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"

Студент Сукочева Алис
Группа ИУ7-53Б
Название предприятия МГТУ им. Н. Э. Баумана, каф. ИУ7
Тема Алгоритмы сортировки

Студент:	_____	Сукочева А.
	подпись, дата	Фамилия, И.О.
Преподаватель:	_____	Волкова Л.Л.
	подпись, дата	Фамилия, И. О.
Преподаватель:	_____	Строганов Ю.В.
	подпись, дата	Фамилия, И. О.

Москва — 2020 г.

Содержание

Введение

В данной лабораторной работе будут рассмотрены алгоритмы сортировки. Данные алгоритмы широко используются в программировании и нуждаются в быстрой реализации.

Алгоритм сортировки — это алгоритм для упорядочивания элементов в списке.

Виды сортировок.

а) Сортировка вставками.
б) Пузырьковая сортировка.
в) Сортировка Шелла.
г) Корневая сортировка.
д) Пирамидальная сортировка.
е) Сортировка слиянием.
ж) Быстрая сортировка.
з) Внешняя многофазная сортировка слиянием.

В данной работе мы рассмотрим только три алгоритма сортировки: вставками, пузырек и быстрая сортировка.

Целью данной работы является изучение трех алгоритмов сортировки и реализации данных алгоритмов.

В рамках выполнения работы необходимо решить следующие задачи.

а) Изучение трех алгоритмов сортировки.
б) Реализация изученных алгоритмов.
в) Получение практических навыков.
г) Сравнительный анализ реализаций алгоритмов сортировки.
д) Экспериментальное подтверждение различий во временной эффективности.

1 Аналитический раздел

1.1 Описание алгоритмов

Сортировка вставками.

Основная идея сортировки вставками состоит в том, что при добавлении нового элемента в уже отсортированный список его стоит сразу вставлять в нужное место вместо того, чтобы вставлять его в произвольное место, а затем заново сортировать весь список. В алгоритме сортировки вставками первый элемент любого списка считается отсортированным списком длиной один. Двухэлементный отсортированный список создается добавлением второго элемента исходного списка в нужное место одноэлементного списка, содержащего первый элемент. Данный процесс вставки продолжается до тех пор, пока все элементы исходного списка не окажутся в расширяющейся отсортированной части списка.

Пузырьковая сортировка.

Алгоритм пузырьковой сортировки совершает несколько проходов по списку. При каждом проходе происходит сравнение соседних элементов. Если порядок соседних элементов неправильный, то они меняются местами. Каждый проход начинается с начала списка.

Быстрая сортировка

В алгоритме быстрой сортировки (Quicksort) используется рекурсивный подход. Выбрав опорный элемент в списке данный алгоритм сортировки делит список на две части, относительно выбранного элемента. Далее в первую часть попадают все элементы, меньшие выбранного, а во вторую — большие элементы. Если в данных частях более двух элементов, рекурсивно запускается для него та же процедура. В конце получится полностью отсортированная последовательность.

1.2 Вывод

Пузырьковая сортировка сравнивает элементы попарно, переставляя между собой элементы тех пар, порядок в которых нарушен. Сортировка вставками, сортирует список, вставляя очередной элемент в нужное место уже отсортированного списка. Быстрая сортировка определяет опорный элемент и далее переставляет элементы, относительно выбранного элемента.

Были рассмотрены основополагающие материалами, которые в дальнейшем потребуются при реализации алгоритмов сортировки.

2 Констукторский раздел

В данном разделе мы рассмотрим схемы алгоритмов сортировки.

На рис. 2.1 представлена схема алгоритма сортировки пузырьком.

