**Министерство образования и науки 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"

# Трудоемкость сортировок

Студент:

Турсунов Жасурбек Рустамович

Группа: ИУ7-56Б

Преподователи:

Волкова Лилия Леонидовна
Строганов Юрий Владимирович

Москва, 2020 г.

# Содержание

# Введение

Целью данной лабораторной работы является изучение применений алгоритмов сортировки и обучение расчету трудоемкости алгоритмов. В данной лабораторной работе рассматриваются алгоритмы сортировки пузырьком, вставкми, а также гномья сортировка. Также требуется сделать сравнение этих алгоритмов, чтобы выбрать наилучше подходящее в конкретном случае.

В ходе лабораторной предстоит выполнить следующие задачи:

1. изучить алгоритмы сортировки;
2. вычислить трудоемкость каждого алгоритма;
3. реализовать три алгоритма сортировки на одном из языков программирования;
4. сравнить алгоритмы сортировки массива.

# 1 Аналитическая часть

Сортировкой массива - одна из самых популярных операций над массивом.[1] Алгоритмы реализуют упорядочивание элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки.
Область применения:

- физика;
- математика;
- экономика;
- информатика;
- и тд.

## 1.1 Сортировка пузырьком

Алгоритм проходит по массиву n-1 раз до тех пор, пока массив не будет полностью отсортирован. В каждом проходе элементы попарно сраниваются и, при необходимости, меняются местами. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент ставится на своё место в конец неотсортированного массива. Таким образом наибольшие элементы "всплывают"как пузырёк.

## 1.2 Сортировка вставками

На каждом шаге выбирается один из элементов неотсортированной части массива (максимальный/минимальный) и помещается на нужную позицию в отсортированную часть массива.

## 1.3 Гномья сортировка

В данном алгоритме поддерживается указатель на текущий элемент, если он больше предыдущего или он первый — указатель смещается на позицию

вправо, иначе текущий и предыдущий элементы меняются местами и указатель сместится влево. Алгоритм похож на сортировку вставками. Главное отличие от сортировки вставками заключается в том, что перед вставкой на нужное место происходит серия обменов, как в сортировке пузырьком.

# 2 Конструкторская часть

**Требования к вводу:** На вход подается массив. **Требования к программе:** На вход подается массив.

1. корректная сортировка массива;
2. при нулевой длине массива программа не должнв аварийно завершаться.

## 2.1 Разработка алгоритмов

В данном разделе будут рассмотрены схемы алгоритмов:

1. сортировка пузырьком;
2. сортировка вставками;
3. гномья сортировка.

На рисунке 1 представлена схема алгоритма сортировки пузырьком.

