МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Отчёт по выполнению практического задания № 2
Тема:
«Эмпирический анализ сложности простых алгоритмов сортировки»
Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Фамилия И.О.
Фамилия И.О.

Группа: АААА-00-00
Номер группы

Москва 2023

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Цель работы

Актуализация знаний и приобретение практических умений по эмпирическому определению вычислительной сложности алгоритмов.

1.2 Задание 1

Оценить эмпирически вычислительную сложность алгоритма простой сортировки на массиве, заполненном случайными числами (средний случай).

1. Составить функцию простой сортировки одномерного целочисленного массива A[n], используя алгоритм согласно варианту индивидуального задания (столбец Алгоритм заданий 1 и 2 в таблице 1). Провести тестирование программы на исходном массиве n=10.

2. Используя теоретический подход, определить для алгоритма:

a. Что будет ситуациями лучшего, среднего и худшего случаев.

b. Функции роста времени работы алгоритма от объёма входа для лучшего и худшего случаев.

3. Провести контрольные прогоны программы массивов случайных чисел при n = 100, 1000, 10000, 100000 и 1000000 элементов с вычислением времени выполнения T(n) – (в миллисекундах/секундах). Полученные результаты свести в сводную таблицу.

4. Провести эмпирическую оценку вычислительной сложности алгоритма, для чего предусмотреть в программе подсчет фактического количества критических операций $T_п$ как сумму сравнений $C_п$ и перемещений $M_п$. Полученные результаты вставить в сводную таблицу 2.

5. Построить график функции роста $T_п$ этого алгоритма от размера массива n.

6. Определить ёмкостную сложность алгоритма.

7. Сделать вывод об эмпирической вычислительной сложности алгоритма на основе скорости роста функции роста.

1.2 Задание 2

Оценить вычислительную сложность алгоритма простой сортировки в наихудшем и наилучшем случаях.

1. Провести дополнительные прогоны программы на массивах при n = 100, 1000, 10000, 100000 и 1000000 элементов, отсортированных:

a. строго в убывающем порядке значений, результаты представить в сводной таблице.

b. строго в возрастающем порядке значений, результаты представить в сводной таблице.

2. Сделать вывод о зависимости (или независимости) алгоритма сортировки от исходной упорядоченности массива.

1.3 Задание 3

Сравнить эффективность алгоритмов простых сортировок.

1. Выполнить разработку и программную реализацию второго алгоритма согласно индивидуальному варианту в столбце Алгоритм задания 3 из таблицы 1.

2. Аналогично заданиям 1 и 2 сформировать таблицы с результатами эмпирического исследования второго алгоритма в среднем, лучшем и худшем случаях в соответствии с форматом Таблицы 2 (на тех же массивах, что и в заданиях 1 и 2).

3. Определить ёмкостную сложность алгоритма от n.

4. На одном сравнительном графике отобразить функции Тп(n) двух алгоритмов сортировки в худшем случае.

5. Аналогично на другом общем графике отобразить функции Тп(n) двух алгоритмов сортировки для лучшего случая.

6. Выполнить сравнительный анализ полученных результатов для двух алгоритмов.

1.4 Индивидуальное задание

Индивидуальный вариант алгоритмов представлен в таблице 1.

Таблица 1 — Вариант индивидуального задания

№	Алгоритм заданий 1 и 2	Алгоритм задания 3
2	Простого обмена («пузырек», *Exchange sort*)	Простой вставки (*Insertion sort*)

2 ЗАДАНИЕ 1

2.1 Алгоритм сортировки простого обмена, тестирование на случайных массивах размером n = 10

Алгоритм сортировки простого обмена представлен на рис. 1. Тестирование на случайном массиве размером n = 10 представлен на рис. 2.

```
void exchangeSort(int *x, int n)
{
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = i; j < n ; j++)
        {
            if (x[i] > x[j])
            {
                int temp = x[i];
                x[i] = x[j];
                x[j] = temp;
            }
        }
    }
}
```

Рисунок 1 - Алгоритм сортировки простого обмена

```
Сортировка простого обмена:
2 1 3 8 4 8 7 5 4 3
n = 10: Сравнений - 45, перемещений - 12, Время: 0 мс.
1 2 3 3 4 4 5 7 8 8
```

Рисунок 2 - Тестирование алгоритма

2.2 Определение функции роста метода сортировки простого обмена

Количество операция зависит только от количества элементов в исходном массиве. Процесс сортировки происходит только при сравнении объектов, поэтому наихудший и наилучший случай будет зависеть только от того, был ли массив отсортирован до этого по возрастанию или по убыванию. Подсчёт количества операторов для определения функции роста предоставлен в таблице 2.

