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 Алгоритмы сортировки

Студент Прохорова Л. А.

Группа ИУ7-53Б

Оценка (баллы) _____

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

Москва — 2020 г.

Введение

Сортировка массива — это процесс распределения всех элементов массива в определенном порядке. Очень часто это бывает полезным. Например, в вашем почтовом ящике электронные письма отображаются в зависимости от времени получения; новые письма считаются более релевантными, чем те, которые вы получили полчаса, час, два или день назад; когда вы переходите в свой список контактов, имена обычно находятся в алфавитном порядке, потому что так легче что-то найти. Все эти случаи включают в себя сортировку данных перед их фактическим выводом.

1 Аналитическая часть

1.1 Детализация задачи

В данной работе я рассмотрю алгоритмы:

- сортировки пузырьком с флагом;
- сортировки вставками;
- сортировки выбором;

Целью данной лабораторной работы является изучение и реализация алгоритмов сортировки, обучение расчету трудоемкости алгоритмов.

Для достижения цели ставятся следующие задачи.

- Изучить алгоритмы сортировки пузырьком с флагом, вставками, выбором.
- Реализовать алгоритмы сортировки пузырьком с флагом, вставками,выбором.
- Дать оценку трудоёмкости в лучшем и худшем случае(для двух алгоритмов сделать вывод трудоёмкости).
- Замерить время работы для лучшего, худшего и произвольного случая.
- Описать и обосновать полученные результаты в отчете о выполненной лабораторной работе, выполненном как расчётно-пояснительная записка к работе.

2 Описание алгоритмов сортировки

Сортировка массива — одна из самых популярных операций над массивом. Алгоритмы реализуют упорядочивание элементов в списке. В слу-

чае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

2.1 Сортировка пузырьком с флагом

Сортировка пузырьком — один из самых известных алгоритмов сортировки.Будем идти по массиву слева направо. Если текущий элемент больше следующего, меняем их местами. Делаем так, пока массив не будет отсортирован. Заметим, что после первой итерации самый большой элемент будет находиться в конце массива, на правильном месте. После двух итераций на правильном месте будут стоять два наибольших элемента, и так далее [2]. Таким образом элементы с большими значениями оказываются в конце списка, а с меньшими остаются в начале.

Этот алгоритм считается учебным и почти не применяется на практике из-за низкой эффективности: он медленно работает на тестах, в которых маленькие элементы (их называют «черепахами») стоят в конце массива. Однако на нём основаны многие другие методы, например, шейкерная сортировка и сортировка расчёской [1].

При сортировке с флагом алгоритм не делает лишних итераций если все элементы стоят на своих местах(за один проход не произошло ни одного обмена).

2.2 Сортировка вставками

Создадим массив, в котором после завершения алгоритма будет лежать ответ. Будем поочередно вставлять элементы из исходного массива так, чтобы элементы в массиве-ответе всегда были отсортированы. Реализовывать алгоритм удобнее по-другому (создавать новый массив и реально что-то вставлять в него относительно сложно): просто сделаем так, чтобы

отсортирован был некоторый префикс исходного массива, вместо вставки будем менять текущий элемент с предыдущим, пока они стоят в неправильном порядке [2].

На рисунке 1 показаны этапы сортировки вставками массива [18, 20, 5, 13, 15].

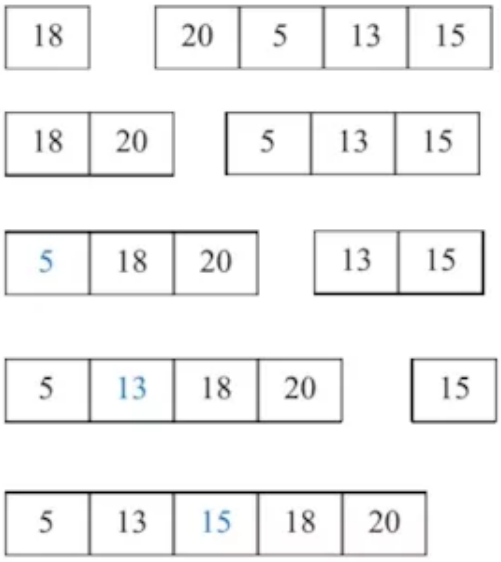

Рисунок 1 – Сортировка вставками.

2.3 Сортировка выбором

Сначала нужно рассмотреть подмножество массива и найти в нём максимум (или минимум). Затем выбранное значение меняют местами со значением первого неотсортированного элемента. Таким образом, после i-ой итерации первые i элементов будут стоять на своих местах. Этот шаг нужно повторять до тех пор, пока в массиве не закончатся неотсортированные подмассивы [1].

Нужно отметить, что эту сортировку можно реализовать двумя способами – сохраняя минимум и его индекс или просто переставляя текущий элемент с рассматриваемым, если они стоят в неправильном порядке [2].

На рисунке 2 показаны этапы сортировки вставками массива [4, 9, 7, 6, 2, 3].

