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»

Лабораторная работа №3
по дисциплине "Анализ Алгоритмов"

Тема Алгоритмы сортировки

Студент Артемьев И. О.

Группа ИУ7-53Б

Преподаватель Волкова Л. Л.

Москва

2021 г.

СОДЕРЖАНИЕ

Введение **4**

1 Аналитическая часть **5**
1.1 Сортировка пузырьком 5
1.2 Сортировка вставками 5
1.3 Сортировка выбором 5
1.4 Вывод 6

2 Конструкторская часть **7**
2.1 Требования к ПО 7
2.2 Трудоемкость алгоритмов 7
2.2.1 Модель вычислений 7
2.3 Расчет трудоемкости 8
2.3.1 Вычисление трудоёмкости алгоритма сортировки пузырьком 8
2.3.2 Вычисление трудоёмкости алгоритма сортировки вставками 8
2.3.3 Вычисление трудоёмкости алгоритма сортировки выбором 9
2.4 Схемы алгоритмов 9
2.5 Описание используемых типов данных 13
2.6 Описание выделенных классов эквивалентности 13
2.7 Структура ПО 13
2.8 Вывод 13

3 Технологическая часть **14**
3.1 Средства реализации 14
3.2 Листинги 14
3.3 Тестирование 16
3.4 Вывод 16

4 Исследовательская часть **17**
4.1 Технические характеристики 17
4.2 Пример работы программы 17
4.3 Сравнительный анализ на основе замеров времени работы алгоритмов 18

4.4 Вывод 21

Заключение **22**

Список литературы **23**

Введение

В данной лабораторной работе будут рассмотрены различные методы сортировок.

Алгоритмы сортировки имеют широкое практическое применение. Сортировки используются в большом спектре задач, включая обработку коммерческих, сейсмических, космических данных. Часто сортировка является просто вспомогательной операцией для упорядочивания данных, упрощения последующих алгебраических действий над данными.

Сортировка применяется во многих областях программирования, например, базы данных или математические программы. Упорядоченные объекты содержатся в телефонных книгах, ведомостях налогов, в библиотеках, в оглавлениях, в словарях.

Во многих вычислительных системах на сортировку уходит больше половины машинного времени. Исходя из этого, можно заключить, что либо сортировка имеет много важных применений, либо ею часто пользуются без нужды, либо применяются в основном неэффективные алгоритмы сортировки.

В настоящее время, в связи с экспоненциально возросшими объемами данных, вопрос эффективной сортировки данных снова стал актуальным.

В настоящее время в сети Интернет можно найти результаты производительности алгоритмов сортировки для ряда ведущих центров данных. При этом используются различные критерии оценки эффективности.

Целью данной работы является реализация и изучение следующих алгоритмов:

- сортировка пузырьком;
- сортировка вставками;
- сортировка выбором.

Для достижения данной цели необходимо решить следующие задачи:

- рассмотреть и изучить сортировки пузырьком, вставками и выбором;
- расчитать их трудоемкость;
- реализовать каждую из этих сортировок;
- сравнить их временные характеристики экспериментально.

1 Аналитическая часть

В данной главе будут рассмотрены алгоритмы сортировки.

1.1 Сортировка пузырьком

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный (возрастание, в случае сортировки по убыванию, и наоборот), выполняется обмен элементов. Проходы по массиву повторяются $N - 1$ раз, но есть модифицированная версия, где если окажется, что обмены больше не нужны, значит проходы прекращаются. При каждом проходе алгоритма по внутреннему циклу очередной наибольший элемент массива ставится на свое место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент массива перемещается на одну позицию к началу массива.

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.

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

1.3 Сортировка выбором

Шаги алгоритмы:

1. Находится номер минимального значения в текущем списке.

2. Производится обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции).
3. Сортируется хвост списка, исключиая при этом из рассмотрения уже отсортированные элементы.

Для реализации устойчивости алгоритма необходимо в пункте 2. минимальный элемент непосредственно вставлять в первую неотсортированную позицию, не меняя порядок остальных элементов.

1.4 Вывод

В данном разделе были рассмотрены алгоритмы сортировок пузырьком, выбором и вставками. Полученных знаний достаточно для разработки выбранных алгоритмов.

В качестве входных данных в программу будет подаваться массив целых чисел, на выходных данных будет отсортированный массив. Ограничениями для работы ПО является то, что входной массив состоит из целых чисел.

Реализуемое ПО будет работать в пользовательском режиме (вывод отсортированного массива тремя способами), а также в экспериментальном (проведение замеров времени выполнения алгоритмов).

2 Конструкторская часть

В данном разделе будут представлены схемы алгоритмов и рассчитана их трудоемкость, а также требования к программному обеспечению (далее – ПО).

2.1 Требования к ПО

Требования к вводу

1. На вход подается массив сравнимых элементов.
2. На выходе — отсортированный массив в заданном порядке, сортировка производится на месте, то есть сортируется тот же самый массив и он же возвращается.
3. Алгоритм сортирует целочисленный тип данных t $-2^{64} < t < 2^{64} - 1$.

