



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 Трудоемкость сортировок

Студент Баринов Н.Ю.

Группа ИУ7-51Б

Оценка (баллы) \_\_\_\_\_

Преподаватель Волкова Л. Л.

Преподаватель Строганов Ю. В.

Москва — 2022 г.

# Содержание

<b>Введение</b>	<b>3</b>
<b>1 Аналитическая часть</b>	<b>5</b>
1.1 Сортировка вставками . . . . .	5
1.2 Сортировка пузырьком . . . . .	5
1.3 Гномья сортировка . . . . .	5
<b>2 Конструкторская часть</b>	<b>7</b>
2.1 Требования к ПО . . . . .	7
2.2 Разработка алгоритмов . . . . .	7
2.3 Модель вычислений для проведения оценки трудоемкости . . .	11
2.4 Трудоемкость алгоритмов . . . . .	11
2.4.1 Алгоритм сортировки пузырьком . . . . .	11
2.4.2 Алгоритм гномьей сортировки . . . . .	12
2.4.3 Алгоритм сортировки вставками . . . . .	12
<b>3 Технологическая часть</b>	<b>13</b>
3.1 Средства реализации . . . . .	13
3.2 Сведения о модулях программы . . . . .	13
3.3 Реализация алгоритмов . . . . .	13
3.4 Функциональные тесты . . . . .	14
<b>4 Исследовательская часть</b>	<b>16</b>
4.1 Технические характеристики . . . . .	16
4.2 Демонстрация работы программы . . . . .	16
4.3 Время выполнения алгоритмов . . . . .	17
<b>Заключение</b>	<b>21</b>
<b>Список использованных источников</b>	<b>22</b>

# Введение

Сортировка - перегруппировка некой последовательности, или кортежа, в определенном порядке. Это одна из главных процедур обработки структурированных данных. Расположение элементов в определенном порядке позволяет более эффективно проводить работу с последовательностью данных, в частности при поиске некоторых данных.

Существует множество алгоритмов сортировки, но любой алгоритм сортировки имеет:

- сравнение, которое определяет, как упорядочена пара элементов;
- перестановка для смены элементов местами;
- алгоритм сортировки, использующий сравнение и перестановки.

Что касается самого поиска, то при работе с отсортированным набором данных время, которое нужно на нахождение элемента, пропорционально логарифму количества элементов. Последовательность, данные которой расположены в хаотичном порядке, занимает время, которое пропорционально количеству элементов, что куда больше логарифма.

**Цель работы:** изучение и исследование трудоемкости алгоритмов сортировки.

## **Задачи работы.**

1. Изучить и реализовать алгоритмы сортировки: пузырьком, вставками, гномья.
2. Провести тестирование по времени для выбранных сортировок.
3. Провести сравнительный анализ трудоемкости алгоритмов на основе теоретических расчетов.
4. Провести сравнительный анализ реализаций алгоритмов по затраченному процессорному времени и памяти.

5. Описать и обосновать полученные результаты в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

# 1 Аналитическая часть

В этом разделе будут рассмотрены алгоритмы сортировок - вставками, пузырьком, гномья.

## 1.1 Сортировка вставками

**Сортировка вставками** [1] - размещение элемента входной последовательности на подходящее место выходной последовательности.

Набор данных условно разделяется на входную последовательность и выходную. В начале отсортированная часть пуста. Каждый  $i$ -ый элемент, начиная с  $i = 2$ , входной последовательности размещается в уже отсортированную часть до тех пор, пока изначальные данные не будут исчерпаны.

## 1.2 Сортировка пузырьком

**Сортировка пузырьком** [2] - Алгоритм сортировки “пузырьком” состоит в повторении проходов по массиву с помощью вложенных циклов. При каждом проходе по массиву сравниваются между собой пары “соседних” элементов. Если элементы какой-то из сравниваемых пар расположены в неправильном порядке – происходит обмен (перезапись) значений ячеек массива. Проходы по массиву повторяются  $N - 1$  раз.

## 1.3 Гномья сортировка

**Гномья сортировка** [3] - алгоритм сортировки, который использует только один цикл, что является редкостью.

В этой сортировке массив просматривается слева-направо, при этом сравниваются и, если нужно, меняются соседние элементы. Если происходит обмен элементов, то происходит возвращение на один шаг назад. Если обмена не было - алгоритм продолжает просмотр массива в поисках неупорядоченных пар.

## Вывод

В данной работе необходимо реализовать алгоритмы сортировки, описанные в данном разделе, а также провести их теоритическую оценку и проверить ее экспериментально.