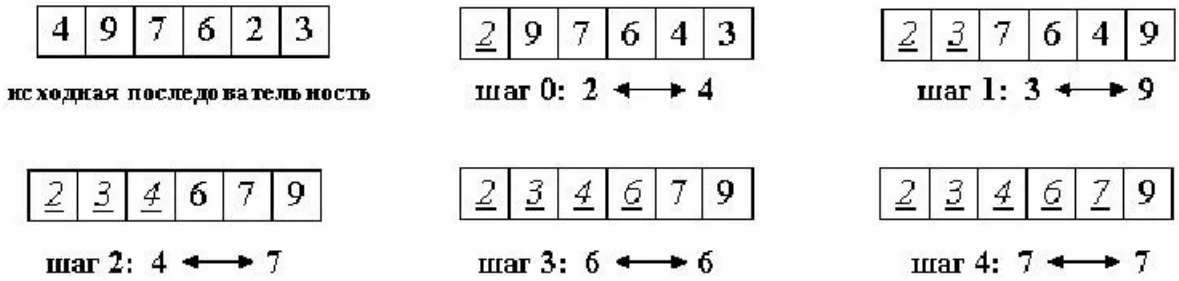

Рисунок 2 – Сортировка выбором.

2.4 Требования к вводу

В разрабатываемом ПО предъявляются следующие требования ко вводу.

1. На вход подаётся массив целых чисел.

2.5 Требования к выводу

Программа выводит массив, отсортированный по неубыванию.

2.6 Требования к программе

Пустой массив - корректный ввод, программа не должна аварийно завершаться.

2.7 Схемы алгоритмов

На рисунках 3,4, 5, 6 показаны схемы алоритмов сортировки.

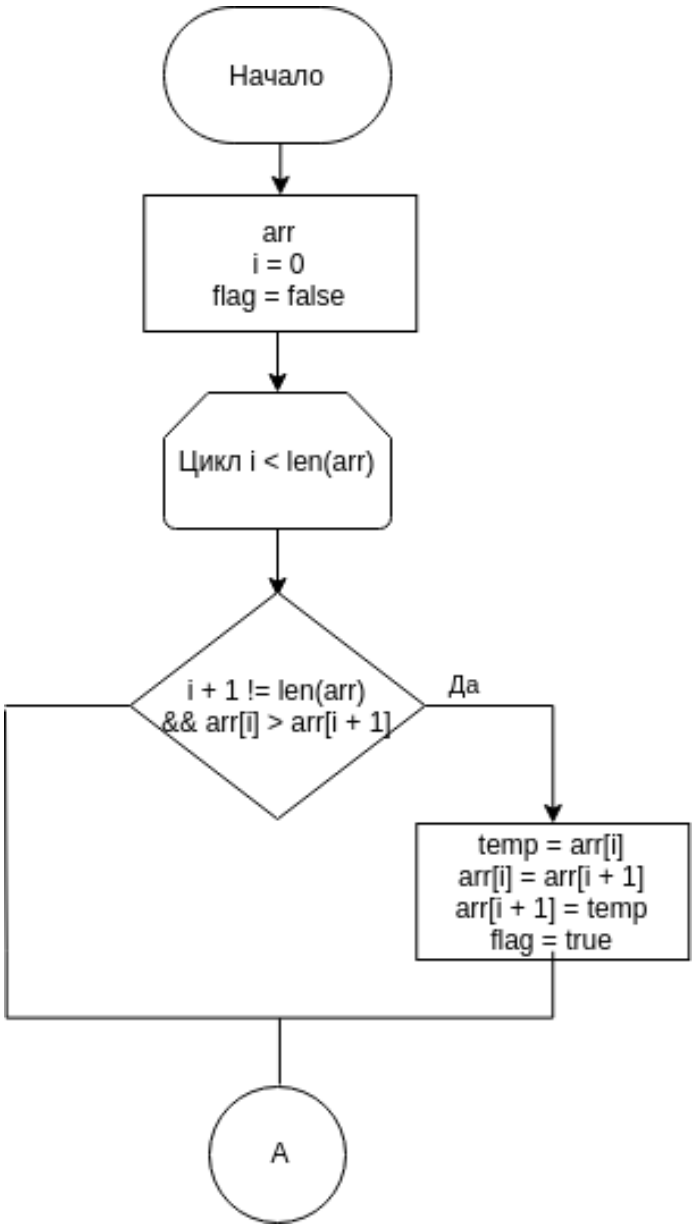

Рисунок 3 – Схема алгоритма сортировки пузырьком с флагом(часть 1).

