

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

**Отчет по выполнению практического задания номер один**

**Тема: Определение эффективного алгоритма сортировки**

**Дисциплина: Структуры и алгоритмы обработки данных**

Выполнил студент Михайлюк Д. С.

Группы ИНБО-07-21

Москва 2021

**СОДЕРЖАНИЕ**

[1 Алгоритм Простой вставки (Insertion sort) 3](#_Toc96562331)

[1.1 Алгоритм сортировки по методу простой вставки 3](#_Toc96562332)

[1.2 Описание процесса определения функции роста времени выполнения сортировки методом простой вставки, при увеличении объема массива n 4](#_Toc96562333)

[1.3 Сводная таблицу результатов выполнения сортировки по указанным объемам, на случайно заполненном массиве для всех указанных объемов 4](#_Toc96562334)

[1.4 Код алгоритма и основной программы 5](#_Toc96562335)

[1.5 График зависимости теоретической Тт=f(С+М)=О(f(n)) и практической Тп=(Сф+Мф) вычислительной сложности алгоритма от размера n массива 7](#_Toc96562337)

[1.6 Анализ результатов для задания 1 8](#_Toc96562338)

[2 Задание 9](#_Toc96562339)

[2.1 Таблица результатов при применении к массиву, упорядоченному по возрастанию 9](#_Toc96562340)

[2.2 Таблица результатов при применении к массиву, упорядоченному по убыванию 9](#_Toc96562341)

[2.3 Код программы, используемой для тестов, результат которых отображен в пунктах 2.1 и 2.2 10](#_Toc96562342)

[2.4 График зависимости теоретической и практической вычислительной сложности алгоритма для трех рассмотренных случаев 12](#_Toc96562343)

[2.5 Емкостная сложность алгоритма от n 13](#_Toc96562344)

[2.6 Анализ результатов и выводы из полученных данных 13](#_Toc96562345)

[3 Алгоритм простого выбора (Selection sort) 14](#_Toc96562346)

[3.1 алгоритм сортировки по методу простого выбора (Selection sort) 14](#_Toc96562347)

[3.2 Описание процесса определения функции роста времени выполнения сортировки методом простого выбора, при увеличении объема массива n 15](#_Toc96562348)

[3.3 Сводная таблицу результатов выполнения сортировки по указанным объемам, на случайно заполненном массиве для всех указанных объемов 15](#_Toc96562349)

[3.5 Код всей программы алгоритма простого выбора 16](#_Toc96562350)

[3.6 График зависимости теоретической и практической вычислительной сложности алгоритма для двух рассмотренных алгоритмов: Алгоритм простой вставки и алгоритм простого выбора 18](#_Toc96562351)

[3.7 Определение эффективности алгоритма 19](#_Toc96562352)

[3.8 Анализ результатов 19](#_Toc96562353)

[4 Выводы 20](#_Toc96562354)

[5 Список информационных источников 21](#_Toc96562355)

# Задание

## Алгоритм простого выбора (Selection sort)

### Постановка задачи

Разработать программу сортирующую массив использующую алгоритм простого выбора, реализованную через процедурную функцию, на языке С++