## 2 Конструкторская часть

В данном разделе будут рассмотрены схемы алгоритмов сортировок (вставками, пузырьком и гномья), а также найдена их трудоемкость

### 2.1 Требования к ПО

Ряд требований к программе: на вход подается массив целых чисел в диапазоне от -10000 до 10000, возвращается отсортированный по месту массив, который был задан в предыдущем пункте.

### 2.2 Разработка алгоритмов

На рисунках 2.1, 2.2 и 2.3 представлены схемы алгоритмов сортировки - вставками, пузырьком и гномья

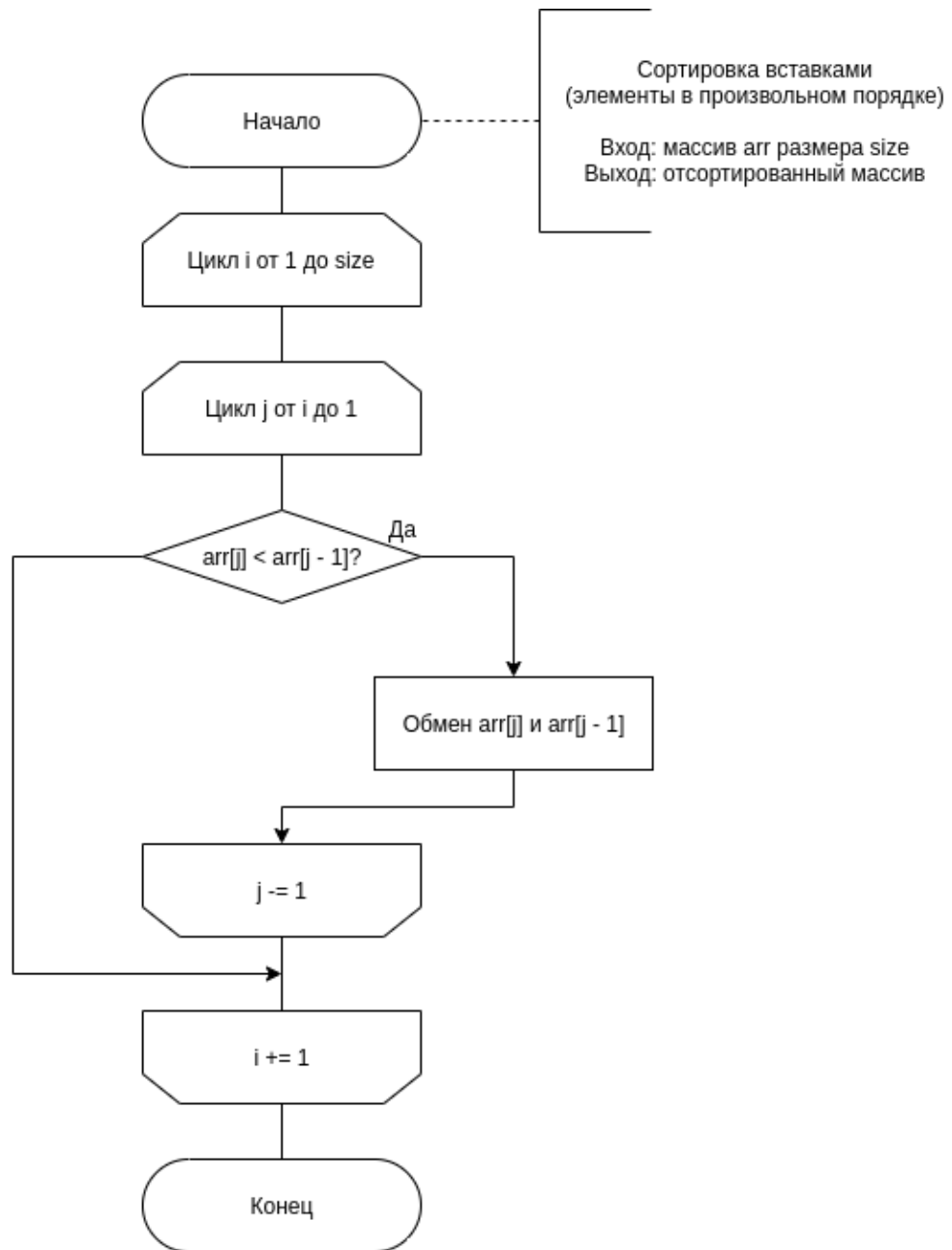