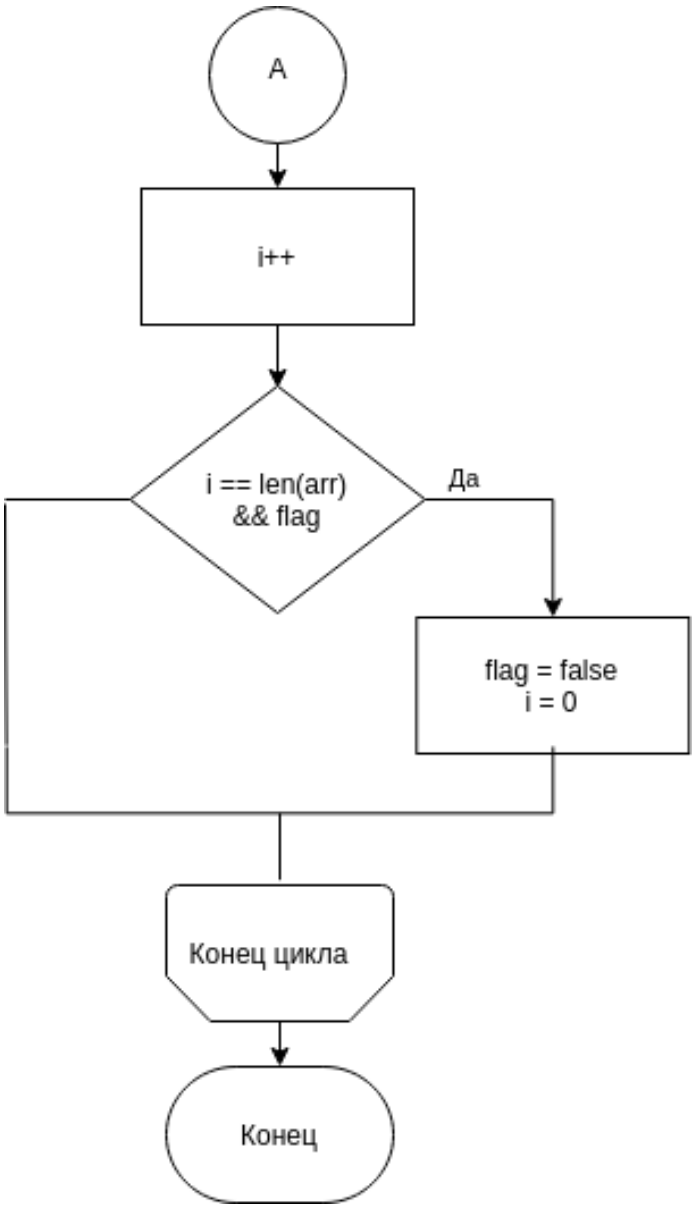

Рисунок 4 – Схема алгоритма сортировки пузырьком с флагом(часть 2).

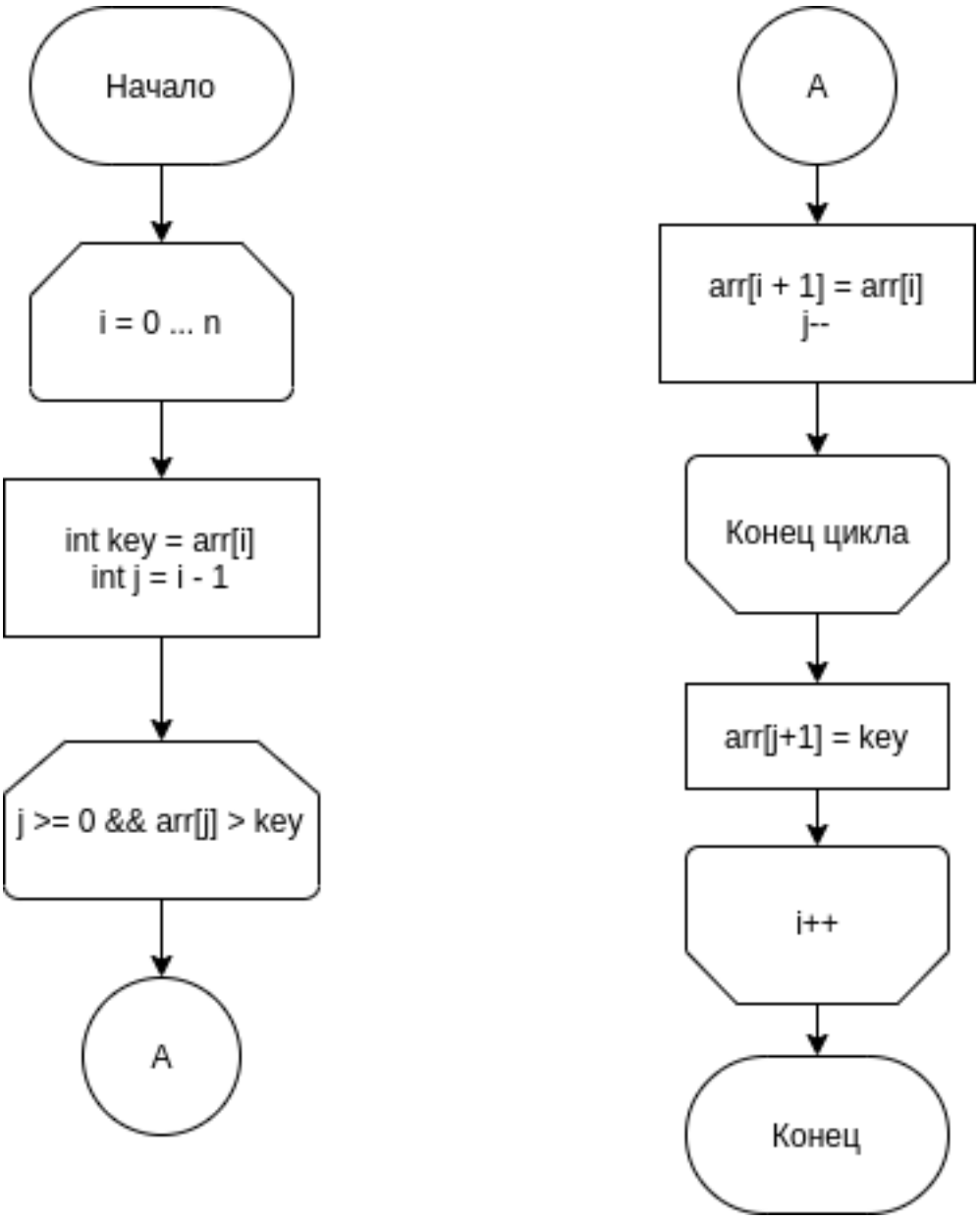

Рисунок 5 – Схема алгоритма сортировки вставками.