### Алгоритм сортировки по методу простого выбора (Selection sort)

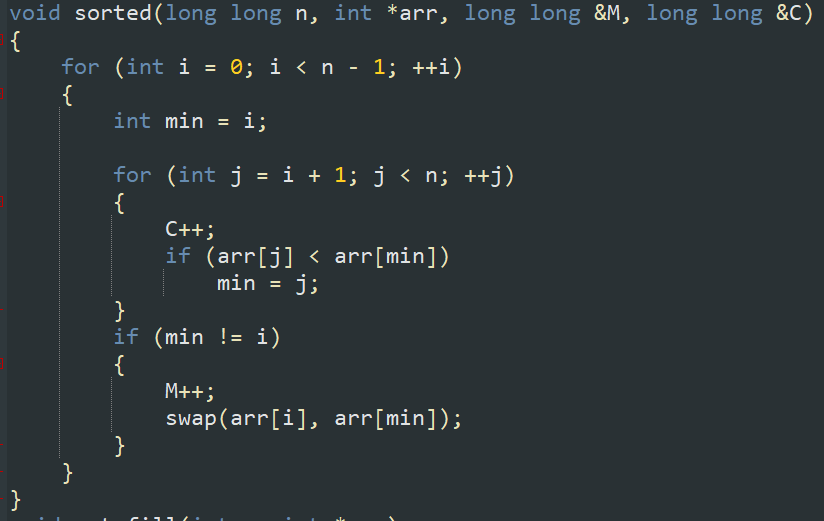


Рисунок 1 - Алгоритм простого выбора реализованный на языке C++

### Описание процесса определения функции роста времени выполнения сортировки методом простого выбора, при увеличении объема массива n

Асимптотическая сложность алгоритма простой вставки: . Что является разновидностью f(n) = O(n^m) полиномиального роста. Кроме того, затраты по памяти программы носит линейный характер: O(n) = n.

График 1 – сложность алгоритма простой вставки

### Сводная таблицу результатов выполнения сортировки по указанным объемам, на случайно заполненном массиве для всех указанных объемов

Таблица 1 - таблица результатов для алгоритма простого выбора

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n) (сек)** | **Тт=f(C+M)** | **Тп=Cф+Mф** |
| 100 | 0.08866 | 5050 | 5046 |
| 1000 | 0.5068 | 500500 | 500494 |
| 10000 | 0.6098 | 50005000 | 50004990 |
| 100000 | 5.114 | 5000050000 | 5000049981 |
| 1000000 | 114.69 | 500000500000 | 500000499924 |

### Код всей программы алгоритма простого выбора

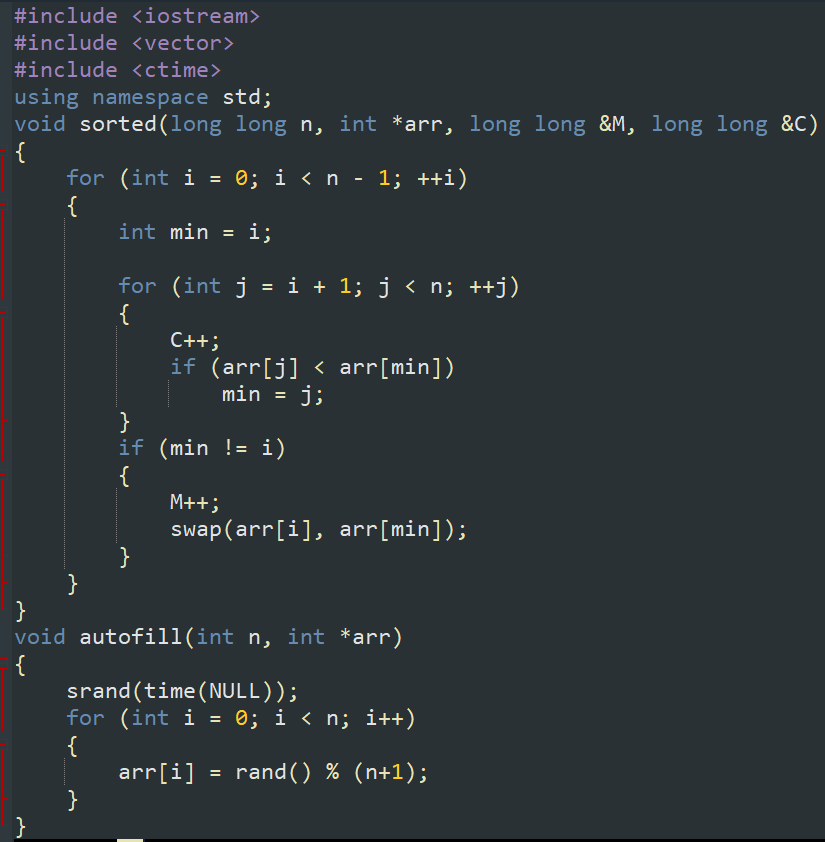


Рисунок 2 – алгоритм простого выбора и функция заполнения массива случайными числами

