

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №7**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Алгоритмы сортировки и оценка сложности алгоритма сортировки

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-06-22 |  | Кликушин В.И. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

ОГЛАВЛЕНИЕ

[1 АТД ЗАДАЧИ 4](#_Toc135145377)

[2 ЗАДАНИЕ 1 6](#_Toc135145378)

[2.1 КОД ПРОГРАММЫ 6](#_Toc135145379)

[2.2 ОЦЕНКА КОЛИЧЕСТВА ВЫПОЛНЯЕМЫХ ОПЕРАЦИЙ 6](#_Toc135145380)

[2.3 ТАБЛИЦА 1 7](#_Toc135145381)

[2.4 ГРАФИК ПО ТРЕМ СТОЛБЦАМ ТАБЛИЦЫ 1 НА КООРДИНАТНОЙ ПРЯМОЙ 8](#_Toc135145382)

[2.5 ТАБЛИЦЫ 2,3 8](#_Toc135145383)

[2.6 ГРАФИКИ СЛУЧАЕВ: ЛУЧШИЙ, ХУДШИЙ, СРЕДНИЙ НА ОДНОЙ КООРДИНАТНОЙ ПЛОСКОСТИ 9](#_Toc135145384)

[2.7 ВЫВОДЫ О ЗАВИСИМОСТИ ВРЕМЕНИ ВЫПОЛНЕНИЯ АЛГОРИТМА ОТ РАЗМЕРА ВХОДНЫХ ДАННЫХ И УПОРЯДОЧЕННОСТИ МАССИВА. АСИМПТОТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМА ДЛЯ ВСЕХ ТРЕХ СЛУЧАЕВ 10](#_Toc135145385)

[2.8 ЗАВИСИМОСТЬ ОБЪЕМА ПАМЯТИ ОТ РАЗМЕРА ОБРАБАТЫВАЕМЫХ ДАННЫХ. НОТАЦИЯ BIG-O 10](#_Toc135145386)

[3 ЗАДАНИЕ 2 12](#_Toc135145387)

[3.1 КОД ПРОГРАММЫ 12](#_Toc135145388)

[3.2 ТАБЛИЦЫ 4,5,6 12](#_Toc135145389)

[3.3 ГРАФИКИ ПО ТРЕМ СТОЛБЦАМ ТАБЛИЦЫ 4 НА ОДНОЙ КООРДИНАТНОЙ ПРЯМОЙ 15](#_Toc135145390)

[3.4 ГРАФИКИ СЛУЧАЕВ: ЛУЧШИЙ, ХУДШИЙ, СРЕДНИЙ НА ОДНОЙ КООРДИНАТНОЙ ПЛОСКОСТИ 16](#_Toc135145391)

[3.5 ВЫВОДЫ О ЗАВИСИМОСТИ ВРЕМЕНИ ВЫПОЛНЕНИЯ АЛГОРИТМА ОТ РАЗМЕРА ВХОДНЫХ ДАННЫХ И УПОРЯДОЧЕННОСТИ МАССИВА. АСИМПТОТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМА ДЛЯ ВСЕХ ТРЕХ СЛУЧАЕВ 16](#_Toc135145392)

[3.6 ЗАВИСИМОСТЬ ОБЪЕМА ПАМЯТИ ОТ РАЗМЕРА ОБРАБАТЫВАЕМЫХ ДАННЫХ. НОТАЦИЯ BIG-O 17](#_Toc135145393)

[4 ЗАДАНИЕ 3 18](#_Toc135145394)

[4.1 КОД ПРОГРАММЫ 18](#_Toc135145395)

[4.2 ТАБЛИЦЫ 8,9,10 18](#_Toc135145396)

[4.3 ГРАФИКИ ПО ТРЕМ СТОЛБЦАМ ТАБЛИЦЫ 7 НА ОДНОЙ КООРДИНАТНОЙ ПРЯМОЙ 20](#_Toc135145397)

[4.4 ГРАФИКИ СЛУЧАЕВ: ЛУЧШИЙ, ХУДШИЙ, СРЕДНИЙ НА ОДНОЙ КООРДИНАТНОЙ ПЛОСКОСТИ 21](#_Toc135145398)

[4.5 ВЫВОДЫ О ЗАВИСИМОСТИ ВРЕМЕНИ ВЫПОЛНЕНИЯ АЛГОРИТМА ОТ РАЗМЕРА ВХОДНЫХ ДАННЫХ И УПОРЯДОЧЕННОСТИ МАССИВА. АСИМПТОТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМА ДЛЯ ВСЕХ ТРЕХ СЛУЧАЕВ 21](#_Toc135145399)

[4.6 ЗАВИСИМОСТЬ ОБЪЕМА ПАМЯТИ ОТ РАЗМЕРА ОБРАБАТЫВАЕМЫХ ДАННЫХ. НОТАЦИЯ BIG-O 22](#_Toc135145400)

[5 ЗАДАНИЕ 4 23](#_Toc135145401)

[5.1 ГРАФИКИ ХУДШИХ СЛУЧАЕВ ДЛЯ СОРТИРОВОК 23](#_Toc135145402)

[5.2 ТАБЛИЦА АССИМПТОТИЧЕСКИХ СЛОЖНОСТЕЙ АЛГОРИТМОВ 24](#_Toc135145403)

[6 ВЫВОДЫ 26](#_Toc135145404)

[7 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ 27](#_Toc135145405)

# 1 АТД ЗАДАЧИ

АТД Data

{

Данные

data – динамический целочисленный массив;

n – количество элементов массива.