2.2 Трудоемкость алгоритмов

В данном разделе будет введена модель вычислений и рассмотрены трудоемкости алгоритмов сортировки выбором, вставки и пузырьком.

2.2.1 Модель вычислений

Для последующего вычисления трудоемкости необходимо ввести модель вычислений [1].

1. $+, -, /, \%, =, \neq, <, >, \leq, \geq, [], *$ — трудоемкость 1.
2. Трудоемкость оператора выбора *if* условие *then A else B* рассчитывается, как:

$$f_{if} = f_{\text{условия}} + \begin{cases} f_A & \text{если условие выполняется,} \\ f_B & \text{иначе.} \end{cases} \quad (1)$$

3. Трудоемкость цикла расчитывается, как:

$$f_{for} = f_{\text{инициализации}} + f_{\text{сравнения}} + N(f_{\text{тела}} + f_{\text{инициализации}} + f_{\text{сравнения}}). \quad (2)$$

4. Трудоемкость вызова функции равна 0.

2.3 Расчет трудоемкости

В данном разделе будут расчитаны трудоемкости алгоритмов методом пузырька, вставками и выбором.

2.3.1 Вычисление трудоёмкости алгоритма сортировки пузырьком

Лучший случай: массив отсортирован, следовательно не произошло ни одного обмена.

Трудоемкость:

$$T(N) = 2 + \frac{N \cdot (N-1)}{2} \cdot 3 = 1.5N^2 - 1.5N + 2 \tag{3}$$

Худший случай: массив отсортирован в обратном порядке, в каждом случае происходил обмен.

Трудоемкость:

$$T(N) = 2 + \frac{N \cdot (N-1)}{2} \cdot 8 = 4N^2 - 4N + 2 \tag{4}$$

2.3.2 Вычисление трудоёмкости алгоритма сортировки вставками

Лучший случай: массив отсортирован, все внутренние циклы состоят всего из одной итерации.

Трудоемкость:

$$T(N) = 2 + 6N \tag{5}$$

Худший случай: массив отсортирован в обратном порядке. Каждый новый элемент сравнивается со всеми в отсортированной последовательности. Все внутренние циклы будут состоять из j итераций.

Трудоемкость.

$$T(N) = \frac{N \cdot (N-1)}{2} \cdot 10 + N + 2 = 5N^2 - 4N + 2 \tag{6}$$

2.3.3 Вычисление трудоёмкости алгоритма сортировки выбором

Лучший случай: массив отсортирован.

Трудоемкость:

$$T(N) = 2 + \frac{N \cdot (N-1)}{2} \cdot 8 + 4N = 4N^2 + 2 \tag{7}$$

Худший случай: массив отсортирован в обратном порядке, в каждом случае происходил обмен.

Трудоемкость:

$$T(N) = \frac{N \cdot (N-1)}{2} \cdot 5 + 5 \cdot N + 3 = 2.5N^2 + 2.5N + 3 \tag{8}$$

2.4 Схемы алгоритмов

На рисунках 1 - 3 изображены схемы реализаций алгоритмов сортировки.

