

МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение*

*высшего образования*

***«МИРЭА - Российский технологический университет»***

**РТУ МИРЭА**

**У к**

Отчёт по выполнению практического задания №1.3

Тема:

«Определение эффективного алгоритма сортировки на основе эмпирического

и асимптотического методов анализа»

Дисциплина: «Структуры и алгоритмы обработки данных»

|  |  |
| --- | --- |
| Выполнил студент: | ???? |
| Группа: | ...ИКБО-52-23......... |

Москва - 2024

**Оглавление**

[1. Цель работы 3](#_Toc24553)

[2. Постановка задачи 1 3](#_Toc9885)

[3. Постановка задачи 2 4](#_Toc19638)

[4. Индивидуальный вариант 4](#_Toc17340)

[5. Решение задачи 1 4](#_Toc23272)

[5.1 Алгоритм шейкерной сортировки с условием Айверсона 4](#_Toc14359)

[5.2 Ёмкостная сложность алгоритма шейкерной сортировки 7](#_Toc12834)

[5.3 Алгоритм сортировки простым слиянием 8](#_Toc25130)

[5.4 Ёмкостная сложность алгоритма сортировки простого выбора 12](#_Toc16072)

[5.5 Эмпирическая оценка вычислительной сложности алгоритма сортировки простого выбора 12](#_Toc19767)

[5.6 Сравнительная характеристика для трёх алгоритмов 12](#_Toc13761)

[5.7 Оценка вычислительной сложности алгоритма шейкерной сортировки и сортировки простого слияния в наихудшем и наилучшем случаях. 14](#_Toc32401)

[5.8 Вывод 15](#_Toc8973)

[6. Решение задачи 2 15](#_Toc26179)

[6.1 Функции роста алгоритма простой сортировки выбором 15](#_Toc16977)

[6.2 Вычисление асимптотической функции вычислительной сложности простого алгоритма 15](#_Toc22018)

[6.3 Вывод 18](#_Toc6285)

1. Цель работы

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

1. Постановка задачи 1

Эмпирическая оценка эффективности алгоритмов

1. Разработать алгоритм ускоренной сортировки, определенной в варианте

(приложение 1), реализовать код на языке С++. Сформировать таблицу результатов эмпирической оценки сложности сортировки по формату табл. 1 для массива, заполненного случайными числами1 .

2. Определить ёмкостную сложность алгоритма ускоренной сортировки.

3. Разработать алгоритм быстрой сортировки, определенной в варианте

(приложение 1), 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.

1. Постановка задачи 2

Асимптотический анализ сложности алгоритмов

1. Из материалов предыдущей практической работы приведите в отчёте формулы Тт(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.

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 усовершенствованного и быстрого алгоритмов сортировки, заданных в вашем варианте.

6. Общие результаты свести в таблицу.

7. Сделать вывод о наиболее эффективном алгоритме из трёх.

1. Индивидуальный вариант

Алгоритмы сортировки:

Задание 1 - Шейкерная с условием Айверсона

Задание 2 – Быстрая сортировка (Хоара)

1. Решение задачи 1
   1. Алгоритм шейкерной сортировки с условием Айверсона

Происходит сравнение элементов, если i элемент больше элемента i+1, то они меняются местами, иначе остаются, операция повторяется, пока не достигнет правого края, затем снова происходит сравнение элементов, если i элемент больше элемента i-1, то они меняются местами, иначе остаются. Обе операции чередуются, пока массив не будет полностью отсортирован. Блок-схема алгоритма представлена на рисунке 1. Реализация алгоритма представлена на рисунке 2. На рисунках 3-8 представлены результаты работы сортировки при разных значениях входных данных. В таблице 1 представлены результаты, полученные ранее.