Операции

1. Создание массива

Предусловие. data – ссылка на элемент типа Data, n – целое число, задающее размер массива.

Постусловие. Создаёт динамический массив из n элементов. Нет возвращаемого значения.

**create\_array(Data &data, int n);**

1. Заполнение массива значениями с клавиатуры

Предусловие. data – ссылка на элемент типа Data.

Постусловие. Заполняет динамический массив значениями с клавиатуры. Нет возвращаемого значения.

**input\_array(Data& data);**

1. Заполнение массива случайными элементами

Предусловие. data – ссылка на элемент типа Data.

Постусловие. Заполняет динамический массив случайными значениями. Нет возвращаемого значения.

**random\_input\_array(Data& data);**

1. Заполнение массива числами по возрастанию

Предусловие. data – ссылка на элемент типа Data.

Постусловие. Заполняет динамический массив n элементами числами по возрастанию – от 1 до n включительно. Нет возвращаемого значения.

**input\_by\_ascending\_order(Data& data);**

1. Заполнение массива числами по убыванию

Предусловие. data – ссылка на элемент типа Data.

Постусловие. Заполняет динамический массив n элементами числами по убыванию – от n до 1 включительно. Нет возвращаемого значения.

**input\_by\_descending\_order(Data& data);**

1. Вывод элементов массива

Предусловие. data – элемент типа Data.

Постусловие. Выводит элементы массива через пробел. Нет возвращаемого значения.

**output\_array(Data data);**

1. Сортировка вставками

Предусловие. data – ссылка на элемент типа Data.

Постусловие. Отсортированный по возрастанию массив из n элементов. Нет возвращаемого значения.

**insert\_sort(Data& data);**

1. Определение количества битонных последовательностей (вспомогательная функция для битонной сортировки)

Предусловие. n – целое число.

Постусловие. Возвращает целое число – количество битонных последовательностей.

**pow\_of\_two(int n);**

1. Генерация битонных последовательностей (вспомогательная функция для битонной сортировки)

Предусловие. data – указатель на целочисленный динамический массив, low – целое число, n – целое число, dir – булева переменная.

Постусловие. Выстраивает элементы массива в битонную последовательность. Нет возвращаемого значения.

**merge(int\* data, int low, int n, bool dir);**

1. Битонная сортировка

Предусловие. data – указатель на целочисленный динамический массив, low – целое число, n – целое число, dir – булева переменная.

Постусловие. Отсортированный по возрастанию массив из n элементов. Нет возвращаемого значения.

**bitonic\_sort(int\* data, int low, int n, bool dir);**

1. Быстрая рандомизированная сортировка

Предусловие. data – ссылка на элемент типа Data, low, high – целые числа.

Постусловие. Отсортированный по возрастанию массив из n элементов. Нет возвращаемого значения.

# 2 ЗАДАНИЕ 1

## 2.1 КОД ПРОГРАММЫ

void insert\_sort(Data& data)

{

for (int i = 1; i < data.n; i++)

{

int x = data.data[i];

int j = i - 1;

while (j >= 0 and data.data[j] >= x)

{

data.data[j + 1] = data.data[j];

j--;

}

data.data[j + 1] = x;

}

}

## 2.2 ОЦЕНКА КОЛИЧЕСТВА ВЫПОЛНЯЕМЫХ ОПЕРАЦИЙ

|  |  |
| --- | --- |
| Оператор | Количество операторов |
| for i ← 1 to n do | n |
| x ← data[i] | n - 1 |
| j ← i - 1 | n - 1 |
| while j >= 0 and data[j] >= x do | n\*n |
| data[j + 1] ← a[j] | n\*n - 1 |
| j ← j - 1 | n\*n - 1 |
| data[j + 1] ← x | n - 1 |
| od  od |  |

Лучший случай - массив отсортирован в соответствии с критерием сортировки. Внутренний цикл не выполняется, а сравнение выполняется. Тогда, за n проходов количество сравнений С(n) = n, а количество перемещений M(n) = 3(n-1) + n = 4n - 3. Сложность О(n).

Худший случай - массив полностью не сортирован по заданному критерию. Внутренний цикл выполнится каждый раз. Тогда, сравнений C(n)=n+n\*n, а перемещений M(n) = n + n – 1 + n - 1 + 2(n\*n – 1) + n – 1 = 2n2 + 4n – 5. Сложность О(n2). Средний случай рассчитаем как среднее арифметическое наилучшего и наихудшего случая: (2n2 + 4n – 5 + 4n – 3)/2 = n2 + 4n – 4. Сложность О(n2).

