|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования |
| **«МИРЭА – Российский технологический университет»** |
| **РТУ МИРЭА** |
|  |

|  |  |
| --- | --- |
| **Отчет по выполнению практического задания № 3** | |
| **Тема:** | |
| **«Определение эффективного алгоритма сортировки на основе эмпирического и асимптотического методов анализа»** | |
| Дисциплина: «Структуры и алгоритмы обработки данных» | |
|  | Выполнил студент: Данюков К.А. |
|  | Группа: ИКБО-74-23 |

Москва – 2024

СОДЕРЖАНИЕ

[1 ЦЕЛЬ 4](#_gjdgxs)

[2 ЗАДАНИЕ №1 5](#_30j0zll)

[2.1 Формулировка задачи 5](#_1fob9te)

[2.2 Описание выполнения и блок-схема алгоритма пирамидальной сортировки 6](#_2et92p0)

[2.3 Определение ситуаций лучшего, среднего и худшего случая и функции роста времени работы алгоритма пирамидальной сортировки 7](#_1t3h5sf)

2.4 Реализация алгоритма на языке C++ и проведение тестирования 7

[2.5 Описание выполнения и блок-схема алгоритма простого слияния 9](#_3rdcrjn)

[2.6 Определение ситуаций лучшего, среднего и худшего случая и функции роста времени работы алгоритма простого слияния 10](#_35nkun2)

[2.7 Реализация алгоритма на языке C++ и проведение тестирования 11](#_1ksv4uv)

[2.8 Сортировка простыми вставками 12](#_3j2qqm3)

[2.9 Сравнение трёх алгоритмов на графике 13](#_1y810tw)

[2.10 Тестирование программы алгоритма пирамидальной сортировки в лучшем и худшем случае 14](#_4i7ojhp)

[2.10.1 Тестирование программы алгоритма пирамидальной сортировки в худшем случае 14](#_2xcytpi)

2.10.2 Тестирование программы алгоритма пирамидальной сортировки в лучшем случае 16

[2.10.3 Заполнение таблицы для алгоритма пирамидальной сортировки в лучшем и худшем случае 17](#_1ci93xb)

[2.11 Тестирование программы алгоритма сортировки простым слиянием в лучшем и худшем случае 17](#_2bn6wsx)

[2.11.1 Тестирование программы алгоритма сортировки простым слиянием в худшем случае 17](#_qsh70q)

[2.11.2 Тестирование программы алгоритма сортировки простым слиянием в лучшем случае 19](#_3as4poj)

[2.11.3 Заполнение таблицы для алгоритма сортировки простым слиянием в лучшем и худшем случае 20](#_1pxezwc)

[2.12 Вывод по заданию №1 20](#_147n2zr)

[3 ЗАДАНИЕ №2 22](#_23ckvvd)

[3.1 Формулировка задачи 22](#_ihv636)

[3.2 Формулы функции роста алгоритма сортировки простой вставкой в худшем и лучшем случае 22](#_32hioqz)

3.3 Асимптотическая оценка вычислительной сложности простого алгоритма сортировки вставкой 23

[3.4 Графическое представление функции роста и полученных асимптотических оценок сверху и снизу 23](#_1hmsyys)

[3.5 Справочная информация о вычислительной сложности алгоритмов пирамидальной сортировки и простым слиянием 24](#_41mghml)

[3.6 Таблица асимптотической сложности трёх алгоритмов 24](#_2grqrue)

[3.7 Выводы по заданию №2 24](#_3fwokq0)

[6 ВЫВОДЫ 26](#_2u6wntf)

[7 ЛИТЕРАТУРА 27](#_19c6y18)

# 1 ЦЕЛЬ

Получить навыки по анализу вычислительной сложности алгоритмов сортировки и определению наиболее эффективного алгоритма.

# 2 ЗАДАНИЕ №1

## 2.1 Формулировка задачи

Вариант 7, в списке 7 (Пирамидальная сортировка, простое слияние)

Эмпирическая оценка эффективности алгоритмов.

1. Разработать алгоритм пирамидальной сортировки, реализовать код на языке С++. Сформировать таблицу 1.1 результатов эмпирической оценки сложности сортировки по формату табл. 1 для массива, заполненного случайными числами.

2. Определить ёмкостную сложность алгоритма пирамидальной сортировки.

3. Разработать алгоритм простого слияния, реализовать код на языке С++. Сформировать таблицу 1.2 результатов эмпирической оценки сортировки по формату табл. 1 для массива, заполненного случайными числами.

4. Определить ёмкостную сложность алгоритма простого слияния.

5. Добавьте в отчёт данные по работе любого из алгоритмов простой сортировки в среднем случае, полученные в предыдущей практической работе (в отчёте – таблица 1.3).

6. Представить на общем сравнительном графике зависимости Тп(n)=Cф+Mф для трёх анализируемых алгоритмов. График должен быть подписан, на нём – обозначены оси.

7. На основе сравнения полученных данных определите наиболее эффективный из алгоритмов в среднем случае (отдельно для небольших массивов при n до 1000 и для больших массивов с n>1000).

8. Провести дополнительные прогоны программ ускоренной и быстрой сортировок на массивах, отсортированных а) строго в убывающем и б) строго возрастающем порядке значений элементов. Заполнить по этим данным соответствующие таблицы 1.4, 1.5 для каждого алгоритма по формату табл. 1.

9. Сделайте вывод о зависимости (или независимости) алгоритмов сортировок от исходной упорядоченности массива на основе результатов, представленных в таблицах.

## 2.2 Описание выполнения и блок-схема алгоритма пирамидальной сортировки

Пирамидальная сортировка - алгоритм сортировки, основанный на структуре данных под названием "куча" (heap). Он был разработан в 1960-х годах и является несравненно эффективным, особенно для больших массивов данных.

Основная идея пирамидальной сортировки заключается в следующем:

1. Сначала строится "пирамида" из исходного массива. Пирамида представляет собой специальную структуру данных, удовлетворяющую условию "родительский элемент

всегда меньше или равен своим дочерним элементам". Построение пирамиды выполняется поэтапно, начиная с последнего родительского элемента и заканчивая корневым элементом массива.

2. После построения пирамиды максимальный элемент (корень пирамиды) находится в начале массива. Он меняется местами с последним элементом массива, который считается отсортированным.

3. Размер пирамиды уменьшается на 1, и выполняется операция "просеивания вниз" (sift down) для нового корня пирамиды. Это означает, что элементы перестраиваются таким образом, чтобы снова удовлетворять условию пирамиды.

