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 Кузнецова А. В.

Группа ИУ7-51Б

Оценка (баллы)

Преподаватель Волкова Л. Л.

Москва — 2022 г.

Оглавление

Введение

В современном мире приходится работать с большими массивами данных. Для упорядочивания данных и последующей их обработки проводится специальная операция по представлению данных в порядке увеличения (или уменьшения) их значения. Данную операцию называют сортировкой. В данной работе приведено сравнение трех известных методов сортировки данных [1].

Целью данной лабораторной работы является изучение алгоритмов сортировки. Для достижения поставленной цели требуется решить задачи, представленные ниже.

1. Реализация алгоритма сортировки бусинами.
2. Реализация алгоритма сортировки подсчетом.
3. Реализация алгоритма гномьей сортировки.
4. Создание схем изучаемых алгоритмов.
5. Оценка трудоёмкости алгоритмов.
6. Определение средств программной реализации алгоритмов.
7. Выполнение замеров процессорного времени работы реализаций алгоритмов сортировки.
8. Проведение сравнительного анализа по времени работы реализаций алгоритмов сортировки.
9. Подготовка отчета о выполненной лабораторной работе.

1 Аналитическая часть

В данном разделе представлено теоретическое описание алгоритма гномьей сортировки, сортировки бусинами и подсчетом.

1.1 Алгоритм сортировки бусинами

Рассмотрим массив arr из n положительных целых чисел, которые нужно отсортировать, и предположим, что максимальный элемент массива arr — число m. Тогда каркас с бусинами должен иметь не менее m стержней и n уровней (см. рис. 1.1 [2]).

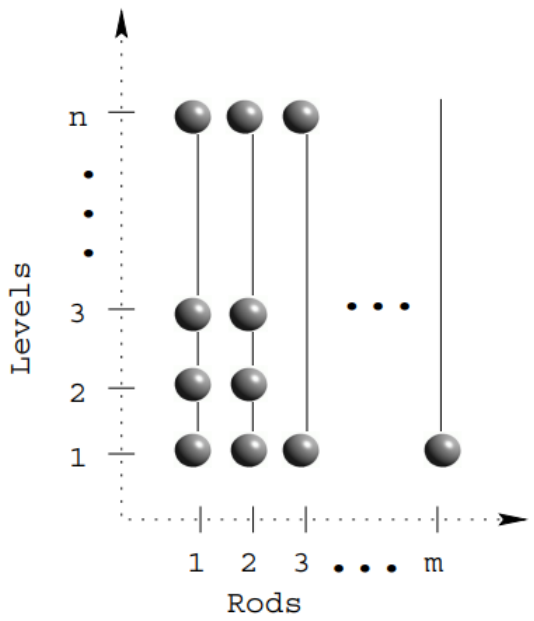

Рисунок 1.1 – Каркас с бусинами

Идея алгоритма сортировки бусинами заключается в том, чтобы для каждого элемента x массива arr надеть на каждый стержень по бусине, начиная от 1-го стержня до x-го стержня. Таким образом, бусины, представленные на каждом уровне, начиная с n-го уровня до 1-го уровня представляют массив arr в порядке возрастания [2].

1.2 Алгоритм сортировки подсчетом

Сортировка подсчетом является сортировкой без сравнений (см. пример на рис.1.2). Рассмотрим массив $A[0..n-1]$, содержащий неотрицательные целые числа меньшие k. Алгоритм сортировки подсчетом состоит из следующих шагов:

- заведем массив $C[0..k-1]$;
- посчитаем в $C[i]$ количество вхождений элемента i в массиве A;
- запишем в массив A все элементы C по $C[i]$ раз.