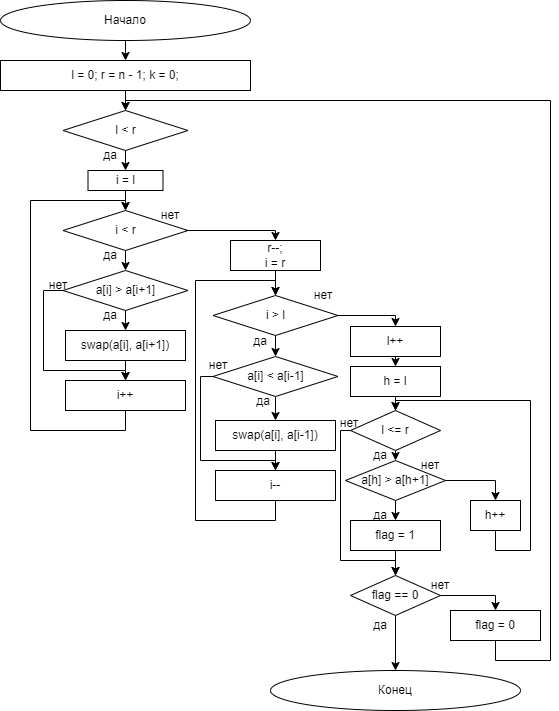


Рисунок 1 - Блок-схема алгоритма

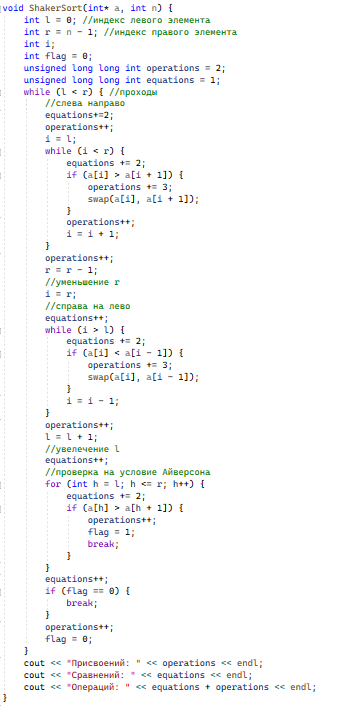


Рисунок 2 - код функции шейкерной сортировки

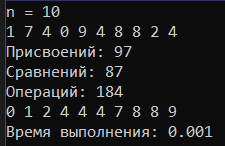


Рисунок 3 - результат тестирования при n = 10

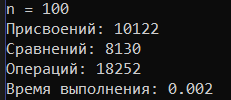


Рисунок 4 - результат тестирования при n = 100

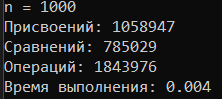


Рисунок 5 - результат тестирования при n = 1000

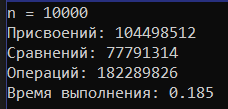


Рисунок 6 - результат тестирования при n = 10000

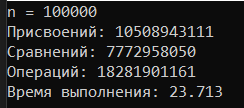


Рисунок 7 - результат тестирования при n = 100000

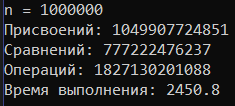


Рисунок 8 - результат тестирования при n = 1000000

Таблица 1 - Сводная таблица результатов для шейкерной сортировки

|  |  |  |
| --- | --- | --- |
| n | T(n), с | Тп = Сп + Мп |
| 100 | 0.001 | 18252 |
| 1000 | 0.002 | 1843976 |
| 10000 | 0.151 | 182289826 |
| 100000 | 16.773 | 18281901161 |
| 1000000 | 1865.9 | 1827130201088 |

* 1. Ёмкостная сложность алгоритма шейкерной сортировки

Ёмкостная сложность равна O(1).

* 1. Алгоритм быстрой сортировки (Хоара)

На очередном шаге выбирается опорный элемент – любой элемент массива. Все остальные элементы массива сравниваются с опорным и делятся на те, которые меньшие опорного, и те, которые больше или равны ему. Для двух получившихся подмассивов производится точно такая же операция пока в подмассиве не останется один элемент. На рисунке 9 представлен код основной функции сортировки. На рисунке 10 представлен код вспомогательной функции. На рисунке 11 представлена блок-схема алгоритма. На рисунках 12-19 представлены результаты работы сортировки при разных значениях входных данных. В таблице 2 представлены результаты, полученные ранее.