4. Шаги 2 и 3 повторяются до тех пор, пока весь массив не будет отсортирован.

Реализация данного описания выполнения алгоритма представлена в виде блок-схемы (рис.1).

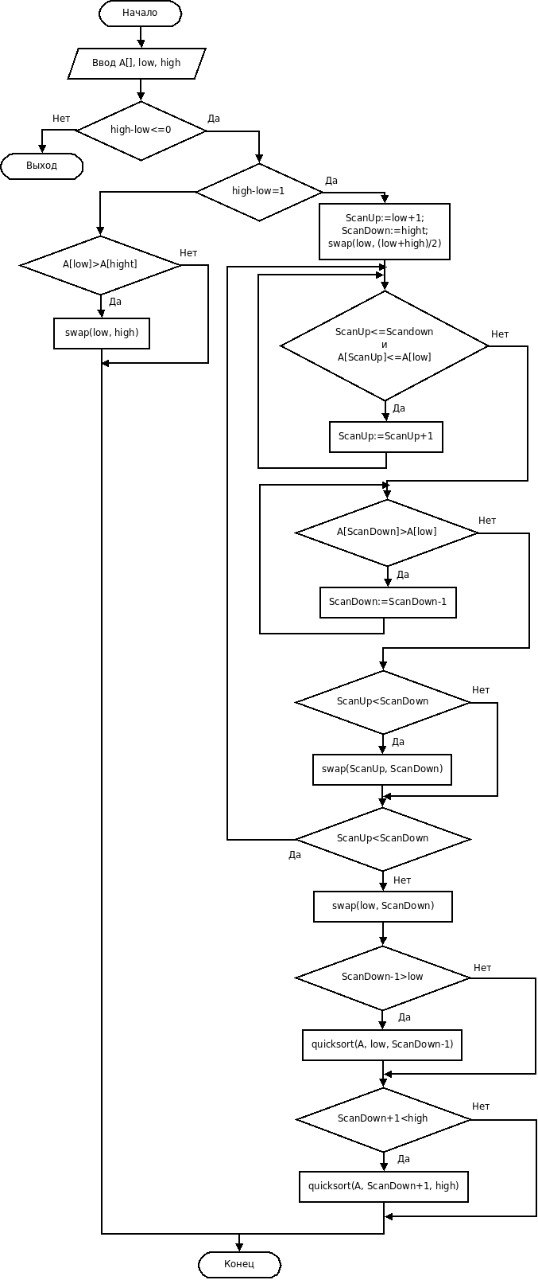


Рисунок 1 – Блок-схема алгоритма пирамидальной сортировки

## 2.3 Определение ситуаций лучшего, среднего и худшего случая и функции роста времени работы алгоритма пирамидальной сортировки

Лучший случай - массив отсортирован по возрастанию. Количество операций сравнений и перемещений равно O(n log2n).

Худший случай - массив отсортирован по убыванию. Количество операций сравнений и перемещений равно O(n log2n).

Средний случай - массив заполнен случайными значениями. Количество операций сравнений и перемещений равно O(n log2n).

Функции роста времени: лучший случай: O(n), худший случай:O(n log2n).

Время исполнения в худшем случае увеличивается квазилинейным ростом размера входного массива. Время исполнения в лучшем случае увеличивается линейно с ростом размера входного массива.

Ёмкостная сложность алгоритма будет равна O(1).

## 2.4 Реализация алгоритма на языке C++ и проведение тестирования

Реализуем алгоритм пирамидальной сортировкой на языке C++(рис.2,3). Воспользуемся библиотеками iostream, random, chrono и vector. Добавим две функции для пирамидальной сортировки.

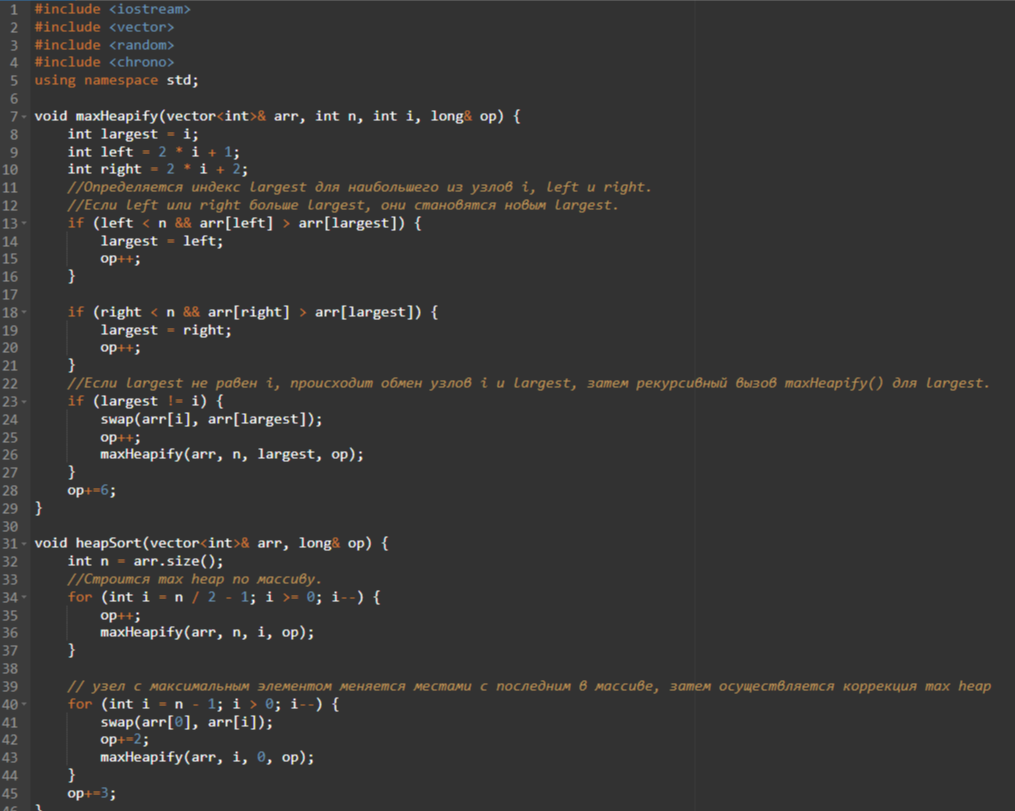


Рисунок 2 – Программа алгоритма пирамидальной сортировки



Рисунок 3 – Функция main для алгоритма пирамидальной сортировки

Проверим работоспособность программы с различными заданными размерами массива n=10 (рис.4), n=100, n=1000, n=10000, n=100000, n=1000000 и продемонстрируем результаты в таблице 1.1.