## 2.3 ТАБЛИЦА 1

Таблица 1. Анализ сортировки для случайно заполненного массива

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) время в мс/сек | Tэт = f(C + M) = n2 + 4n - 4 | Tэп = Cф + Mф - количество |
| 100 | 0,006 мс | 10404 | Сравнений: 4837  Перемещений: 5134  Общее: 9971 |
| 1000 | 0, 526 мс | 1004004 | Сравнений: 505205  Перемещений: 508202  Общее: 1013407 |
| 10000 | 46 мс | 100040004 | Сравнений: 50090679  Перемещений: 50120676  Общее: 100211355 |
| 100000 | 4542 мс | 10000400004 | Сравнений: 724896597  Перемещений: 725196594  Общее: 1450093191 |
| 1000000 | 572875 мс | 1000004000004 | Сравнений: 500497087379  Перемещений: 500500087376  Общее: 1000997174755 |

## 2.4 ГРАФИК ПО ТРЕМ СТОЛБЦАМ ТАБЛИЦЫ 1 НА КООРДИНАТНОЙ ПРЯМОЙ

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Рисунок 1 – Графики по трём столбцам таблицы 1

## 2.5 ТАБЛИЦЫ 2,3

Таблица 2. Анализ сортировки массива, заполненного по убыванию

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) – время в мс/сек | Tэт = f(C + M) = 2n2 + 4n - 5 | Tэп = Cф + Mф |
| 100 | 0,053 мс | 20395 | Сравнений: 9999  Перемещений: 10296  Общее: 20295 |
| 1000 | 1 мс | 2003995 | Сравнений: 999999  Перемещений: 1002996  Общее: 2002995 |
| 10000 | 134 мс | 200039995 | Сравнений: 99999999  Перемещений: 100029996  Общее: 200029995 |
| 100000 | 14106 мс | 20000399995 | Сравнений: 9999999999  Перемещений: 10000299996  Общее: 20000299995 |
| 1000000 | 1481818 мс | 2000003999995 | Сравнений: 999999999999  Перемещений: 1000002999996  Общее: 2000002999995 |

Таблица 3. Анализ сортировки массива, заполненного по возрастанию

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) – время в мс/сек | Tэт = f(C + M) =  4n - 3 | Tэп = Cф + Mф |
| 100 | 0,0007 мс | 397 | Сравнений: 198  Перемещений: 396  Общее: 594 |
| 1000 | 0,0037 мс | 3997 | Сравнений: 1998  Перемещений: 3996  Общее: 5994 |
| 10000 | 0,036 мс | 39997 | Сравнений: 19998  Перемещений: 39996  Общее: 59994 |
| 100000 | 0,377 мс | 399997 | Сравнений: 199998  Перемещений: 399996  Общее: 599994 |
| 1000000 | 3 мс | 3999997 | Сравнений: 1999998  Перемещений: 3999996  Общее: 5999994 |

## 2.6 ГРАФИКИ СЛУЧАЕВ: ЛУЧШИЙ, ХУДШИЙ, СРЕДНИЙ НА ОДНОЙ КООРДИНАТНОЙ ПЛОСКОСТИ

На рисунке 2 продемонстрирована зависимость количества операций от размера задачи в трёх случаях. Лучший случай на графике выглядит как прямая горизонтальная линия, что говорит о том, насколько быстро алгоритм проходит уже отсортированный массив по сравнению со средним и худшим случаями.

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рисунок 2 – График трёх случаев

## 2.7 ВЫВОДЫ О ЗАВИСИМОСТИ ВРЕМЕНИ ВЫПОЛНЕНИЯ АЛГОРИТМА ОТ РАЗМЕРА ВХОДНЫХ ДАННЫХ И УПОРЯДОЧЕННОСТИ МАССИВА. АСИМПТОТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМА ДЛЯ ВСЕХ ТРЕХ СЛУЧАЕВ

Глядя на таблицы сортировки методом простой вставки, можно сделать вывод о том, что скорость сортировки массива напрямую зависит не только от размера входных данных, но и непосредственно и от самих входных данных. Так, например, массив из 100 000 элементов, заполненный числами по возрастанию был отсортирован менее чем за секунду, а такой же массив, заполненный в обратном порядке, сортировался уже порядка 14 секунд.

Асимптотическая сложность в худшем и среднем случаях: O(n2); в лучшем случае: O(n).

## 2.8 ЗАВИСИМОСТЬ ОБЪЕМА ПАМЯТИ ОТ РАЗМЕРА ОБРАБАТЫВАЕМЫХ ДАННЫХ. НОТАЦИЯ BIG-O

Выделение дополнительной памяти для хранения массива не требуется, поэтому ёмкостная сложность O(1).