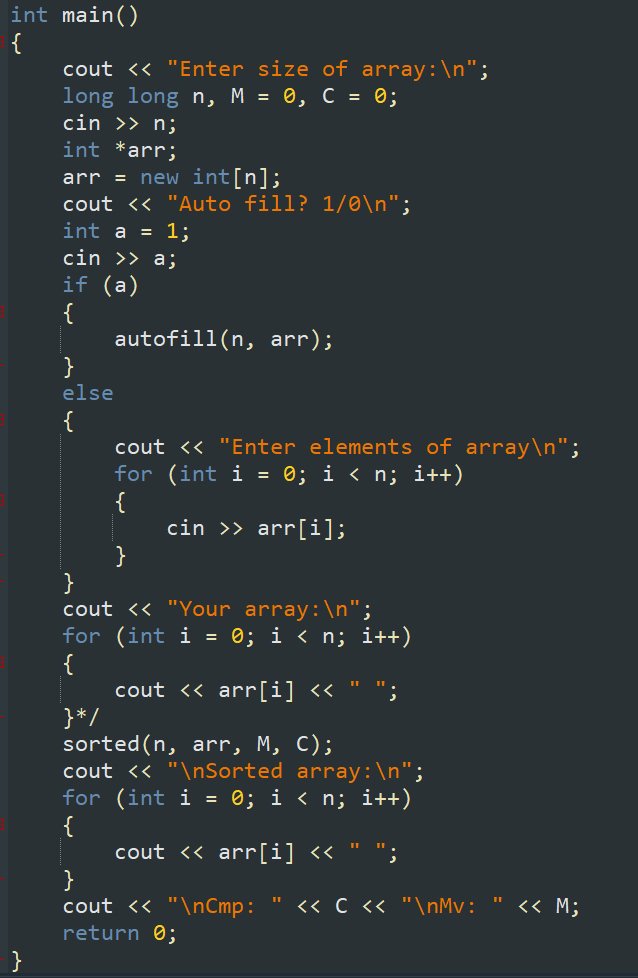


Рисунок 3 – Код основной программы

## Пирамидальная сортировка (Heap sort)

### Постановка задачи

Разработать программу сортирующую массив использующую алгоритм пирамидальной сортировки, реализованную через процедурную функцию, на языке С++

### Алгоритм сортировки по методу простого выбора (Selection sort)

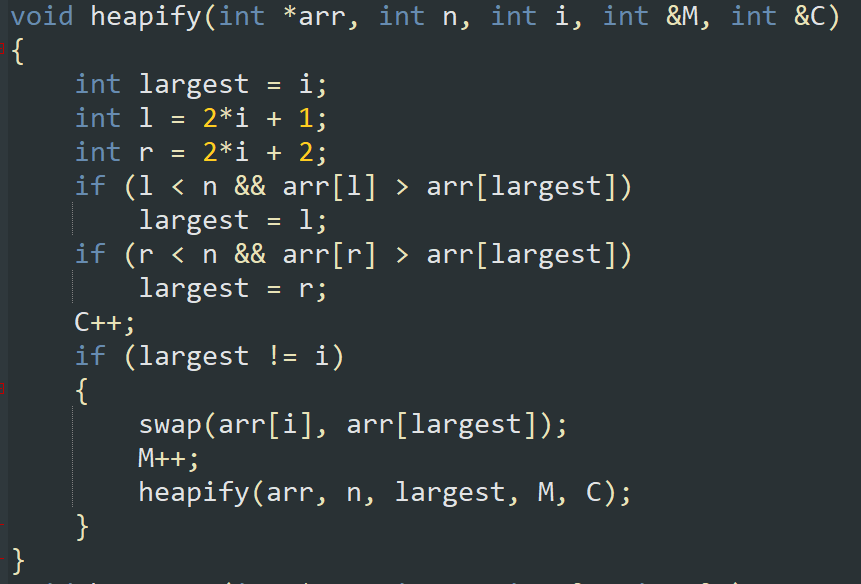


Рисунок 4 - Вспомогательный алгоритм формирования куч, реализованный на языке C++