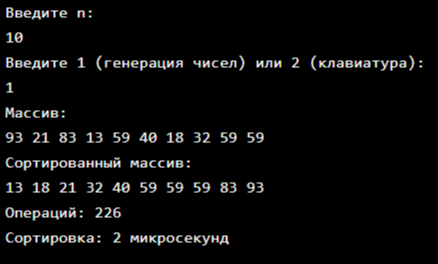


Рисунок 4 - Тестирование программы при n=10

Таблица 1.1. Сводная таблица результатов

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **Тп(n)=Cф+Mф** |
| 100 | 0.036 | 5183 |
| 1000 | 0.538 | 79143 |
| 10000 | 3.841 | 1072440 |
| 100000 | 65.567 | 13519432 |
| 1000000 | 570.111 | 162883355 |

## 2.5 Описание выполнения и блок-схема алгоритма простого слияния

Алгоритм простого слияния (merge sort) является одним из самых эффективных и популярных алгоритмов сортировки. Он основан на принципе "разделяй и властвуй", который заключается в разбиении задачи на более мелкие подзадачи, их решении и объединении результатов.

Описание выполнения алгоритма простого слияния:

* Разделение: Исходный массив разбивается на две равные (или практически равные) части.
* Рекурсивное выполнение: Каждая из двух частей сортируется отдельно с использованием того же алгоритма простого слияния. Этот шаг повторяется до тех пор, пока все части не будут отсортированы.
* Слияние: Отсортированные части объединяются обратно в исходный массив. В процессе слияния каждый элемент сравнивается с элементами другой части и помещается в соответствующую позицию.

Реализация данного описания выполнения алгоритма представлена в виде блок-схемы (рис.5).

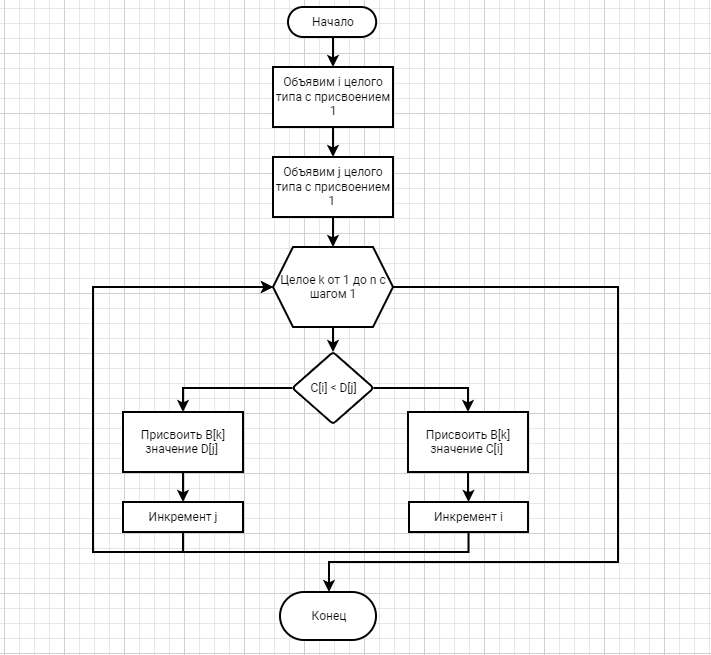


Рисунок 5 – Блок-схема алгоритма простого слияния

## 2.6 Определение ситуаций лучшего, среднего и худшего случая и функции роста времени работы алгоритма простого слияния

Лучший случай - массив уже отсортирован. Количество операций равно O(nlog2n).

Средний случай - массив заполнен случайными числами. Количество операций равно O(nlog2n).

Худший случай - массив отсортирован в обратном порядке. Количество операций равно O(n2).

Функции роста времени в лучшем случае равна O(nlog2n), а в худшем случае O(n2).

Для данного метода сортировки, время исполнения в худшем случае увеличивается квадратично с ростом размера входного массива. Следовательно, можно использовать квадратичную функцию для описания функции роста данного сортировочного метода. Время исполнения в лучшем случае увеличивается квазилинейным ростом размера входного массива.

Ёмкостная сложность алгоритма будет равна O(log2n).

## 2.7 Реализация алгоритма на языке C++ и проведение тестирования

Реализуем алгоритм пирамидальной сортировки на языке C++(рис.6,7). Используем встроенные библиотеки iostream, random, chrono и vector.