# 3 ЗАДАНИЕ 2

## 3.1 КОД ПРОГРАММЫ

int pow\_of\_two(int n)

{

int k = 1;

while (k > 0 && k < n)

{

k = k << 1;

}

return k >> 1;

}

void merge(int\* data, int low, int n, bool dir)

{

if (n > 1)

{

int m = pow\_of\_two(n);

for (int i = low; i < low + n - m; i++)

{

if (dir == (data[i] > data[i + m]))

{

swap(data[i], data[i + m]);

}

}

merge(data, low, m, dir);

merge(data, low + m, n - m, dir);

}

}

void bitonic\_sort(int\* data, int low, int n, bool dir)

{

int m;

if (n > 1)

{

m = n / 2;

bitonic\_sort(data, low, m, !dir);

bitonic\_sort(data, low + m, n - m, dir);

merge(data, low, n, dir);

}

}

## 3.2 ТАБЛИЦЫ 4,5,6

Таблица 4. Анализ сортировки для случайно заполненного массива

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) – время в мс/сек | Tэт = f(C + M)  n \* log2n | Tэп = Cф + Mф |
| 100 | 0,0323 мс | 2121 | Сравнений: 4978  Перемещений: 3712  Общее: 8690 |
| 1000 | 0,5193 мс | 47717 | Сравнений: 92874  Перемещений: 77539  Общее: 170413 |
| 10000 | 8,0954 мс | 848304 | Сравнений: 1432274  Перемещений: 1272713  Общее: 2704987 |
| 100000 | 109,881 мс | 13254745 | Сравнений: 20748738  Перемещений: 19460543  Общее: 40209281 |
| 1000000 | 1460,31 мс | 190868332 | Сравнений: 284667778  Перемещений: 276258386  Общее: 560926164 |

Таблица 5. Анализ сортировки массива, заполненного по убыванию

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) – время в мс/сек | Tэт = f(C + M)  n \* log2n | Tэп = Cф + Mф |
| 100 | 0,0927 мс | 2121 | Сравнений: 4978  Перемещений: 3648  Общее: 8626 |
| 1000 | 0,6684 мс | 47717 | Сравнений: 92874  Перемещений: 76406  Общее: 169280 |
| 10000 | 8,1611 мс | 848304 | Сравнений: 1432274  Перемещений: 1235676  Общее: 2667950 |
| 100000 | 110,145 мс | 13254745 | Сравнений: 20748738  Перемещений: 18863136  Общее: 39611874 |
| 1000000 | 1404 мс | 190868332 | Сравнений: 284667778  Перемещений: 269648288  Общее: 554316066 |

Таблица 6. Анализ сортировки массива, заполненного по возрастанию

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) – время в мс/сек | Tэт = f(C + M)  n \* log2n | Tэп = Cф + Mф |
| 100 | 0,0865 мс | 2121 | Сравнений: 4978  Перемещений: 3526  Общее: 8504 |
| 1000 | 1,2005 мс | 47717 | Сравнений: 92874  Перемещений: 75170  Общее: 168044 |
| 10000 | 7,2128 мс | 848304 | Сравнений: 1432274  Перемещений: 1218292  Общее: 2650566 |
| 100000 | 104 мс | 13254745 | Сравнений: 20748738  Перемещений: 18645456  Общее: 39394194 |
| 1000000 | 1447 мс | 190868332 | Сравнений: 284667778  Перемещений: 267515520  Общее: 552183298 |