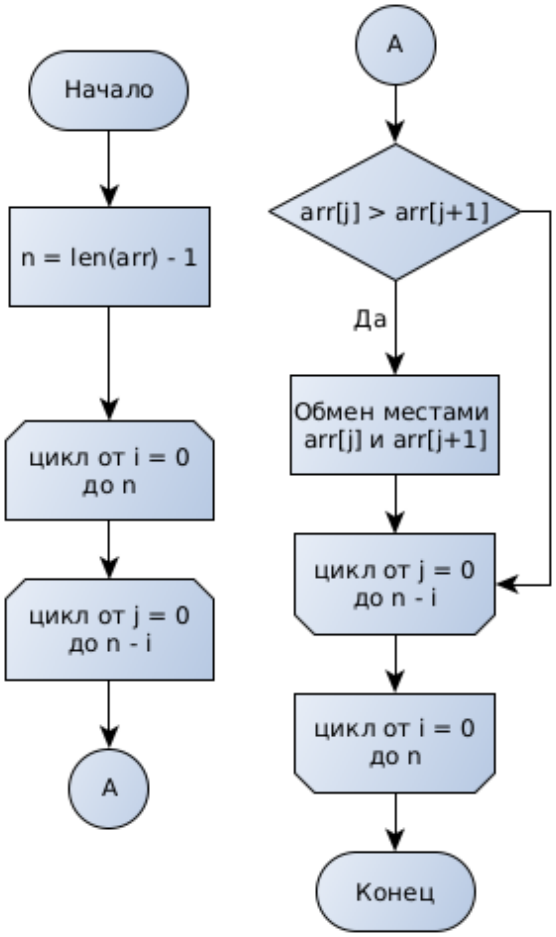

Рисунок 2.1 — Схема алгоритма сортировки пузырьком

На рис. 2.2 представлена схема алгоритма сортировки вставками.

На рис. 2.3 представлена схема алгоритма quicksort.

2.1 Вывод

В данном разделе были рассмотрены схемы (рис. 2.1 - 2.3) алгоритмов сортировки.