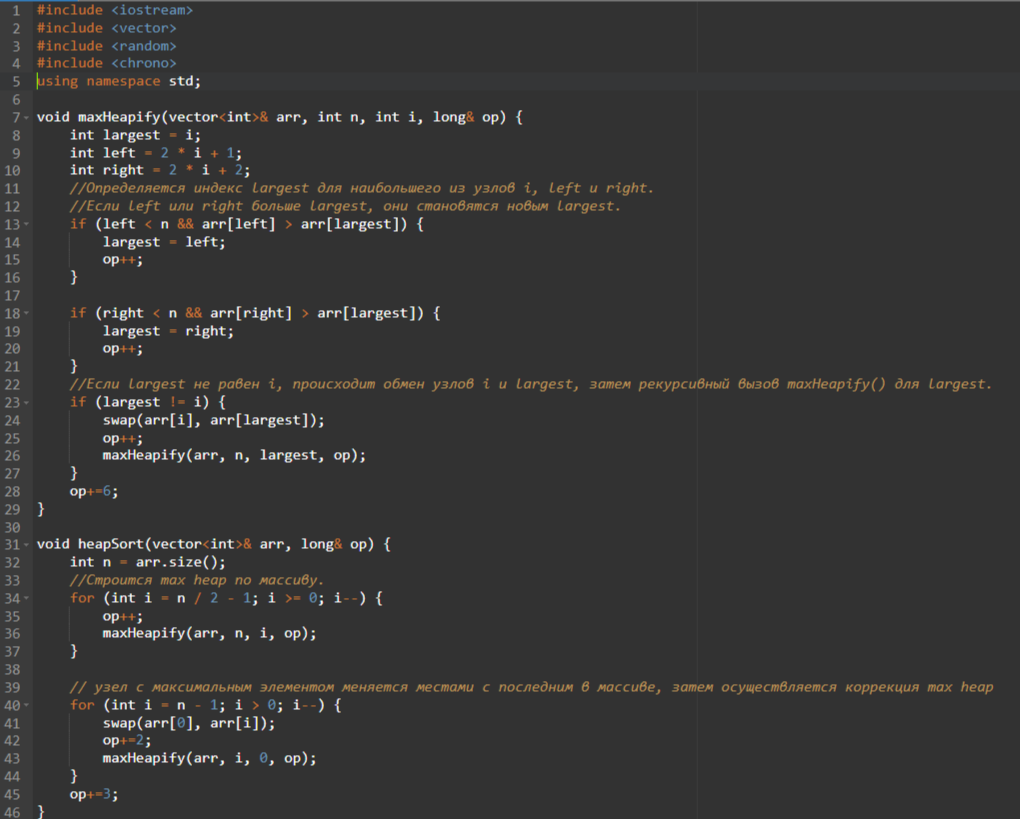


Рисунок 6 – Программа алгоритма простого слияния

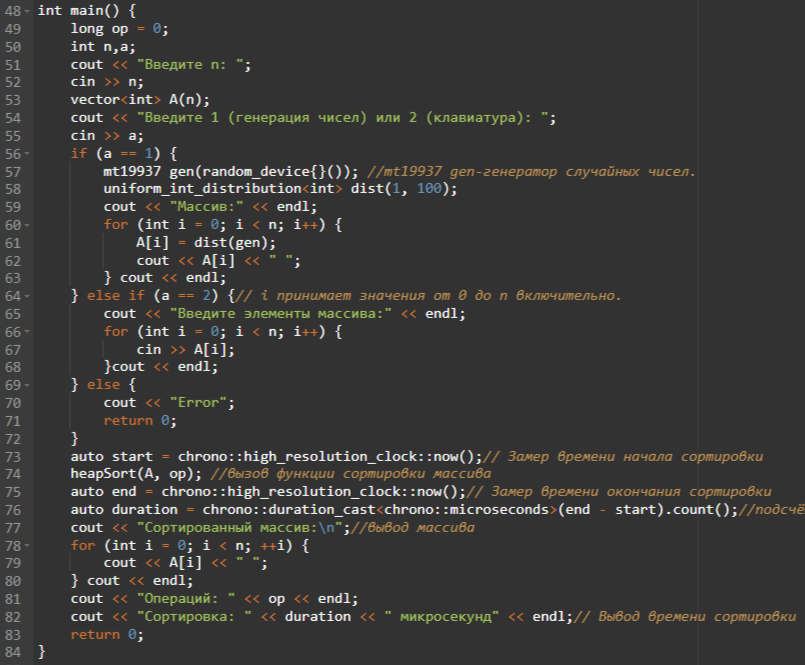


Рисунок 7 – Функция main для алгоритма простого слияния

Протестируем программу с заданным размером массива n=10 (рис.8), n=100, n=1000, n=10000, n=100000, n=1000000. Продемонстрируем результаты тестирования от n=100 до n=1000000 в таблице 1.2.

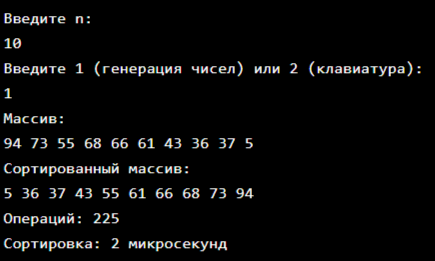


Рисунок 8 - Тестирование программы при n=10

Таблица 1.2. Сводная таблица результатов

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **Тп(n)=Cф+Mф** |
| 100 | 0.035 | 5201 |
| 1000 | 0.367 | 79231 |
| 10000 | 5.554 | 1071320 |
| 100000 | 64.809 | 13509538 |
| 1000000 | 1699.144 | 162910099 |

## 2.8 Сортировка простыми вставками

Добавим из предыдущей работы таблицу результатов тестирования простой сортировки вставками в среднем случае(табл.1.3).

Таблица 1.3. Сводная таблица результатов

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **Тп(n)=Cф+Mф** |
| 100 | 0.08 | 7099 |
| 1000 | 8.93 | 680764 |
| 10000 | 844.17 | 67125598 |
| 100000 | 87105.1 | 6767452771 |
| 1000000 | 4747060.69 | 676542874612 |

## 2.9 Сравнение трёх алгоритмов на графике

На основании данных из таблиц для среднего случая, мы посмотрим графики функции роста для трёх алгоритмов сортировки: слиянием, со смещением Дональда Кнута, а также простыми вставками. Первый график будет показывать значения до 1000 элементов, а второй - от 10000 до 1000000 элементов. Это позволит провести более детальное сравнение алгоритмов.