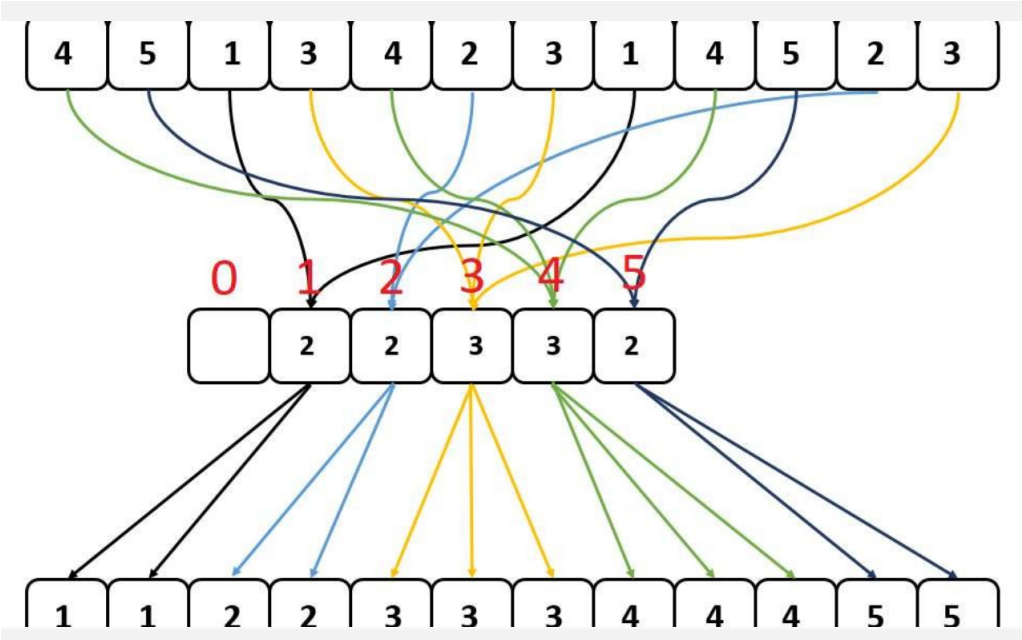

Рисунок 1.2 – Сортировка подсчетом

1.3 Алгоритм гномьей сортировки

Для начала сравнения указатель ставится на второй элемент массива. После этого происходит сравнение текущего и предыдущего элементов. Если порядок соблюден — происходит переход к следующему элементу, если нет, то элементы меняются местами и указатель в цикле переходит к предыдущему элементу. Цикл сортировки заканчивается в тот момент, когда номер указателя становится равным длине массива [3].

Вывод

В данном разделе были описаны основные положения алгоритмов сортировки: бусинами, подсчетом и гномьей сортировки.

2 Конструкторская часть

В данном разделе будут рассмотрены схемы вышеизложенных алгоритмов, а также вычислена их трудоемкость.

2.1 Разработка алгоритмов

На рисунке 2.1 представлен алгоритм сортировки бусинами.

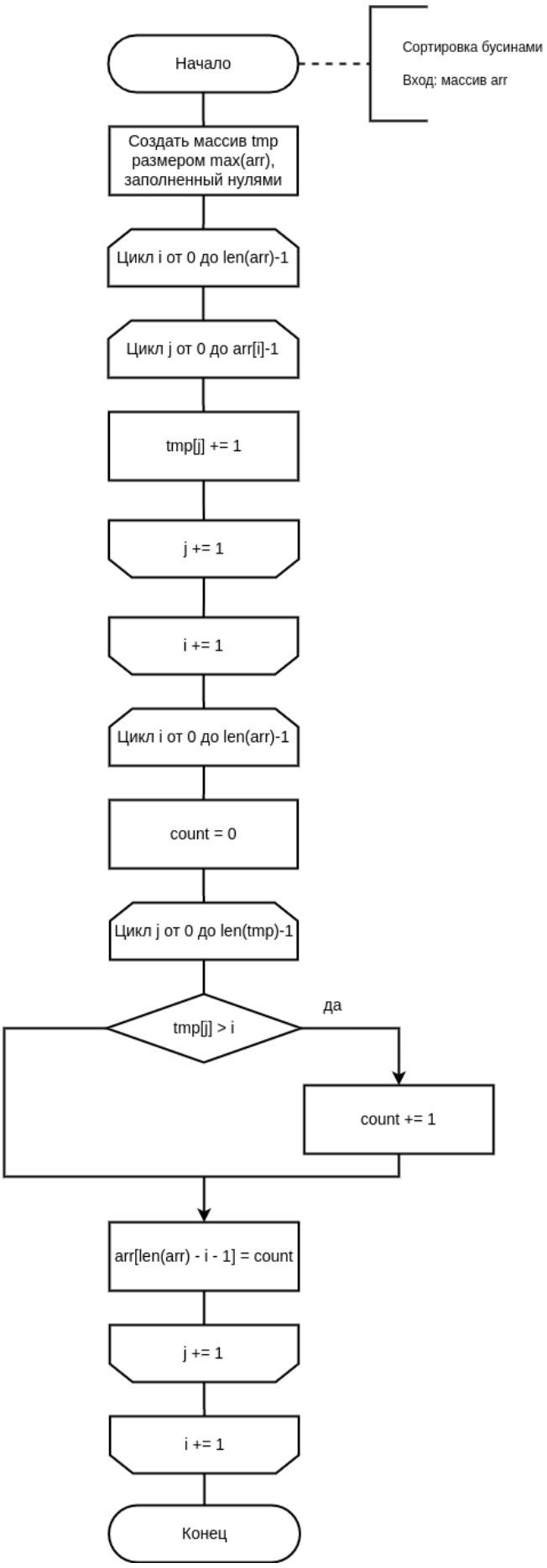

Рисунок 2.1 – Схема алгоритма сортировки бусинами

На рисунке 2.2 представлен алгоритм сортировки подсчетом.