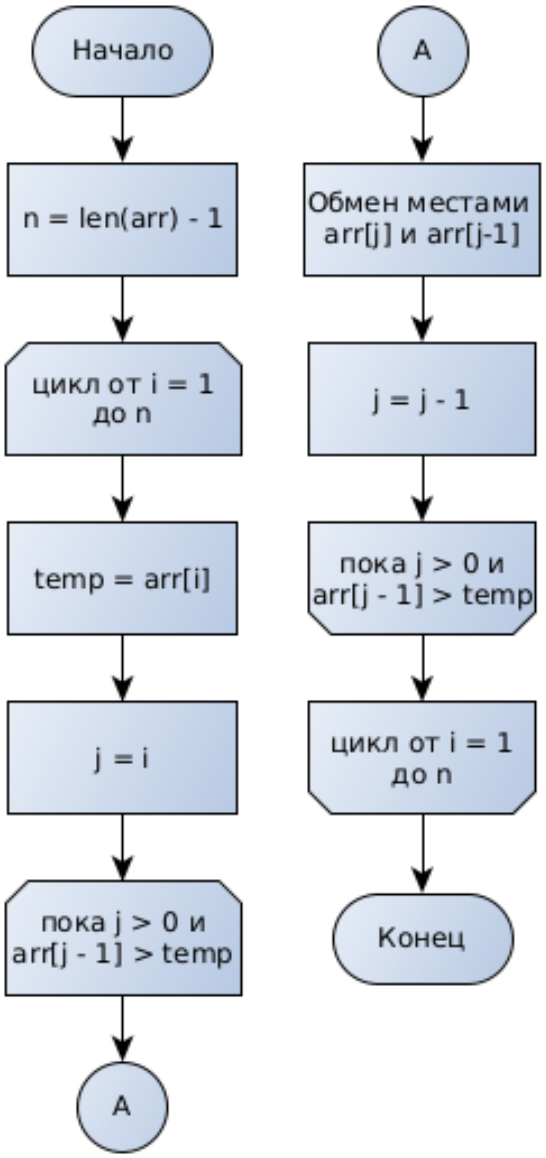

Рисунок 2.2 — Схема алгоритма сортировки вставками

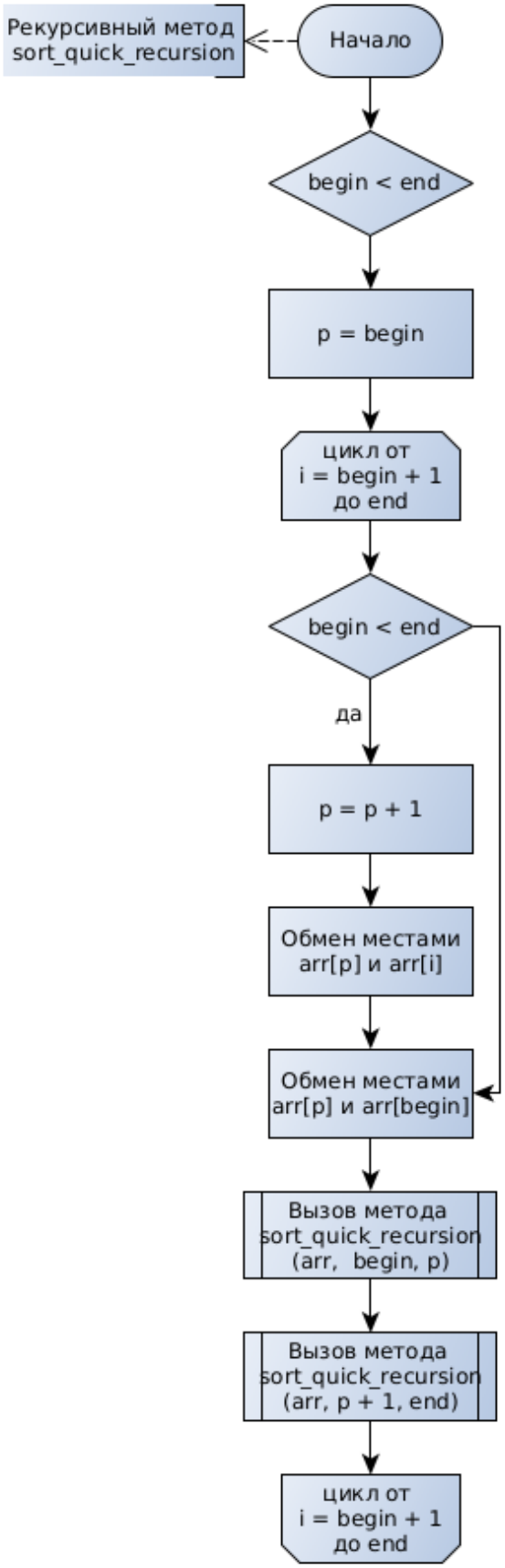

Рисунок 2.3 — Схема алгоритма сортировки quicksort

3 Технологический раздел

3.1 Выбор ЯП

В данной лабораторной работе использовался язык программирования - python [1]. Я знакома с данным языком. Поэтому он был выбран. В качестве среды разработки я использовала Visual Studio Code [2], т.к. считаю его достаточно удобным и легким. Visual Studio Code подходит не только для Windows [3], но и для Linux [4], это еще одна причина, по которой я выбрала VS code, т.к. у меня установлена ОС Ubuntu 18.04.4 [5].

3.2 Требования к программному обеспечению

Входными данными являются:

а) размерность массива n;
б) n элементов массива.

На выходе получается отсортированный массив.

3.3 Сведения о модулях программы

Данная программа разбита на модули:

— main.py - Файл, содержащий точку входа в программу. В нем происходит общение с пользователем и вызов алгоритмов;
— sort.py - Файл, содержащий алгоритмы сортировок.
— arr.py - файл, содержащие методы для работы с массивом.

На листингах 3.1-3.5 представлен код программы.

Листинг 3.1 — Главная функция main

```
1  def main():
2      arr = inputArr()
3
4      output("Введенный массив:", GREEN)
5      outputArr(arr)
6
7      arr_bubble = sort_bubble(arr)
8
9      output("Отсортированный массив (Пузырек):", GREEN)
10     outputArr(arr_bubble)
11
12     arr_insert = sort_insert(arr)
13     output("Отсортированный массив (Вставками):", GREEN)
14     outputArr(arr_insert)
15
```

```
16      arr_quick = quick_sort(arr)
17      output("Отсортированный массив (Quick sort):", GREEN)
18      outputArr(arr_quick)
```

Листинг 3.2 — Сортировка пузырьком

```
1       def sort_bubble(arr):
2       n = len(arr) - 1
3
4       for i in range(n):
5           for j in range(n - i):
6               if arr[j] > arr[j + 1]:
7                   arr[j], arr[j+1] = arr[j+1], arr[j]
8
9       return arr
```

Листинг 3.3 — Сортировка вставками

```
1   def sort_insert(arr):
2       n = len(arr)
3
4       for i in range(1, n):
5           temp = arr[i]
6           j = i
7           while j > 0 and arr[j - 1] > temp:
8               arr[j] = arr[j - 1]
9               j -= 1
10          arr[j] = temp
11
12      return arr
```