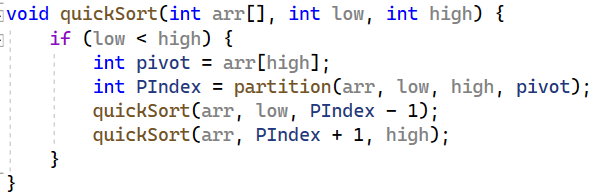


Рисунок 9 - реализация основной функции быстрой сортировки (Хоара)

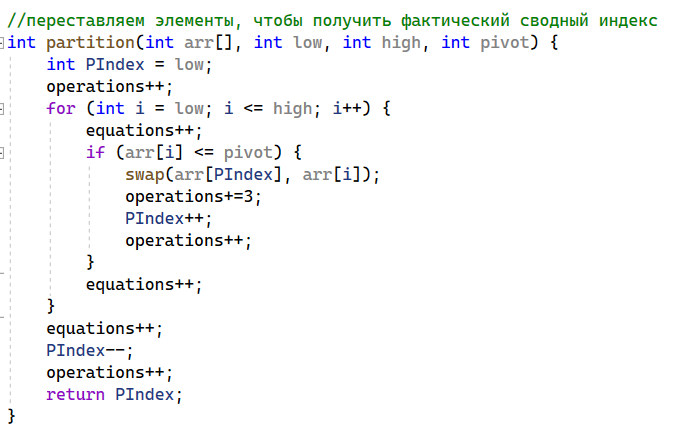


Рисунок 10 - Реализация вспомогательной функции

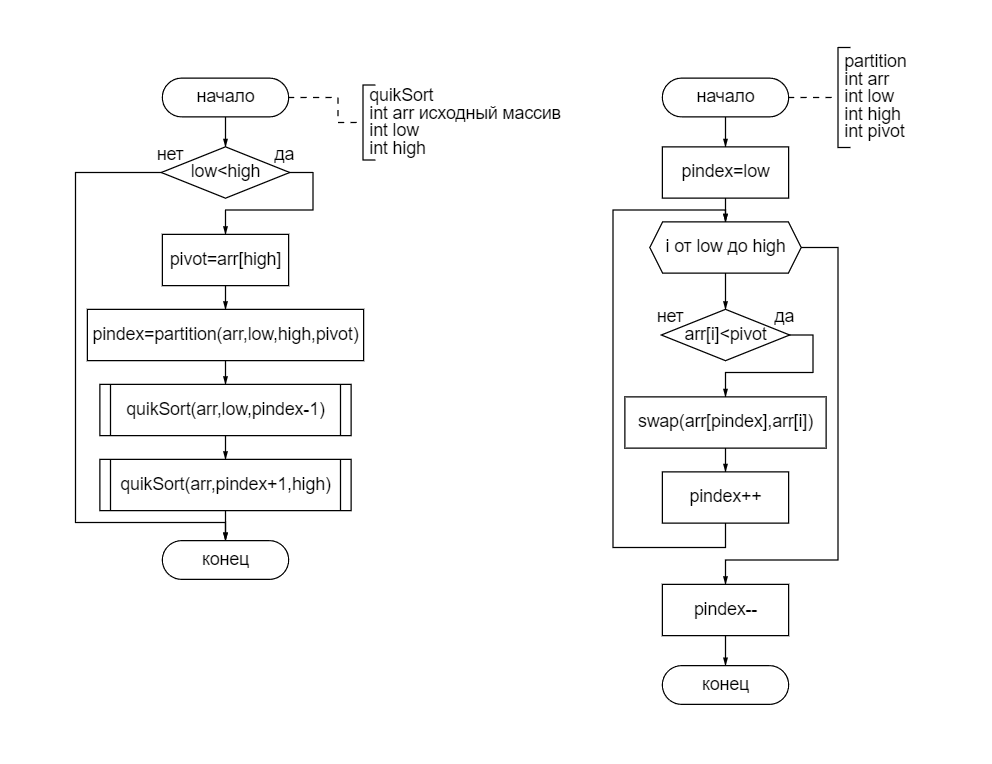


Рисунок 11 - Блок-схема алгоритма быстрой сортировки Хоара

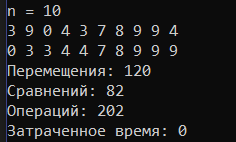


Рисунок 12 - результат тестирования при n = 10

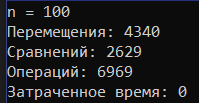


Рисунок 13 - результат тестирования при n = 100

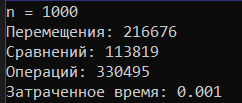


Рисунок 14 - результат тестирования при n = 1000

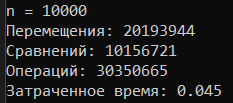


Рисунок 15 - результат тестирования при n = 10000

Таблица 2 - Сводная таблица результатов для быстрой сортировки

|  |  |  |
| --- | --- | --- |
| n | T(n), с | Тп = Сп + Мп |
| 100 | 0 | 6969 |
| 1000 | 0.001 | 330495 |
| 10000 | 0.045 | 30350665 |
| 100000 | 0.1 | 3003409665 |
| 1000000 | 4.3 | 300035006995 |

* 1. Ёмкостная сложность алгоритма сортировки простого выбора

1. Ёмкостная сложность равна O(logn).
   1. Эмпирическая оценка вычислительной сложности алгоритма сортировки простого выбора