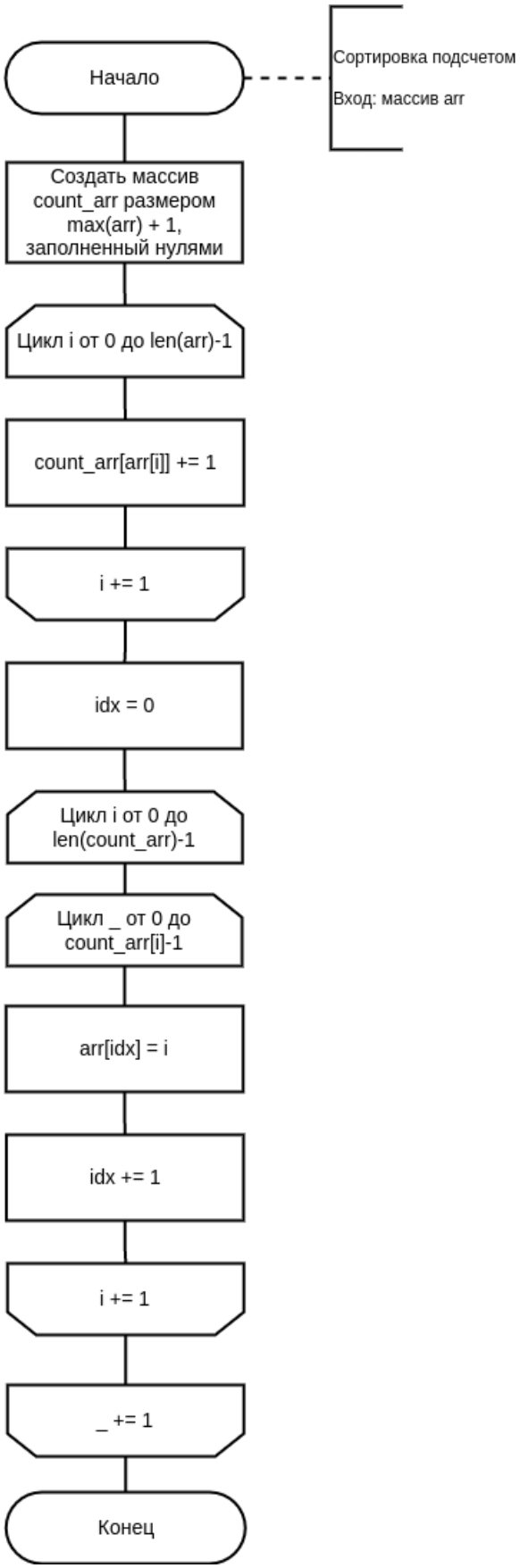

Рисунок 2.2 – Схема алгоритма сортировки подсчетом

На рисунке 2.3 представлен алгоритм гномьей сортировки.