Листинг 3.4 — QuickSort

```
1   def sort_quick_recursion(arr, begin, end):
2       if begin < end:
3           # Опорная точка. pivot.
4           p = begin
5           for i in range(begin + 1, end):
6               if arr[i] < arr[begin]:
7                   p += 1
8                   arr[p], arr[i] = arr[i], arr[p]
9           arr[begin], arr[p] = arr[p], arr[begin]
10
11          sort_quick_recursion(arr,  begin, p)
12          sort_quick_recursion(arr, p + 1, end)
13
14      return arr
15
```

```
16 def quick_sort(arr):
17     return sort_quick_recursion(arr, 0, len(arr) - 1)
```

Листинг 3.5 — Методы для работы с массивом

```
1  def inputArr():
2      n = int(input("Введите n: "))
3      arr = list()
4
5      if (not n):
6          return
7      for _ in range(n):
8          arr.append(int(input("Введите элемент: ")))
9
10     return arr
11
12
13 def outputArr(arr):
14     for i in range(len(arr)):
15         print(arr[i], end=" ")
16     print()
```

3.4 Тестирование

В данном разделе будет приведена таблица с тестами (таблица 3.1).

Таблица 3.1 — Таблица тестов

Массив	Результат	Результат
5 4 3 2 1	1 2 3 4 5	Ответ верный
1 2 3 4 5	1 2 3 4 5	Ответ верный
1 -1 2 0	-1 0 1 2	Ответ верный
1 -1 2 0	-1 0 1 2	Ответ верный
-1 -2 -3	-3 -2 -1	Ответ верный
1 1 1 1	1 1 1 1	Ответ верный
0	0	Ответ верный
		Ответ верный

Все тесты пройдены.

3.5 Вывод

В данном разделе был выбран и обоснован язык программирования. Были разобраны листинги 3.1-3.5 с кодом программы. А также приведены тесты (таблица 3.1).

4 Экспериментальная часть

В данном разделе будет произведено сравнение вышеизложенных алгоритмов.

4.1 Временные характеристики

Для сравнения возьмем массивы размерностью [100, 200, 300,. . . ,1000]. Так как подсчет сортировки массива считается короткой задачей, воспользуемся усреднением массового эксперимента. Для этого сложим результат работы алгоритма n раз (n >= 10), после чего поделим на n. Тем самым получим достаточно точные характеристики времени. Сравнение произведем при n = 100. Результаты можно увидеть на рис. 4.1 - 4.3.

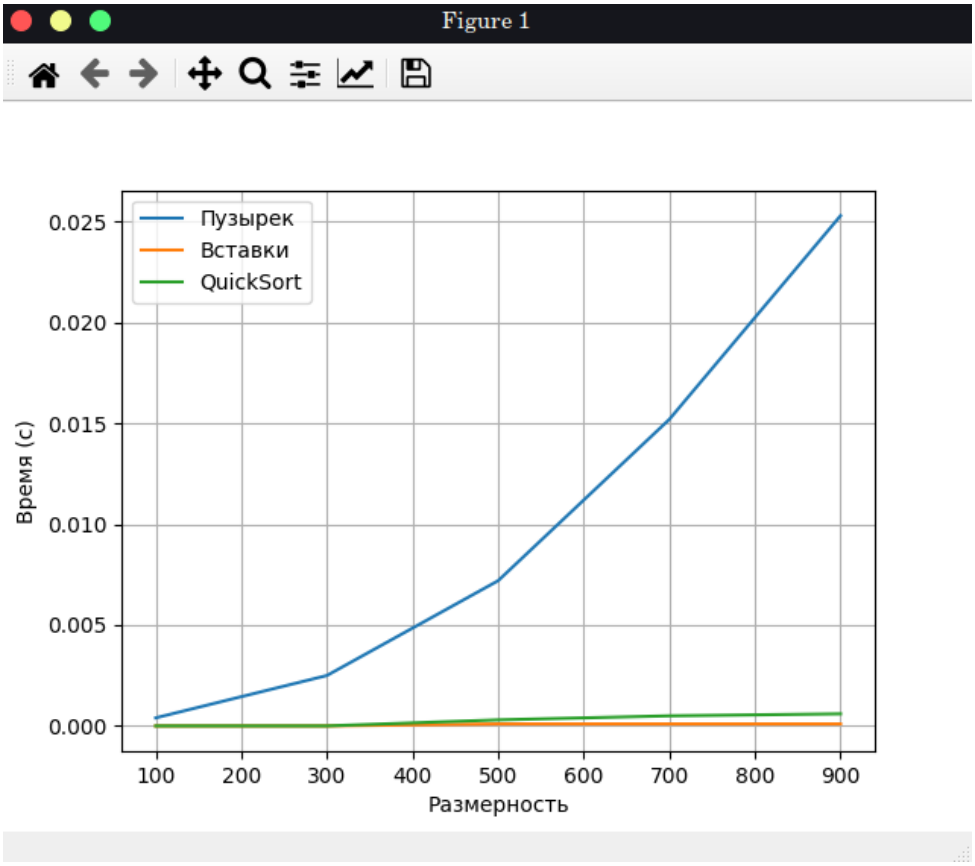

Рисунок 4.1 — Время работы алгоритмов на лучших данных.