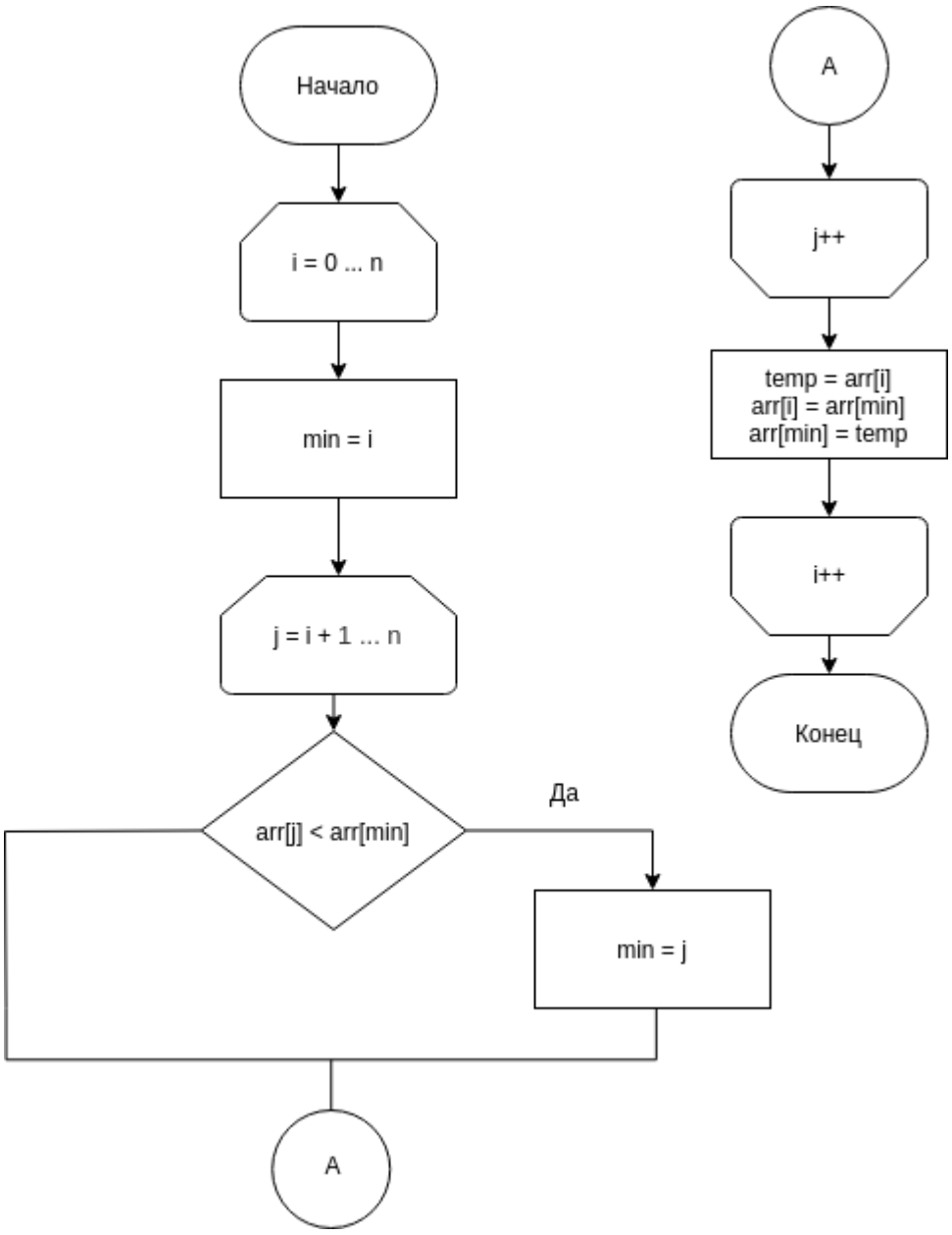

Рисунок 6 – Схема алгоритма сортировки выбором.

3 Трудоёмкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов.

1. Базовые операции стоимостью 1: +, -, *, /, =, ==, <=, >=, !=, +=, [], получение полей класса

2. Оценка трудоемкости цикла: f_цикла = f_инициализации + f_сравнения + N * (f_инкремента + f_сравнения + f_тела.

3. Стоимость условного перехода возьмем за 0, стоимость вычисления условия остаётся.

Вычисление трудоёмкости алгоритма сортировки пузырьком с флагом.

Лучший случай: Массив отсортирован; не произошло ни одного обмена за 1 проход -> выходим из цикла

Трудоемкость:

$$T(N) = 1 + 1 + N * (1 + 7 + 1 + 3) + 1 = 12N + 3 = O(N) \qquad (1)$$

Худший случай: Массив отсортирован в обратном порядке; в каждом случае происходил обмен

Трудоемкость:

$$T(N) = 1 + 1 + N * (N * (1 + 7 + 5 + 1 + 1 + 3) +$$

$$+ 1 + 1) + 1 = \mathrm{N} * (18\mathrm{N} + 2) + 3 = 18\mathrm{N}^2 + 2N + 3 = O(N^2)(2)$$

Вычисление трудоёмкости алгоритма сортировки вставками.

Сортировка вставками

Лучший случай: отсортированный массив. При этом все внутренние циклы состоят всего из одной итерации.

