

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: Боргачев Т.М.

Группа: ИНБО-10-23

Москва – 2024

# СОДЕРЖАНИЕ

[1 ФОРМУЛИРОВКА ЗАДАЧИ 3](#_Toc160120892)

[1.2 Задание 1 Эмпирическая оценка эффективности алгоритмов 3](#_Toc160120895)

[1.3 Задание 2 Асимптотический анализ сложности алгоритмов 4](#_Toc160120896)

[2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ 5](#_Toc160120897)

[2.1 Задание 1 5](#_Toc160120898)

[2.1.1 Реализация алгоритма в виде функции 5](#_Toc160120899)

[2.1.2 Ёмкостная сложность алгоритма сортировки Шелла 7](#_Toc160120903)

[2.1.3 Реализация второго алгоритма в виде функции 7](#_Toc160120904)

[2.1.4 Ёмкостная сложность алгоритма простого слияния 10](#_Toc160120905)

[2.1.5 Данные по работе алгоритма вставками 10](#_Toc160120906)

[2.1.6 Представление данных в виде графиков 10](#_Toc160120907)

[2.2 Задание 2 13](#_Toc160120908)

[2.2.1 Функция роста алгоритма простой вставки 13](#_Toc160120909)

[2.2.2 Асимптотическая оценка сложности алгоритмов 14](#_Toc160120910)

[2.2.3 Графическое представление функции роста и полученных асимптотических оценок сверху и снизу 14](#_Toc160120911)

[3 ВЫВОДЫ 16](#_Toc160120912)

[4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ 16](#_Toc160120913)

# 

## Цель

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

# 1 ФОРМУЛИРОВКА ЗАДАЧИ

## Задание 1 Эмпирическая оценка эффективности алгоритмов

* 1. Разработать алгоритм ускоренной сортировки, определенной в варианте 5: Сортировка Шелла со смещениями Д. Кнута. способ 2, реализовать код на языке С++. Сформировать таблицу результатов эмпирической оценки сложности сортировки для массива, заполненного случайными числами.
  2. Определить ёмкостную сложность алгоритма ускоренной сортировки.
  3. Разработать алгоритм быстрой сортировки, определенной в варианте 5: Простое слияние, реализовать код на языке С++. Сформировать таблицу результатов эмпирической оценки сортировки для массива, заполненного случайными числами.
  4. Определить ёмкостную сложность алгоритма быстрой сортировки
  5. Добавьте в отчёт данные по работе любого из алгоритмов простой сортировки в среднем случае, полученные в предыдущей практической работе.
  6. Представить на общем сравнительном графике зависимости 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 для каждого алгоритма.
  9. Сделайте вывод о зависимости (или независимости) алгоритмов сортировок от исходной упорядоченности массива на основе результатов, представленных в таблицах.

## 1.3 Задание 2 Асимптотический анализ сложности алгоритмов

* 1. Из материалов предыдущей практической работы приведите в отчёте формулы (n) функций роста алгоритма простой сортировки в лучшем и худшем случае (того же алгоритма, что и в задании 1).
  2. На основе определений соответствующих нотаций получите асимптотическую оценку вычислительной сложности простого алгоритма сортировки:

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

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

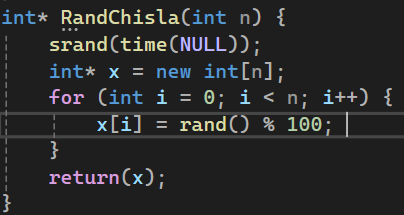
* 1. Получите (если это возможно) асимптотически точную оценку вычислительной сложности алгоритма в нотации θ.
  2. Реализуйте графическое представление функции роста и полученных асимптотических оценок сверху и снизу.
  3. Привести справочную информацию о вычислительной сложности усовершенствованного и быстрого алгоритмов сортировки, заданных в вашем варианте.
  4. Общие результаты свести в таблицу.
  5. Сделать вывод о наиболее эффективном алгоритме из трёх.

# ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

## Задание 1

### 2.1.1 Реализация алгоритма в виде функции

Напишем отдельно функции для заполнения массива случайными значениями рис. 1 и вывода массива на экран рис. 2.