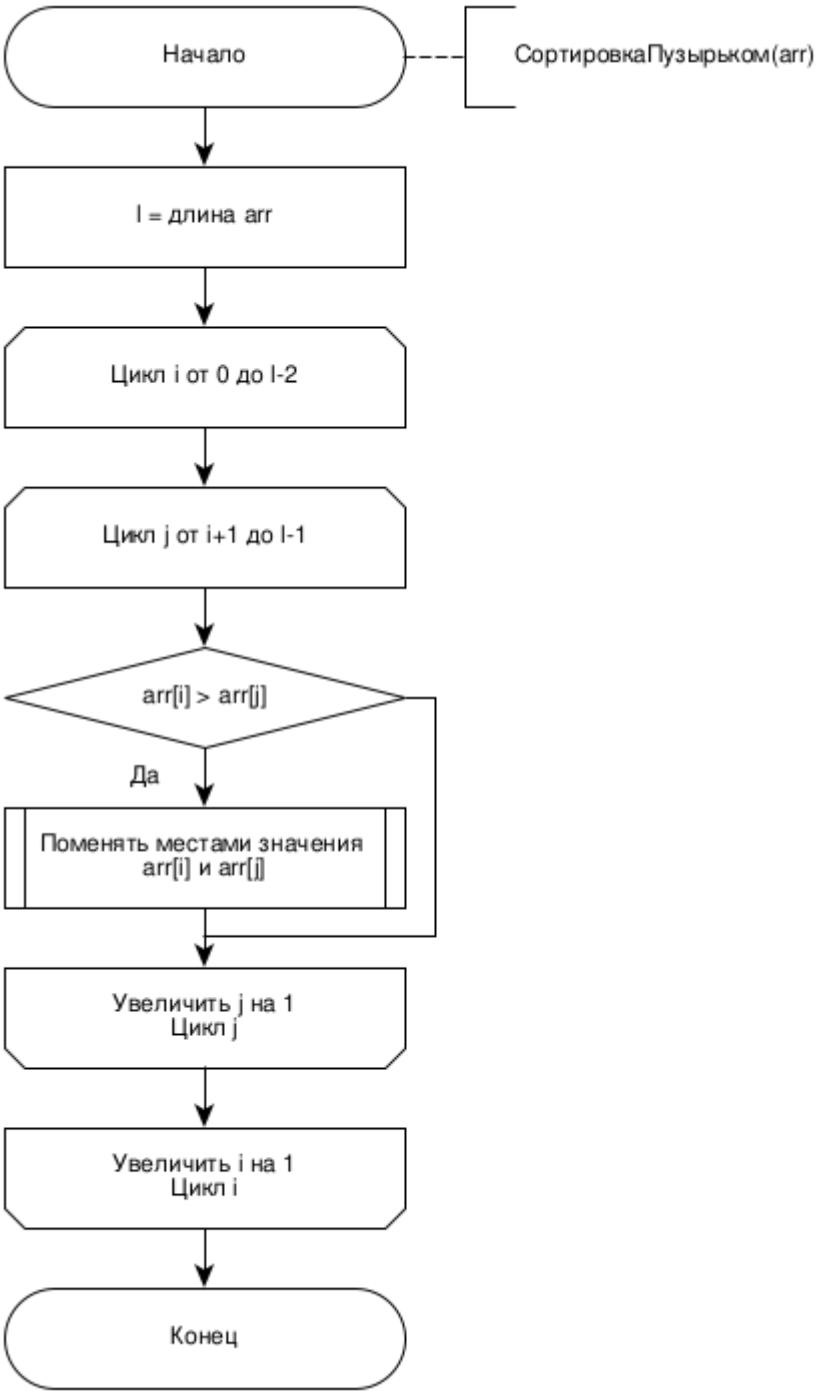

Рис. 1 — Схема алгоритма сортировки пузырьком

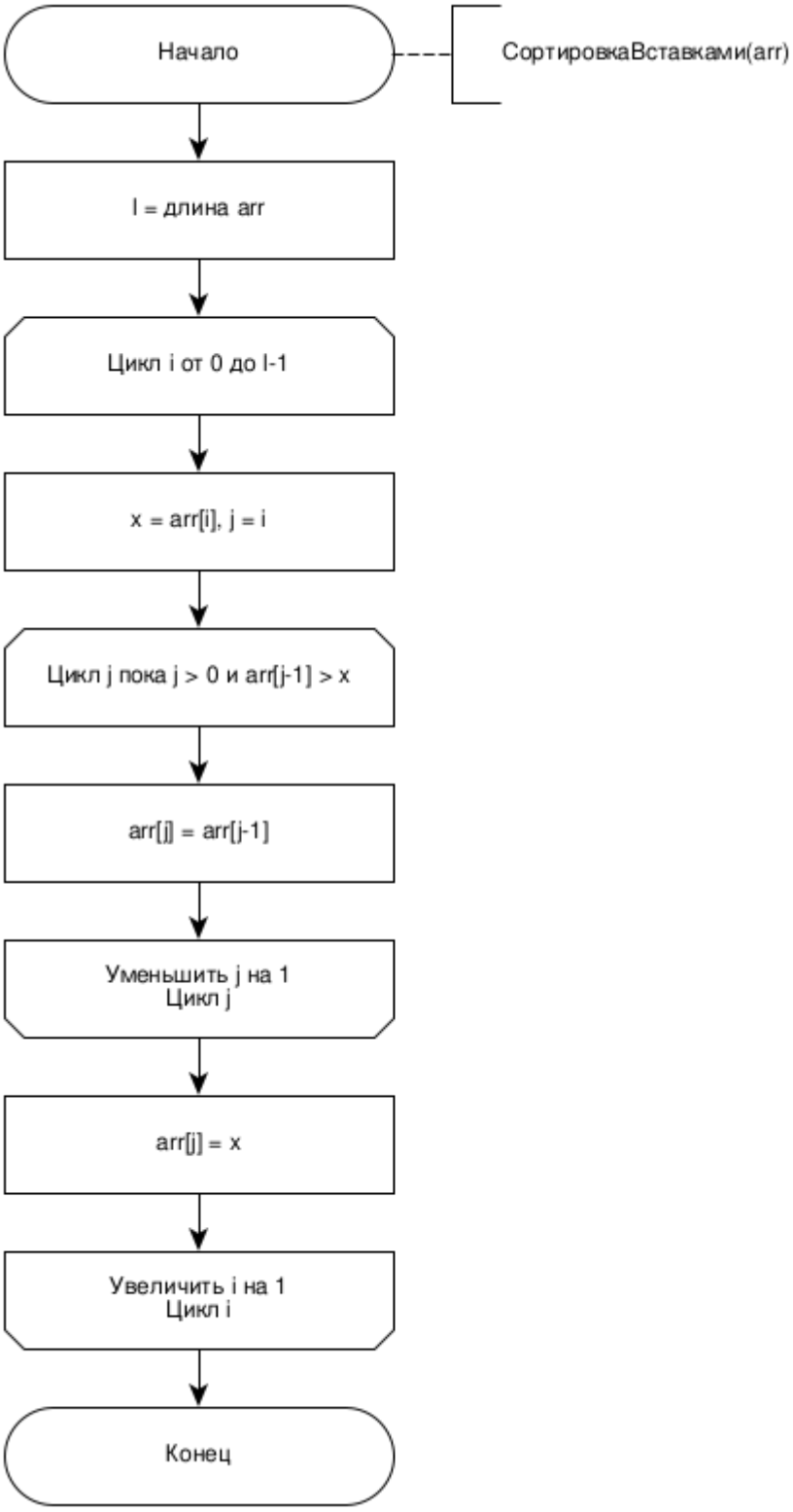

Рис. 2 — Схема алгоритма сортировки вставками

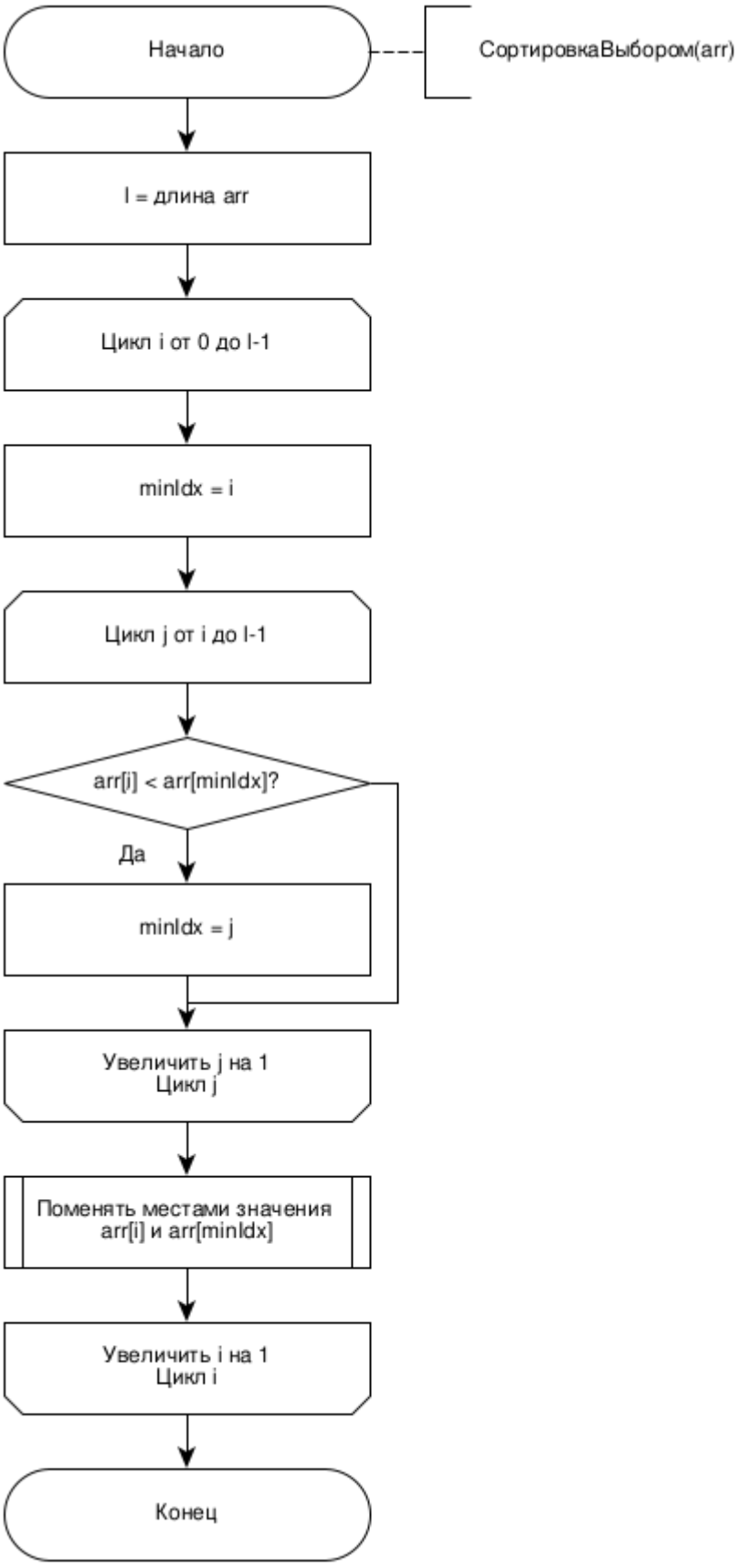

Рис. 3 — Схема алгоритма сортировки выбором

2.5 Описание используемых типов данных

При описании алгоритмов будут использованы следующие структуры данных:

1. массив сравнимых элементов типа int;
2. длина массива - целое число типа int.