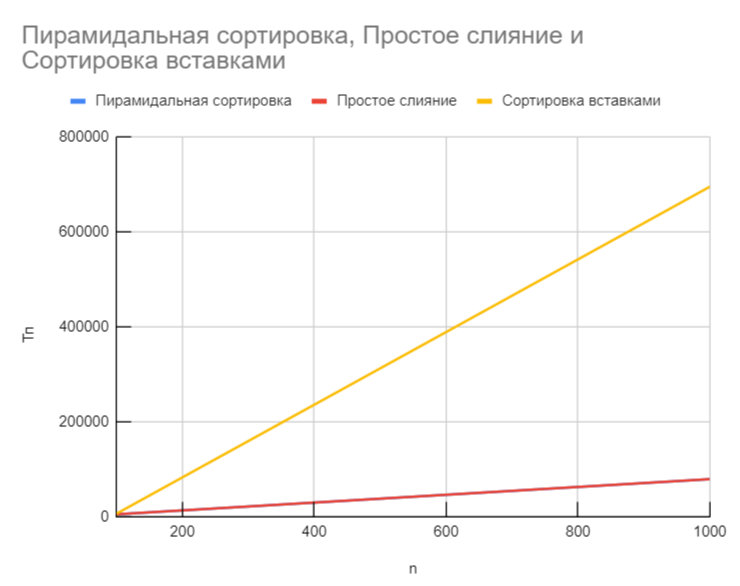


Рисунок 9 - График сравнения трёх сортировок в среднем случае при n до 1000

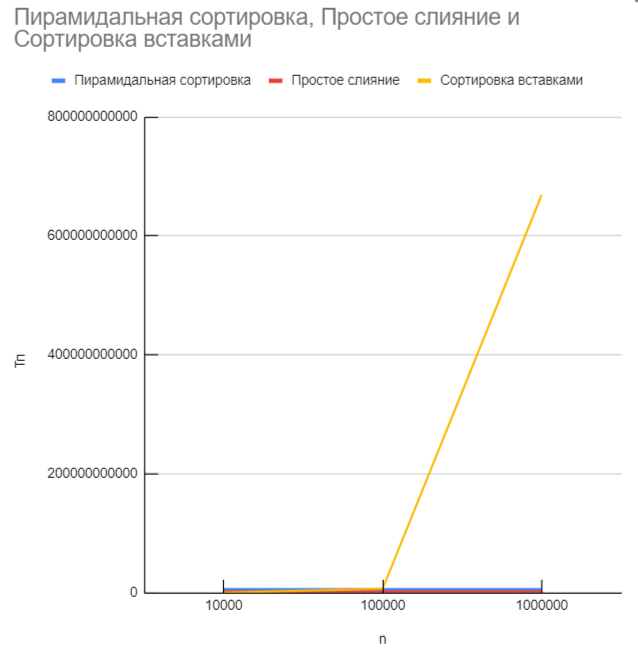


Рисунок 10 - График сравнения трёх сортировок в среднем случае при n от 10000 до 1000000

Таким образом, сложность алгоритмов в средних случаях будет различаться. Алгоритм простых вставок характеризуется наименьшей эффективностью, в то время как алгоритм сортировки простым слиянием занимает среднюю позицию по эффективности. Наиболее эффективным же алгоритмом является сортировка Шелла с использованием смещения по методу Д.Кнута вторым способом.

## 2.10 Тестирование программы алгоритма пирамидальной сортировки в лучшем и худшем случае

### **2.10.1 Тестирование программы алгоритма пирамидальной сортировки в худшем случае**

Протестируем программу с заданным размером массива n=10 (рис.12), n=100, n=1000, n=10000, n=100000, n=1000000 и отсортированными значениями по убыванию. Продемонстрируем результаты тестирования от n=100 до n=1000000 в таблице 1.4. Воспользуемся библиотекой Algorithm для сортировки по убыванию. В функцию main добавим функцию sort(рис.13). Алгоритм пирамидальной сортировки не изменяется и соответствует продемонстрированному на рисунке 2.

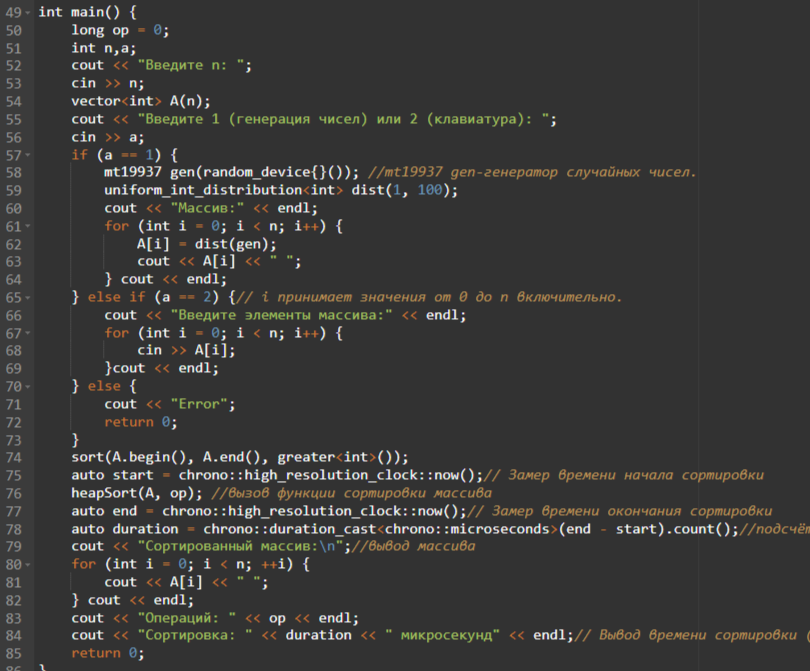


Рисунок 11 – Функция main с сортировкой по убыванию

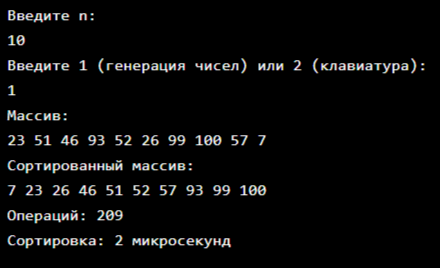


Рисунок 12 – Результаты тестирования программы при n=10 и с отсортированными значениями по убыванию

### **2.10.2 Тестирование программы алгоритма пирамидальной сортировки в лучшем случае**

Протестируем программу с заданным размером массива n=10 (рис.13), n=100, n=1000, n=10000, n=100000, n=1000000 и отсортированными значениями по возрастанию. Продемонстрируем результаты тестирования от n=100 до n=1000000 в таблице 1.4. Воспользуемся библиотекой Algorithm для сортировки по возрастанию. В функцию main добавим функцию sort(рис. 14). Алгоритм пирамидальной сортировки не изменяется и соответствует продемонстрированному на рисунке 2.