# 

Рисунок 1 – Функция заполнения массива случайными значениями

# 

# 

Рисунок 2 – Функция вывода массива на экран

При сортировке Шелла сначала сравниваются и сортируются между собой значения, стоящие один от другого на некотором расстоянии d. После этого процедура повторяется для некоторых меньших значений d, а завершается сортировка Шелла упорядочиванием элементов при d = 1 (то есть обычной сортировкой вставками). Схема работы алгоритма представлена на рис. 3.

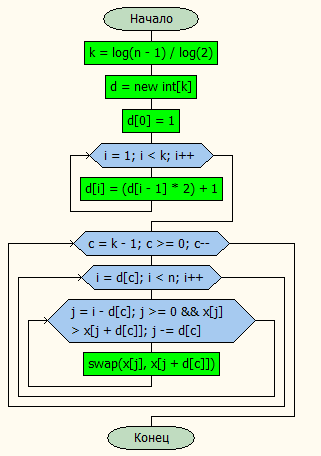


Рисунок 3 – Блок-схема сортировки Шелла

Функция, написанная на языке C++ и реализующая сортировку Шелла представлена на рис. 4.

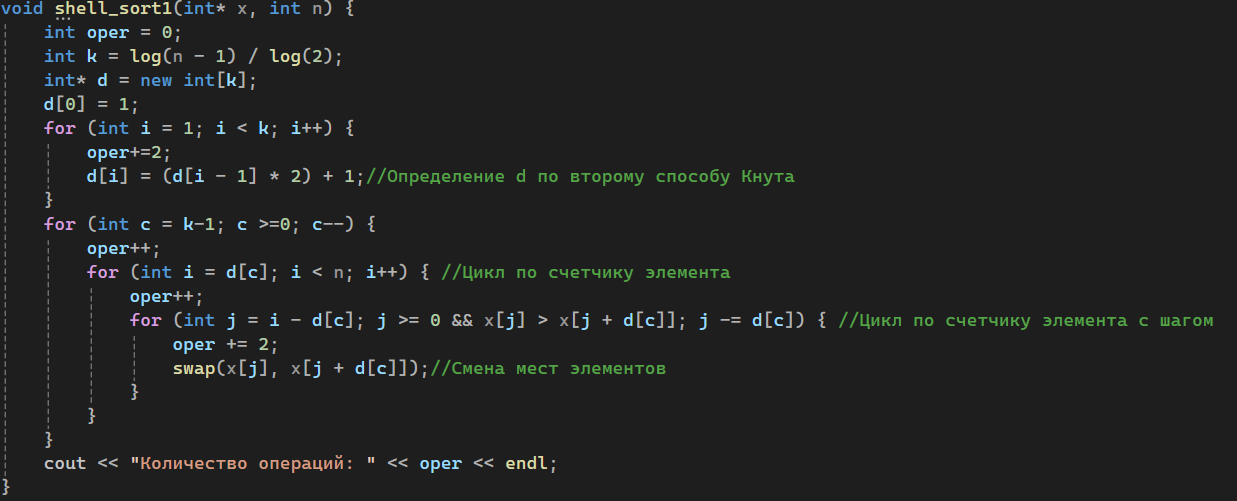


Рисунок 4 – Функция сортировки Шелла

Проведем тестирование алгоритма для различных n и занесем результаты в табл. 1.

Таблица 1 – Результаты тестирования сортировки Шелла

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** |  |
| 100 | 1 | 1138 |
| 1000 | 1 | 18480 |
| 10000 | 2 | 252166 |
| 100000 | 16 | 3104824 |
| 1000000 | 171 | 38635862 |

### 2.1.2 Ёмкостная сложность алгоритма сортировки Шелла

В течение работы алгоритма – сортировка охватывает и преобразовывает один массив x[n], а также берет значения d из массива, следовательно ёмкостная сложность алгоритма равна n+1.