Трудоемкость:

$$T(N) = 2+((N-1)*(2+2+2+3+3)) = 12*(N-1)+2 = 12N-10 = O(N) \qquad (3)$$

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

Трудоемкость:

$$T(N) = 2 + (N-1) * (2+2+2+1+3) + \sum_{1}^{N}(4+1+3) =$$

$$= 2 + 10\text{N} - 10 + 8 * \left(\frac{N*(N+1)}{2}\right) = 4N^2 + 14N + 2 = O(N^2)(4)$$

Сортировка выбором

Трудоёмкость сортировки выбором в лучшем случае $O(N^2)$. Трудоёмкость сортировки выбором в худшем случае $O(N^2)$ [3].

4 Технологическая часть

4.1 Выбор языка программирования

Я выбрала языком программирования Python.

Время работы алгоритмов было замерено с помощью функции process_time() из библиотеки time.

4.2 Сведения о модулях программы

Программа состоит из следующих модулей:

- main.py - главный файл программы, в котором располагаются алгоритмы и меню
- test.py - файл с замерами времени

В листингах 1,2,3 представлен код используемых алгоритмов.

Листинг 1 – Сортировка пузырьком с флагом.

```
def bubble_sort(arr):
    i = 0
    flag = False
    while (i < len(arr)):
        if (i + 1 != len(arr) and arr[i] > arr[i + 1]):
            arr[i], arr[i + 1] = arr[i + 1], arr[i]
            flag = True
        i += 1

        if (i == len(arr) and flag):
            flag = False
            i = 0
    return arr

```

Листинг 2 – Сортировка вставками.

```
def insertion_sort(arr):
    for i in range(1, len(arr)):
```

```
        j = i - 1
        key = arr[i]

        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr

```

Листинг 3 – Сортировка выбором.

```
def selection_sort(arr):
    for i in range(0, len(arr) - 1):
        min = i
        for j in range(i + 1, len(arr)):
            if arr[j] < arr[min]:
                min = j
        arr[i], arr[min] = arr[min], arr[i]

```

4.3 Тесты для проверки корректности программы

В таблице 1 представлены функциональные тесты для проверки работы программы. Все тесты пройдены успешно.

Таблица 1 – Функциональные тесты

Входные данные	Ожидаемый результат
Пустой массив	Пустой массив
0 0 0 0 0	0 0 0 0 0
1 2 3 4 5	1 2 3 4 5
5 4 3 2 1	1 2 3 4 5
4 2 3 5 1	1 2 3 4 5
3 2 3 2 3 2	2 2 2 3 3 3

Вывод

В этом разделе была представлена реализация алгоритмов сортировки вставками, пузырьком, выбором.

5 Исследовательская часть

5.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- Операционная система Ubuntu 18.04 64-bit.
- Память 8 GiB Процессор Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz

5.2 Демонстрация работы программы

На рисунке 13 представлен результат работы программы.

```
Введите массив:
1 4 6 2 5 3 4

Результат сортировки пузырьком:
[1, 2, 3, 4, 4, 5, 6]

Результат сортировки вставками:
[1, 2, 3, 4, 4, 5, 6]

Результат сортировки выбором:
[1, 2, 3, 4, 4, 5, 6]
```

Рисунок 7 – Результат работы программы.

5.3 Время работы алгоритмов.

Алгоритмы тестировались при помощи функции process_time() [4] из библиотеки time языка Python. time.process_time() всегда возвращает значение времени типа float в секундах. Возвращает значение (в долях секунды) суммы системного и пользовательского процессорного времени текущего процесса. Не включает время, прошедшее во время сна. Контрольная точка возвращаемого значения не определена, поэтому допустима только разница между результатами последовательных вызовов. Замеры времени для каждой длины слов проводились 100 раз. В качестве результата взято среднее время работы сортировки на каждой длине массива.

В таблице 2 представлены замеры времени работы алгоритмов в лучшем случае.

Таблица 2 – Время работы алгоритмов в лучшем случае

длина	пузырьком с флагом	вставками	выбором
10	4.80692	3.43440	9.49087
50	18.40953	9.80128	115.08382
100	32.44425	17.23233	343.78603
300	105.62383	44.51905	3097.13863
500	176.58764	78.06553	8431.19578
1000	368.06116	166.90630	35400.43560