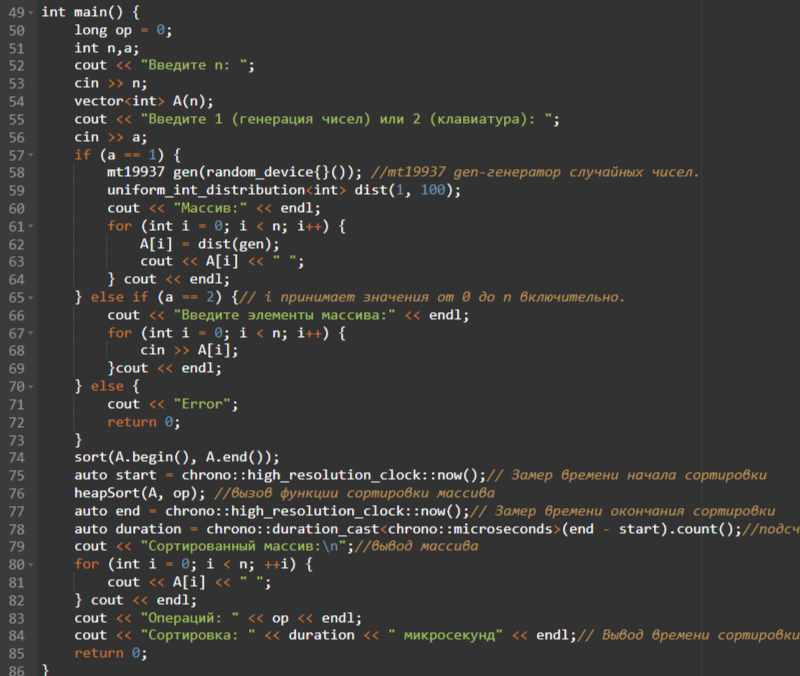


Рисунок 13 – Функция main с сортировкой по возрастанию

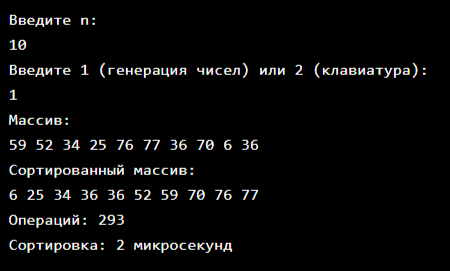


Рисунок 14 – Результаты тестирования программы при n=10 и с отсортированными значениями по возрастанию

### **2.10.3 Заполнение таблицы для алгоритма пирамидальной сортировки в лучшем и худшем случае**

Заполним таблицу 1.4 данными, полученными в результате тестирования программ.

Таблица 1.4. Сводная таблица результатов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **T(n), мс** | **Тп(n)=Cф+Mф в худшем случае** | **T(n), мс** | **Тп(n)=Cф+Mф в лучшем случае** |
| 100 | 0.022 | 4536 | 0.036 | 5595 |
| 1000 | 0.379 | 72406 | 0.562 | 81407 |
| 10000 | 4.662 | 1001240 | 5.1 | 1108712 |
| 100000 | 42.434 | 12761516 | 40.883 | 12373258 |
| 1000000 | 517.358 | 155561786 | 654.237 | 144923850 |

## 2.11 Тестирование программы алгоритма сортировки простым слиянием в лучшем и худшем случае

### **2.11.1 Тестирование программы алгоритма сортировки простым слиянием в худшем случае**

Протестируем программу с заданным размером массива n=10 (рис.16), n=100, n=1000, n=10000, n=100000, n=1000000 и отсортированными значениями по убыванию. Продемонстрируем результаты тестирования от n=100 до n=1000000 в таблице 1.5. Воспользуемся библиотекой Algorithm для сортировки по убыванию. В функцию main добавим функцию sort(рис. 15). Алгоритм простого слияния не изменяется и соответствует продемонстрированному на рисунке 6.

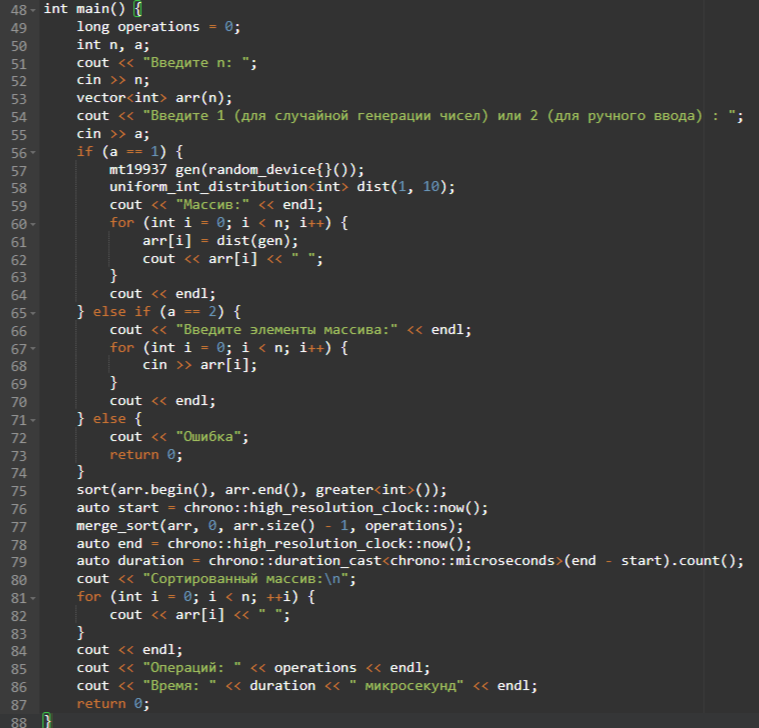


Рисунок 15 – Функция main с сортировкой по убыванию

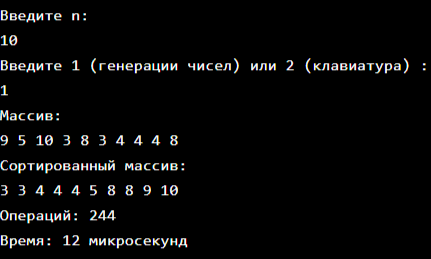


Рисунок 16 – Результаты тестирования программы при n=10 и с отсортированными значениями по убыванию

### **2.11.2 Тестирование программы алгоритма сортировки простым слиянием в лучшем случае**

Протестируем программу с заданным размером массива n=10 (рис.18), n=100, n=1000, n=10000, n=100000, n=1000000 и отсортированными значениями по возрастанию. Продемонстрируем результаты тестирования от n=100 до n=1000000 в таблице 1.5. Воспользуемся библиотекой Algorithm для сортировки по возрастанию. В функцию main добавим функцию sort(рис. 17). Алгоритм простого слияния не изменяется и соответствует продемонстрированному на рисунке 6.