## 3.3 ГРАФИКИ ПО ТРЕМ СТОЛБЦАМ ТАБЛИЦЫ 4 НА ОДНОЙ КООРДИНАТНОЙ ПРЯМОЙ

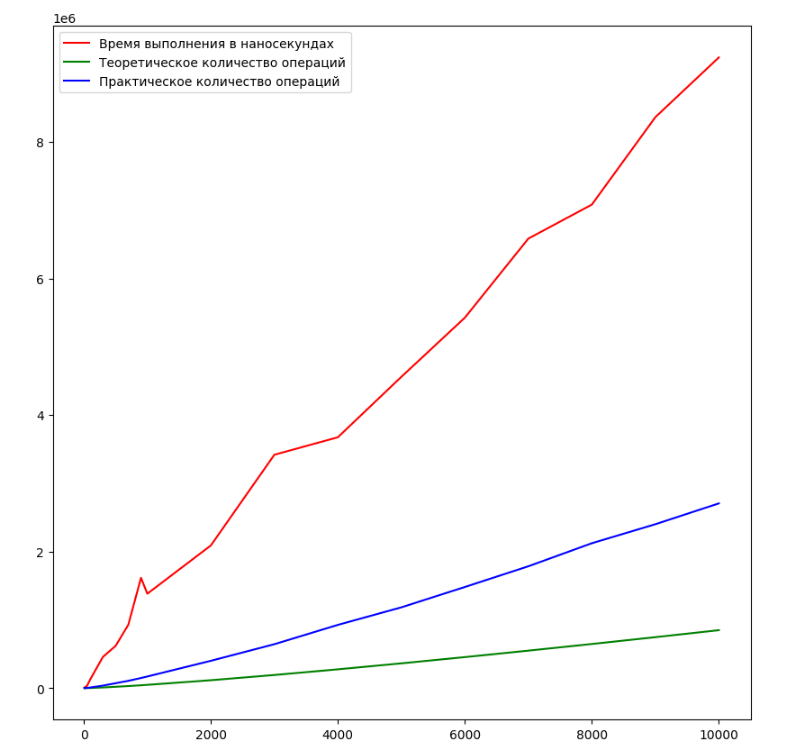


Рисунок 3 – График по таблице 4

## 3.4 ГРАФИКИ СЛУЧАЕВ: ЛУЧШИЙ, ХУДШИЙ, СРЕДНИЙ НА ОДНОЙ КООРДИНАТНОЙ ПЛОСКОСТИ

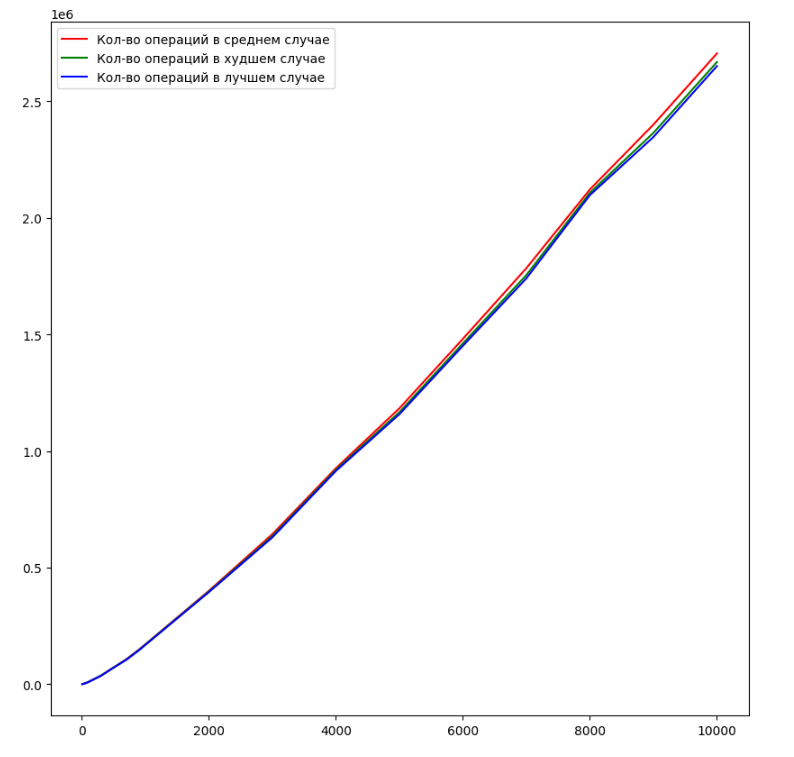


Рисунок 4 – График по трём случаям

## 3.5 ВЫВОДЫ О ЗАВИСИМОСТИ ВРЕМЕНИ ВЫПОЛНЕНИЯ АЛГОРИТМА ОТ РАЗМЕРА ВХОДНЫХ ДАННЫХ И УПОРЯДОЧЕННОСТИ МАССИВА. АСИМПТОТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМА ДЛЯ ВСЕХ ТРЕХ СЛУЧАЕВ

Можно сделать вывод, что сортировка выполняет практически константное количество операций для массивов, заполненных по возрастанию, убыванию и в случайном порядке. Таким образом, для алгоритма не имеет значение, массив с какими элементами подан для сортировки. Асимптотическая сложность во всех трёх случаях: O(n\*log2n).

## 3.6 ЗАВИСИМОСТЬ ОБЪЕМА ПАМЯТИ ОТ РАЗМЕРА ОБРАБАТЫВАЕМЫХ ДАННЫХ. НОТАЦИЯ BIG-O

Алгоритм не требует выделения дополнительной памяти для хранения данных, поэтому ёмкостная сложность: O(1). Однако алгоритм является рекурсивным, поэтому его выполнение заполняет стек рекурсивных вызовов. В общем случае, в стеке максимально может находиться log(n) вызовов.