Таблица 2 - Подсчёт количества операторов

Оператор	Кол-во выполнения оператора в строке	
	В лучшем случае (массив отсортирован)	В худшем случае (массив отсортирован по убыванию)
for (int i = 0; i < n - 1; i++)	n	n
for (int j = i + 1; j < n ; j++)	$\frac{n^2-n}{2}$	$\frac{n^2-n}{2}$
if (x[i] > x[j])	$\frac{n^2-n}{2}$	$\frac{n^2-n}{2}$
int temp = x[i];	0	$\frac{n^2-n}{2}$
x[i] = x[j];	0	$\frac{n^2-n}{2}$
x[j] = temp;	0	$\frac{n^2-n}{2}$

В лучшем случае $T(n) = n^2$ — квадратичная зависимость.

В худшем случае $T(n) = 2.5n^2 - 1.5n \Rightarrow n^2$ — квадратичная зависимость

2.3 Сводная таблица результатов тестирования

Произведем тестирование и сведем результаты в таблицу. Результаты тестирования приведены на рис.3.

```
Сортировка простого обмена:
n = 100: Сравнений - 4950, перемещений - 370, Время: 13 мс.

n = 1000: Сравнений - 499500, перемещений - 3929, Время: 959 мс.

n = 10000: Сравнений - 49995000, перемещений - 39848, Время: 85495 мс.

n = 100000: Сравнений - 4999950000, перемещений - 400627, Время: 8050076 мс.
```

Рисунок 3 - Результаты тестирования сортировки простого обмена

Сведённые результаты представлены в таблице 3.

Таблица 3 - Сводная таблица результатов

n	T(n), мс	$T_T = C + M$	$T_N = C_N + M_N$
100	13		5320
1000	959		503 429
10000	85494		50 034 848
100000	8050076		5 000 350 627
1000000	∞		∞

2.4 Построение графика фактического количества операций

По данным из табл. 3 построим график фактического количества операций. График изображен на рис. 4.