### 2.1.3 Реализация второго алгоритма в виде функции

Алгоритм простого слияния решает задачу сортировки так: сначала задача разбивается на несколько подзадач меньшего размера. Затем эти задачи решаются с помощью рекурсивного вызова или непосредственно, если их размер достаточно мал. Наконец, их решения комбинируются, и получается решение исходной задачи.

Схема алгоритма представлена на рис. 5.

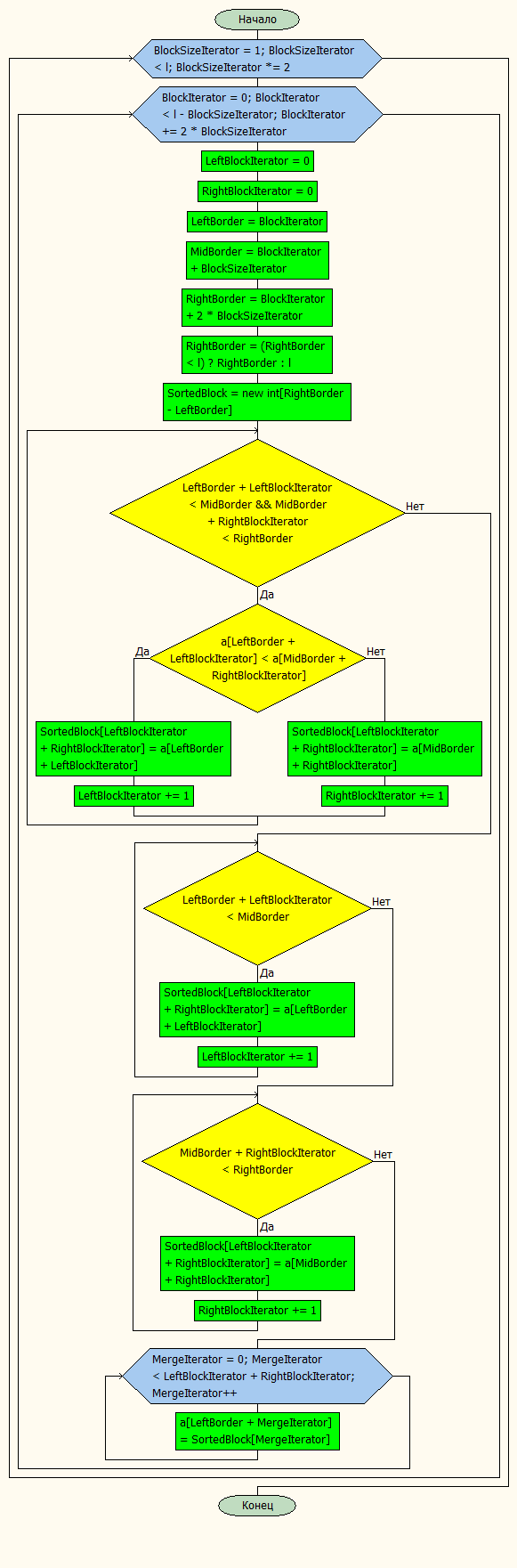


Рисунок 5 – Блок схема алгоритма простого слияния

Код алгоритма на языке C++ представлен на рис. 6 и 7.