Рисунок 2.1 – Схема алгоритма сортировки вставками



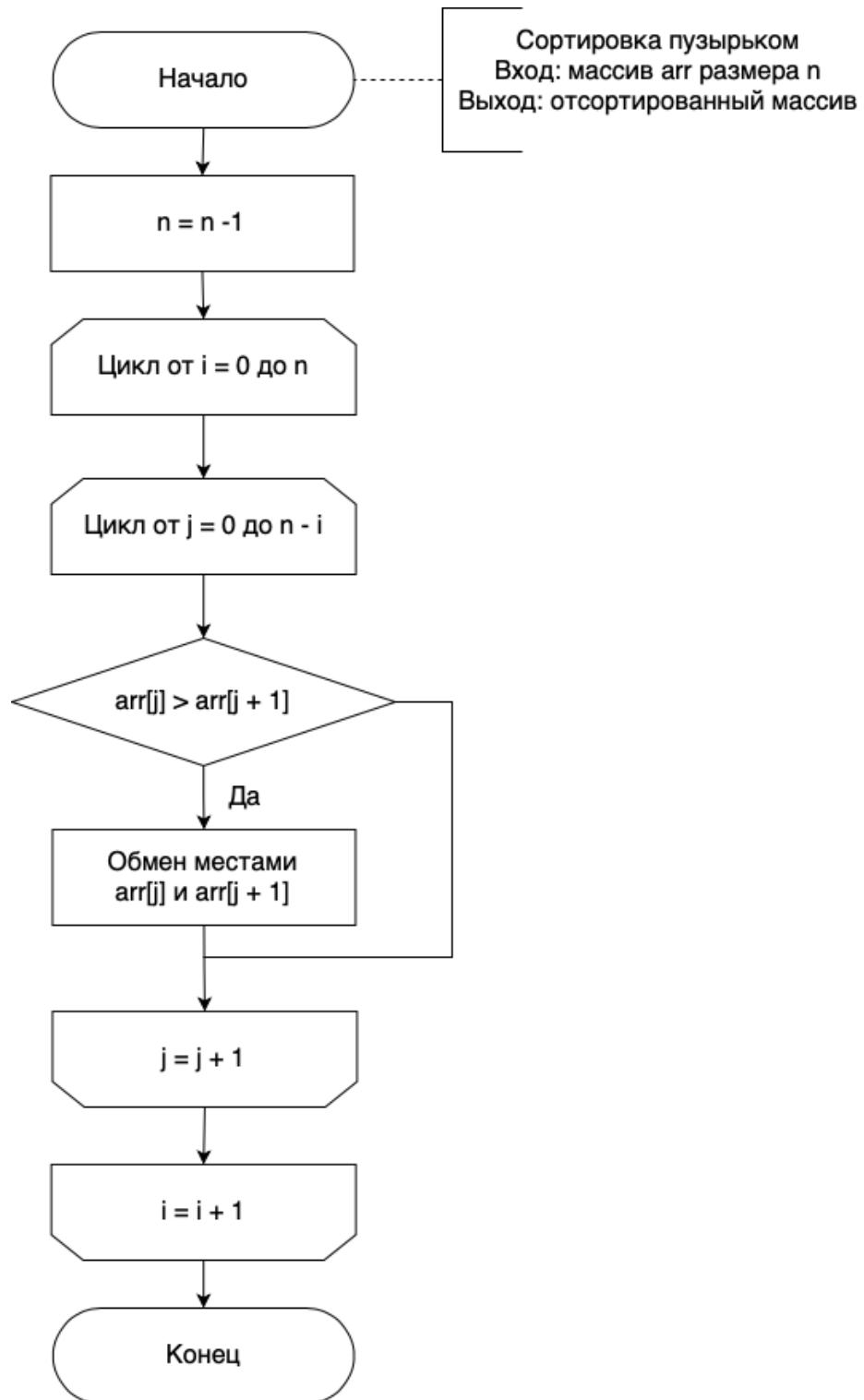


Рисунок 2.2 – Схема алгоритма сортировки пузырьком

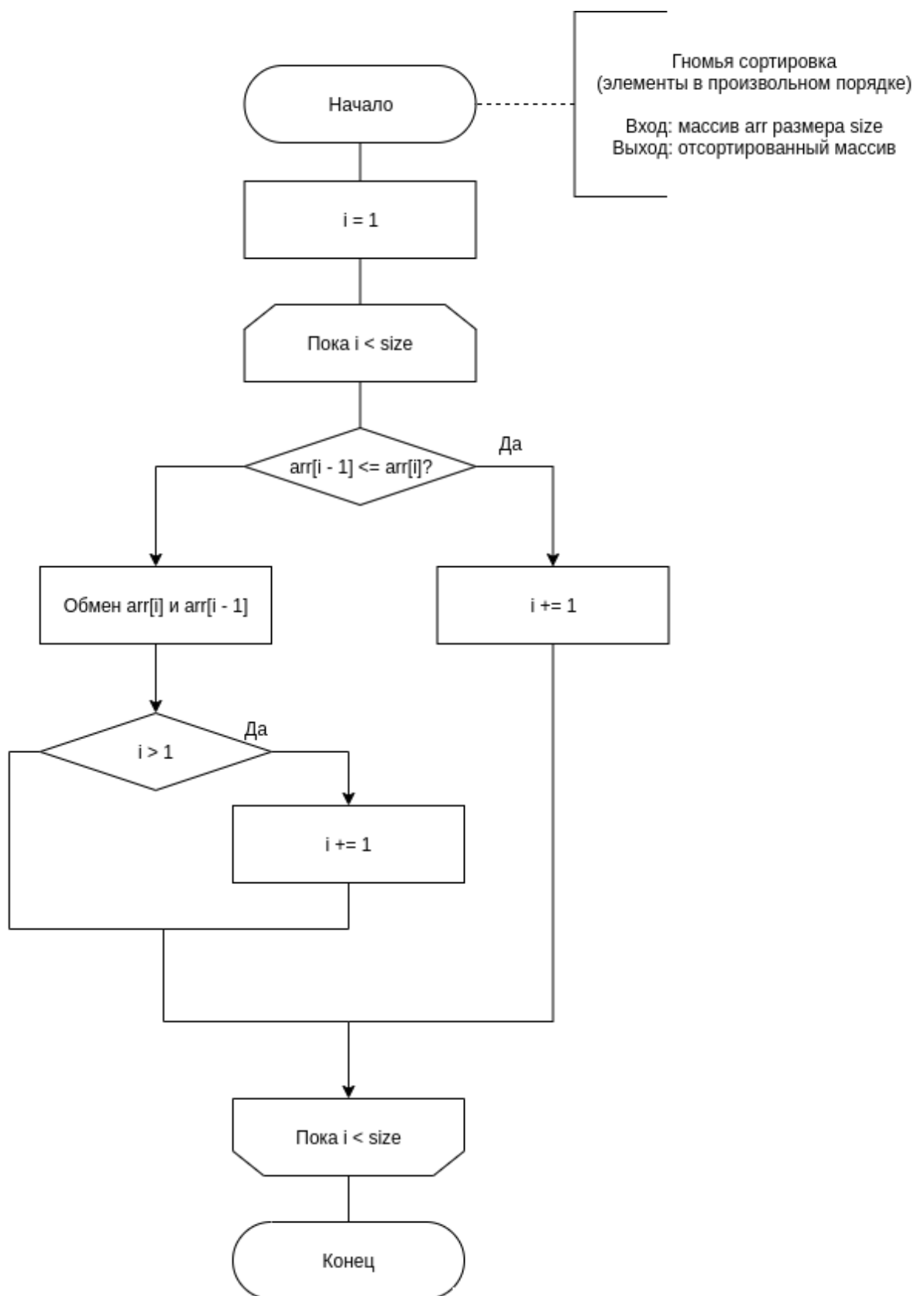


Рисунок 2.3 – Схема алгоритма гномьей сортировки

## 2.3 Модель вычислений для проведения оценки трудоемкости

Введем модель вычислений [4], которая потребуется для определения трудоемкости каждого отдельно взятого алгоритма сортировки:

1. операции из списка (2.1) имеют трудоемкость равную 1:

$$\begin{aligned} +, -, /, *, \%, =, + =, - =, * =, / =, \% =, \\ ==, ! =, <, >, < =, > =, [], ++, --; \end{aligned} \quad (2.1)$$

2. трудоемкость оператора выбора `if условие then A else B` рассчитывается, как:

$$f_{if} = f_{условия} + \begin{cases} f_A, & \text{если условие выполняется,} \\ f_B, & \text{иначе;} \end{cases} \quad (2.2)$$

3. трудоемкость цикла рассчитывается, как:

$$f_{for} = f_{инициализации} + f_{сравнения} + N(f_{тела} + f_{инкремент} + f_{сравнения}); \quad (2.3)$$

4. трудоемкость вызова функции равна 0.

## 2.4 Трудоемкость алгоритмов

Определим трудоемкость выбранных алгоритмов сортировки по коду.

