



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 №2
по дисциплине
«Структуры и алгоритмы обработки данных»

Тема: ЭМПИРИЧЕСКИЙ АНАЛИЗ СЛОЖНОСТИ ПРОСТЫХ
АЛГОРИТМОВ СОРТИРОВКИ

Выполнил студент группы ИКБО-15-23

Гольд Д.В.

Принял старший преподаватель

Скворцова Л.А.

ОГЛАВЛЕНИЕ

1.ЗАДАНИЕ №1	3
1.1. Алгоритм сортировки по методу варианта при $n=10$	3
1.2. Процесс определения функции роста метода сортировки.	4
1.3. Сводная таблица результатов выполнения сортировки	5
1.4. График.	5
2.ЗАДАНИЕ №2	5
2.1. Таблицу по возрастанию, по убыванию.....	5
2.2. Представить код программы и результат работы при $n=10$	6
3.ЗАДАНИЕ №3	7
3.1 Алгоритм сортировки при $n=10$	7
3.2 Процесс определения функции роста метода сортировки.	8
3.3 Сводные таблицы результатов. Для различных случаев.....	8
3.4 Графики	9
4.ВЫВОД	10
5. СПИСОК ИСТОЧНИКОВ	11

1.3АДАНИЕ №1

1.1. Алгоритм сортировки по методу варианта при n=10.

Номер строки инструкции алгоритма	Инструкция (оператор) алгоритма	Количество выполнений инструкции
1	For i ← 0; i < n – 1 :	n + 1
2	minimaseInd ← i	n
3	For j ← i + 2; j < n :	n (n - 1) /2
4	If array[j] < arrayComp[minimaseInd]	n (n - 1) /2 – 1
5	minimaseInd ← j	n (n - 1) /2 - 1
6	temporaryChange ← arrayComp[i]	n
7	arrayComp[i] ← arrayComp[minimaseInd]	n
8	arrayComp[minimaseInd] ← temporaryChange	n

$$T(n) = n + 1 + n + 3 * n * (n - 1) / 2 + 3 * n = 3 * n^2 + 2 * n + 1$$

Код программы

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <chrono>

using namespace std;

void sortedArraySelection(int arrayComp[], int size, int& compCount, int& swapCount) {
    compCount = 0;
    swapCount = 0;

    for (int i = 0; i < size - 1; ++i) {
        int minimaseInd = i;
        for (int j = i + 1; j < size; ++j) {
            compCount++;
            if (arrayComp[j] < arrayComp[minimaseInd]) {
                minimaseInd = j;
            }
        }
        int temporaryChange = arrayComp[i];
        arrayComp[i] = arrayComp[minimaseInd];
        arrayComp[minimaseInd] = temporaryChange;
        swapCount++;
    }
}

int main() {
    int num;
```

```

cout << "enter the number of elements in the array: ";
cin >> num;

int* data = new int[num];

srand(time(0));
for (int i = 0; i < num; ++i) {
    data[i] = rand() % 100;
}

cout << "original array: ";
for (int i = 0; i < num; ++i) {
    cout << data[i] << " ";
}
cout << endl;

int comparisons, swaps;

auto startTime = chrono::steady_clock::now();
sortedArraySelection(data, num, comparisons, swaps);
auto endTime = chrono::steady_clock::now();

cout << "sorted array: ";
for (int i = 0; i < num; ++i) {
    cout << data[i] << " ";
}
cout << endl;

auto timeDiff = endTime - startTime;
cout << "algorithm execution time: " << chrono::duration <double,
std::milli>(timeDiff).count() << " milliseconds" << endl;

cout << "number of comparisons: " << comparisons << endl;
cout << "number of swaps: " << swaps << endl;