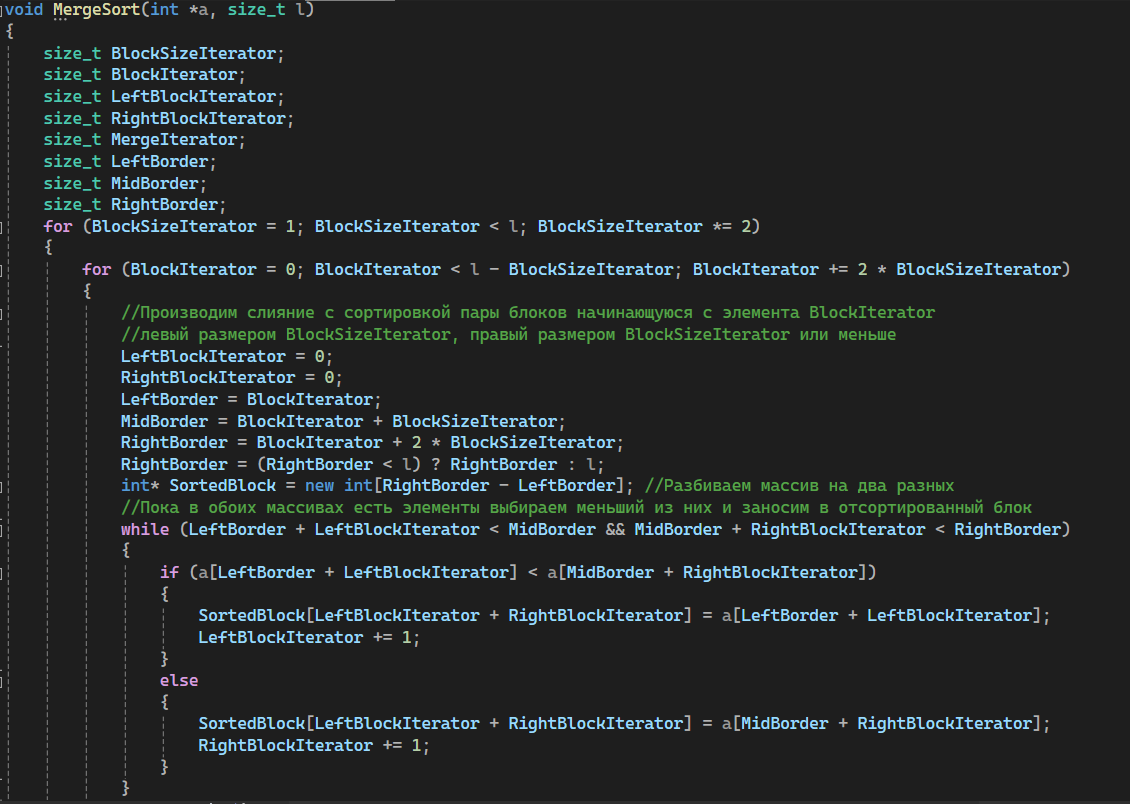


Рисунок 6 – Первая часть кода алгоритма простого слияния

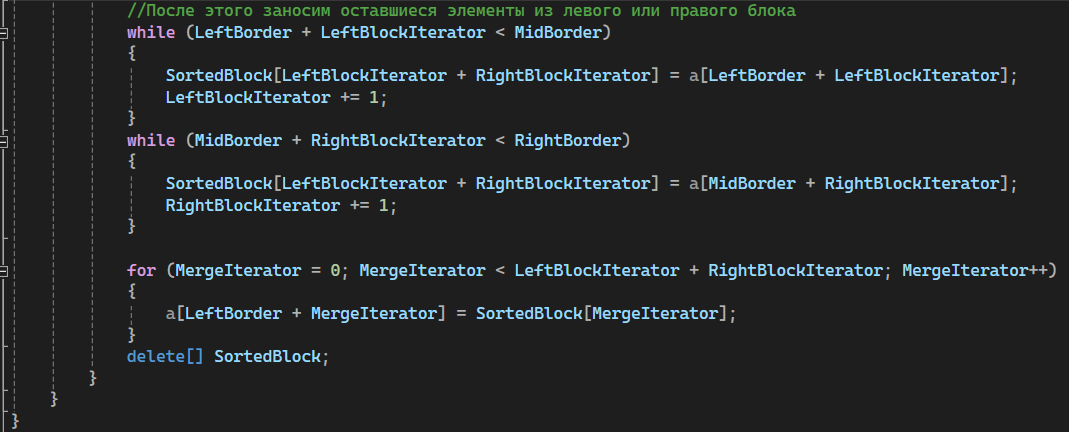


Рисунок 7 – Вторая часть кода алгоритма простого слияния

Проведем тестирование алгоритма для различных n и занесем результаты в табл. 2.