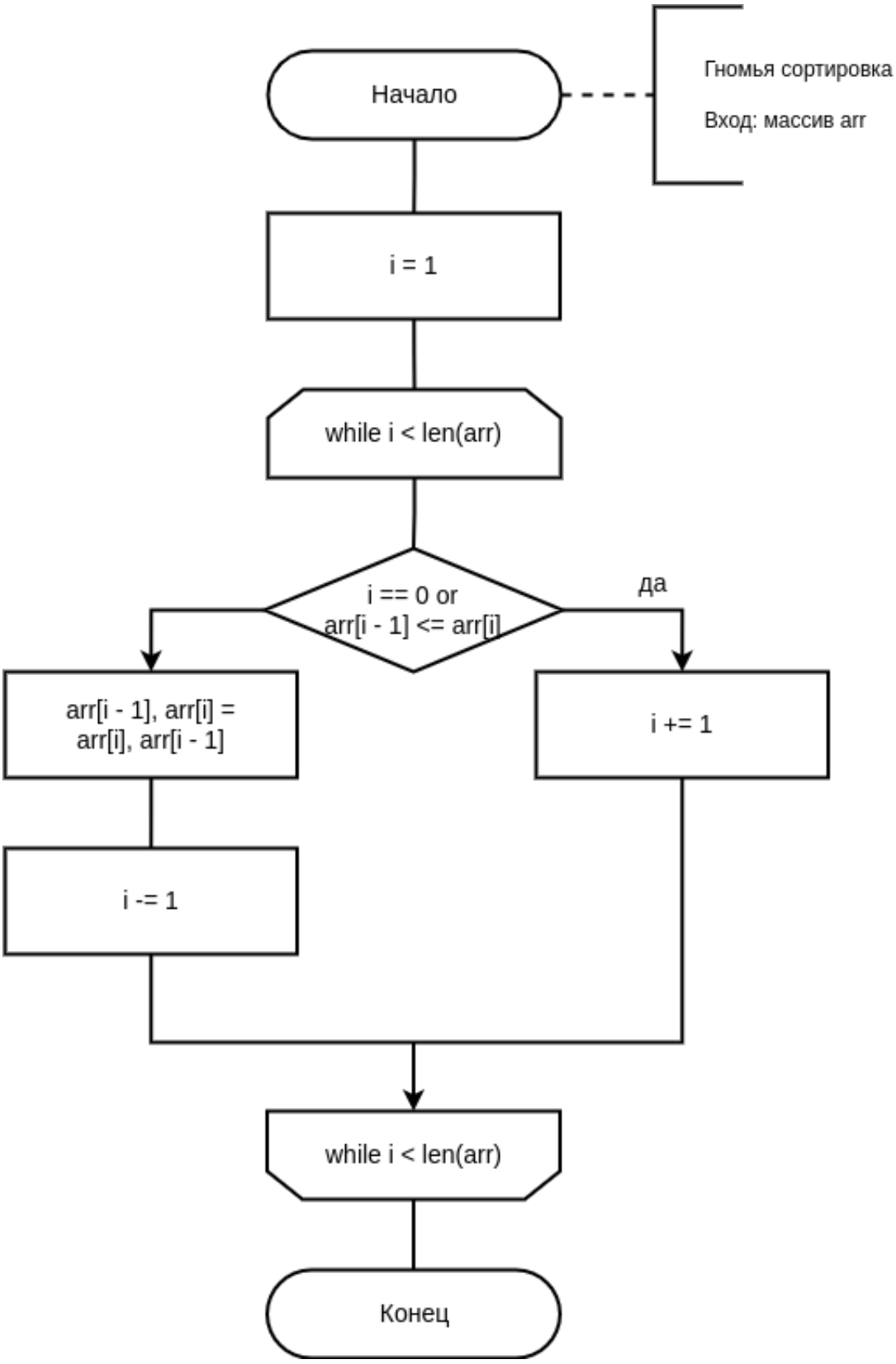

Рисунок 2.3 – Схема алгоритма гномьей сортировки

2.2 Вычислительная модель оценки трудоемкости

1. Трудоемкость следующих базовых операций равна 2:

$$/, \%, * \tag{2.1}$$

2. Трудоемкость следующих базовых операций единична:

$$=, + =, - =, ++, --, ==, ! =, <, <=, >, >=, +, -, <<, >>, [] \tag{2.2}$$

3. Трудоемкость цикла:

$$f_{\text{цикла}} = f_{\text{иниц.}} + f_{\text{сравнения}} + N_{\text{итераций}}(f_{\text{тела}} + f_{\text{инкремент}} + f_{\text{сравнения}}) \tag{2.3}$$

4. Трудоемкость условного оператора:

$$f_{if} = f_{\text{выч. условия}} + \begin{bmatrix} min(f_1, f_2), & \text{лучший случай,} \\ max(f_1, f_2), & \text{худший случай.} \end{bmatrix} \tag{2.4}$$

2.3 Вычисление трудоемкости алгоритмов

2.3.1 Алгоритм сортировки бусинами

Трудоёмкость алгоритма сортировки бусинами в лучшем случае, когда массив заполнен нулями, равна $O(n)$, в худшем, когда массив запоолнен различными положительными числами, $O(S)$, где S — сумма всех чисел в массиве, n — количество элементов в массиве [2].

2.3.2 Алгоритм сортировки подсчетом

Трудоёмкость алгоритма сортировки подсчетом равна

$$f_{counting\_sort} = 5 + n + 2 + n(2+3) + 1 + 2 + k(3+2) + \\ (3+3) \cdot (n+k) = 10 + 6n + 5k + 6(n+k) \approx 6(n+k) = O(n+k), \quad (2.5)$$

где n — количество элементов в массиве, k — диапазон значений.

В лучшем случае, когда $n >> k$, трудоемкость равна $O(n)$, в худшем — $O(n+k)$.

2.3.3 Алгоритм гномьей сортировки

Трудоёмкость алгоритма гномьей сортировки в лучшем случае, когда массив отсортирован, равна

$$f_{gnome\_sort\_best} = 1 + 1 + (n-1)(6+1) = 7n - 5 \approx 7n = O(n) \quad (2.6)$$

Трудоёмкость в худшем случае, когда массив отсортирован в обратном порядке, равна (2.7):

$$f_{gnome\_sort\_worst} = 1 + 1 + (n-1)(6 + (n-1) \cdot (7+1)) = \\ 2 + (n-1) \cdot (8n-2) = 8n^2 - 10n + 4 \approx 8n^2 = O(n^2), \quad (2.7)$$

где n — количество элементов в массиве.

Вывод

На основе теоретических данных, полученных из аналитического раздела, были построены схемы требуемых алгоритмов. Кроме того, была произведена оценка трудоемкости алгоритмов.

3 Технологическая часть

В данном разделе приведены средства реализации, сведения о модулях программы, листинги кода, тесты.

3.1 Средства реализации

В качестве языка программирования для реализации данной лабораторной работы был выбран язык Python [4]. Данный выбор обусловлен наличием массивов и необходимой функциональности для замеров времени.

Замеры времени проводились при помощи функции process_time_ns из библиотеки time [5].