# 4 ЗАДАНИЕ 3

## 4.1 КОД ПРОГРАММЫ

void quick\_random\_sort(Data& data, int low, int high)

{

srand(time(NULL));

if (low < high)

{

int pivotIndex = low + rand() % (high - low + 1);

int pivotValue = data.data[pivotIndex];

swap(data.data[pivotIndex], data.data[high]);

int i = low - 1;

for (int j = low; j < high; j++)

{

if (data.data[j] < pivotValue)

{

i++;

swap(data.data[i], data.data[j]);

}

}

swap(data.data[i + 1], data.data[high]);

quick\_random\_sort(data, low, i);

quick\_random\_sort(data, i + 2, high);

}

}

## 4.2 ТАБЛИЦЫ 8,9,10

Таблица 7. Анализ сортировки для случайно заполненного массива

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) – время в мс/сек | Tэт = f(C + M)  n \* logn | Tэп = Cф + Mф |
| 100 | 0,0358 мс | 460 | Сравнений: 1543  Перемещений: 1760  Общее: 3303 |
| 1000 | 0,6064 мс | 6907 | Сравнений: 22357  Перемещений: 23980  Общее: 46337 |
| 10000 | 5,0676 мс | 92103 | Сравнений: 406785  Перемещений: 408427  Общее: 815212 |
| 100000 | 51,495 мс | 1151290 | Сравнений: 12988807  Перемещений: 8200563  Общее: 21189370 |
| 1000000 | 1236,71 мс | 69077553 | Сравнений: 1028618973  Перемещений: 530657780  Общее: 1559276753 |

Таблица 8. Анализ сортировки массива, заполненного по убыванию

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) – время в мс/сек | Tэт = f(C + M)  n \* logn | Tэп = Cф + Mф |
| 100 | 0.0293 мс | 460 | Сравнений: 1398  Перемещений: 1504  Общее: 2902 |
| 1000 | 0,635 мс | 6907 | Сравнений: 22737  Перемещений: 25847  Общее: 48584 |
| 10000 | 4,3926 мс | 92103 | Сравнений: 326328  Перемещений: 379663  Общее: 705991 |
| 100000 | 114,669 мс | 1151290 | Сравнений: 9359847  Перемещений: 12855781  Общее: 22215628 |
| 1000000 | 5790,01 мс | 69077553 | Сравнений: 597018649  Перемещений: 883440080  Общее: 1480458729 |

Таблица 9. Анализ сортировки массива, заполненного по возрастанию

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) – время в мс/сек | Tэт = f(C + M)  n \* logn | Tэп = Cф + Mф |
| 100 | 0.0788 мс | 460 | Сравнений: 1477  Перемещений: 1699  Общее: 3176 |
| 1000 | 0.8437 мс | 6907 | Сравнений: 22243  Перемещений: 25675  Общее: 47918 |
| 10000 | 1.4172мс | 92103 | Сравнений: 149055  Перемещений: 131343  Общее: 280398 |
| 100000 | 4,1846 мс | 1151290 | Сравнений: 419746  Общее: 21189370 |
| 1000000 | 648,081 мс | 69077553 | Сравнений: 407070638  Перемещений: 219991088  Общее: 627061726 |