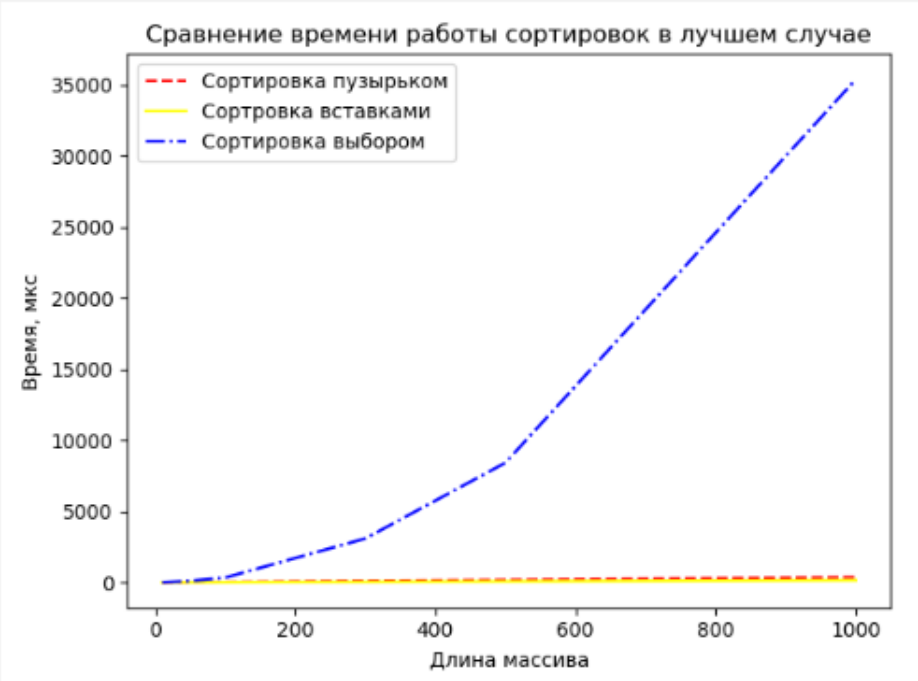

Рисунок 8 – Время работы сортировок в лучшем случае.

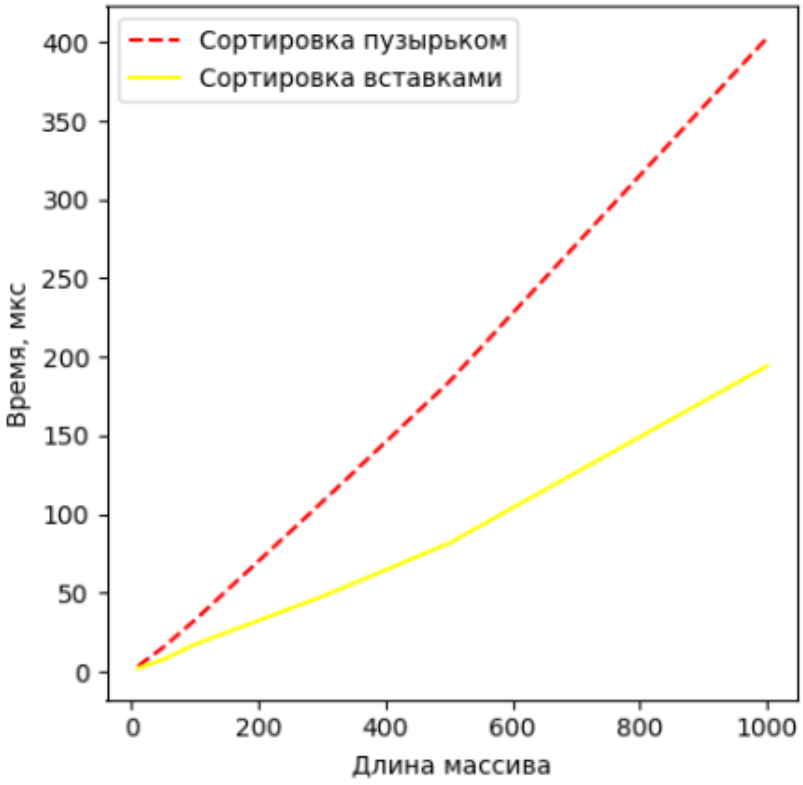

Рисунок 9 – Время работы сортировок вставками и пузырьком в лучшем случае.

В таблице 3 представлены замеры времени работы алгоритмов в худшем случае.

Таблица 3 – Время работы алгоритмов в худшем случае

длина	пузырьком с флагом	вставками	выбором
10	44.44178	9.53536	10.37062
50	1086.48139	205.88859	128.04513
100	3615.00233	663.28282	368.75568
300	37140.38387	5684.85509	3187.57168
500	103053.70644	15914.03027	8893.61887
1000	438259.28284	69039.85004	36950.06734

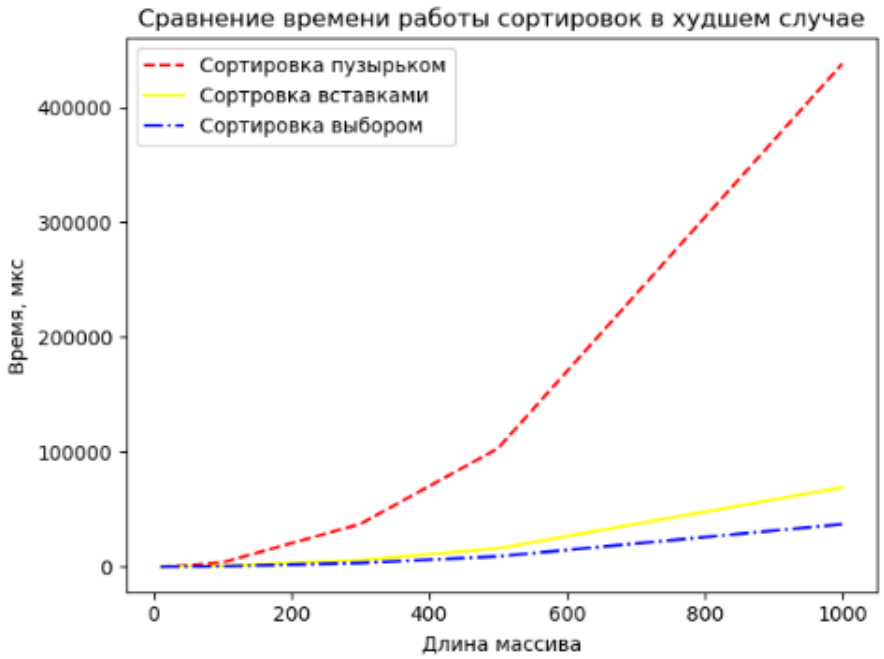

Рисунок 10 – Время работы сортировок в худшем случае.