3.2 Сведения о модулях программы

Данная программа разбита на следующие модули:

- main.py — файл, содержащий точку входа в программу, в нем происходит вызов алгоритмов;
- sort.py — файл, содержащий алгоритмы сортировки;
- array.py — файл, содержащий функции работы с массивом;
- measurements.py — файл, содержащий функции замеров времени работы алгоритмов;

3.3 Реализация алгоритмов

В листингах 3.1–3.3 представлены реализации алгоритмов сортировки: бусинами, подсчетом, гномьей.

Листинг 3.1 – Алгоритм сортировки бусинами

```
def bead_sort(arr):
    tmp = [0] * max(arr)

    for i in range(len(arr)):
        for j in range(arr[i]):
            tmp[j] += 1

    for i in range(len(arr)):
        count = 0
        for j in range(len(tmp)):
            if tmp[j] > i:
                count += 1
        arr[len(arr) - i - 1] = count
```

Листинг 3.2 – Алгоритм сортировки подсчетом

```
def counting_sort(arr):
    count_arr = [0] * (max(arr) + 1)

    for i in range(len(arr)):
        count_arr[arr[i]] += 1

    idx = 0
    for i in range(len(count_arr)):
        for _ in range(count_arr[i]):
            arr[idx] = i
            idx += 1
```

Листинг 3.3 – Алгоритм гномьей сортировки

```
def gnome_sort(arr):
    i = 1
    while i < len(arr):
        if i == 0 or arr[i - 1] <= arr[i]:
            i += 1
        else:
            arr[i - 1], arr[i] = arr[i], arr[i - 1]
            i -= 1
```

3.4 Тестирование

В таблице 3.1 приведены функциональные тесты для алгоритмов сортировки.

Таблица 3.1 – Функциональные тесты

Входной массив	Ожидаемый результат
[]	[]
[1]	[1]
[1, 2, 3, 4, 5]	[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]	[1, 2, 3, 4, 5]
[3, 7, 1, 5, 9]	[1, 3, 5, 7, 9]
[5, 3, 5, 6, 3, 3]	[3, 3, 3, 5, 5, 6]

При проведении функционального тестирования, полученные результаты работы программы совпали с ожидаемыми. Таким образом, функциональное тестирование пройдено успешно.

Вывод

Были реализованы алгоритмы сортировки бусинами, сортировки подсчетом и гномьей сортировки и проведено тестирование их реализаций.

4 Исследовательская часть

В данном разделе будут приведены примеры работы программы, и будет проведен сравнительный анализ реализованных алгоритмов сортировки по затраченному процессорному времени.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось исследование, приведены ниже:

- операционная система Manjaro Linux [6];
- память 7,6 ГБ;
- процессор 8 × Intel® Core™ i5-10210U CPU @ 1.60 ГГц [7].

4.2 Демонстрация работы программы

На рисунке 4.1 приведен пример работы программы. На этом рисунке пользователь выбирает из меню пункт 2 — сортировку подсчетом, вводит исходный массив и получает на выходе отсортированный массив.

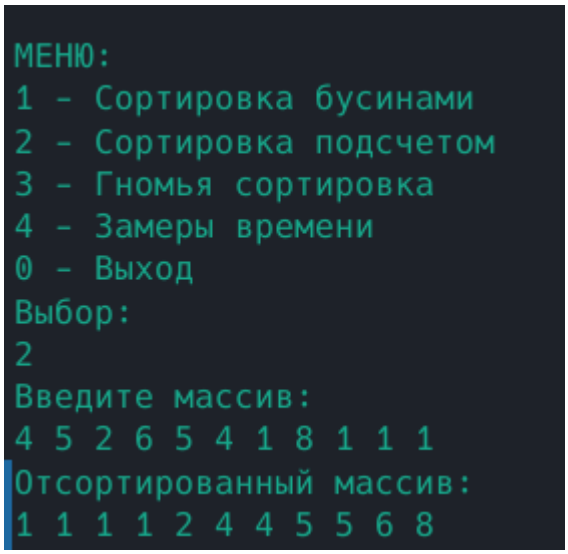

Рисунок 4.1 – Пример работы программы

4.3 Временные характеристики

Функция process_time_ns из библиотеки time языка программирования Python возвращает процессорное время в наносекундах.

Замеры проводились для массивов длиной от 100 до 1000.

На рисунке 4.2 приведены результаты замеров времени алгоритмов сортировки для лучшего случая.