Таблица 2 – Результаты тестирования сортировки слиянием

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** |  |
| 100 | 1 | 4130 |
| 1000 | 1 | 58062 |
| 10000 | 3 | 783542 |
| 100000 | 28 | 9641569 |
| 1000000 | 266 | 112967595 |

### 2.1.4 Ёмкостная сложность алгоритма простого слияния

В течение работы алгоритма – сортировка охватывает и преобразовывает один массив x[n] так, что в нем остается половина элементов, а также создает дополнительный массив, состоящий из оставшейся половины элементов x, следовательно n/2 + n/2 = n - ёмкостная сложность алгоритма.

### 2.1.5 Данные по работе алгоритма вставками

Из предыдущей работы возьмем результаты тестирования алгоритма и занесем в табл. 3.

Таблица 3 – Результаты работы алгоритма сортировки вставками

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** |  |
| 100 | 1 | 5366 |
| 1000 | 1 | 499076 |
| 10000 | 43 | 49605128 |
| 100000 | 4198 | 648864620 |
| 1000000 | 428095 | 1228144492 |

### 2.1.6 Представление данных в виде графиков

Данные тестирования алгоритмов при n <= 1000 и n > 1000 представлены на рис. 8 и 9 соответственно.

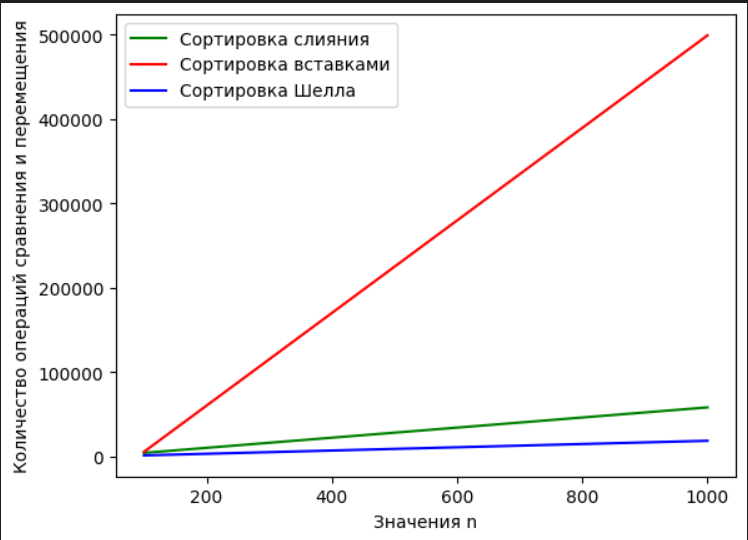


Рисунок 8 – Тестирование алгоритмов для n<=1000

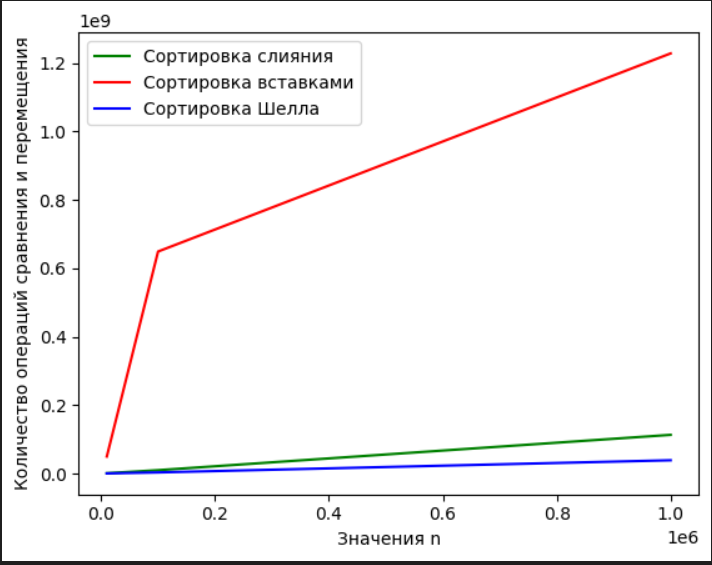


Рисунок 9 – Тестирование алгоритмов для n>1000

Так как алгоритм сортировки Шелла в обоих случаях (для n <= 1000 и n>1000) справился быстрее и за меньшее количество операций, то можно сделать вывод: он эффективнее двух других в среднем случае со случайными значениями элементов массива

2.1.7 Лучший и худший случаи для алгоритмов

Результаты тестирования алгоритма сортировки Шелла в лучшем случае представлены в табл. 4.