### 2.4.1 Алгоритм сортировки пузырьком

Трудоемкость в лучшем случае (при уже отсортированном массиве):

$$f_{best} = 2 + 1 + N \cdot \left(1 + 1 + 1 + \frac{N-1}{2} \cdot (1 + 1 + 4)\right) = 3N^2 + 3 = O(N^2). \quad (2.4)$$

Трудоёмкость в худшем случае (при массиве, отсортированном в обратном порядке):

$$\begin{aligned} f_{worst} &= 2 + 1 + N \cdot \left(1 + 1 + 1 + \frac{N-1}{2} \cdot (1 + 1 + 11)\right) = \\ &= 6.5N^2 - 3.5N + 3 = O(N^2). \end{aligned} \quad (2.5)$$

### 2.4.2 Алгоритм гномьей сортировки

Трудоёмкость в лучшем случае (при уже отсортированном массиве):

$$f_{best} = 1 + N(4 + 1) = 5N + 1 = O(N) \quad (2.6)$$

Трудоёмкость в худшем случае (при массиве, отсортированном в обратном порядке):

$$f_{worst} = 1 + N(4 + (N - 1) * (7 + 2)) = 9N^2 - 5N + 1 = O(N^2) \quad (2.7)$$

### 2.4.3 Алгоритм сортировки вставками

Трудоёмкость данного алгоритма может быть рассчитана с использованием той же модели подсчета трудоемкости.

Трудоёмкость алгоритма сортировки вставками: в лучшем случае -  $O(N)$ , в худшем случае -  $O(N^2)$  [1].

## Вывод

Были разработаны схемы всех трех алгоритмов сортировки. Также для каждого из них были рассчитаны и оценены лучшие и худшие случаи.

## 3 Технологическая часть

В данном разделе будут рассмотрены средства реализации, а также представлены листинги сортировок.

### 3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *Python*[5]. В текущей лабораторной работе требуется замерить процессорное время для выполняемой программы, а также построить графики. Все эти инструменты присутствуют в выбранном языке программирования.

Время работы было замерено с помощью функции *process\_time(...)* из библиотеки *time*[6].

### 3.2 Сведения о модулях программы

Программа состоит из двух модулей: *main.py* - файл, содержащий весь служебный код, *sorts.py* - файл, содержащий код всех сортировок.

### 3.3 Реализация алгоритмов

В листингах 3.1, 3.2, 3.3 представлены реализации алгоритмов сортировок (пузырьком, вставками и гномьей).

Листинг 3.1 – Алгоритм сортировки пузырьком

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

Листинг 3.2 – Алгоритм сортировки вставками

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

Листинг 3.3 – Алгоритм гномьей сортировки

```
1  def gnomme_sort(arr, n):
2
3  i = 1
4
5  while (i < n):
6      if (arr[i] < arr[i - 1]):
7          tmp = arr[i]
8          arr[i] = arr[i - 1]
9          arr[i - 1] = tmp
10
11         if (i > 1):
12             i -= 1
13     else:
14         i += 1
15
16     return arr
```

## 3.4 Функциональные тесты

В таблице 3.1 приведены тесты для функций, реализующих алгоритмы сортировки. Тесты *для всех сортировок* пройдены успешно.

Таблица 3.1 – Функциональные тесты

Входной массив	Ожидаемый результат	Результат
[1, 2, 3, 4, 5]	[1, 2, 3, 4, 5]	[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]	[1, 2, 3, 4, 5]	[1, 2, 3, 4, 5]
[9, 7, −5, 1, 4]	[−5, 1, 4, 7, 9]	[−5, 1, 4, 7, 9]
[5]	[5]	[5]
[]	[]	[]

## Вывод

Были разработаны схемы всех трех алгоритмов сортировки. Для каждого алгоритма была вычислена трудоемкость и оценены лучший и худший случаи.

## 4 Исследовательская часть

В данном разделе будут приедены примеры работы программа, а также проведен сравнительный анализ адгоритмов при различных ситуациях на основе полученных данных.