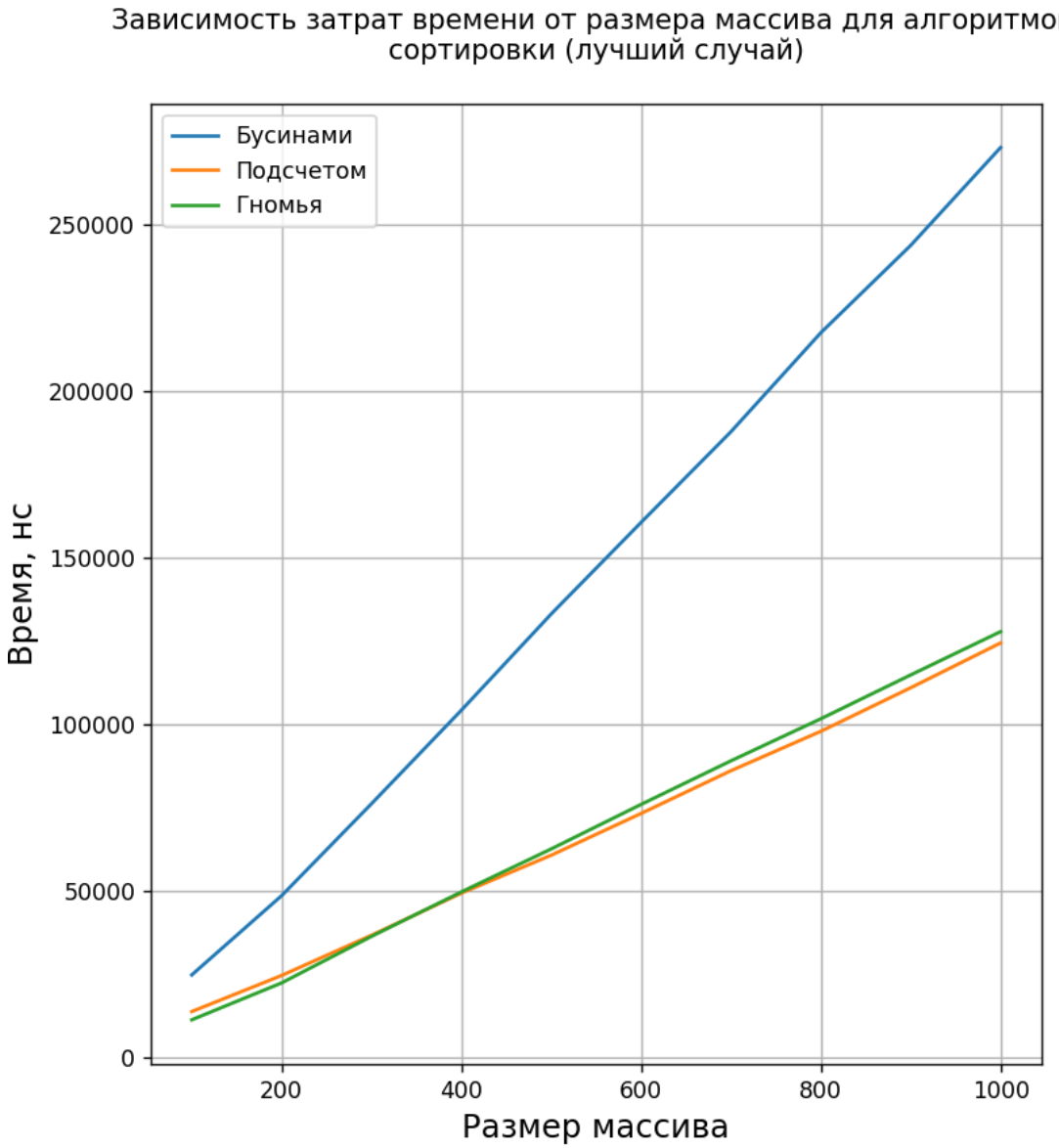

Рисунок 4.2 – Сравнение по времени алгоритмов сортировки (лучший случай)

На рисунке 4.3 приведены результаты замеров времени алгоритмов сортировки для худшего случая.