Таблица 4 – Тестирование сортировки Шелла в лучшем случае

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **=+** |
| 100 | 1 | 496 |
| 1000 | 1 | 8012 |
| 10000 | 1 | 113668 |
| 100000 | 4 | 1468992 |
| 1000000 | 51 | 17951500 |

Результаты тестирования алгоритма сортировки Шелла в худшем случае представлены в табл. 5.

Таблица 5 – Тестирование сортировки Шелла в худшем случае

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **=+** |
| 100 | 1 | 840 |
| 1000 | 1 | 12976 |
| 10000 | 1 | 152904 |
| 100000 | 6 | 1879590 |
| 1000000 | 58 | 22537680 |

Результаты тестирования алгоритма сортировки слиянием в лучшем случае представлены в табл. 6.

Таблица 6 – Тестирование алгоритма простого слияния в наилучшем случае

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **=+** |
| 100 | 1 | 4001 |
| 1000 | 1 | 54542 |
| 10000 | 3 | 721063 |
| 100000 | 23 | 8715708 |
| 1000000 | 234 | 100177379 |

Результаты тестирования алгоритма сортировки слиянием в худшем случае представлены в табл. 7.

Таблица 7 – Тестирование алгоритма простого слияния в наихудшем случае

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **=+** |
| 100 | 1 | 3570 |
| 1000 | 1 | 49873 |
| 10000 | 3 | 659897 |
| 100000 | 22 | 8087004 |
| 1000000 | 223 | 94807183 |

Таким образом, сортировка Шелла зависит от упорядочивания первоначального массива, и работает в 1.5 раза эффективнее с уже упорядоченным массивом. В то же время сортировка слиянием не зависит от упорядочивания первоначального массива, и в моем случае сработала с отсортированным по убыванию массивом эффективнее, нежели чем с отсортированным по возрастанию.

## Задание 2

### 2.2.1 Функция роста алгоритма простой вставки

Из материалов предыдущей практической работы, функция роста алгоритма простой вставки T(n) в лучшем случае равна 3, в худшем случае T(n)= 0.5n^2 +2.5n.

### 2.2.2 Асимптотическая оценка сложности алгоритмов

О-нотация – это оценка сверху, представленная в виде функции О(n), где то, что под знаком О – это часть полинома T(n), вносящая наибольший вклад в скорость роста.

В таком случае вычислительная сложность простого алгоритма в О-нотации равна O(n^2).

Ω-нотация – это оценка снизу, представленная в виде функции Ω(n) для значения n в лучшем случае работы алгоритма

В таком случае вычислительная сложность простого алгоритма в Ω-нотации равна 3.

θ-нотация: для некоторой функции g(n) запись f(n)=θ(g(n)) обозначает множество функций {f(n), для которых: существуют положительные константы c1, c2, такие что 0 ≤ с1g(n) ≤ f(n) ≤ c2 g(n) для всех n > n0}.

В θ-обозначениях функция f(n) асимптотически ограничивается сверху и снизу: для всех n > n0 f(n) = g(n) с точностью до постоянного множителя.

Докажем, что для простой сортировки T(n)= θ(n^2):

c1 <= 0.5+2.5/n выполняется для всех n>=1 при c1 = 0.5.

c2>=0.5 +2.5/n выполняется для всех n>=1 при c2 = 3.

Тогда найдены c1 = 0.5, c2 = 3 и n0=1, а, значит, по определению, T(n)=θ(n^2 ), что и требовалось доказать.

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

Графики функции роста и асимптотических оценок представлен на рис. 10.