Таблица 3 - Таблица результатов для алгоритма сортировки простого выбора

|  |  |  |
| --- | --- | --- |
| n | T(n), с | Тп = Сп + Мп |
| 100 | 0.001 | 11156 |
| 1000 | 0.004 | 1056013 |
| 10000 | 0.183 | 105059547 |
| 100000 | 18.247 | 10500574429 |
| 1000000 | 1027.03 | 1050006741074 |

* 1. Сравнительная характеристика для трёх алгоритмов

В таблице 4 представлены результаты сортировки для трёх алгоритмов: «Простого выбора», «Шейкерная сортировка», «Быстрая сортировка».

Таблица 4 - Сводная таблица результатов

|  |  |  |  |
| --- | --- | --- | --- |
| n | Простого выбора | Шейкерная сортировка | Быстрая сортировка |
| 100 | 11156 | 18252 | 6969 |
| 1000 | 1056013 | 1843976 | 330495 |
| 10000 | 105059547 | 182289826 | 30350665 |
| 100000 | 10500574429 | 18281901161 | 3003409665 |
| 1000000 | 1050006741074 | 1827130201088 | 300035006995 |

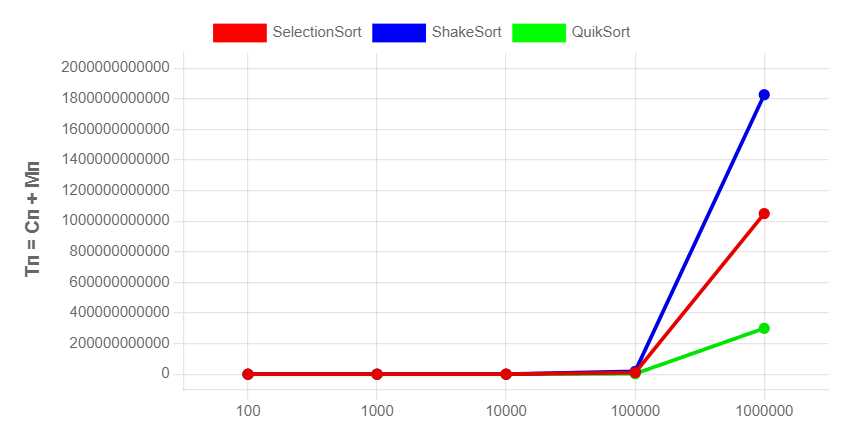


Рисунок 16 - График функции роста алгоритмов

Заметим, что при разных значениях n, алгоритм быстрой сортировки требует наименьшее количество операций, а алгоритм шейкерной сортировки - наибольшее.