2.6 Описание выделенных классов эквивалентности

Для дальнейшего тестирования работы программы были выделены следующие классы эквивалентности:

1. отсортированный массив;
2. обратно отсортированный массив;
3. массив случайных значений.

2.7 Структура ПО

ПО будет состоять из пяти модулей:

1. main.py - модуль, вызывающий загрузку меню (является модулем запуска);
2. menu.py - модуль, содержащий меню пользователя;
3. list_form.py - модуль, содержащий функции для генерации списков;
4. sorts.py модуль, содержащий сортировки;
5. plot.py - модуль, содержащий функции для построения графиков сложности сортировок.

2.8 Вывод

По аналитическим расчетам можно сделать вывод, что в лучшем случае быстрее всего отработает алгоритм вставками. А в худшем случае быстрее всего отработает алгоритм пузырька.

3 Технологическая часть

В данном разделе будут представлены средства реализации и листинги.

3.1 Средства реализации

Для реализации программ я выбрал язык программирования Python, так как я очень хорошо знаком с этим языком и пишу на нем давно, также этот язык является удобным, безопасным и в нем присутствуют инструменты замера времени.

3.2 Листинги

В листингах 1 - 3 представлены реализации трех алгоритмов сортировки.

Листинг 1: Функция сортировки массива методом пузырька

```
1  def bubble(a: list) -> list:
2      l = len(a)
3      ret_a = a.copy()
4
5      for i in range(l - 1):
6          for j in range(i + 1, l):
7              if ret_a[i] > ret_a[j]:
8                  ret_a[i], ret_a[j] = ret_a[j], ret_a[i]
9
10     return ret_a
```

Листинг 2: Функция сортировки массива методом вставок

```
1  def insertion(a: list) -> list:
2      l = len(a)
3      ret_a = a.copy()
4
5      for i in range(l):
6          x = ret_a[i]
7          j = i
8
```

```
 9            while j > 0 and ret_a[j - 1] > x:
10                ret_a[j] = ret_a[j - 1]
11                j -= 1
12
13            ret_a[j] = x
14
15        return ret_a
```

Листинг 3: Функция сортировки массива методом выбора

```
 1 def selection(a: list) -> list:
 2     l = len(a)
 3     ret_a = a.copy()
 4
 5     for i in range(l):
 6         minIdx = i
 7
 8         for j in range(i, l):
 9             if ret_a[j] < ret_a[minIdx]:
10                 minIdx = j
11
12         ret_a[minIdx], ret_a[i] = ret_a[i], ret_a[minIdx]
13
14     return ret_a
```

3.3 Тестирование

В таблице 1 приведены результаты тестирования.

Таблица 1 — Результаты функционального тестирования.

Вход	Результат	Ожидаемый результат
1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
4, 3, 2, 1	1, 2, 3, 4	1, 2, 3, 4
1, 3, 2, 4	1, 2, 3, 4	1, 2, 3, 4
1	1	1
пустой	пустой	пустой

3.4 Вывод

В данном разделе были представлены требования к программному обеспечению, выбор языка программирования, листинги реализаций алгоритмов и результаты тестирования.

4 Исследовательская часть

В данном разделе будут представлены замеры времени работы реализаций алгоритмов.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование представлены далее.

— Операционная система: macOS Catalina версия 10.15.6;

— Память: 8 GB 1600 MHz DDR3;

— Процессор: 1,8 GHz 2-ядерный процессор Intel Core i5.

Тестирование проводилось на ноутбуке, включенном в сеть электропитания. Во время Тестирования ноутбук был нагружен только встроенными приложениями окружения рабочего стола.

4.2 Пример работы программы

Демонстрация работы программы приведена на рисунке 4.