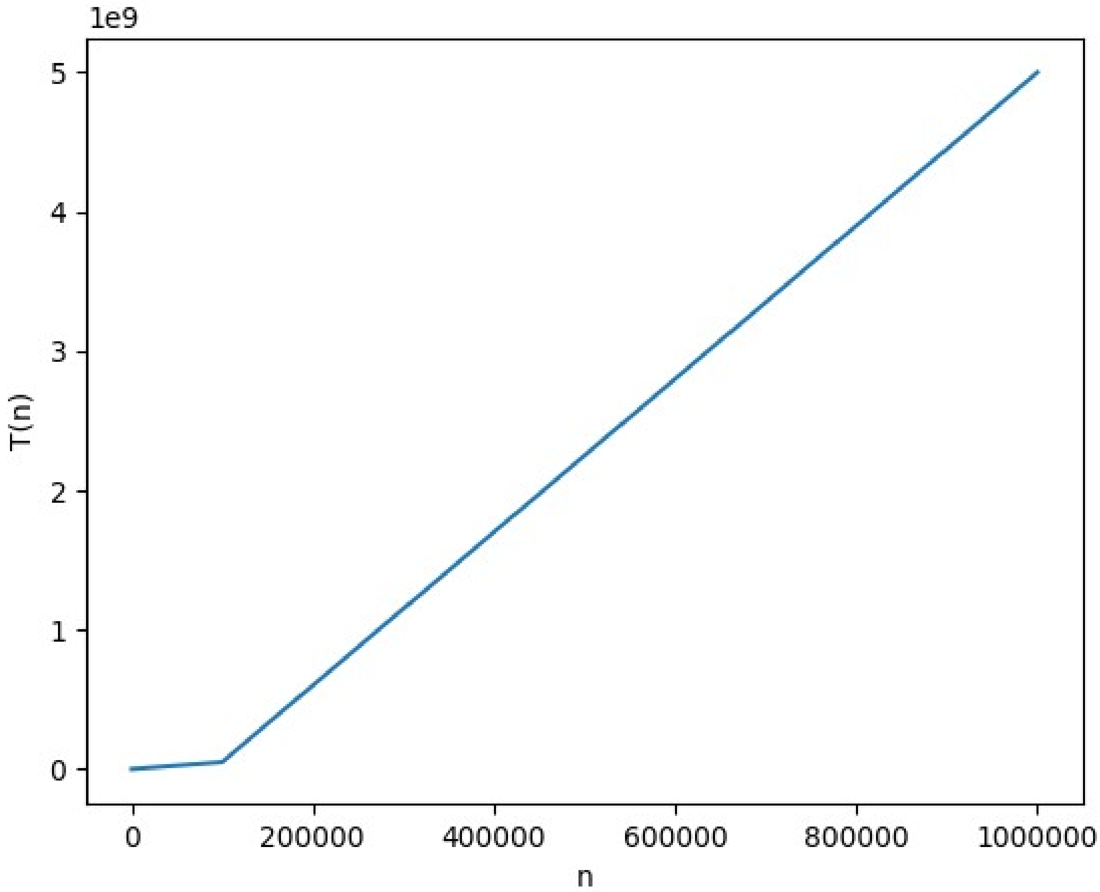


Рисунок 4 - График фактического количества операций

3 ЗАДАНИЕ 2

3.1 Результаты тестирования алгоритма сортировки простого обмена на убывающих и возрастающих массивах

Проведём тестирование алгоритма на убывающих и возрастающих массивах, и запишем результаты в сводную таблицу.

Результаты тестирования представлены на рис. 5.

```
Сортировка простого обмена (Возрастающий массив):
n = 10: Сравнений - 45, Перемещений - 0, Время: 0 мс.
n = 100: Сравнений - 4950, Перемещений - 0, Время: 9 мс.
n = 1000: Сравнений - 499500, Перемещений - 0, Время: 965 мс.
n = 10000: Сравнений - 49995000, Перемещений - 0, Время: 85885 мс.
n = 100000: Сравнений - 4999950000, Перемещений - 0, Время: 8034348 мс.

Сортировка простого обмена (Убывающий массив):
n = 10: Сравнений - 45, Перемещений - 45, Время: 0 мс.
n = 100: Сравнений - 4950, Перемещений - 4950, Время: 15 мс.
n = 1000: Сравнений - 499500, Перемещений - 499500, Время: 1492 мс.
n = 10000: Сравнений - 49995000, Перемещений - 49995000, Время: 147126 мс.
n = 100000: Сравнений - 4999950000, Перемещений - 4999950000, Время: 14821707 мс.
```

Рисунок 5 - Результаты тестирования алгоритма

Сводная таблица результатов на массиве по возрастанию представлена в табл. 4, по убыванию в табл. 5

Таблица 4 - Сводная таблица для возрастающего массива

n	**T(n), мс**	$T_T = C + M$	$T_N = C_N + M_N$
10	0	45	45
100	139	4950	4950
1000	965	499500	499500
10000	85885	49995000	49995000
100000	8034348	4999950000	4999950000

Таблица 5 - Сводная таблица для убывающего массива

n	**T(n), мс**	$T_T = C + M$	$T_N = C_N + M_N$
10	0	90	90
100	15	9900	9900
1000	1492	999000	999000
10000	147126	99990000	99990000
100000	14821707	9999900000	9999900000