delete[] data;
return 0;
}

```

Скрин работы программы при n = 10

```

enter the number of elements in the array: 10
original array: 73 17 82 47 43 55 11 98 13 67
sorted array: 11 13 17 43 47 55 67 73 82 98
algorithm execution time: 0.0012 milliseconds
number of comparisons: 45
number of swaps: 9

```

1.2. Процесс определения функции роста метода сортировки.

$$T_{\text{худший}}(n) = n + 1 + n + 3 \cdot n \cdot (n - 1) / 2 + 3 \cdot n = 3 \cdot n^2 + 2 \cdot n + 1$$

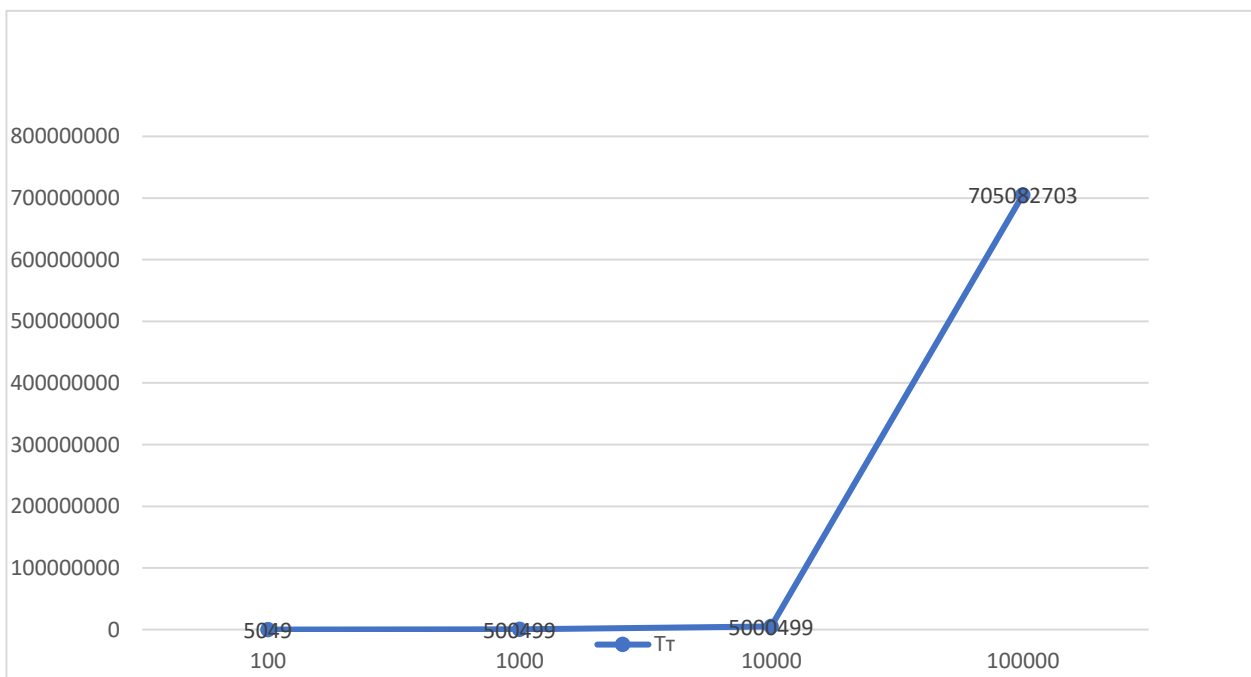
$$T_{\text{лучший}}(n) = n + 1 + n + 2 \cdot n \cdot (n - 1) / 2 + 3 \cdot n = 2 \cdot n^2 + 3 \cdot n + 1$$

$$T_{\text{средний}}(n) = n + 1 + n + 3 \cdot n \cdot (n - 1) / 2 + 3 \cdot n = 3 \cdot n^2 + 2 \cdot n + 1$$

1.3. Сводная таблица результатов выполнения сортировки

n	T(n), мс	T_T= C+ M	T_П= C_П+M_П
100	30_401	5049	5049
1000	3_004_001	500499	500499
10000	300_040_001	5000499	5000499
100000	30_000_400_001	705082703	705082703
1000000	3_000_004_000_001	_	_

1.4. График.



2.ЗАДАНИЕ №2

2.1. Таблицу по возрастанию, по убыванию.

n	T(n), мс	T_T= C+ M	T_П= C_П+M_П
100	30_301	5049	5049
1000	3_003_001	500499	500499
10000	300_030_001	5000499	5000499
100000	30_000_300_001	705082703	705082703
1000000	3_000_003_000_001	_	_

n	T(n), мс	T_T= C+ M	T_П= C_П+M_П
----------	-----------------	----------------------------	---

100	30_301	5049	5049
1000	3_003_001	500499	500499
10000	300_030_001	5000499	5000499
100000	30_000_300_001	705082703	705082703
1000000	3_000_003_000_001	_	_

2.2. Представить код программы и результат работы при n=10.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <chrono>

using namespace std;

void sortedArraySelection(int arrayComp[], int size, int& compCount, int& swapCount) {
    compCount = 0;
    swapCount = 0;

    for (int i = 0; i < size - 1; ++i) {
        int minimaseInd = i;
        for (int j = i + 1; j < size; ++j) {
            compCount++;
            if (arrayComp[j] > arrayComp[minimaseInd]) {
                minimaseInd = j;
            }
        }
        int temporaryChange = arrayComp[i];
        arrayComp[i] = arrayComp[minimaseInd];
        arrayComp[minimaseInd] = temporaryChange;
        swapCount++;
    }
}

int main() {
    int num;
    cout << "enter the number of elements in the array: ";
    cin >> num;

    int* data = new int[num];

    srand(time(0));
    for (int i = 0; i < num; ++i) {
        data[i] = rand() % 100;
    }

    cout << "original array: ";
    for (int i = 0; i < num; ++i) {
        cout << data[i] << " ";
    }
    cout << endl;

    int comparisons, swaps;

    auto startTime = chrono::steady_clock::now();
    sortedArraySelection(data, num, comparisons, swaps);
    auto endTime = chrono::steady_clock::now();

    cout << "sorted array: ";
    for (int i = 0; i < num; ++i) {
        cout << data[i] << " ";
    }
}
```

```

    cout << endl;

    auto timeDiff = endTime - startTime;
    cout << "algorithm execution time: " << chrono::duration <double,
std::milli>(timeDiff).count() << " milliseconds" << endl;

    cout << "number of comparisons: " << comparisons << endl;
    cout << "number of swaps: " << swaps << endl;

    delete[] data;
    return 0;
}