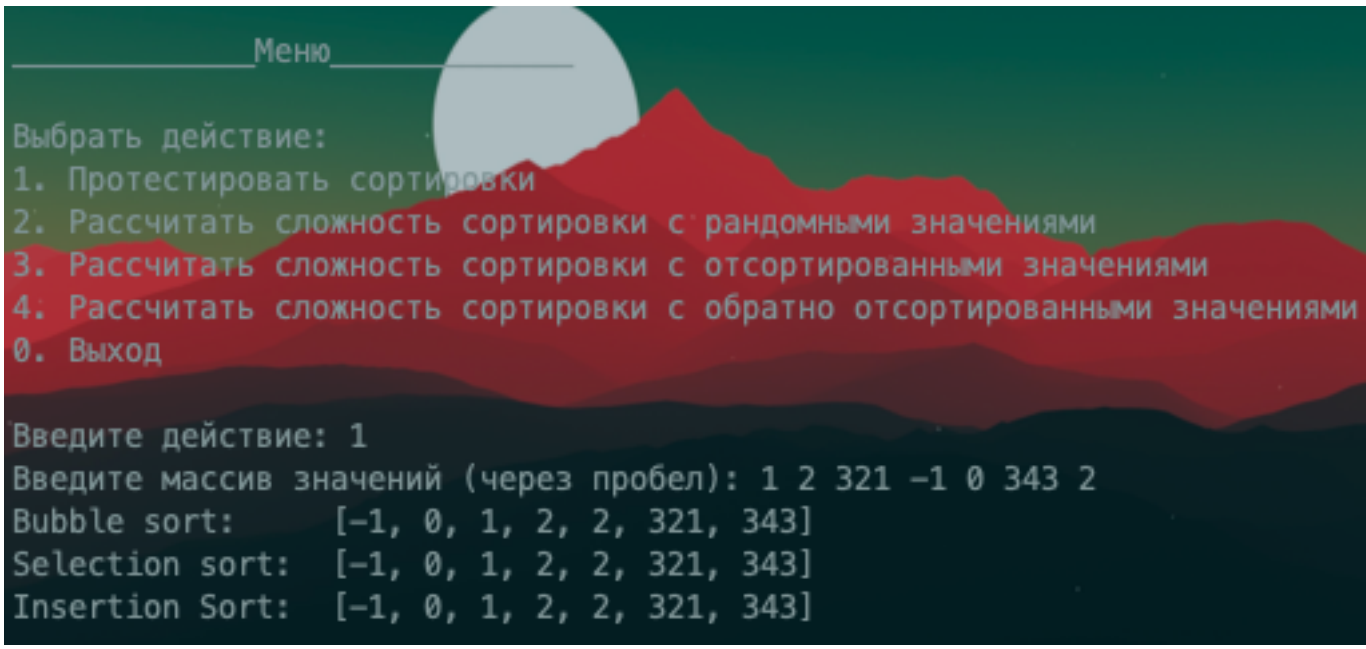

Рис. 4 — Пример работы программы.

4.3 Сравнительный анализ на основе замеров времени работы алгоритмов

Был проведен замер времени работы каждого из алгоритмов с помощью функции process_time(), которая находится в модуле time языка Python. Для сравнения алгоритмов сортировки используются массивы размерностью [10, 100, 500, 1000]. Замеры времени усреднялись, для каждого алгоритма сортировки проводилось по 30 итераций.

Таблица 2 — Результаты времени выполнения алгоритмов сортировок на отсортированном массиве (в секундах)

Размер массива	Пузырек	Вставки	Выбором
10	1,73E-05	2,71E-06	1,37E-05
100	1,43E-04	1,76E-05	9,38E-04
500	1,86E-02	1,56E-04	1,82E-02
1000	1,18E-01	3,68E-04	8,36E-02