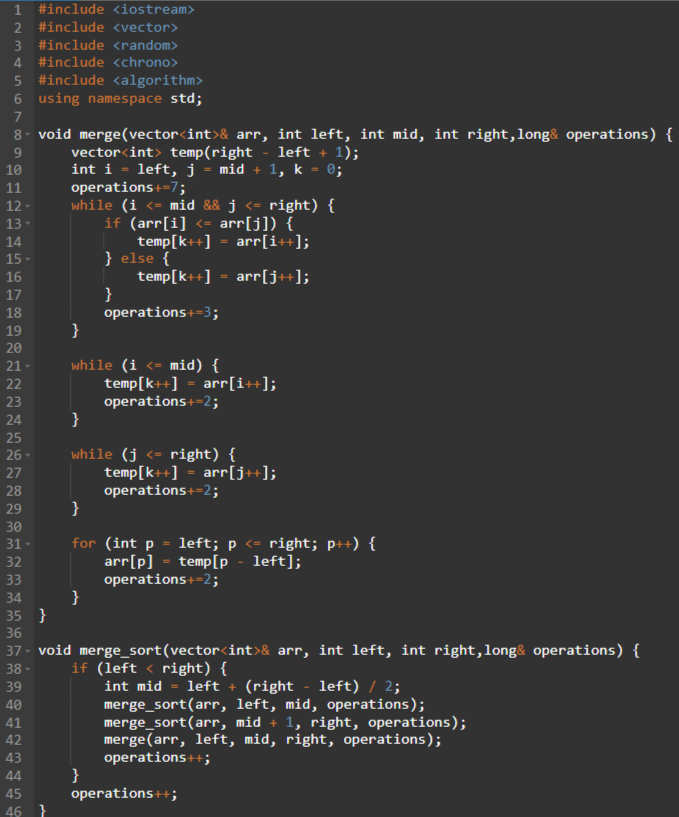


Рисунок 17 – Функция main с сортировкой по возрастанию

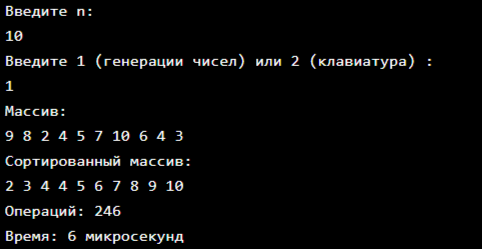


Рисунок 18 – Результаты тестирования программы при n=10 и с отсортированными значениями по возрастанию

### **2.11.3 Заполнение таблицы для алгоритма сортировки простым слиянием в лучшем и худшем случае**

Заполним таблицу 1.5 данными, полученными в результате тестирования программ.

Таблица 1.5. Сводная таблица результатов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **T(n), мс** | **Тп(n)=Cф+Mф в худшем случае** | **T(n), мс** | **Тп(n)=Cф+Mф в лучшем случае** |
| 100 | 0.057 | 4114 | 0.85 | 4035 |
| 1000 | 0.285 | 55774 | 0.295 | 54939 |
| 10000 | 3.459 | 711877 | 3.235 | 703463 |
| 100000 | 54.563 | 8633916 | 76.033 | 8529607 |
| 1000000 | 416.014 | 100678516 | 589.931 | 99872119 |

## 2.12 Вывод по заданию №1

Алгоритмы пирамидальной сортировки и простого слияния являются независимыми от исходной упорядоченности массива. Оба алгоритма имеют сложность O(n log n) в среднем и худшем случаях, что означает, что время выполнения не зависит от того, насколько массив изначально упорядочен.

Пирамидальная сортировка и простое слияние обрабатывают данные поэтапно и используют различные методы сортировки, поэтому они могут эффективно сортировать как неупорядоченные, так и частично упорядоченные массивы. Однако в случае, если массив уже упорядочен или частично отсортирован, эти алгоритмы могут работать немного быстрее из-за определенных оптимизаций, например, простое слияние может более эффективно объединять уже упорядоченные части массива.

Таким образом, пирамидальная сортировка и простое слияние не зависят от исходной упорядоченности массива в плане времени выполнения, но могут иметь различное поведение на частично упорядоченных данных.

# 

# 3 ЗАДАНИЕ №2

## 3.1 Формулировка задачи

Асимптотический анализ сложности алгоритмов

Требования по выполнению задания

1. Из материалов предыдущей практической работы приведите в отчёте формулы Тт(n) функций роста алгоритма простой сортировки вставками в лучшем и худшем случае.

2. На основе определений соответствующих нотаций получите асимптотическую оценку вычислительной сложности простого алгоритма сортировки вставками:

- в О-нотации (оценка сверху) для анализа худшего случая;

- в Ω-нотации (оценка снизу) для анализа лучшего случая.

3. Получите (если это возможно) асимптотически точную оценку вычислительной сложности алгоритма в нотации θ.

4. Реализуйте графическое представление функции роста и полученных асимптотических оценок сверху и снизу.

5. Привести справочную информацию о вычислительной сложности алгоритмов пирамидальной сортировки и простым слиянием.

6. Общие результаты свести в табл. 2.

7. Сделать вывод о наиболее эффективном алгоритме из трёх.

## 3.2 Формулы функции роста алгоритма сортировки простой вставкой в худшем и лучшем случае

Лучший случай - массив уже отсортирован. В этом случае количество операций сравнения и перемещения будет минимальным и будет составлять Тт(n)=n.

Худший случай - массив отсортирован в обратном порядке. В этом случае количество операций также будет Тт(n)=(n2-n)/2.

Средний случай - массив отсортирован в обратном порядке. В этом случае количество операций также будет Тт(n)=n2/4

## 3.3 Асимптотическая оценка вычислительной сложности простого алгоритма сортировки вставкой

Асимптотическая оценка вычислительной сложности простого алгоритма сортировки вставкой для худшего случая равна О(n2).

Асимптотическая оценка вычислительной сложности простого алгоритма сортировки вставкой для лучшего случая равна Ω(n).

Асимптотическая оценка вычислительной сложности простого алгоритма сортировки вставкой для среднего случая равна θ(n2).

Ёмкостная сложность алгоритма простой сортировки вставкой O(1).

## 3.4 Графическое представление функции роста и полученных асимптотических оценок сверху и снизу

Построим график на основе пунктов 3.2 и 3.3(рис.19).