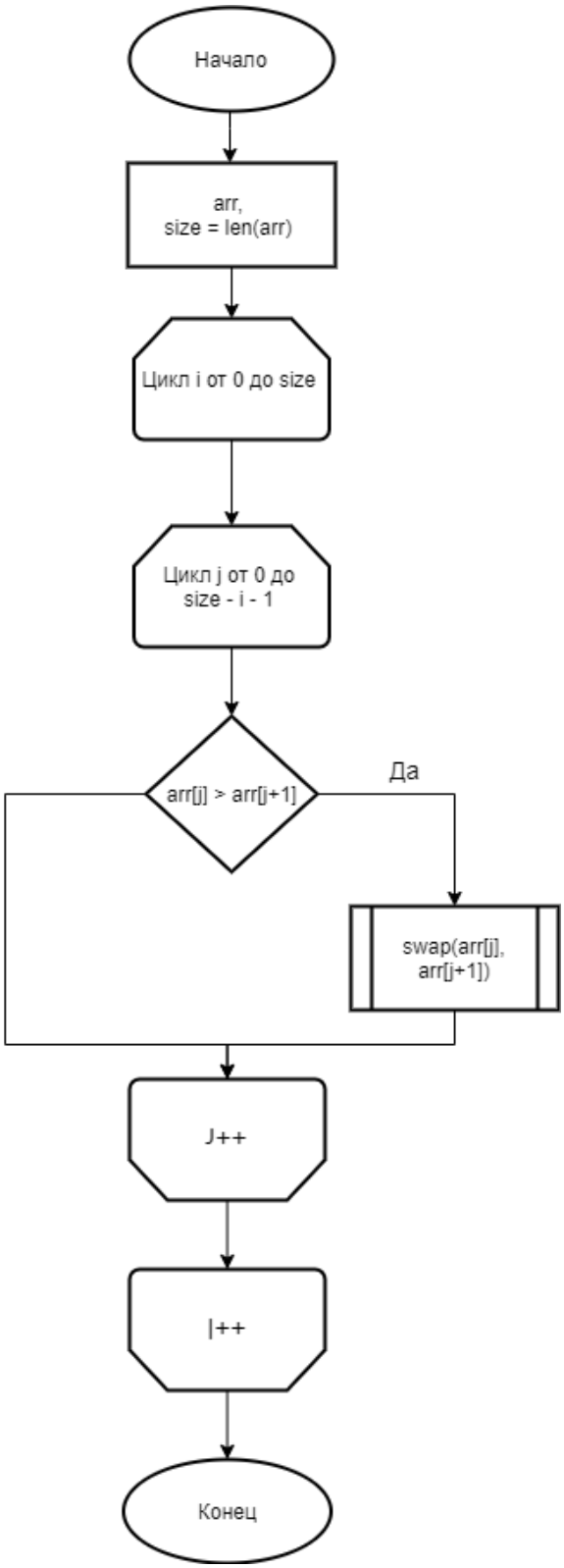


Рис. 1: Схема алгоритма сортировки пузырьком

На рисунке 2 представлена схема алгоритма сортировки вставками.

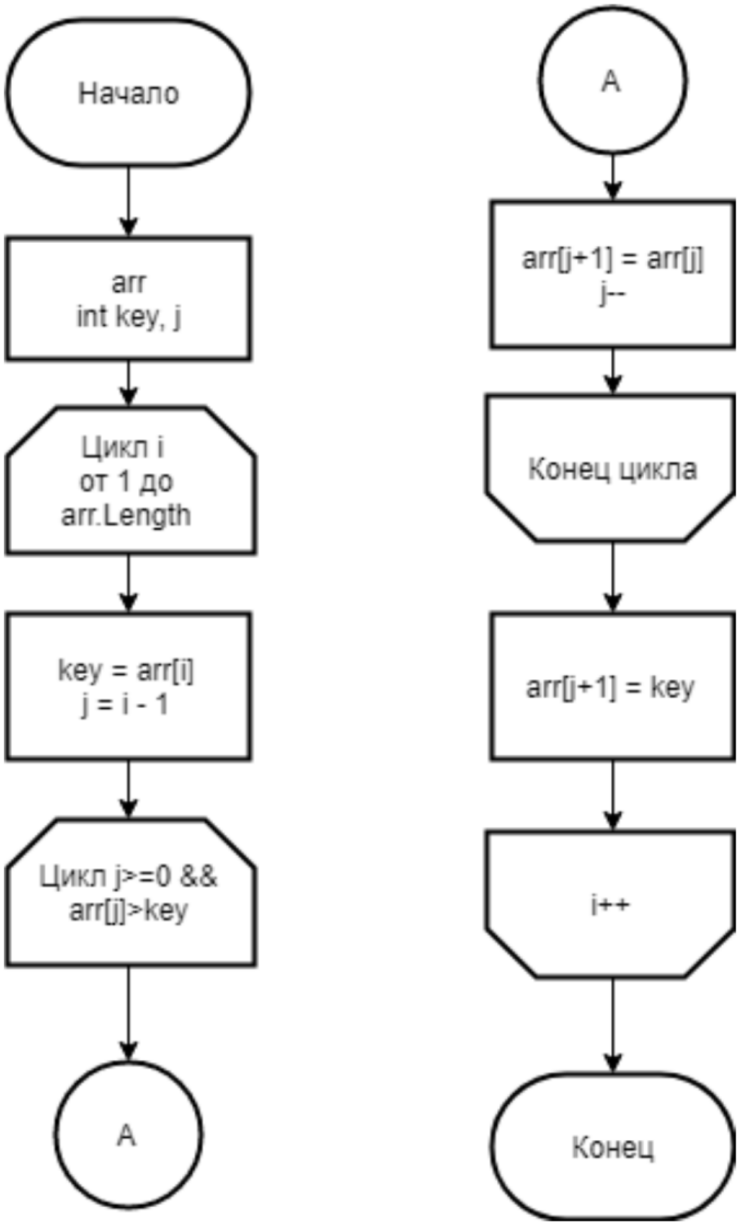


Рис. 2: Схема алгоритма сортировки вставками

На рисунке 3 представлена схема алгоритма гномьи сортировки.