## 4.3 ГРАФИКИ ПО ТРЕМ СТОЛБЦАМ ТАБЛИЦЫ 7 НА ОДНОЙ КООРДИНАТНОЙ ПРЯМОЙ

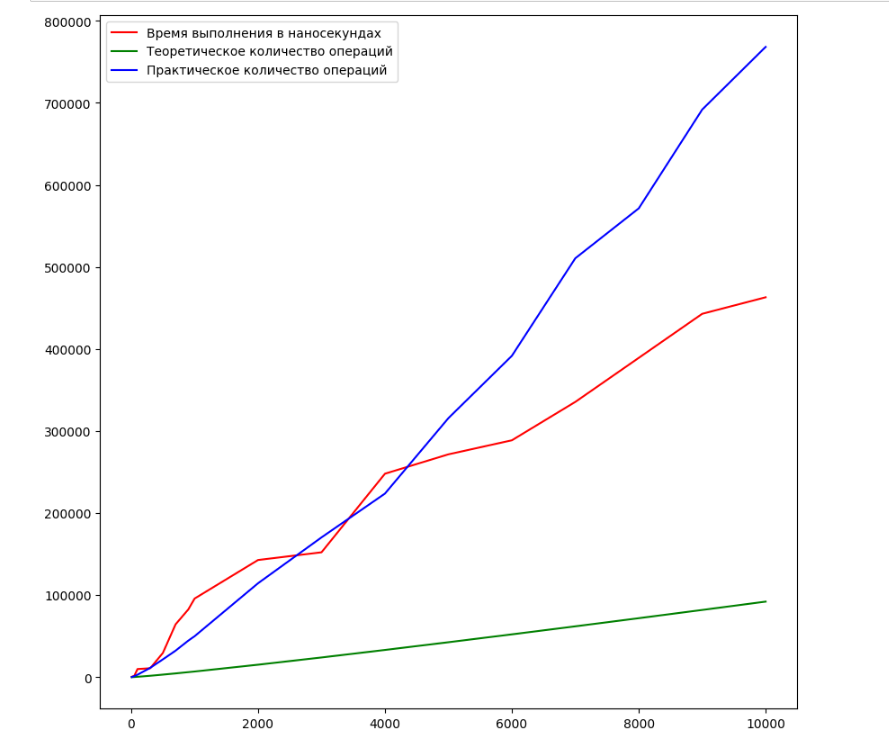


Рисунок 5 – График по таблице 7

## 4.4 ГРАФИКИ СЛУЧАЕВ: ЛУЧШИЙ, ХУДШИЙ, СРЕДНИЙ НА ОДНОЙ КООРДИНАТНОЙ ПЛОСКОСТИ

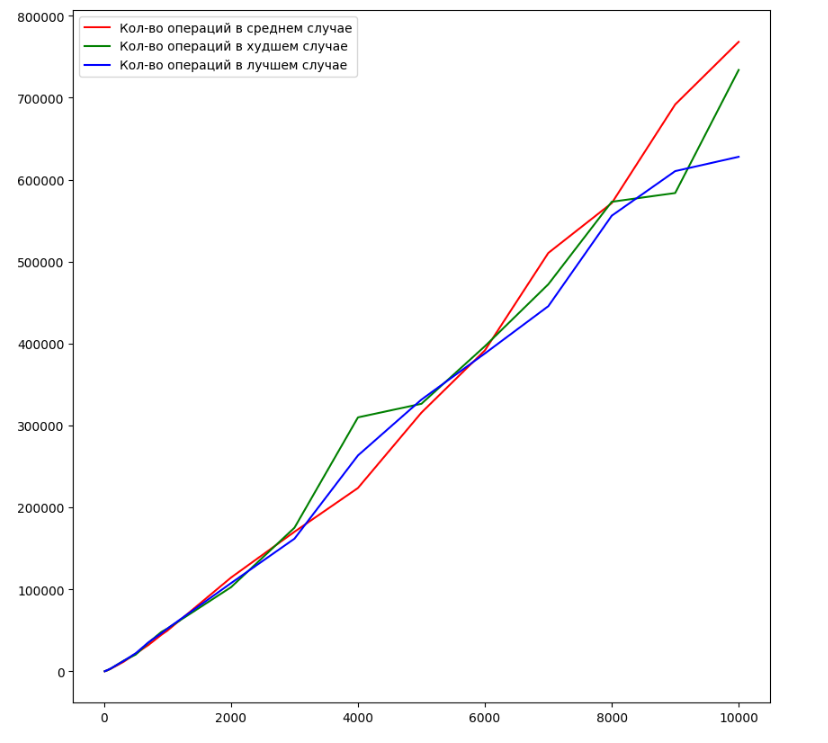


Рисунок 6 – График по трём случаям

## 4.5 ВЫВОДЫ О ЗАВИСИМОСТИ ВРЕМЕНИ ВЫПОЛНЕНИЯ АЛГОРИТМА ОТ РАЗМЕРА ВХОДНЫХ ДАННЫХ И УПОРЯДОЧЕННОСТИ МАССИВА. АСИМПТОТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМА ДЛЯ ВСЕХ ТРЕХ СЛУЧАЕВ

У данного алгоритма может быть асимптотическая сложность O(n2) лишь в единственном случае: опорный элемент равен максимальному элементу в массиве. Тогда разбиение приводит к тому, что левая часть состоит из n-1 элемента, а правая из одного. Тогда вместо logn разбиений требуется n разбиений и скорость выполнения сортировки имеет порядок n2. Во всех обобщенных же случаях асимптотическая сложность O(n\*logn). Разница между отсортированным по убыванию и отсортированным по возрастанию массивами невелика, т.к. в качестве опорного элемента выбирается случайный.

## 4.6 ЗАВИСИМОСТЬ ОБЪЕМА ПАМЯТИ ОТ РАЗМЕРА ОБРАБАТЫВАЕМЫХ ДАННЫХ. НОТАЦИЯ BIG-O

Алгоритм не требует выделения дополнительной памяти для хранения данных, поэтому ёмкостная сложность: O(1). Однако алгоритм является рекурсивным, поэтому его выполнение заполняет стек рекурсивных вызовов. В общем случае, в стеке максимально может находиться log(n) вызовов.