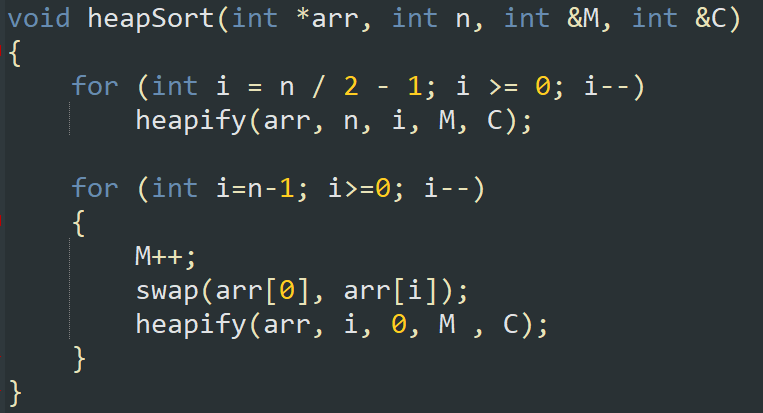


Рисунок 5 – Основной алгоритм пирамидальной сортировки, реализованный на языке C++

### Описание процесса определения функции роста времени выполнения пирамидальной сортировки, при увеличении объема массива n

Асимптотическая сложность алгоритма пирамидальной сортировки: . Что является разновидностью f(n) = O(n\*log(n)) квазилинейного роста Кроме того, затраты по памяти программы носит линейный характер: O(n) = n.

График 2 – сложность алгоритма пирамидальной сортировки

### Сводная таблицу результатов выполнения сортировки по указанным объемам, на случайно заполненном массиве для всех указанных объемов

Таблица 2 - таблица результатов для алгоритма пирамидальной сортировки

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n) (сек)** | **Тт=f(C+M)** | **Тп=Cф+Mф** |
| 100 | 0.03717 | 1300 | 1198 |
| 1000 | 0.4253 | 20500 | 18654 |
| 10000 | 0.5468 | 273000 | 253364 |
| 100000 | 0.55 | 3420000 | 3200278 |
| 1000000 | 0.7218 | 41000000 | 38596414 |