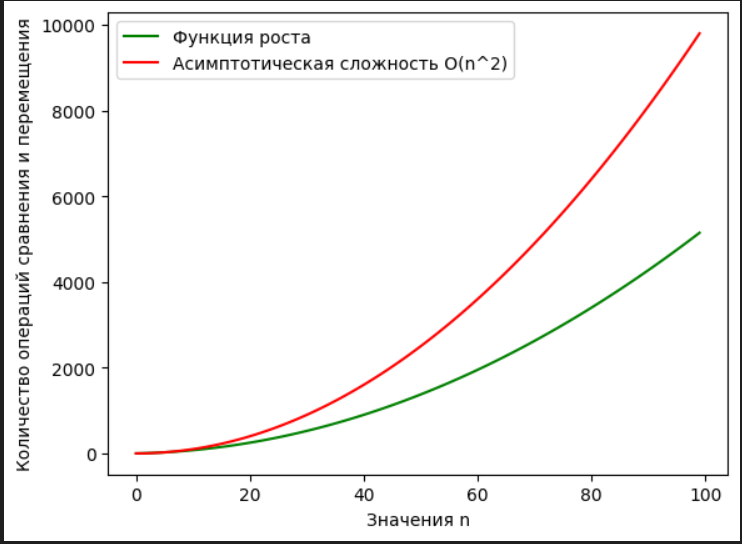


Рисунок 10 – График сложностей простого алгоритма

2.2.4 Асимптотическая сложность алгоритмов быстрой и усовершенствованной сортировки

Асимптотическая сложность сортировки Шелла равна O(n\*log^2(n)) для лучшего времени и O(n^2) для худшего.

Асимптотическая сложность сортировки простого слияния равна O(n\*log(n)) для любого времени.

Запишем результаты в табл. 8.

Таблица 8 – Сводная таблица результатов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм | Асимптотическая сложность алгоритма | | | |
| Наихудший случай (сверху) | Наилучший случай (снизу) | Средний случай (точная оценка) | Ёмкостная сложность |
| Простой | n^2 | 3 | 0.5n^2+2.5n | n |
| Усовершен. | n^2 | n\*log^2(n) | n^2 | n+1 |
| Быстрый | n\*log(n) | n\*log(n) | n\*log(n) | n |

Таким образом, более эффективным (в асимптотическом смысле) является быстрый алгоритм (простого слияния).

# 3 ВЫВОДЫ

На основе тестирования алгоритмов для разных значений n, был сделан вывод о том, что наиболее эффективным алгоритмом является алгоритм сортировки Шелла, в то время как самым неэффективным является алгоритм сортировки простой вставки для любых случаев.

Так же была выявлена зависимость усовершенствованного алгоритма от начальной сортировки массива, в то время как у быстрого этой зависимости нет.

С асимптотической же точки зрения, наиболее эффективным алгоритмом можно назвать алгоритм сортировки простого слияния для любых из трех случаев (худшего, лучшего, среднего).

# 4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Сартаков М.В., ПР-1.1 (Теоретическая сложность алгоритма) М., МИРЭА — Российский технологический университет – 12 с. - URL: <https://online-edu.mirea.ru/pluginfile.php?file=%2F1042738%2Fmod_assign%2Fintroattachment%2F0%2FПР1.1%20%28Теоретическая%20сложность%20алгоритма%29.pdf&amp;forcedownload=1> (дата обращения: 15.02.2024). - Режим доступа: Электронно-облачная система – Cloud MIREA РТУ МИРЭА. - Текст: электронный.
2. Рысин М.Л., Сартаков М.В., Туманова М.Б., Введение в структуры и алгоритмы обработки данных. Ч. 1 - учебное пособие, 2022, МИРЭА – Российский технологический университет. – 2022, 109с. – URL: [file:///C:/Users/borga/Downloads/Рысин%20М.Л.%20и%20др.%20Введение%20в%20структуры%20и%20алгоритмы%20обработки%20данных.%20Ч.%201%20-%20учебное%20пособие,%202022.pdf](C://Users/borga/Downloads/Рысин%20М.Л.%20и%20др.%20Введение%20в%20структуры%20и%20алгоритмы%20обработки%20данных.%20Ч.%201%20-%20учебное%20пособие,%202022.pdf) (дата обращения: 15.02.2024 ). – Режим доступа: Электронно-облачная система – Cloud MIREA РТУ МИРЭА. - Текст: электронный.