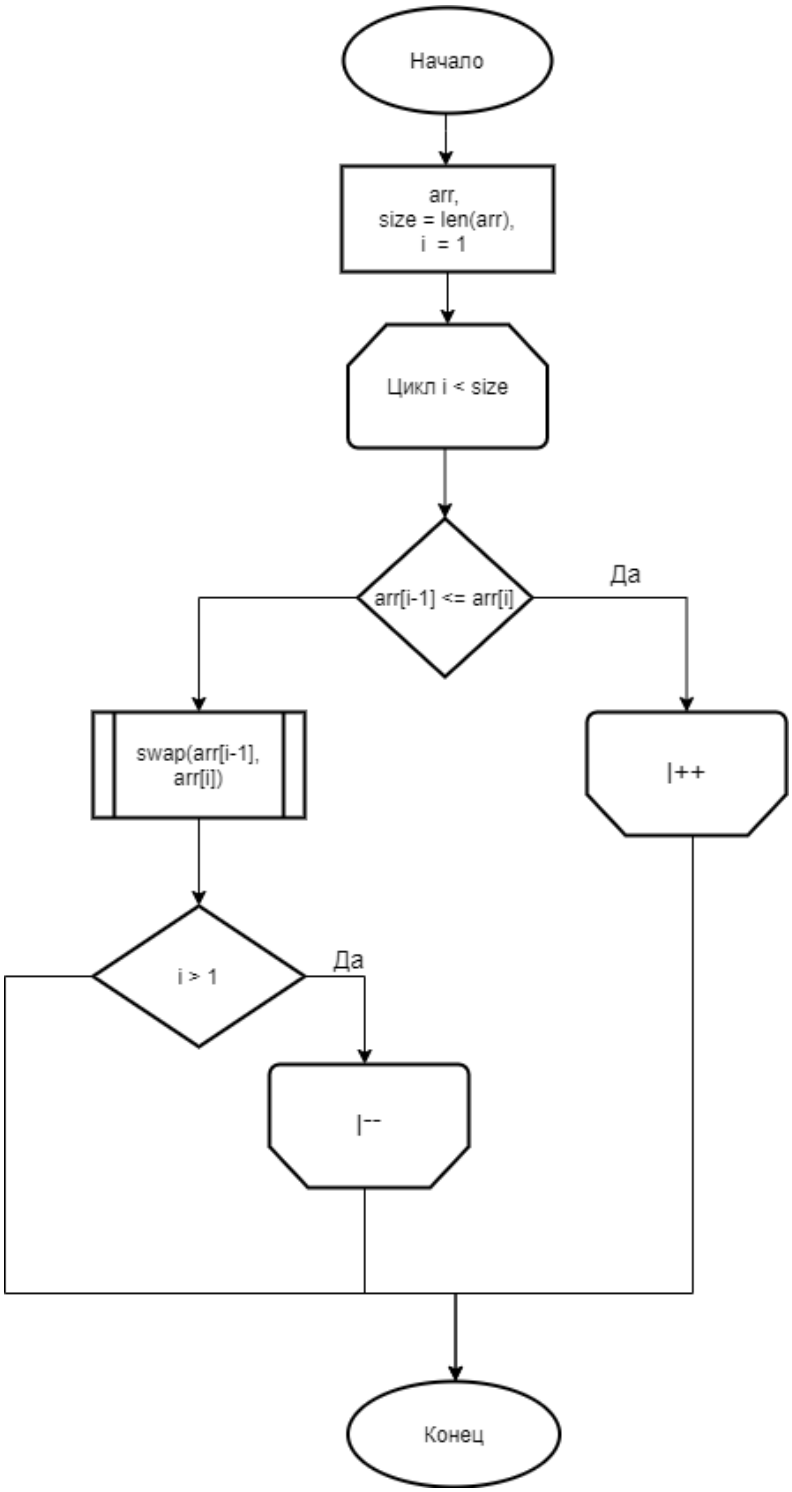


Рис. 3: Схема алгоритма гномьей сортировки

## 2.2 Трудоемкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов:

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

2. оценка трудоемкости цикла: Fц = init + N * (a + Fтела + post) + a, где a - условие цикла, init - предусловие цикла, post - постусловие цикла;

3. стоимость условного перехода применим за 0, стоимомть вычисления условия остаётся.