### Код всей программы алгоритма пирамидальной сортировки

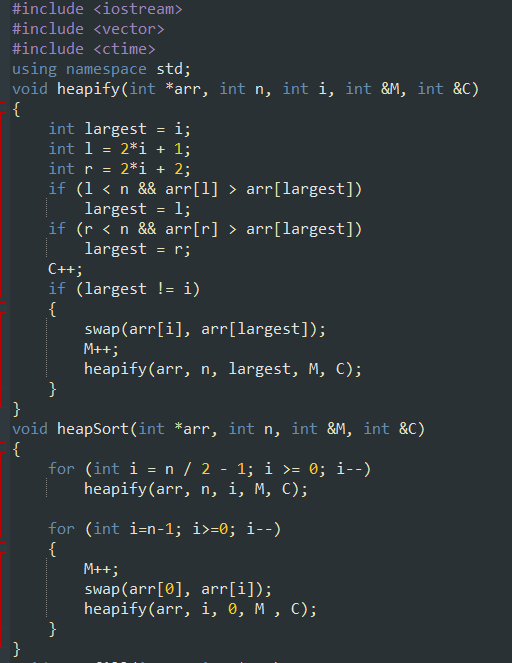


Рисунок 6 – Функция пирамидальной сортировки

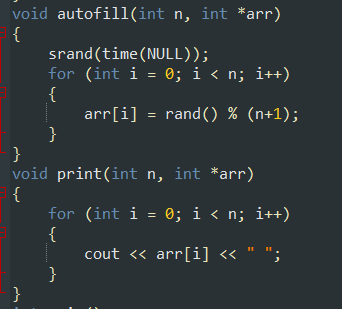


Рисунок 7 – дополнительные функции вывода массива на экран и автозаполнения

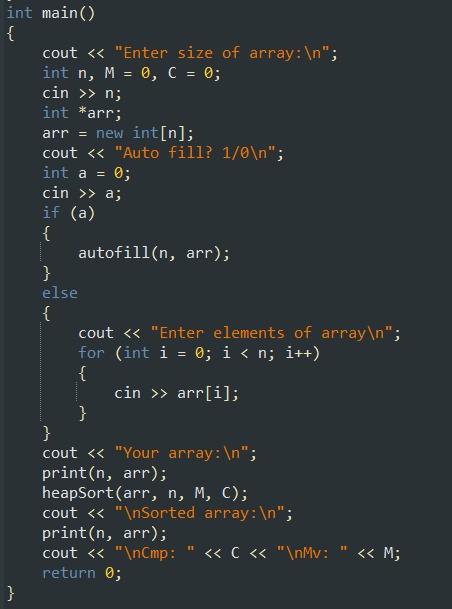


Рисунок 8 – код основной программы

## Анализ полученных результатов

В ходе проведенных тестов, можно сказать, что алгоритм простой вставки эффективен лишь на небольших размерах массива (не более 60) при том даже более эффективен чем алгоритм пирамидальной сортировки, однако на больших массивах алгоритм пирамидальной сортировки намного быстрее алгоритма простой вставки.

График 3 - график зависимости Сф+Мф для анализируемых алгоритмов

## Ускоренная сортировка простого слияния (Merge sort)

### Постановка задачи

Разработать программу сортирующую массив использующую алгоритм сортировки слиянием, реализованную через процедурную функцию, на языке С++

### Алгоритм сортировки слиянием

Рисунок 9 - Алгоритм сортировки слиянием, реализованный на языке C++

### Описание процесса определения функции роста времени выполнения сортировки слиянием, при увеличении объема массива n

Асимптотическая сложность алгоритма пирамидальной сортировки: . Что является разновидностью f(n) = O(n\*log(n)) квазилинейного роста Кроме того, затраты по памяти программы носит линейный характер: O(n) = 2\*n.

График 2 – сложность алгоритма сортировки слиянием

### Сводная таблицу результатов выполнения сортировки по указанным объемам, на случайно заполненном массиве для всех указанных объемов

Таблица 3 - таблица результатов для алгоритма сортировки слиянием

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n) (сек)** | **Тт=f(C+M)** | **Тп=Cф+Mф** |
| 100 | 0.5282 | 1300 | 1244 |
| 1000 | 0.03928 | 20500 | 18714 |
| 10000 | 0.5531 | 273000 | 259875 |
| 100000 | 0.4996 | 3420000 | 3259520 |
| 1000000 | 0.2418 | 41000000 | 38696802 |