* 1. Оценка вычислительной сложности алгоритма шейкерной сортировки и сортировки простого слияния в наихудшем и наилучшем случаях.

Таблица 5 - Таблица результатов для шейкерной сортировки с условием Айверсона

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | В убывающем  порядке | | В возрастающем  порядке | |
| T(n), с | Тп = Сп + Мп | T(n), с | Тп = Сп + Мп |
| 100 | 0.002 | 30350 | 0.002 | 800 |
| 1000 | 0.005 | 3003500 | 0.002 | 8000 |
| 10000 | 0.29 | 300035000 | 0.002 | 80000 |
| 100000 | 28.021 | 30000350000 | 0.002 | 800000 |
| 1000000 |  | 3000003500000 | 0.008 | 8000000 |

Таблица 6 - Таблица результатов для быстрой сортировки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | В убывающем  порядке | | В возрастающем  порядке | |
| T(n), с | Тп = Сп + Мп | T(n), с | Тп = Сп + Мп |
| 100 | 0 | 20988 | 0 | 30988 |
| 1000 | 0.003 | 2009988 | 0.005 | 3009988 |
| 10000 | 0.006 | 1030960 | 0.01 | 300099988 |
| 100000 | 0.025 | 12643672 | 0.04 | 30000999988 |
| 1000000 | 0.267 | 149478592 | 0.48 | 3000009999988 |

* 1. Вывод

Мы можем наблюдать, что временные и вычислительные затраты шейкерной сортировки напрямую зависят от изначальной упорядоченности массива. Быстрая сортировка не так сильно зависит от изначальной отсортированности массива.

1. Решение задачи 2
   1. Функции роста алгоритма простой сортировки выбором

Таблица 7 - Теоретическая оценка функции роста алгоритма

|  |  |  |
| --- | --- | --- |
| Оператор кода | Количество выполнений оператора | |
| В лучшем случае | В худшем случае |
| for (int i = 0; i < n-1; i++) | n | n |
| imax = i; | n-1 | n-1 |
| for (int j = i + 1; j < n; j++) | (n2-2\*n+1)/2+n-1 | (n2-2\*n+1)/2+n-1 |
| if (a[j] <= a[imax]) | (n2-2\*n+1)/2+n-1 | (n2-2\*n+1)/2+n-1 |
| imax = j; | 0 | (n2-2\*n+1)/2+n-1 |
| swap(a[i], a[imax]); | 3\*(n-1) | 3\*(n-1) |

T(n)(в лучшем случае) = n2 +5n - 5 - квадратичная зависимость - T(n) = n2

T(n)(в худшем случае) = 2n2+6n - 2 - квадратичная зависимость - T(n) = n2

* 1. Вычисление асимптотической функции вычислительной сложности простого алгоритма

T(n) имеет порядок роста O(g(n)), если имеется константа с и счетчик n0, такие что 0 < T(n) ≤ с·g(n), для n>=n0.

0 < 2n2+6n - 2 ≤ с·g(n)

Пусть, c\*g(n) =c\*n2

2n2+6n - 2 ≤ с·n2

(2-c)n2+6n - 2≤0

D=52-8\*c=4(13-2c)

При c>6,5, D<0, значит неравенство не будет иметь решений, а соответсвенно будет справедливо для всех n>=n0. Пусть c = 7, а n0=1, тогда всё неравенство 0<2n2+6n - 2 ≤ с·n2 - верно, значит T(n) = O(n2)

T(n) имеет порядок роста Ω(g(n)), если имеется константа с и счетчик n0, такие что 0 < с·g(n) ≤ T(n), для n>=n0.