3.2 Код программы и тестирование при n = 10

Код представлен на рис. 6. Тестирование представлено на рис. 7

```
void exchangeSort(int *x, int n)
{
    long long int move_count = 0;
    long long int comp_count = 0;
    auto start = high_resolution_clock::now();
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
            comp_count += 1;
            if (x[i] > x[j])
            {
                move_count += 1;
                int temp = x[i];
                x[i] = x[j];
                x[j] = temp;
            }
        }
    }
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stop - start).count();

    cout << "n = " << n << ": "
         << "Сравнений - " << comp_count
         << ", Перемещений - " << move_count
         << ", Время: " << duration << " мс."
         << endl;
}
```

Рисунок 6 - Код алгоритма простой сортировки обменом

```
Сортировка простого обмена (Возрастающий массив):
n = 10: Сравнений - 45, Перемещений - 0, Время: 0 мс.

Сортировка простого обмена (Убывающий массив):
n = 10: Сравнений - 45, Перемещений - 45, Время: 0 мс.
```

Рисунок 7 - Тестирование алгоритма при n = 10

3.3 Вывод о вычислительной сложности алгоритма

Алгоритм обладает квадратичной сложностью. У алгоритма есть лучший и худший случаи, однако порядок функции роста не зависит от упорядоченности входных значений.

4 ЗАДАНИЕ 3

4.1 Алгоритм сортировки простой вставки, тестирование на случайных массивах размером n = 10

Сортировка вставкой итерируется по массиву, сравнивая один входной элемент при каждом повторении, и формирует отсортированный выходной список. На каждой итерации алгоритм удаляет один элемент из входных данных, находит место, которому он принадлежит, в отсортированном списке и вставляет его туда. Это повторяется рекурсивно до последнего элемента.

Алгоритм сортировки простого представлен на рис. 8. Тестирование на случайном массиве размером n = 10 представлен на рис. 9.

```
void insertionSort(int *x, int n)
{
    int i, key, j;
    for (i = 1; i < n; i++)
    {
        key = x[i];
        j = i - 1;
        while (j >= 0 && x[j] > key)
        {
            x[j + 1] = x[j];
            j -= 1;
        }

        x[j + 1] = key;
    }
}
```

Рисунок 8 - Алгоритм сортировки простой вставки

```
Сортировка простой вставки:
5 7 9 1 8 7 4 2 1 2
1 1 2 2 4 5 7 7 8 9
```

Рисунок 9 - Результат тестирования при n = 10

4.2 Определение функции роста метода сортировки простой вставки

Определим ситуации лучшего и худшего случая.

Количество выполнения циклов зависит только от объема данных. Перемещение элементов происходит только если соседние элементы не отсортированы.

В следствие, лучшим случаем является подача на вход отсортированного массива, худший случаем будет, когда на вход поступит массив, отсортированный по убыванию.

Подсчёт количества операций для худшего и лучшего случая представлен в таблице 6.

Таблица 6 - Количество операторов алгоритма сортировки простыми вставками

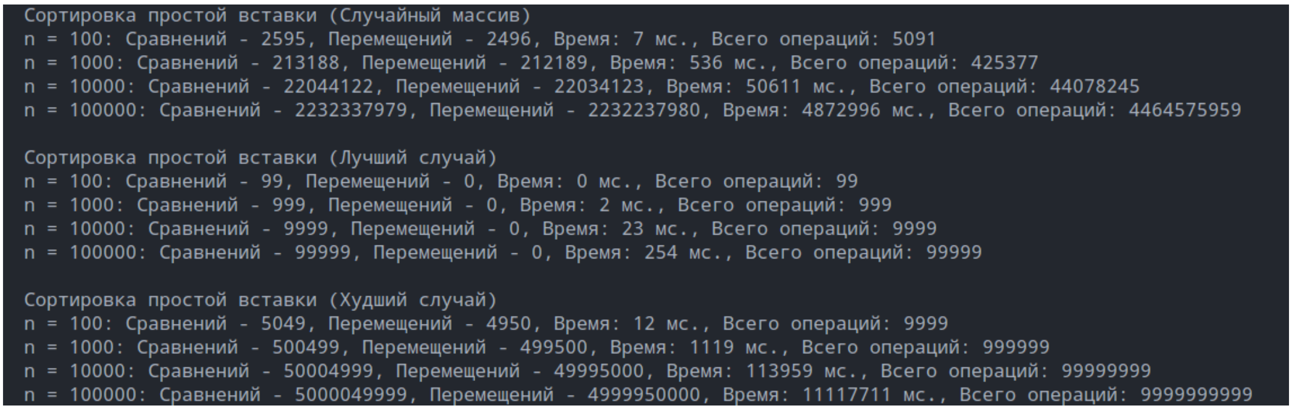
Оператор	Кол-во выполнения оператора в строке	
	В лучшем случае (массив отсортирован)	В худшем случае (массив отсортирован по убыванию)
for (i = 1; i < n; i++)	n	n
key = x[i];	n - 1	n - 1
j = i - 1;	n - 1	n - 1
while (j >= 0 && x[j] > key)	1	$\frac{n^2+n}{2}-1$
x[j + 1] = x[j];	0	$\frac{n^2-n}{2}$
j -= 1;	0	$\frac{n^2-n}{2}$
x[j + 1] = key;	n - 1	n - 1