### 4.1 Технические характеристики

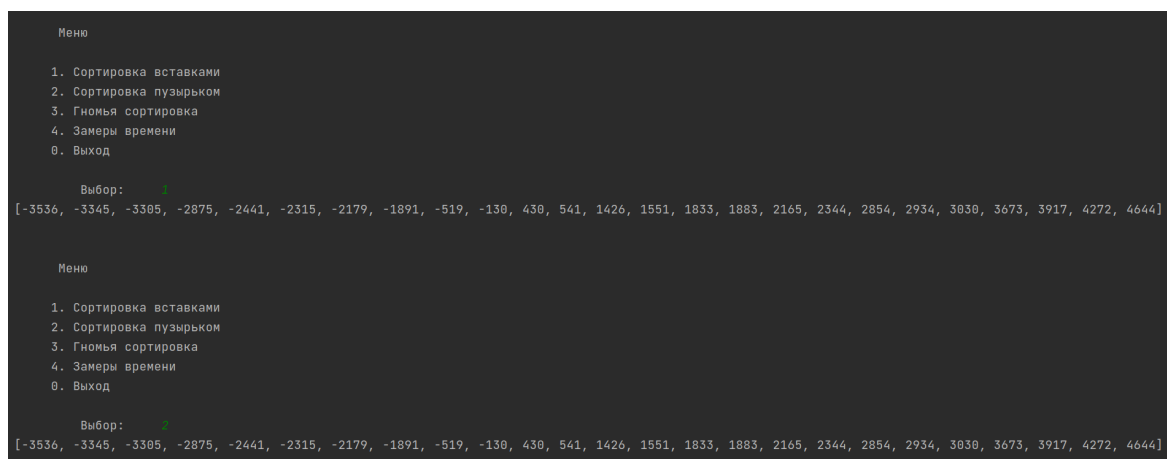
Технические характеристики устройства, на котором выполнялось тестирование представлены далее:

- операционная система – Ubuntu 22.04.1 [7] Linux x86\_64;
- память – 8 GiB;
- процессор – AMD RYZEN 3500U

При тестировании ноутбук был включен в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, а также системой тестирования.

### 4.2 Демонстрация работы программы

На рисунке 4.1 представлен результат работы программы.



```
Меню
1. Сортировка вставками
2. Сортировка пузырьком
3. Гномья сортировка
4. Замеры времени
0. Выход

Выбор: 1
[-3536, -3345, -3305, -2875, -2441, -2315, -2179, -1891, -519, -130, 430, 541, 1426, 1551, 1833, 1883, 2165, 2344, 2854, 2934, 3030, 3673, 3917, 4272, 4644]

Меню
1. Сортировка вставками
2. Сортировка пузырьком
3. Гномья сортировка
4. Замеры времени
0. Выход

Выбор: 1
[-3536, -3345, -3305, -2875, -2441, -2315, -2179, -1891, -519, -130, 430, 541, 1426, 1551, 1833, 1883, 2165, 2344, 2854, 2934, 3030, 3673, 3917, 4272, 4644]
```

Рисунок 4.1 – Пример работы программы



## 4.3 Время выполнения алгоритмов

Как было сказано выше, используется функция замера процессорного времени `process_time(...)` из библиотеки `time` на Python. Функция возвращает пользовательское процессорное время типа `float`.

Использовать функцию приходится дважды, затем из конечного времени нужно вычесть начальное, чтобы получить результат.

Результаты замеров времени работы алгоритмов сортировки на различных входных данных (в мс) приведены в таблицах 4.1, 4.2 и 4.3.

Таблица 4.1 – Отсортированные данные

Размер	Пузырьком	Вставками	Гномья
100	0.3020	0.0120	0.0088
200	1.2018	0.0243	0.0171
300	3.0260	0.0359	0.0247
400	5.0592	0.0571	0.0370
500	8.2762	0.0672	0.0468
600	12.5456	0.0831	0.0583
700	17.6357	0.0941	0.0746
800	22.8576	0.1246	0.0984
900	29.9132	0.1273	0.0911
1000	36.0504	0.1522	0.1007

Таблица 4.2 – Отсортированные в обратном порядке данные

Размер	Пузырьком	Вставками	Гномья
100	0.8783	0.5253	1.3620
200	3.8468	2.3479	5.9857
300	8.6238	4.9829	12.9845
400	15.2310	9.1723	24.2409
500	24.8823	14.7606	39.4460
600	35.7643	21.4536	58.5193
700	52.4496	30.1513	80.8845
800	72.5080	41.5019	117.2655
900	87.3379	54.1696	139.8114
1000	111.8675	58.8111	175.6119

Таблица 4.3 – Случайные данные

Размер	Пузырьком	Вставками	Гномья
100	0.6363	0.3118	0.8291
200	2.3936	1.1767	3.0757
300	5.2481	2.5084	6.2609
400	10.6879	4.8457	12.7158
500	15.7345	7.8693	20.7163
600	24.9607	10.5869	29.7342
700	33.6753	14.7531	41.2334
800	45.1266	20.5407	55.1073
900	58.8964	25.3347	71.1348
1000	72.3063	29.9543	84.1493

Также на рисунках 4.2, 4.3, 4.4 приведены графические результаты замеров работы сортировок в зависимости от размера входного массива.