Далее будет приведены оценки трудоемкости алгоритмов.

### 2.2.1 Сортировка пузырьком

**Лучший случай:** Массив отсортирован; не произошло ни одного обмена за 1 проход -> выходим из цикла.
Трудоемкость: 1 + 1 + 2 + $n$ * (2 + 7 + 1 + 3) = 13n + 4 = $O(n)$.

**Худший случай:** Массив отсортирован в обратном порядке; в каждом случае происходил обмен.
Трудоемкость: 1 + 1 + 2 + $n$ * ($n$ * (7 + 5 + 1 + 3) + 1 + 1) = $16n^2$ + 2n + 4 = $O(n^2)$.

### 2.2.2 Сортировка вставками

**Лучший случай:** Массив отсортирован. При этом все внутренние циклы состоят всего из одной итерации.
Трудоемкость: T(n) = 3n + ((2 + 2 + 4 + 2) * (n - 1)) = 3n + 10(n - 1) = 13n - 10 = $O(n)$.

**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) = 3n + (2 + 2)(n - 1) + 4($\frac{n(n+1)}{2}$) - 1) + 5$\frac{n(n-1)}{2}$ + 3(n-1) = 3n + 4n - 4 + $2n^2$ + 2n - 4 + $2.5n^2$ - 2.5n + 3n - 3 = $4.5n^2$ + 9.5n - 11 = $O(n^2)$.

### 2.2.3 Гномья сортировка

**Лучший случай:** Массив отсортирован; не произошло ни одного обмена.
Трудоемкость: $1 + 1 + 1 + n * (1 + 4 + 1) = 6n + 3 = O(n)$.

**Худший случай:** Массив отсортирован в обратном порядке; в каждом случае происходил обмен.
Трудоемкость: $1 + 1 + 1 + n * (2 + n * (3 + 1 + 1)) = 5n^2 + 2n + 3 = O(n^2)$.

## 2.3 Вывод

В данном разделе были рассмотрены схемы алгоритмов сортировки массива. Введена модель оценки трудоемкости алгоритма, были расчитаны трудоемкости алгоритмов в соответствии с этой моделью.

Сортировка пузырьком: лучший - $O(n)$, худший - $O(n^2)$.
Сортировка вставками: лучший - $O(n)$, худший - $O(n^2)$.
Гномья сортировка: лучший - $O(n)$, худший - $O(n^2)$.

При этом сортировка вставками быстрее пузырька в худшем случае т.к. имеет меньший коэффициент. Вставки $4.5n^2$, пузырек $16n^2$

# 3 Технологическая часть

В данном разделе будут рассмотрены требования к программному обеспечению, средства реализации и представлен листинг кода.

## 3.1 Требования к программному обеспечению

Входные данные: массив.

Выходные данные: отсортированный массив.

На рисунке 4 представлена IDEF0-диаграмма, показывающая функциональную схему сортировки массива.