0 < с·g(n) ≤ n2 +5n - 5

Пусть, c\*g(n) =c\*n2

c\*n2 ≤ n2 +5n - 5

0≤ (1-с) n2 +5n - 5

При c = 0,5 , n0=1, неравенство будет справедливо для всех n>=n0. Значит T(n) = Ω(n2)

Воспользуемся теоремой. Для любых двух функций f(n) и g(n) соотношение f(n) = θ(g(n)) выполняется тогда и только тогда, когда f(n) = O(g(n)) и f(n) = Ω(g(n)). Таким образом. T(n) = O(n2)=Ω(n2)=θ(n2)

Графическое представление функции роста и полученных асимптотических оценок сверху и снизу представлено на рисунке 19.

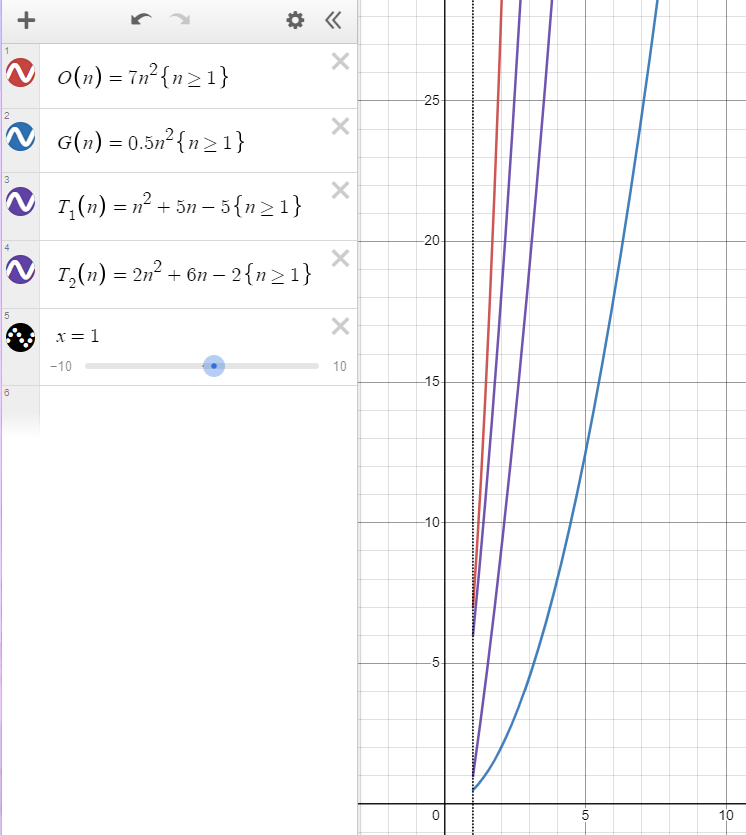


Рисунок 17 - Графическое представление функций роста и асимптотических оценок

Вычислительная сложность шейкерной сортировки.

O(n2) = θ(n2) - средний и наихудший случай

Ω(n) - наилучший случай

Вычислительная сложность быстрой сортировки

O(n\*log2n) наилучший случай

O(n\*logn) средний случай

Ω(n2) – наихудший случай

Таблица 7 - Сводная таблица результатов вычислительной сложности алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм | Асимптотическая сложность алгоритма | | | |
| Наихудший  случай  (сверху) | Наилучший  случай  (снизу) | Средний  случай  (точная  оценка) | Ёмкостная  сложность |
| Простого выбора | O(n2) | Ω(n2) | θ(n2) | O(1) |
| Шейкерная сортировка | O(n2) | Ω(n) | θ(n2) | O(1) |
| Быстрая сортировка | O(n2) | Ω(n\*log2n) | θ(n\*logn) | O(logn) |

* 1. Вывод

Из полученных данных можно сделать вывод о том, что наиболее эффективной сортировкой является быстрая сортировка Хоара.