В лучшем случае T(n) = 4n — 2 => n — линейная зависимость.

В худшем случае $T(n) = 2{,}5n^2 + 3.5n - 4 \Rightarrow n^2$ — квадратичная зависимость

4.3 Тестирование алгоритма для среднего, худшего и лучшего случаев

Результаты тестирования представлены на рис. 10.



```
Сортировка простой вставки (Случайный массив)
n = 100: Сравнений - 2595, Перемещений - 2496, Время: 7 мс., Всего операций: 5091
n = 1000: Сравнений - 213188, Перемещений - 212189, Время: 536 мс., Всего операций: 425377
n = 10000: Сравнений - 22044122, Перемещений - 22034123, Время: 50611 мс., Всего операций: 44078245
n = 100000: Сравнений - 2232337979, Перемещений - 2232237980, Время: 4872996 мс., Всего операций: 4464575959

Сортировка простой вставки (Лучший случай)
n = 100: Сравнений - 99, Перемещений - 0, Время: 0 мс., Всего операций: 99
n = 1000: Сравнений - 999, Перемещений - 0, Время: 2 мс., Всего операций: 999
n = 10000: Сравнений - 9999, Перемещений - 0, Время: 23 мс., Всего операций: 9999
n = 100000: Сравнений - 99999, Перемещений - 0, Время: 254 мс., Всего операций: 99999

Сортировка простой вставки (Худший случай)
n = 100: Сравнений - 5049, Перемещений - 4950, Время: 12 мс., Всего операций: 9999
n = 1000: Сравнений - 500499, Перемещений - 499500, Время: 1119 мс., Всего операций: 999999
n = 10000: Сравнений - 50004999, Перемещений - 49995000, Время: 113959 мс., Всего операций: 99999999
n = 100000: Сравнений - 5000049999, Перемещений - 4999950000, Время: 11117711 мс., Всего операций: 9999999999
```

Рисунок 10 - Результаты тестирования

Сводные таблицы для данных случаев (табл. 7, 8, 9).

Таблица 7 - Сводная таблица для случайного массива

n	**T(n), мс**	$T_T = C + M$	$T_N = C_N + M_N$
100	7	5049	5091
1000	536	500 499	425 377
10000	50611	50 004 999	44 078 245
100000	4872996	5 000 049 999	4 464 575 959

Таблица 8 - Сводная таблица для лучшего случая

n	**T(n), мс**	$T_T = C + M$	$T_N = C_N + M_N$
100	0	99	99
1000	2	999	999
10000	23	9999	9999
100000	254	99999	99999

Таблица 9 - Сводная таблица для худшего случая

n	**T(n), мс**	$T_T = C + M$	$T_N = C_N + M_N$
100	12	9999	9999
1000	1119	999999	999999
10000	113959	99999999	99999999
100000	11117711	9999999999	9999999999

4.4 Построение графика фактического количества операций

По данным из сводных таблиц 7, 8, 9 построим графики фактического количества операций. График изображён на рис. 11.