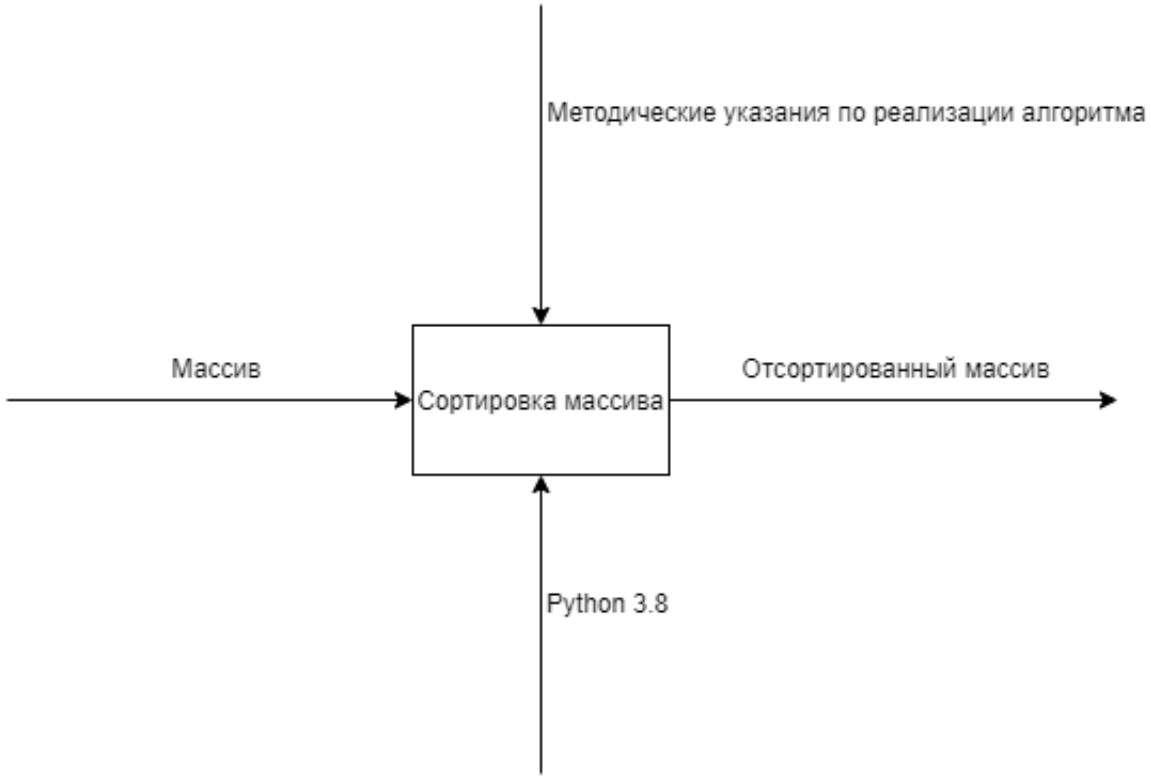


Рис. 4: IDEF0-диаграмма, описывающая функциональную схему сортировки массива.

## 3.2 Средства реализации

В данной работе используется язык программирования Python, так как ЯП позволяет написать программу за кратчайшее время. Проект выполнен в среде разработки Visual Studio Code.

## 3.3 Листинг кода

В данном пункте представлен листинг кода алгоритмов сортировик.[2] А именно:

- алгоритм сортировки пузырьком;
- алгоритм сортировки вставками;
- алгоритм гномьей сортировки.

На листинге 1 представлен код алгоритма сортировки массива пузырьком.

```
1       def bubbleSort(arr):
2           size = len(arr)
3           for i in range(size):
4               for j in range(0, size-i-1):
5                   if arr[j] > arr[j+1]:
6                       arr[j], arr[j+1] = arr[j+1], arr[j]
7           return arr
8
```

Листинг 1: Алгоритм сортировки пузырьком

На листинге 2 представлен код алгоритма сортировки массива вставками.

```
1       def insertSort(arr):
2           size = len(arr)
3           for i in range(size):
4               j = i - 1
5               key = arr[i]
6               while arr[j] > key and j >= 0:
7                   arr[j+1] = arr[j]
8                   j -= 1
9               arr[j+1] = key
10          return arr
11
```

Листинг 2: Алгоритм сортировки вставками

На листинге 3 представлен код алгоритма сортировки массива гномьей сортировкой.

```
        def gnomeSort(arr):
            i, size = 1, len(arr)
            while i < size:
                if arr[i - 1] <= arr[i]:
                    i += 1
                else:
                    arr[i - 1], arr[i] = arr[i], arr[i - 1]
                    if i > 1:
                        i -= 1
            return arr

```

Листинг 3: Алгоритм гномьей сортировки

## 3.4 Вывод

В данном разделе была представлена структура ПО и листинги кода программы.

# 4 Исследовательская часть

В данном разделе будет проведен эксперимент и сравнительный анализ.