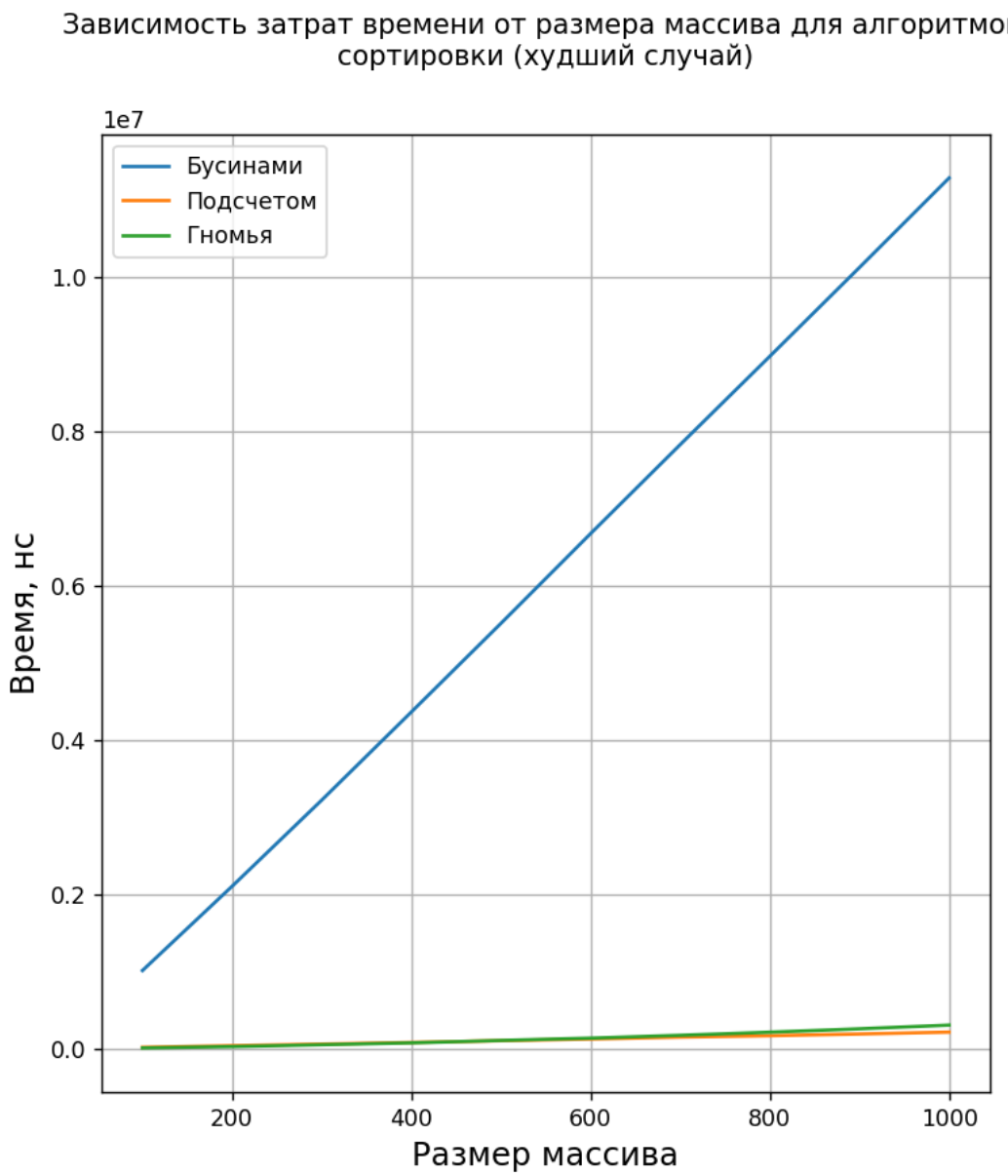

Рисунок 4.3 – Сравнение по времени алгоритмов сортировки (худший случай)

На рисунке 4.4 приведены результаты замеров времени алгоритмов сортировки для произвольного случая.