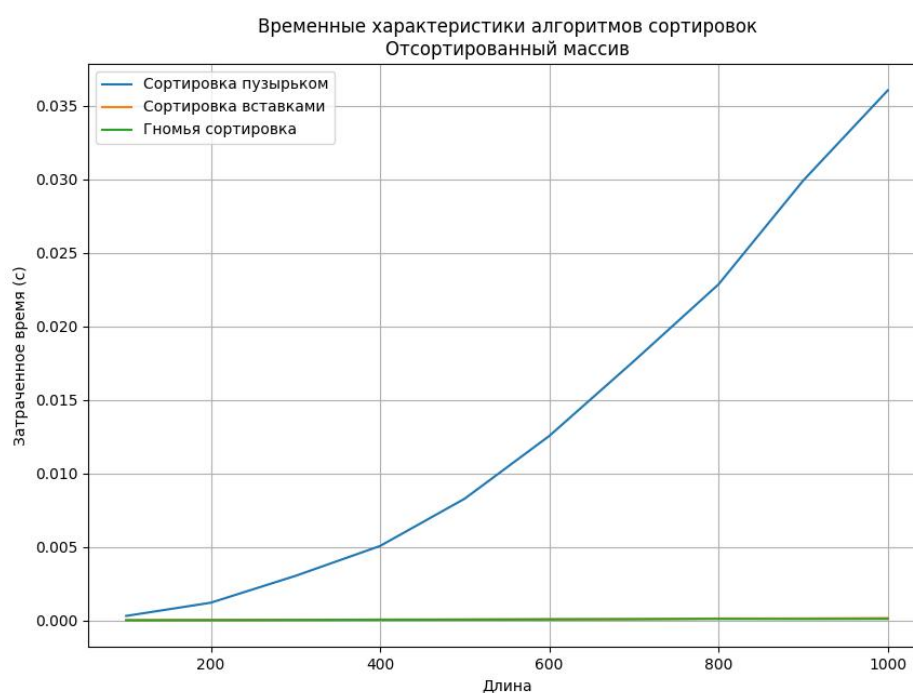


Рисунок 4.2 – Отсортированный массив

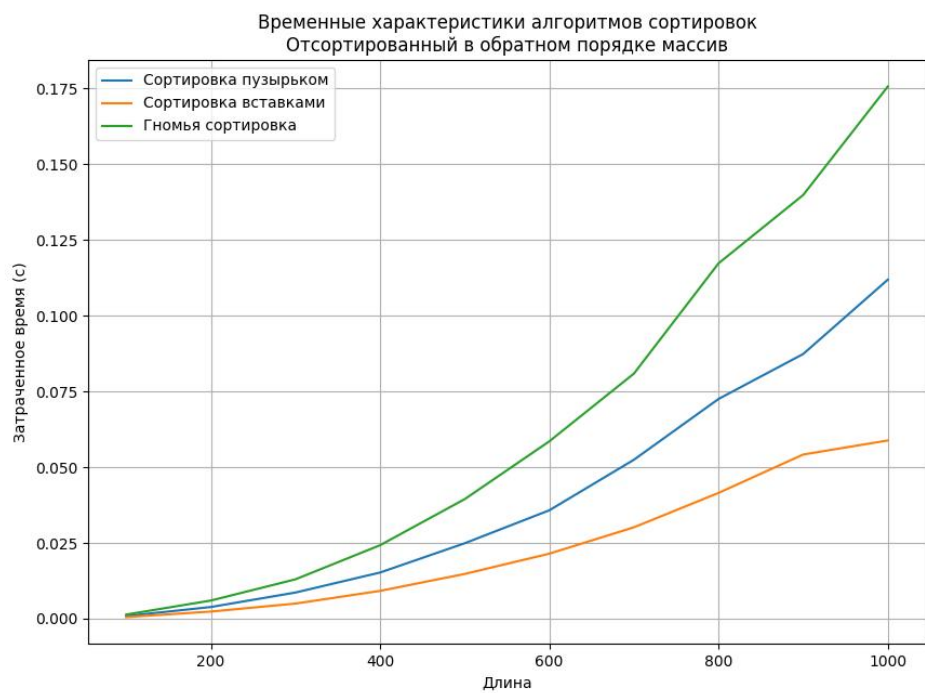


Рисунок 4.3 – Отсортированный в обратном порядке массив

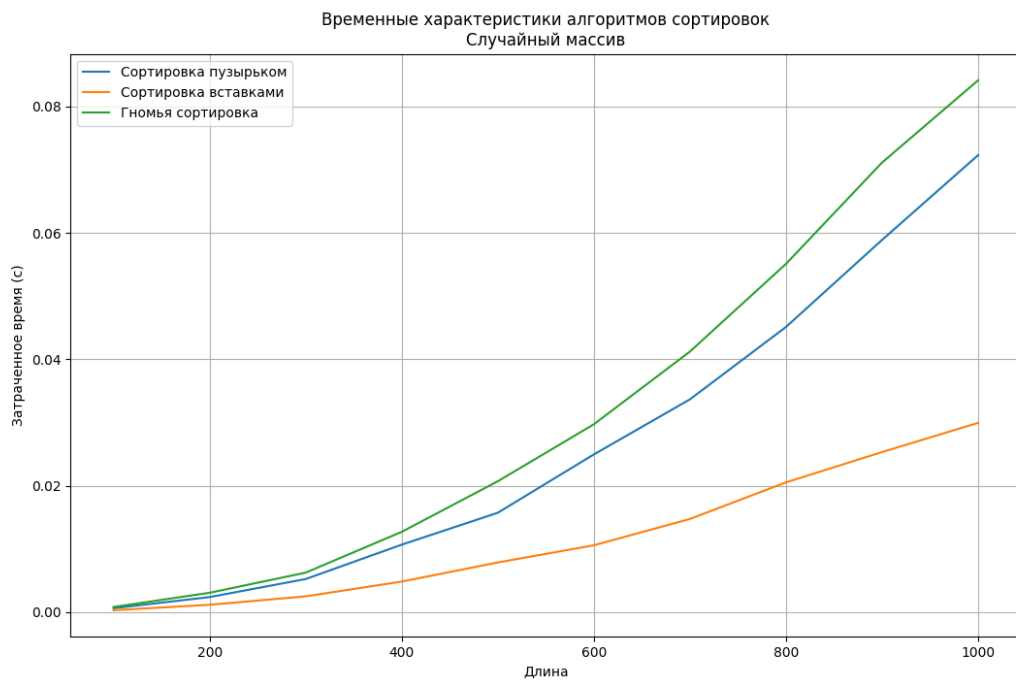


Рисунок 4.4 – Случайный массив

## Вывод

Исходя из полученных результатов, гномья сортировка при случайно заполненном массиве, а также при обратно отсортированном работает дольше всех (примерно в 1.5 дольше, чем сортировка пузырьком и в 2 раза дольше сортировки вставками), при этом сортировка методом вставок показала себя лучше всех. Что касается уже отсортированных данных, то лучше всего себя здесь показала гномья сортировка, в то время, как метод пузырьком оказался худшим.

Теоретические результаты замеров и полученные практически результаты совпадают.

# Заключение

Цель, которая была поставлена в начале лабораторной работы была достигнута, а также в ходе выполнения лабораторной работы были решены следующие задачи:

- изучены и реализованы алгоритмы сортировки: пузырьком, вставками, гномья;
- проведено тестирование по времени для выбранных сортировок;
- проведен сравнительный анализ трудоемкости алгоритмов на основе теоретических расчетов;
- проведен сравнительный анализ реализаций алгоритмов по затраченному процессорному времени и памяти;