# 5 ЗАДАНИЕ 4

## 5.1 ГРАФИКИ ХУДШИХ СЛУЧАЕВ ДЛЯ СОРТИРОВОК

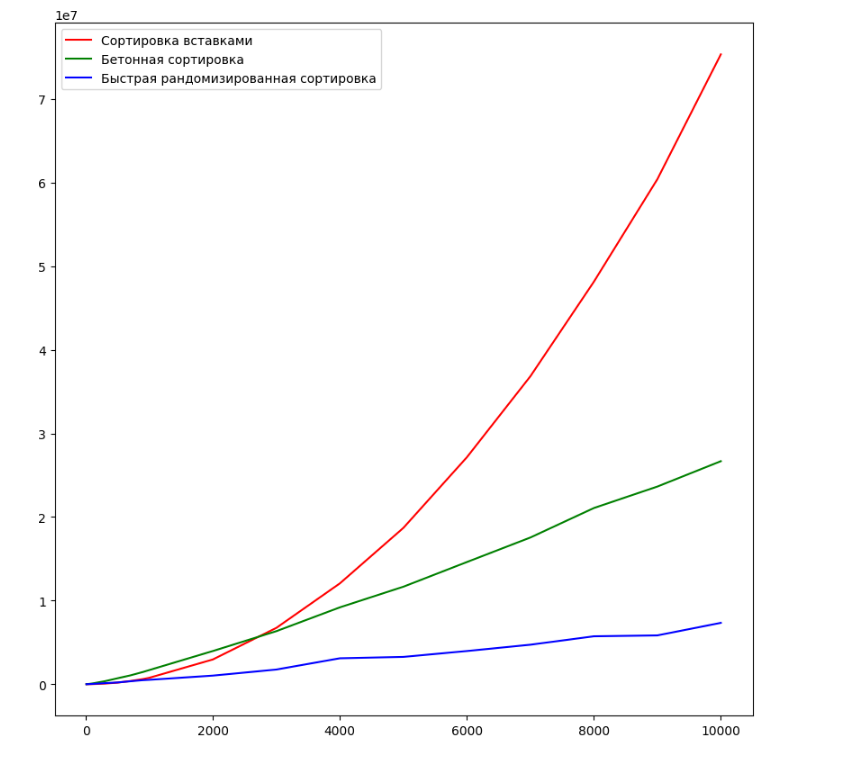


Рисунок 7 – Графики количества операций при худших случаях

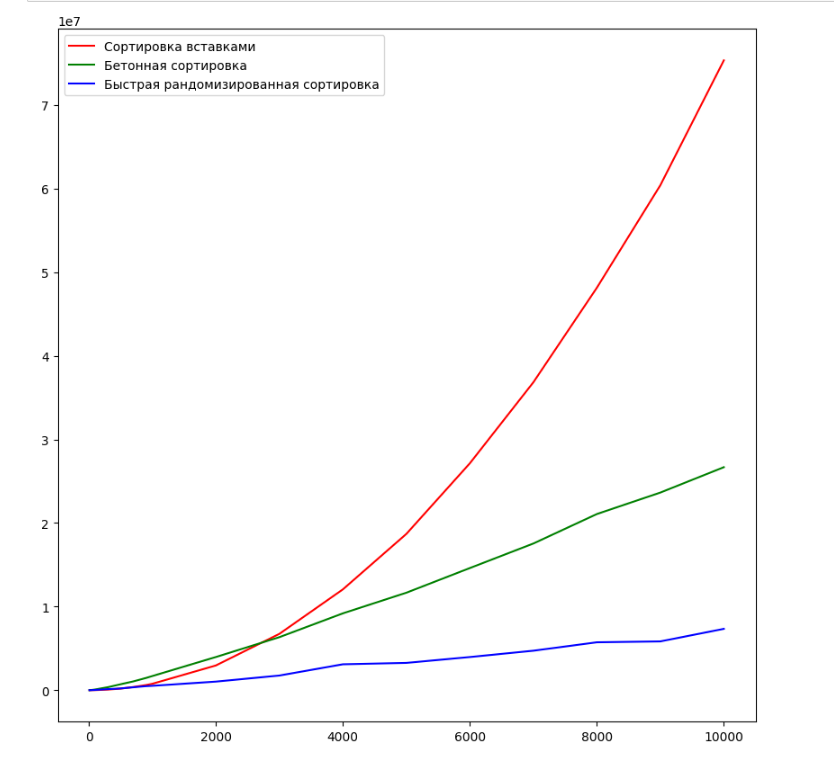


Рисунок 8 – Графики времени выполнения сортировок в худших случаях

По графикам видно, что с ростом количества элементов массива наиболее эффективной сортировкой является быстрая рандомизированная. Можно сделать вывод о том, что начиная с n = 1000 скорость роста времени выполнения сортировки простыми вставками резко возрастает.

## 5.2 ТАБЛИЦА АССИМПТОТИЧЕСКИХ СЛОЖНОСТЕЙ АЛГОРИТМОВ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название алгоритма | Асимптотическая сложность | | | |
| Массив упорядочен по убыванию | Массив упорядочен по возрастанию | Массив из случайных элементов | Ёмкостная сложность |
| Сортировка вставками | O(n2) | O(n) | O(n2) | O(1) |
| Битонная сортировка | O(n\*log2n) | O(n\*log2n) | O(n\*log2n) | O(1) |
| Быстрая рандомизированная сортировка | O(n\*logn) | O(n\*logn) | O(n\*logn) | O(logn)) |

# 6 ВЫВОДЫ

Рассмотрев три сортировки в соответствие с вариантом, можно сделать вывод, о том, что для массивов небольшого размера наиболее эффективной является сортировка простыми вставками. Однако, при подаче большего количества элементов время выполнения этой сортировки резко возрастает, более эффективными становятся битонная и быстрая рандомизированная сортировка.

# 7 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Учебно-методическое пособие СиАОД (часть 1)

2. Приложение к практическим работам – СДО (online-edu.mirea.ru)