### Код всей программы алгоритма пирамидальной сортировки

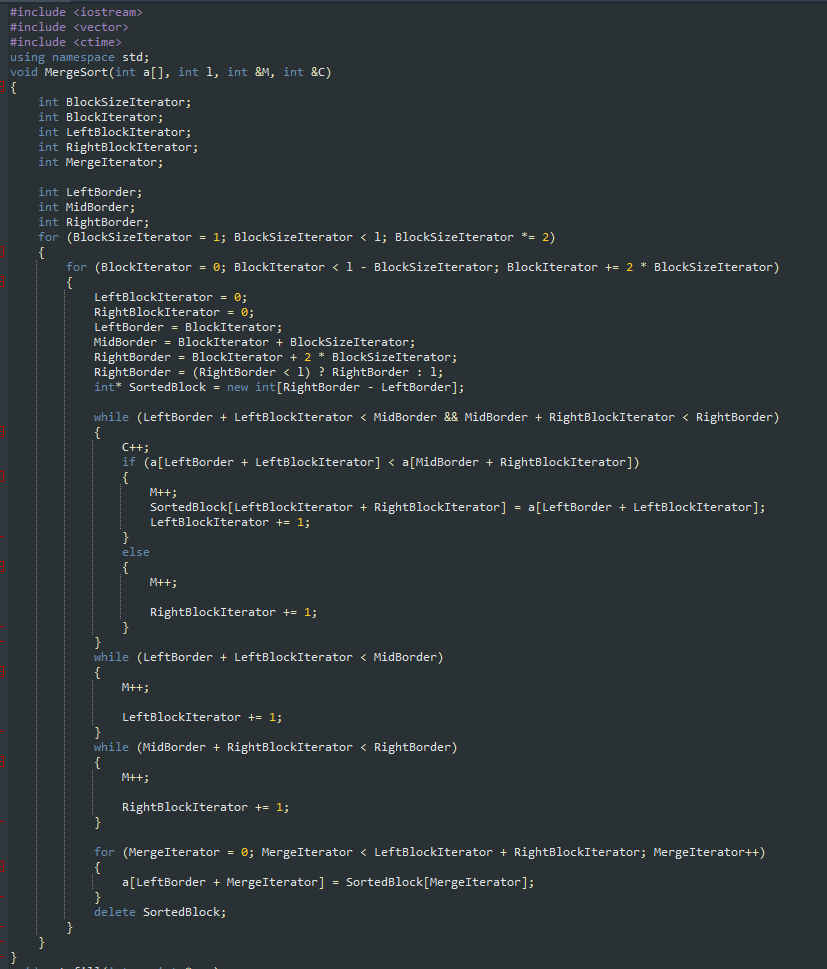


Рисунок 6 – Функция сортировки слиянием

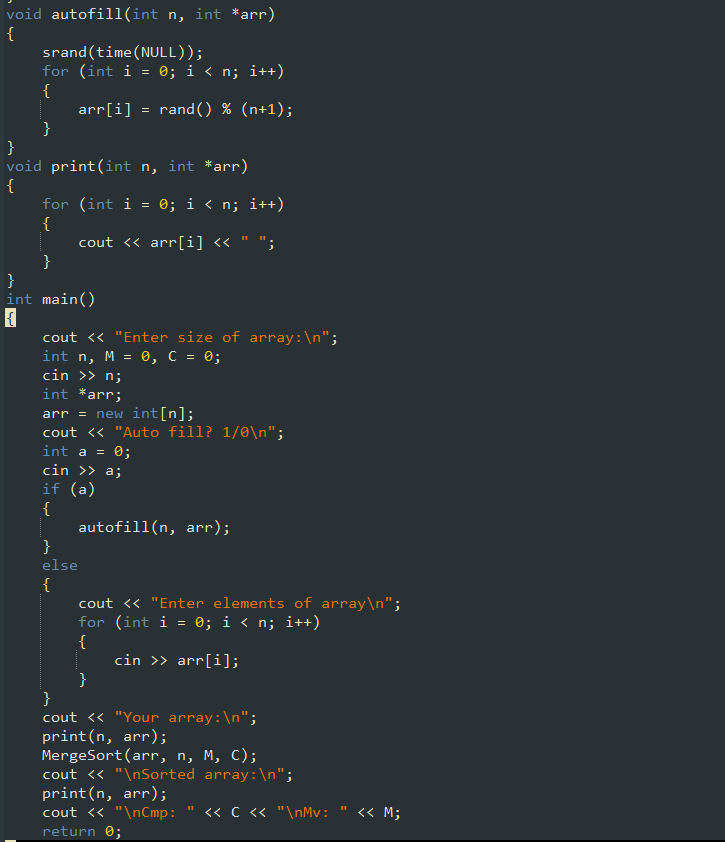


Рисунок 7 – дополнительные функции вывода массива на экран и автозаполнения, а так же основная программа

## Анализ результатов

В ходе проведенных тестов, на основе полученных результатов, отображенных в таблице 2 и 3 можно утверждать, что оба рассматриваемых алгоритма примерно одинаковы, и однозначно сказать какой алгоритм более эффективен невозможно.