4.2 Сравнительный анализ алгоритмов

Введем модель вычислений трудоемкости алгоритма. Пусть трудоемкость 1 у следующих базовых операций: +, -, *, /, %, =, ==, !=, <, <=, >, >=, []. Трудоемкость цикла: fцикла = finиц + fсравн + Nитер ∗ (fтела + finкрем + fсравн).Трудоемкость условного перехода 1.

Алгоритм сортировки пузырьком обладает трудоемкостью 4.1.

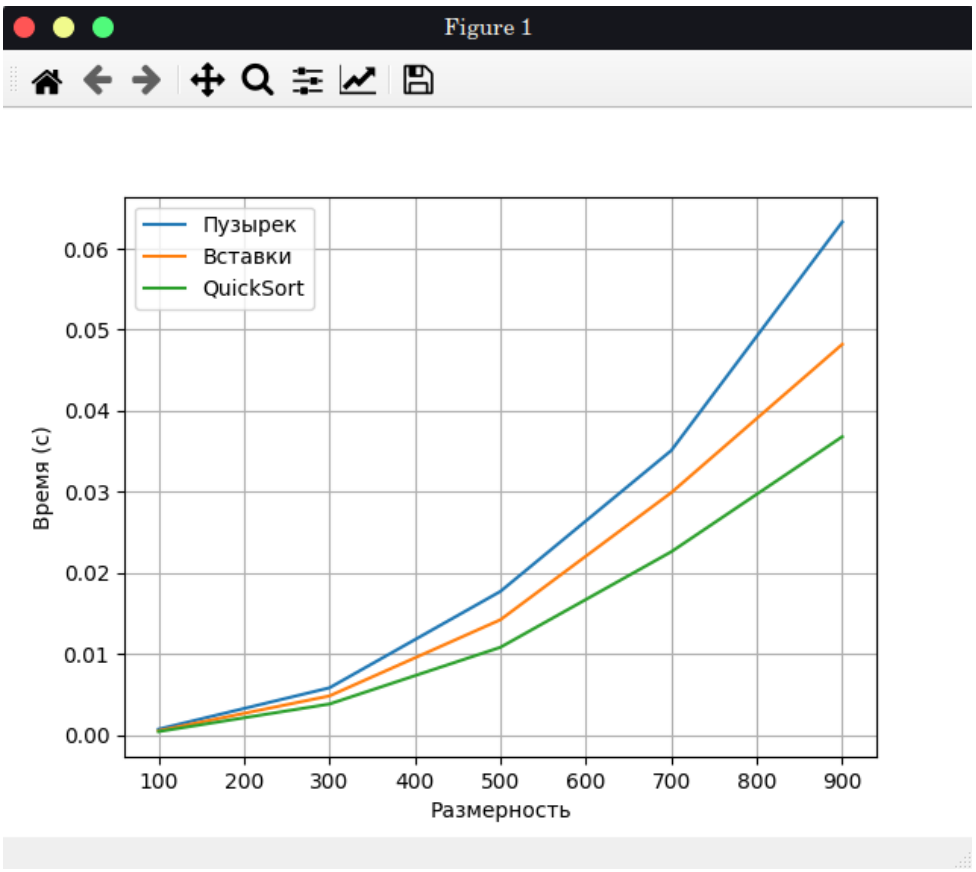

Рисунок 4.2 — Время работы алгоритмов на худших данных.

$$N^2 * \begin{bmatrix} 4 \text{ л.с.} \\ 8.5 \text{ х.с} \end{bmatrix} + N * \begin{bmatrix} 3 \text{ л.с.} \\ \text{-1.5 х.с} \end{bmatrix} - 4 \tag{4.1}$$

Трудоемкость квадратичная от размера массива.