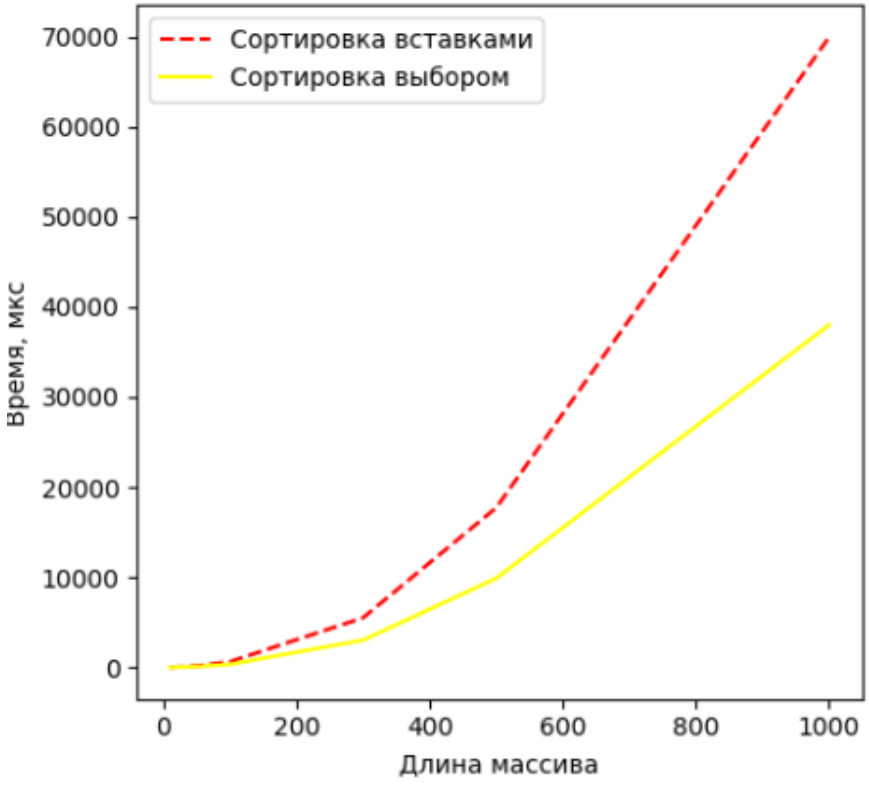

Рисунок 11 – Время работы сортировок вставками и выбором в худшем случае.

В таблице 4 представлены замеры времени работы алгоритмов в произвольном случае.

Таблица 4 – Время работы алгоритмов в произвольном случае

длина	пузырьком с флагом	вставками	выбором
10	27.80954	7.07086	10.97786
50	751.76795	108.30140	106.85882
100	2838.44984	281.08474	322.75758
300	31363.73485	2882.51491	3051.78592
500	92107.79662	8491.61305	8939.50457
1000	396258.78831	35201.84220	36957.40646

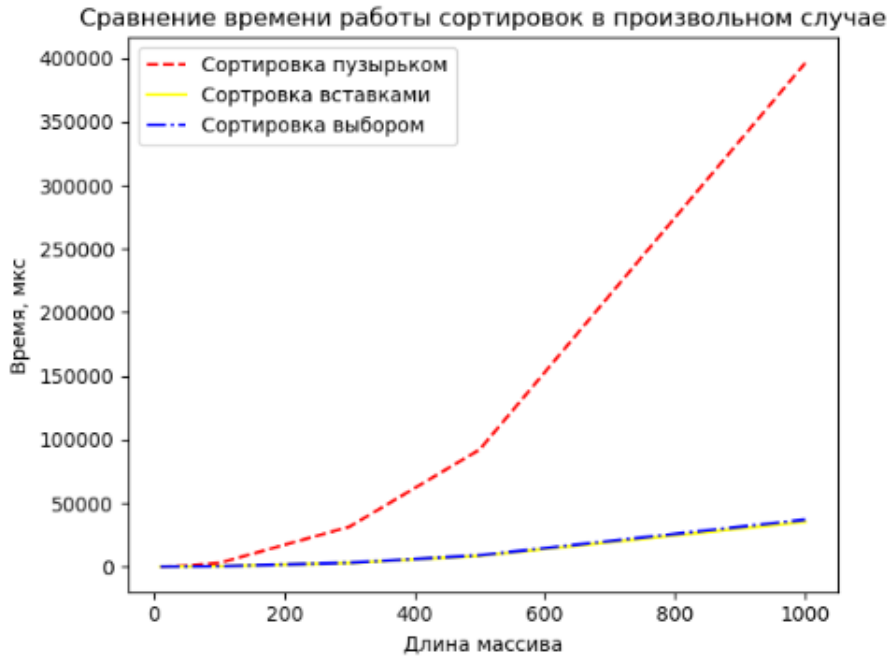

Рисунок 12 – Время работы сортировок в произвольном случае.