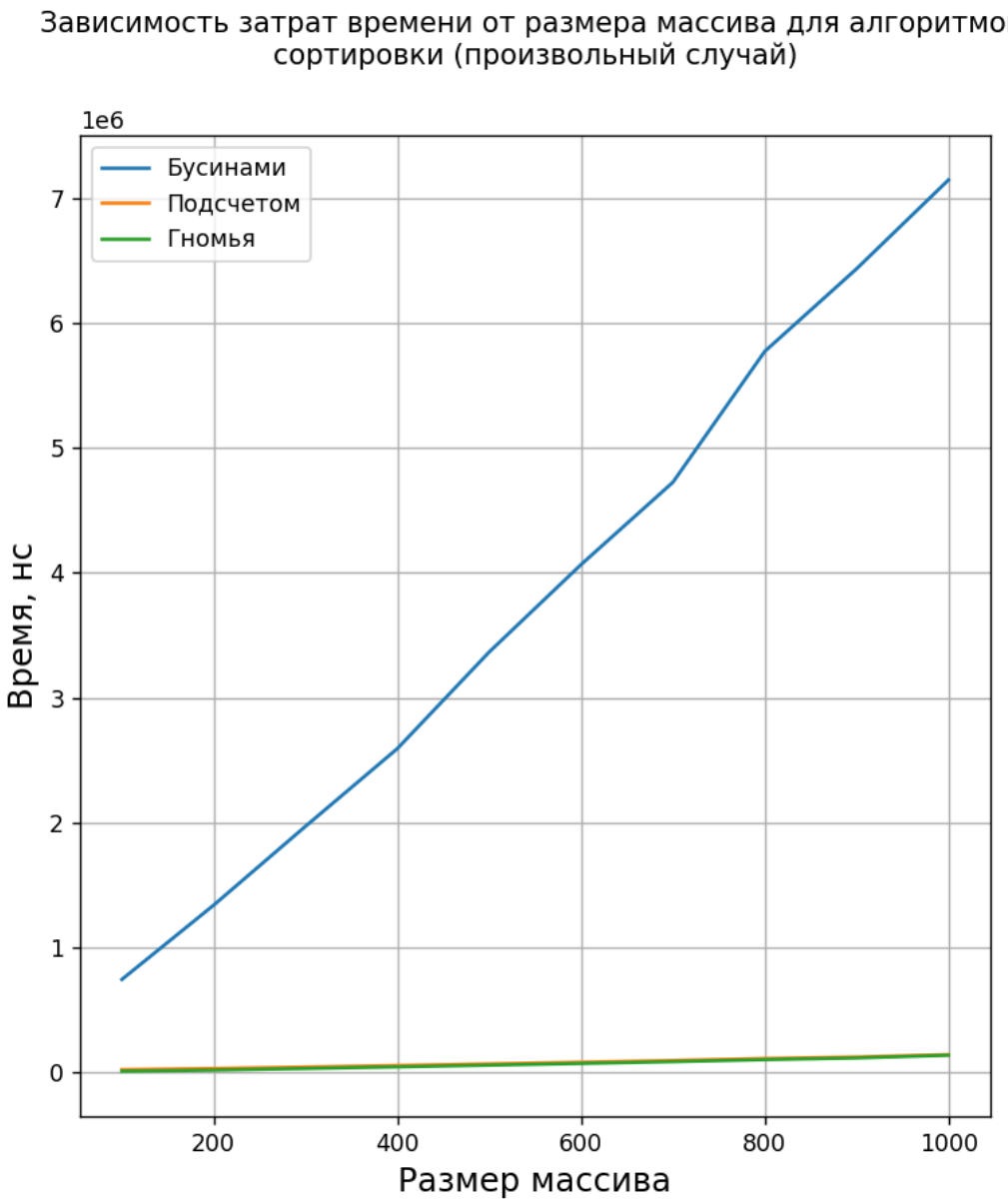

Рисунок 4.4 – Сравнение по времени алгоритмов сортировки (произвольный случай)

4.4 Вывод

Приведенные характеристики времени показывают, что наименее эффективным по времени алгоритмом является сортировка бусинами. При этом результаты замеров времени для сортировки подсчетом и гномьей сортировки практически одинаковые для массивов, заполненных случайными элементами. Но для лучшего и худшего случаев выигрывает по времени

сортировка подсчетом.

Заключение

В результате выполнения данной лабораторной работы были рассмотрены алгоритмы сортировки (бусинами, подсчетом, гномьей), построены схемы, соответствующие данным алгоритмам. Можно сделать вывод, что сортировку подсчетом стоит применять для массивов, у которых количество элементов намного превышает диапазон значений. Сортировка бусинами малоэффективна в программной реализации. Гномья сортировка работает быстрее с отсортированными массивами и медленнее с массивами отсортированными в обратном порядке. Из результатов замеров времени следует, что сортировка подсчетом и гномья сортировка для случайно заполненных массивов работают практически с одинаковой скоростью. При этом в лучшем и худшем случае выигрывает по времени сортировка подсчетом и поэтому рекомендуется к применению. Сортировка бусинами значительно медленнее.

В рамках выполнения работы цель достигнута: изучены алгоритмы сортировки.

Решены все задачи:

- реализован алгоритм сортировки бусинами;
- реализован алгоритм сортировки подсчетом;
- реализован алгоритм гномьей сортировки;
- созданы схемы изучаемых алгоритмов;
- оценены трудоёмкости алгоритмов;
- определены средства программной реализации алгоритмов;
- выполнены замеры процессорного времени работы реализаций алгоритмов сортировки;
- проведен сравнительный анализ по времени работы реализаций алгоритмов сортировки;
- подготовлен отчета о выполненной лабораторной работе.

Список используемых источников

[1] Сравнительный анализ времени выполнения сортировки по алгоритму Шелла, гномьей сортировки и сортировки перемешиванием [Электронный ресурс]. Режим доступа: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-vremeni-vypolneniya-sortirovki-po-algoritmu-shella-gnomiey-sortirovki-i-sortirovki-peremeshivaniem/viewer (дата обращения: 21.10.2022).

[2] Bead-Sort: A Natural Sorting Algorithm [Электронный ресурс]. Режим доступа: https://www.cs.auckland.ac.nz/research/groups/CDMTCS/researchreports/171joshua.pdf (дата обращения: 21.10.2022).

[3] Гномья сортировка [Электронный ресурс]. Режим доступа: https://kvodo.ru/gnome-sorting.html (дата обращения: 21.10.2022).

[4] Welcome to Python [Электронный ресурс]. Режим доступа: https://www.python.org (дата обращения: 21.10.2022).

[5] time — Time access and conversions [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/time.html#functions (дата обращения: 21.10.2022).

[6] Manjaro Linux [Электронный ресурс]. Режим доступа: https://manjaro.org (дата обращения: 21.10.2022).

[7] Процессор Intel® Core™ i5-10210U [Электронный ресурс]. Режим доступа: https://ark.intel.com/content/www/ru/ru/ark/products/195436/intel-core-i510210u-processor-6m-cache-up-to-4-20-ghz.html (дата обращения: 21.10.2022).