## 4.1 Системные характеристики

Характеристики компьютера на котором проводился замер времени сортировки массива:

1. операционная система - Windows 10;
2. процессор - Intel(R) Core(TM) i7-10510U CPU @1.80GHz 2.30GHz;
3. оперативная память - 16 ГБ.

## 4.2 Постановка эксперимента

В рамках данного проекта были проведены эксперименты, описанные ниже:

1. сравнение времени работы алгоритмов сортировки при разных заполнениях.

## 4.3 Сравнительный анализ на основе замеров времени работы алгоритмов

Был проведен замер времени работы каждого из алгоритмов.

На рисунке 5 представлен первый эксперимент, где он производится для лучшего случая, когда массив уже отсортирован.


Рис. 5: Сравнение времени работы алгоритмов сортировки, в уже отсортированном массиве

На рисунке 6 представлен второй эксперимент, где он производится для худшего случая, когда массив отсортирован в обратном порядке. Ниже приведена полученная диаграмма:

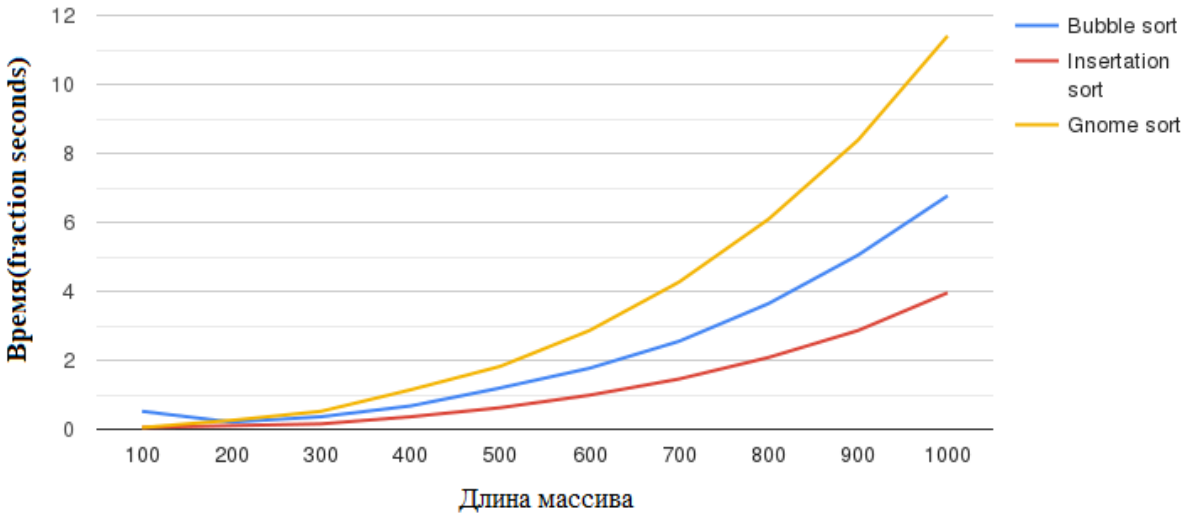


Рис. 6: Сравнение времени работы алгоритмов в обратно отсортированном массиве

На рисунке 7 производится третий эксперимент для общего случая, когда массив заполнен рандомными значениями. Ниже приведена полученная диаграмма:

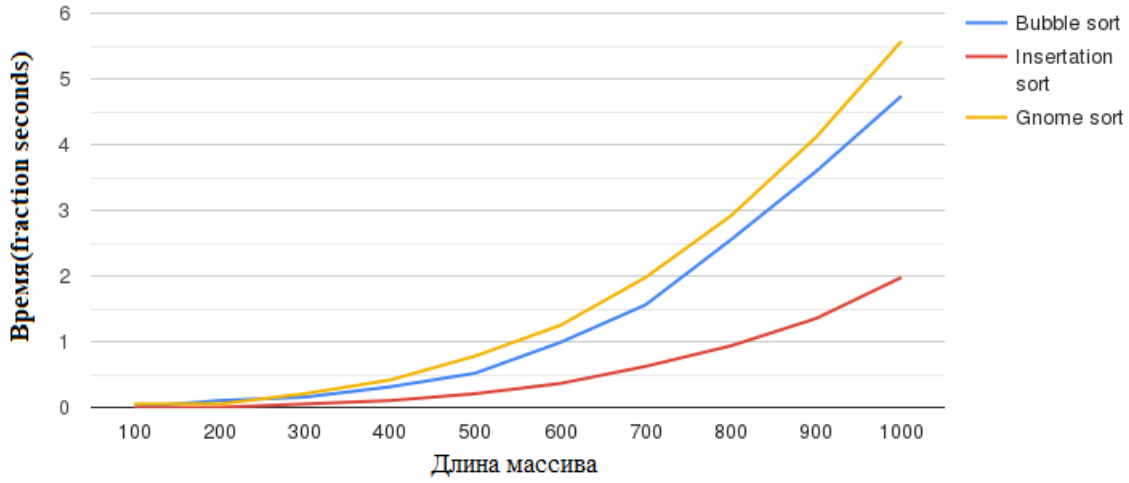


Рис. 7: Сравнение времени работы алгоритмов при случайн заполнении массива

## 4.4 Тестирование программы

В данном разделе будут показаны результаты тестирования

Всего было реализовано 5 тестовых случаев:

1. отрицательный размер массива;
2. размер матрицы равен = 0;
3. сравнение работы все трех алгоритмов на уже отсортированных значениях массива;
4. сравнение работы все трех алгоритмов на обратно отсортированных значениях массива;
5. сравнение работы все трех алгоритмов на случайных значениях массива;

На рисунке 8 предоставлен результат программы при вводе отрицательного значения для размера массива.

```
Input size of array: -7
Size of array can not be less than zero
```

Рис. 8: Результат программы при отрицательном размере массива

На рисунке 9 предоставлен результат программы при вводе размера массива равного нулю.

```
Input size of array: 0
You input n = 0. That's why your array is empty
```

Рис. 9: Результат программы при нулевом размере массива

На рисунке 10 предоставлен результат программы при уже отсортированных значениях массива.

```
Array with sorted elements:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Bubble Sort:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Insertation Sort:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Gnome Sort:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Рис. 10: Результат программы при уже отсортированных значениях массива

На рисунке 11 предоставлен результат программы при обратно отсортированных значениях массива.

```
Array with reversed elements:  [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
Bubble Sort:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Insertation Sort:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Gnome Sort:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Рис. 11: Результат программы при обратно отсортированных значениях массива

На рисунке 12 предоставлен результат программы при случайных значениях массива.

```
Array with random elements:  [75, 76, 33, 83, 4, 30, 34, 96, 75, 22]
Bubble Sort:  [4, 22, 30, 33, 34, 75, 75, 76, 83, 96]
Insertation Sort:  [4, 22, 30, 33, 34, 75, 75, 76, 83, 96]
Gnome Sort:  [4, 22, 30, 33, 34, 75, 75, 76, 83, 96]
```

Рис. 12: Результат программы при случайных значениях массива

## 4.5 Вывод

По результатам тестирования все рассматриваемые алгоритмы сортировки были реализованы верно. Самым быстрым алгоритмом при случайном заполнении, оказался алгоритм сортировки вставками, а самым медленным - алгоритм гномьей сортировки. А алгоритм сортировки пузырьком, показал схожий результат с алгоритмом гномьей сортировки.

# Заключение

В ходе работы были изучены алгоритмы сортировки массива: пузырьком, вставки, гномья. Выполнено сравнение всех рассматриваемых алгоритмов. Входе исследования был найден отпимальный алгоритм. Изучены зависимости выполнения алгоритмов от длины массива. Также был реализован программный код продукта.

# Список литературы

[1] Marcus Sanatan. *Sorting Algorithms in Python [ЭЛ. РЕСУРС] Режим доступа:* URL: `https://stackabuse.com/sorting-algorithms-in-python/`. (дата обращения: 10.11.2020).

[2] Yandex. *Основные виды сортировок и примеры их реализации[ЭЛ. РЕСУРС] Режим доступа:* URL: `https://academy.yandex.ru/posts/osnovnye-vidy-sortirovok-i-primery-ikh-realizatsii`. (дата обращения: 10.11.2020).