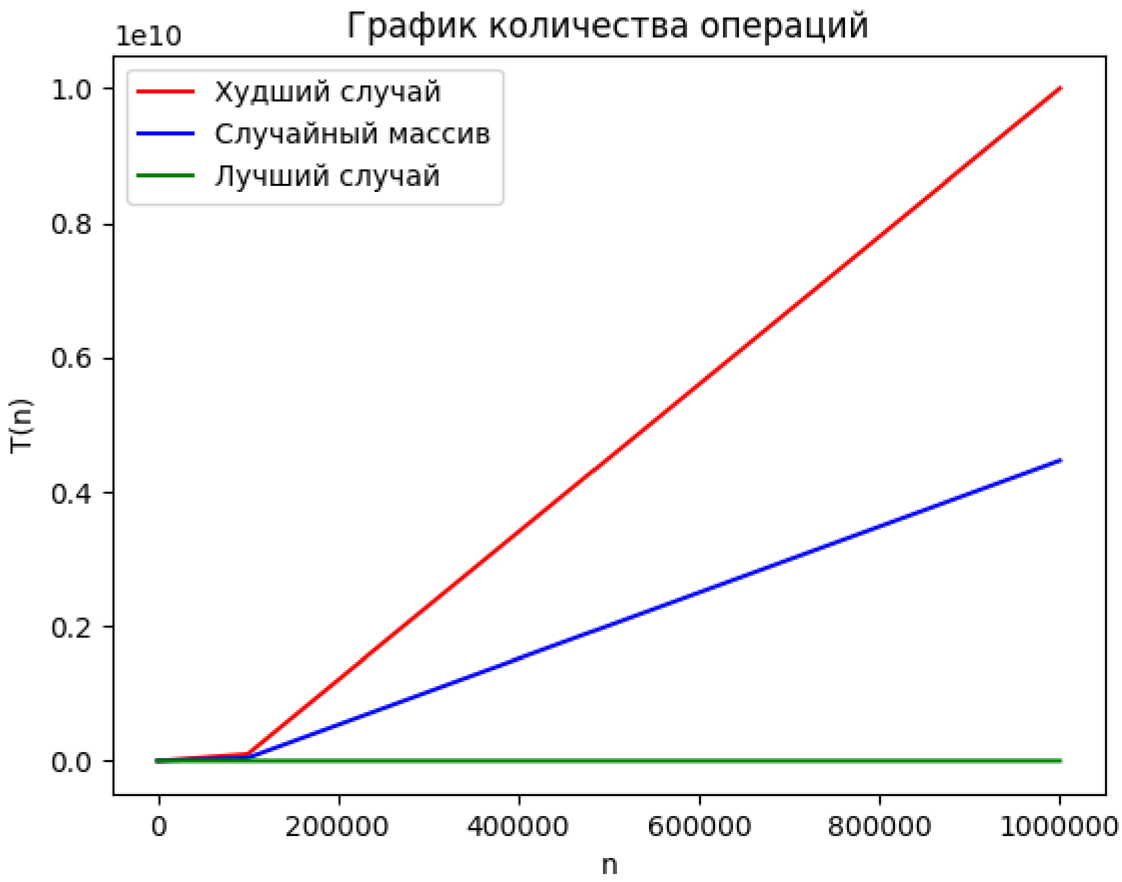


Рисунок 11 - График фактического количества операций

5 ВЫВОД

Второй алгоритм сортировки простыми вставками эффективнее чем первый алгоритм сортировки простого обмена, поскольку второй алгоритм в лучшем случае имеет линейную сложность, в среднем случае близок к линейной сложности, когда первый алгоритм в любом случае имеет квадратичную сложность.

4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. AlgoList – алгоритмы, методы, исходники [Электронный ресурс]. URL: http://algolist.manual.ru/ (дата обращения 05.03.2024).
2. Insertion Sort - Data Structure and Algorithm Tutorials [Электронный ресурс]. URL: https://www.geeksforgeeks.org/insertion-sort/ (дата обращения 05.03.2024).
3. Сортировка вставками - Материал из Википедии — свободной энциклопедии [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Сортировка_вставками