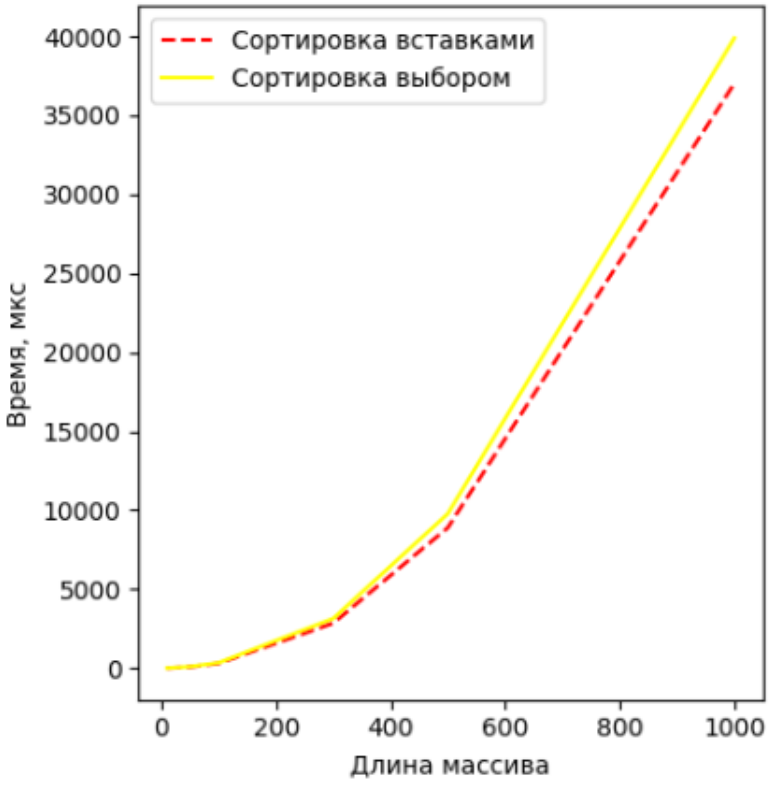

Рисунок 13 – Время работы сортировок вставками и выбором в произвольном случае.

Вывод

По результатам тестирования выявлено, что все рассмотренные алгоритмы реализованы правильно.

В лучшем случае сортировка выбором оказалась самой медленной. На длине массива 500 сортировка выбором работает в 108 раз медленнее чем сортировка вставками и в 47 раз медленнее, чем сортировка пузырьком с флагом. Сортировка вставками в лучшем случае работает быстрее всего.

В худшем случае самой медленной сортировкой является сортировка пузырьком с флагом. На длине массива 500 она работает 6,5 раз дольше, чем сортировка вставками и в 11,5 раз дольше, чем сортировка выбором. Сортировка выбором является самой быстрой в худшем случае.

В произвольном случае время работы сортировок вставками и выбором сопоставимо. Самой медленной является сортировка пузырьком с

флагом. На длине массива 500 она работает почти в 11 раз медленнее, чем эти сортировки.

Заключение

Цель лабораторной работы достигнута.

В ходе выполнения работы решены следующие задачи. Для достижения цели ставятся следующие задачи.

- Изучены алгоритмы сортировки пузырьком с флагом, вставками, выбором.
- Реализованы алгоритмы сортировки пузырьком с флагом, вставками,выбором.
- Дана оценка трудоёмкости в лучшем и худшем случае(для двух алгоритмов сделать вывод трудоёмкости).
- Проведены замеры времени работы для лучшего, худшего и произвольного случая.
- Описаны и обоснованы полученные результаты в отчете о выполненной лабораторной работе, выполненном как расчётно-пояснительная записка к работе.

Эксперементально было подтверждено различие по временной эффективности сортировок пузырьком с флагом, вставками, выбором на материале замеров процессорного времени выполнения реализации на варьирующихся длинах массивов.

В лучшем случае сортировка выбором оказалась самой медленной. На длине массива 500 сортировка выбором работает в 108 раз медленнее чем сортировка вставками и в 47 раз медленнее, чем сортировка пузырьком с флагом. Сортировка вставками в лучшем случае работает быстрее всего.

В худшем случае самой медленной сортировкой является сортировка пузырьком с флагом. На длине массива 500 она работает 6,5 раз дольше,

чем сортировка вставками и в 11,5 раз дольше, чем сортировка выбором. Сортировка выбором является самой быстрой в худшем случае.

В произвольном случае время работы сортировок вставками и выбором сопоставимо. Самой медленной является сортировка пузырьком с флагом. На длине массива 500 она работает почти в 11 раз медленнее, чем эти сортировки.

В результате исследований я пришла к выводу, что самой медленной сортировкой в произвольном и худшем случаях является сортировка пузырьком. Сортировка выбором ялвяется самой быстрой в худшем случае, но работает медленнее остальных сортировок в лучшем случае. В произвольном случае время работы сортировок вставками и выбором сопоставимо.

Список литературы

[1] Основные виды сортировок и примеры их реализации. [Электронный ресурс]. Режим доступа:

https://academy.yandex.ru/posts/osnovnye-vidy-sortirovok-i-primery-ikh-realizatsii Дата обращения: 24.09.2020

[2] Описание алгоритмов сортировки и сравнение их производительности. [Электронный ресурс.] Режим доступа:

https://habr.com/ru/post/335920/ Дата обращения: 24.09.2020

[3] Знай сложности алгоримтмов. [Электронный ресурс.] Режим доступа:

https://habr.com/ru/post/188010/ Дата обращения: 24.09.2020

[4] Python documentation. [Электронный ресурс.] Режим доступа:

https://www.python.org/doc/ Дата обращения: 26.09.2020