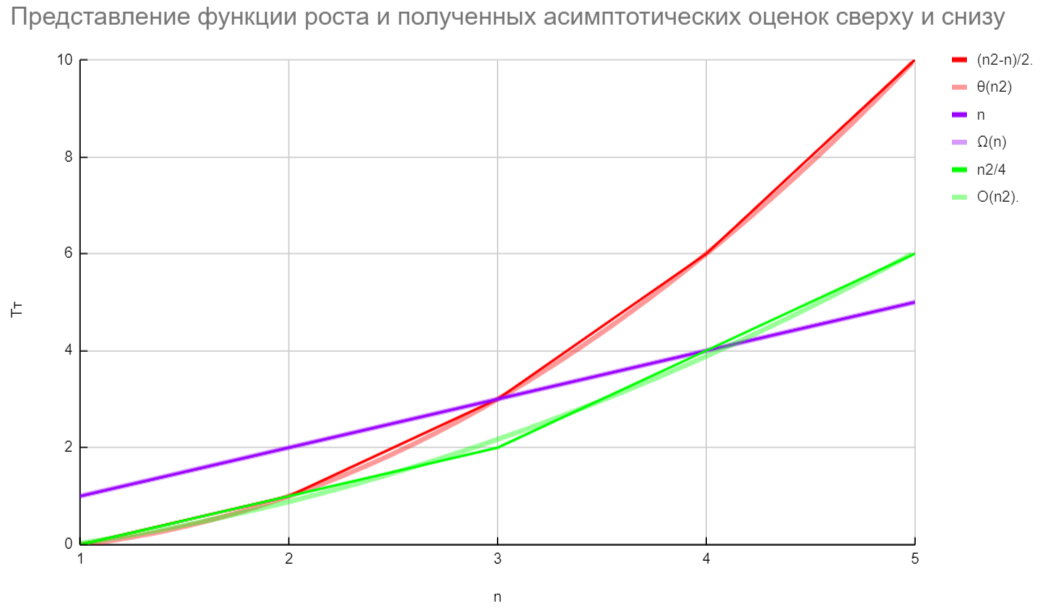


Рисунок 19 - Графическое представление функции роста и полученных асимптотических оценок сверху и снизу

## 3.5 Справочная информация о вычислительной сложности алгоритмов пирамидальной сортировки и простым слиянием

Алгоритм пирамидальной сортировки имеет временную сложность O(nlog2n) в худшем случае, Ω(nlog2n) в лучшем случае и θ(nlog2n) в среднем случае.

Ёмкостная сложность алгоритма пирамидальной сортировки равна O(1).

Алгоритм простого слияния имеет временную сложность O(nlog2n) в худшем случае, Ω(nlog2n) в лучшем случае и θ(nlog2n) в среднем случае. Емкостная сложность этого алгоритма равна O(n).

## 3.6 Таблица асимптотической сложности трёх алгоритмов

Заполним таблицу 2 асимптотической сложности алгоритма для алгоритмов сортировки простой вставки,пирамидальной сортировки и простым слиянием.

Таблица 2. Сводная таблица результатов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм | Асимптотическая сложность алгоритма | | | |
| Наихудший случай (сверху) | Наилучший случай (снизу) | Средний случай (точная оценка) | Ёмкостная сложность |
| Простая вставка | О(n2) | Ω(n) | θ(n2) | О(1) |
| Пирамидальная сортировка | О(nlog2n) | Ω(nlog2n) | θ(nlog2n) | О(1) |
| Простым слиянием | О(nlog2n) | Ω(nlog2n) | θ(nlog2n) | О(n) |

## 

## 3.7 Выводы по заданию №2

Наиболее эффективным алгоритмом из перечисленных является пирамидальная сортировка. Она имеет сложность O(nlogn) в худшем и среднем случая, что делает ее быстрой и эффективной для больших объемов данных. Простое слияние также имеет сложность O(nlogn), но требует дополнительной памяти для временного массива, что может быть недостатком при работе с большими данными. Простые вставки имеют сложность O(n2) в худшем случае, что делает их менее эффективными для сортировки больших объемов данных.

Таким образом, при необходимости выбора из перечисленных алгоритмов для сортировки данных, рекомендуется отдавать предпочтение пирамидальной сортировке в силу ее более высокой эффективности и скорости работы.

# 

# 6 ВЫВОДЫ

В ходе практической работы были выполнены следующие задачи:

- Получены навыки по анализу вычислительной сложности алгоритмов сортировки и определению наиболее эффективного алгоритма;

- Проведён анализ алгоритмов сортировки простым слиянием и Шелла со смещением Д.Кнута вторым способом;

- Были реализованы программы алгоритмов сортировки простым слиянием и Шелла со смещением Д.Кнута вторым способом;

- Проведено тестирование программ для алгоритмов сортировки простым слиянием и Шелла со смещением Д.Кнута вторым способом;

- Построены графики функции роста Тп алгоритмов сортировки простым слиянием и Шелла со смещением Д.Кнута вторым способом;

- Произведено сравнение алгоритмов простой сортировки вставками, сортировки простым слиянием и Шелла со смещением Д.Кнута вторым способом;

- Проведен анализ асимптотической сложности алгоритмов сортировки простыми вставками, сортировки простым слиянием и Шелла со смещением Д.Кнута вторым способом;

-Произведено сравнение асимптотической сложности алгоритмов сортировки простыми вставками, сортировки простым слиянием и Шелла со смещением Д.Кнута вторым способом;

- Проведено определение наиболее эффективного алгоритма.

Таким образом, главную цель практической работы, а именно получение навыков по анализу вычислительной сложности алгоритмов сортировки и определению наиболее эффективного алгоритма, можно считать выполненной.

# 7 ЛИТЕРАТУРА

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. – СПб: Питер, 2017. – 288 с.

2. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 406 с.

3. Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск, 2-е изд. – М.: ООО «И.Д. Вильямс», 2018. – 832 с.

4. Кораблин Ю.П. Структуры и алгоритмы обработки данных: учебно-методическое пособие / Ю.П. Кораблин, В.П. Сыромятников, Л.А. Скворцова. – М.: РТУ МИРЭА, 2020. — 219 с.

5. Кормен Т.Х. и др. Алгоритмы: построение и анализ, 3-е изд. – М.: ООО «И.Д.Вильямс», 2013. – 1328 с.

6. Макконнелл Дж. Основы современных алгоритмов. Активный обучающий метод. 3-е доп. изд., - М.: Техносфера, 2018. – 416 с.

7. Седжвик Р. Фундаментальные алгоритмы на C++. Анализ/Структуры данных/Сортировка/Поиск. – К.: Издательство «Диасофт», 2001. – 688 с.

8. Скиена С. Алгоритмы. Руководство по разработке, - 2-е изд. – СПб: БХВ-Петербург, 2011. – 720 с.

9. Хайнеман Д. и др. Алгоритмы. Справочник с примерами на C, C++, Java и Python, 2-е изд. – СПб: ООО «Альфа-книга», 2017. – 432 с.

10. AlgoList – алгоритмы, методы, исходники [Электронный ресурс]. URL: http://algolist.manual.ru/ (дата обращения 15.03.2022).

11. Алгоритмы – всё об алгоритмах / Хабр [Электронный ресурс]. URL: https://habr.com/ru/hub/algorithms/ (дата обращения 15.03.2022).

12. НОУ ИНТУИТ | Технопарк Mail.ru Group: Алгоритмы и структуры данных [Электронный ресурс]. URL: https://intuit.ru/studies/courses/3496/738/info (дата обращения 15.03.2022).