График 4 -график зависимости Сф+Мф для анализируемых алгоритмов

# Задание

## Алгоритм простого выбора в разных случаях

Таблица 4 – Таблица результатов алгоритма простого выбора в худшем случае

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n) (сек)** | **Тт=f(C+M)** | **Тп=Cф+Mф** |
| 100 | 0.5422 | 5050 | 5049 |
| 1000 | 0.05849 | 500500 | 500499 |
| 10000 | 0.6043 | 50005000 | 50004999 |
| 100000 | 5.124 | 5000050000 | 5000049999 |
| 1000000 | 115.6863 | 500000500000 | 500000499999 |

Таблица 5 – Таблица результатов алгоритма простого выбора в лучшем случае

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n) (сек)** | **Тт=f(C+M)** | **Тп=Cф+Mф** |
| 100 | 0.4922 | 5050 | 99 |
| 1000 | 0.4775 | 500500 | 999 |
| 10000 | 0.04784 | 50005000 | 9999 |
| 100000 | 0.03021 | 5000050000 | 99999 |
| 1000000 | 0.04102 | 500000500000 | 999999 |

## Пирамидальная сортировка (Heap sort)

Таблица 4 – Таблица результатов алгоритма пирамидальной сортировки в худшем случае

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n) (сек)** | **Тт=f(C+M)** | **Тп=Cф+Mф** |
| 100 | 0.05096 | 1300 | 1084 |
| 1000 | 0.539 | 20500 | 17134 |
| 10000 | 0.079 | 273000 | 238394 |
| 100000 | 0.04323 | 3420000 | 3044870 |
| 1000000 | 0.2356 | 41000000 | 37166818 |

Таблица 5 – Таблица результатов алгоритма пирамидальной сортировки в лучшем случае

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n) (сек)** | **Тт=f(C+M)** | **Тп=Cф+Mф** |
| 100 | 0.4922 | 1300 | 1232 |
| 1000 | 0.05388 | 20500 | 19918 |
| 10000 | 0.06399 | 273000 | 268914 |
| 100000 | 0.493 | 3420000 | 3351710 |
| 1000000 | 0.1994 | 41000000 | 40075586 |

## Ускоренная сортировка простого слияния (Merge sort)

Таблица 6 – Таблица результатов алгоритма сортировки простого слияния в худшем случае

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n) (сек)** | **Тт=f(C+M)** | **Тп=Cф+Mф** |
| 100 | 0.4973 | 1300 | 1102 |
| 1000 | 0.539 | 20500 | 16915 |
| 10000 | 0.06679 | 273000 | 220608 |
| 100000 | 0.0403 | 3420000 | 2708002 |
| 1000000 | 0.1956 | 41000000 | 32044950 |

Таблица 7 – Таблица результатов алгоритма сортировки простого слияния в лучшем случае

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n) (сек)** | **Тт=f(C+M)** | **Тп=Cф+Mф** |
| 100 | 0.45 | 1300 | 1126 |
| 1000 | 0.388 | 20500 | 16959 |
| 10000 | 0.09039 | 273000 | 225502 |
| 100000 | 0.3893 | 3420000 | 2772804 |
| 1000000 | 0.134 | 41000000 | 31735331 |

## Выводы по эффективному алгоритму

В ходе проделанной работы опытным путем было выяснено, что асимптотическая сложность алгоритма простого выбора зависит от “сложности” входных данных и при наилучшем случае имеет сложность O(f(n))= n, в то время как сложность алгоритма пирамидальной сортировки и сортировки слиянием не зависит от “сложности” входных данных и сохраняет асимптотическую сложность.

Отсюда можно сделать вывод что для лучшего случая наиболее эффективным будет являться алгоритм простого выбора, а для все остальных случаем равно подойдут алгоритмы пирамидальной сортировки и сортировки слиянием.

В силу в целом меньшей асимптотической сложности и более предсказуемого времени выполнения лучшем для всех случаем можно назвать как алгоритм пирамидальной сортировки, так и алгоритм сортировки слиянием.

Таблица 7 – таблицы для рассматриваемых в задании алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм | Асимптотическая сложность алгоритма | | | |
| Наихудший случай | Наилучший случай | Средний случай | Емкостная сложность |
| Простого выбора | O(n) | O(n2) | O(n2) | O(n) |
| Пирамидальная сортировка | O(n\*log(n)) | O(n\*log(n)) | O(n\*log(n)) | O(n) |
| Простое слияние | O(n\*log(n)) | O(n\*log(n)) | O(n\*log(n)) | O(2n) |