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

Быстрая сортировка в лучшем случае: $(N * log(N))$. В худшем случае: (N^2).

4.3 Вывод

Все алгоритмы в худшем случае обладают квадратичной сложностью. А в лучшем случае меньше всего сложность у алгоритма сортировки вставками.

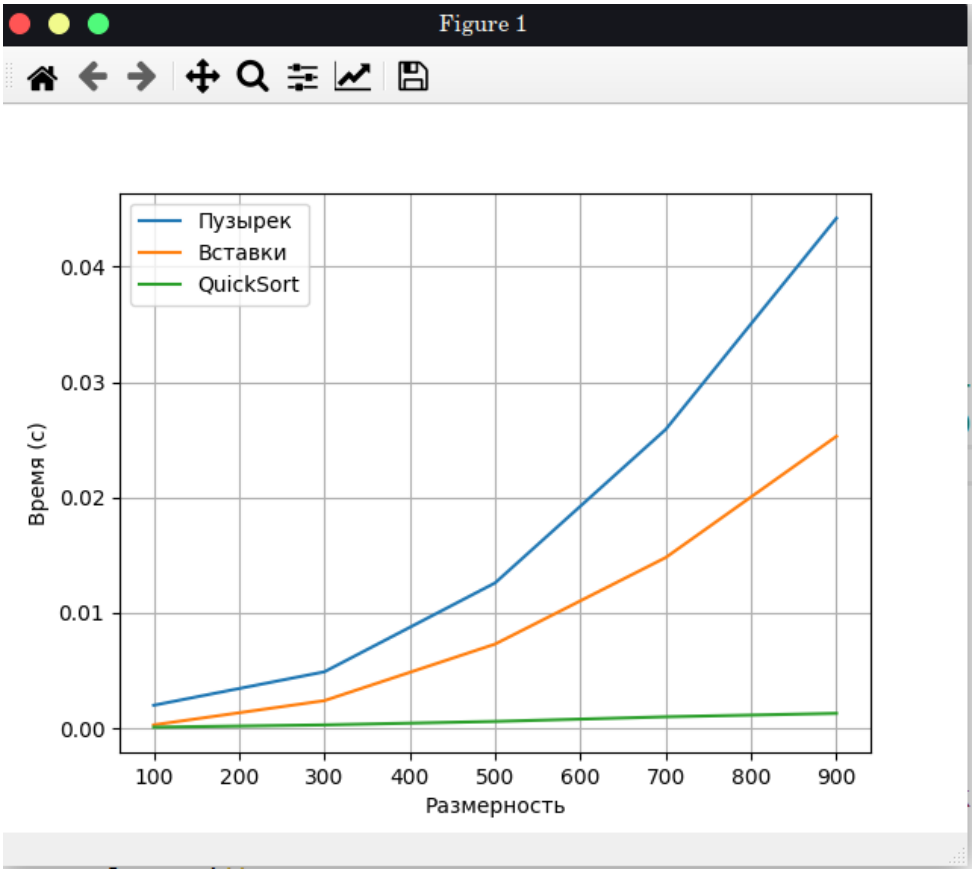

Рисунок 4.3 — Время работы алгоритмов на рандомных данных.

Заключение

В данной работе было рассмотрено три алгоритма сортировки: вставками, пузырек и быстрая сортировка. Был описан и реализован каждый каждый алгоритм (листинги 3.1-3.5). Также были показаны схемы работы алгоритмов (рис. 4.1 - 4.3) Был выбран и обоснован язык программирования. А также приведены тесты (таблица 3.1) и сравнительный анализ алгоритмов.

В рамках выполнения работы решены следующие задачи.

а) Изучены 3 алгоритма сортировки.
б) Реализованы изученные алгоритмы.
в) Получены практические навыки.
г) Произведен сравнительный анализ реализаций алгоритмов сортировки.
д) Экспериментально подтверждены различия во временной эффективности.

Список использованных источников

1. *Майкл, Доусон.* Python Programming for the Absolute Beginner, 3rd Edition / Доусон Майкл. — Прогресс книга, 2019. — P. 416.

2. Visual Studio Code. — Microsoft, 2005. https://code.visualstudio.com/.

3. Windows. — Microsoft, 1985. https://www.microsoft.com/ru-ru/windows.

4. Linux. — 1991. https://www.linux.org.ru/.

5. Ubuntu 18.04. — 2018. https://releases.ubuntu.com/18.04/.