- описаны и обоснованы полученные результаты в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

## Список использованных источников

- [1] Сортировка вставками [Электронный ресурс]. Режим доступа: <https://kvodo.ru/sortirovka-vstavkami-2.html> (дата обращения: 11.11.2022).
- [2] Сортировка пузырьком [Электронный ресурс]. Режим доступа: <https://kvodo.ru/puzyirkovaya-sortirovka.html> (дата обращения: 11.11.2022).
- [3] Гномья сортировка [Электронный ресурс]. Режим доступа: <https://kvodo.ru/gnome-sorting.html> (дата обращения: 11.11.2022).
- [4] М. В. Ульянов Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ. 2007.
- [5] Welcome to Python [Электронный ресурс]. Режим доступа: <https://www.python.org> (дата обращения: 11.11.2022).
- [6] time — Time access and conversions [Электронный ресурс]. Режим доступа: <https://docs.python.org/3/library/time.html#functions> (дата обращения: 11.11.2022).
- [7] Ubuntu 22.04.1 LTS (Jammy Jellyfish) [Электронный ресурс]. Режим доступа: <https://releases.ubuntu.com/22.04/> (дата обращения: 11.11.2022).
- [8] Межгосударственный стандарт. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. ГОСТ 19.701–90.
- [9] Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. ГОСТ 7.32–2017.