алгоритмы. Построение и анализ. Издание 3-е

[2] Алгоритмы: разработка и применение Авторы: Джон Клейнберг, Эва Тардос. Год издания: 2016.

[3] Введение в анализ алгоритмов Автор: Майкл Солтис. Год издания: 2019.

[4] Vijay V. Vazirani, «Approximation Algorithms» — о том, как быстро находить достаточно хорошие решения для трудных оптимизационных задач.

[5] Совершенный алгоритм. Основы Автор: Тим Рафгарден. Год издания: 2019.