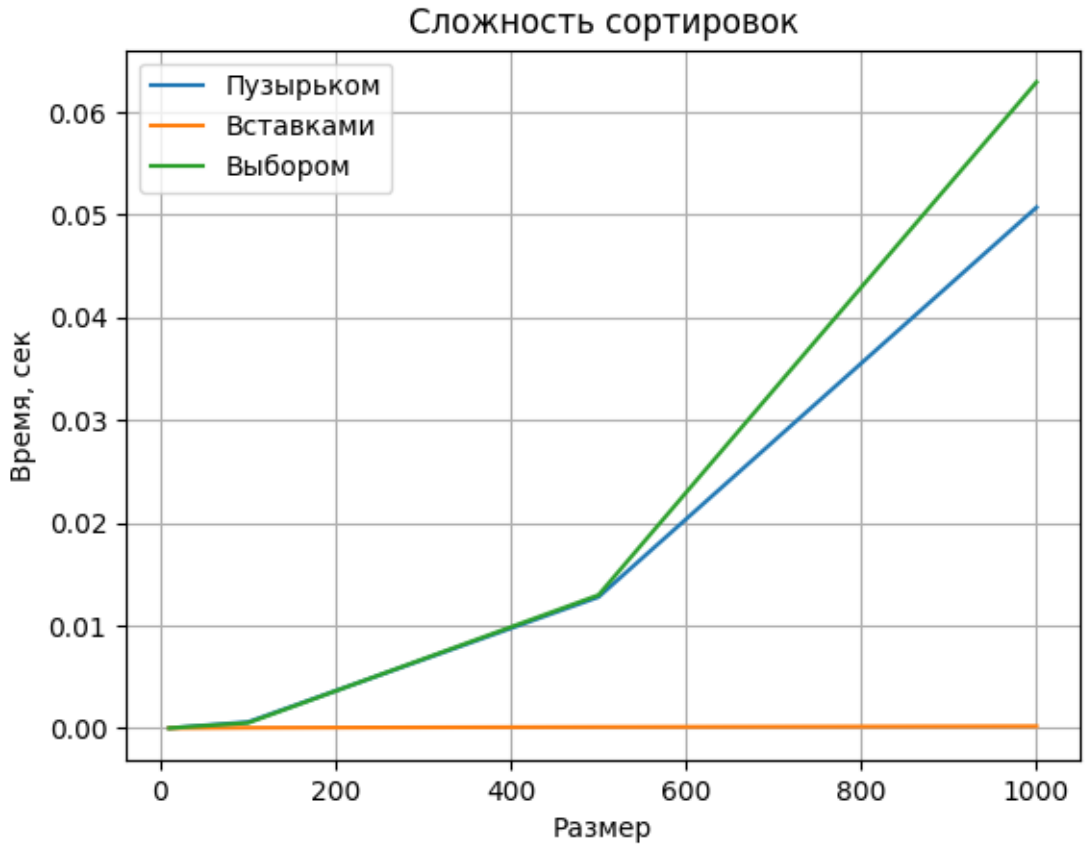

Рис. 5 — Время работы алгоритмов для отсортированного массива

Таблица 3 — Результаты времени выполнения алгоритмов сортировок на отсортированном в обратном порядке массиве (в секундах)

Размер массива	Пузырек	Вставки	Выбором
10	2,86E-05	2,47E-05	2,14E-05
100	1,81E-03	2,00E-03	8,75E-04
500	4,04E-02	5,93E-02	1,90E-02
1000	1,87E-01	2,27E-01	1,01E-01

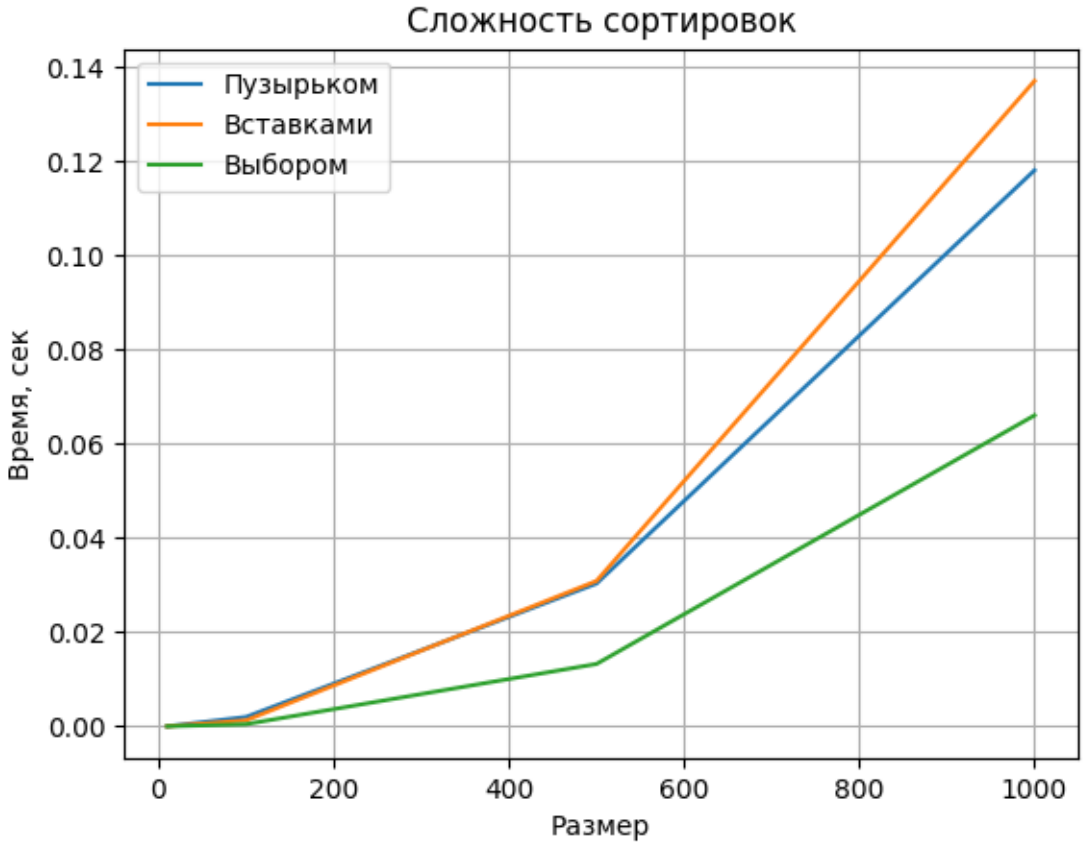

Рис. 6 — Время работы алгоритмов для отсортированного в обратном порядке массива

Таблица 4 — Результаты времени выполнения алгоритмов сортировок на случайном массиве (в секундах)

Размер массива	Пузырек	Вставки	Выбором
10	2,02E-05	8,53E-06	3,11E-05
100	1,01E-03	1,32E-03	8,29E-04
500	2,79E-02	2,20E-02	2,35E-02
1000	1,38E-01	1,09E-01	7,46E-02

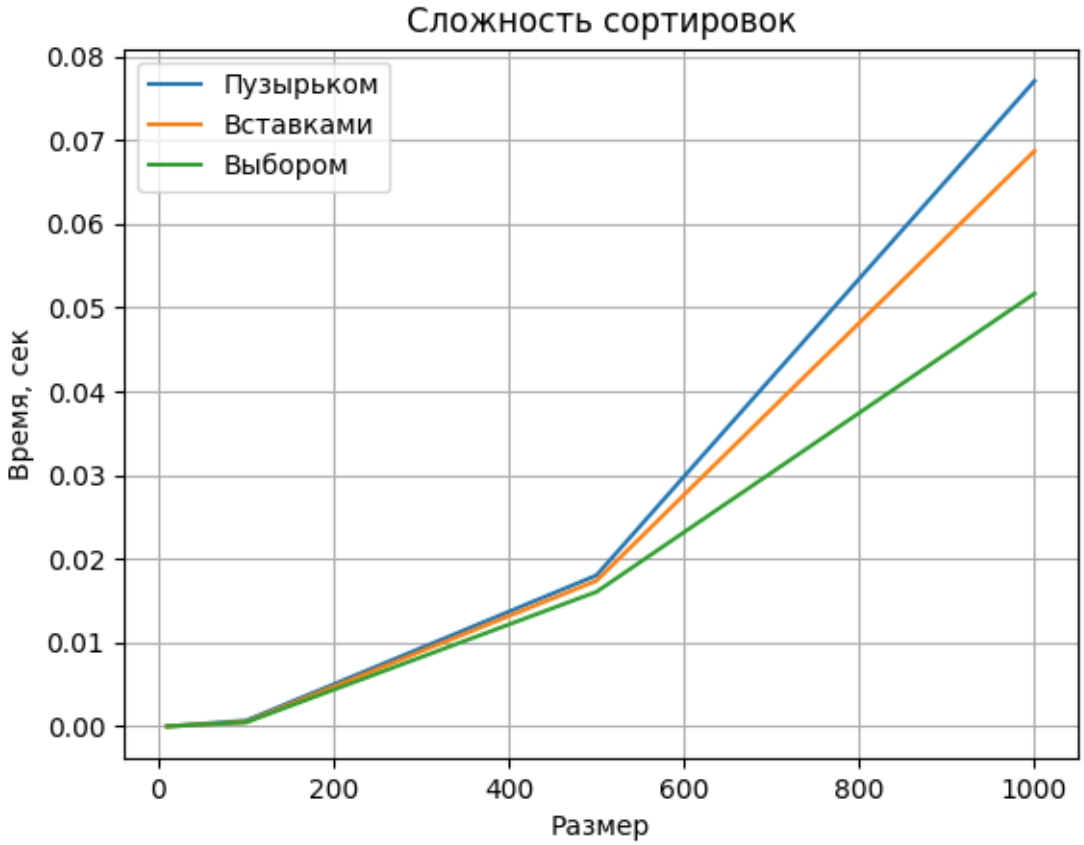

Рис. 7 — Время работы алгоритмов для случайного массива

4.4 Вывод

В данном разделе было произведено сравнение количества затраченного времени вышеизложенных алгоритмов.

Исходя из полученных результатов, можно сделать вывод, что пузырьковая сортировка приблизительно равно работает с сортировкой вставками на обратно отсортированном массиве и на массиве случайных чисел ±0.2 сек на 1000 элементах, а сортировка выбором на этих массивах является самой быстрой. Также сортировка вставками на отсортированном массиве является самой быстрой и сортирует практически на 0 сек.

Заключение

В рамках данной лабораторной работы была достигнута цель и выполнены следующие задачи:

— были изучены и реализованы три алгоритма сортировки: методом пузырька, вставок и выбором;

— был проведен сравнительный анализ трудоемкости алгоритмов на основе теоретических расчетов и выбранной модели вычислений;

— был проведен сравнительный анализ алгоритмов на основе экспериментальных данных.

Экспериментально были установлены различия в производительности различных алгоритмов сортировки.

Список литературы

1. Welcome to Python [Электронный ресурс] Режим доступа: https://www.python.org/, (дата обращения: 13.10.2021)
2. time - Time access and conversions [Электронный ресурс] Режим доступа: https://docs.python.org/3/library/time.html, (дата обращения: 13.10.2021)
3. Сортировка выбором [Электронный ресурс] Режим доступа: https://kvodo.ru/sortirovka-vyiborom-2.html, (дата обращения: 13.10.2021)
4. Сортировка вставками [Электронный ресурс] Режим доступа: https://habr.com/ru/post/415935/, (дата обращения: 13.10.2021)
5. Сортировка Пузырьком [Электронный ресурс] Режим доступа: https://habr.com/ru/post/204600/, (дата обращения: 13.10.2021)