```

```

enter the number of elements in the array: 10
original array: 9 7 6 42 62 89 61 0 38 95
sorted array: 95 89 62 61 42 38 9 7 6 0
algorithm execution time: 0.0011 milliseconds
number of comparisons: 45
number of swaps: 9

```

3.ЗАДАНИЕ №3

3.1 Алгоритм сортировки при n=10

Номер строки инструкции алгоритма	Инструкция (оператор) алгоритма	Количество выполнений инструкции
1	For i ← 0; i < n – 1 :	n + 1
2	For j ← 0; i < n – 1 :	n*(n – 1) /2
3	If arrayComp[j] > arrayComp[j + 1]	n*(n – 1) /2 - 1
4	temporaryChange = arrayComp[j]	n*(n – 1) /2 - 1
5	arrayComp[j] = arrayComp[j+1]	n*(n – 1) /2 - 1
6	arrayComp[j+1] = temporaryChange	n*(n – 1) /2 - 1

$$T(n) = 1 + n + 5*n*(n - 1) /2 = (5*n**2 - 5*n) /2 + n + 1 = 3*n**2 - 2*n + 1$$

Код программы

```

#include <iostream>
#include <vector>
#include <ctime>
#include <chrono>
#include <cstdlib>

using namespace std;

```

```

void bubblesSortedArray(vector<int>& arrayComp) {
    int n = arrayComp.size();
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (arrayComp[j] > arrayComp[j + 1]) {
                int temporaryChange = arrayComp[j];
                arrayComp[j] = arrayComp[j + 1];
                arrayComp[j + 1] = temporaryChange;
            }
        }
    }
}

int main() {
    vector<int> arrayComp = {84, 45, 12, 72, 7};

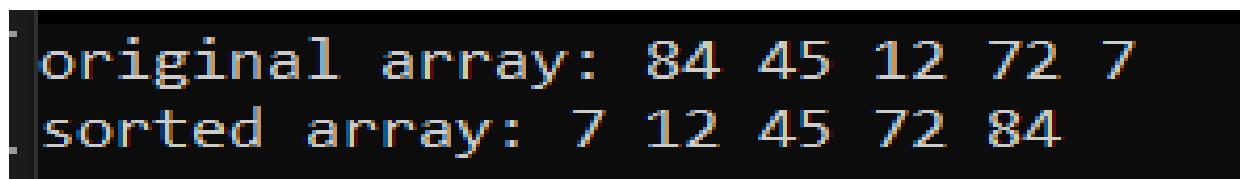
    cout << "original array: ";
    for (int i : arrayComp) {
        cout << i << " ";
    }
    cout << endl;

    bubblesSortedArray(arrayComp);

    cout << "sorted array: ";
    for (int i : arrayComp) {
        cout << i << " ";
    }
    cout << endl;

    return 0;
}

```



```

original array: 84 45 12 72 7
sorted array: 7 12 45 72 84

```

3.2 Процесс определения функции роста метода сортировки.

$$T_{\text{худший}}(n) = 1 + n + 5 \cdot n \cdot (n - 1) / 2 = (5 \cdot n^2 - 5 \cdot n) / 2 + n + 1 = 3 \cdot n^2 - 3 \cdot n + 1$$

$$T_{\text{лучший}}(n) = 1 + n + 2 \cdot n \cdot (n - 1) / 2 = n^2 + 1$$

$$T_{\text{средний}}(n) = (4 \cdot n^2 - 3 \cdot n + 2) / 2 = 2 \cdot n^2 - 2 \cdot n + 1$$

3.3 Сводные таблицы результатов. Для различных случаев.

n	T _{п(средний)}	T _{п(Лучший)}	T _{П(Худший)}
100	9_801	7280	7495
1000	998_001	728564	749862
10000	9_980_001	72476294	74616874
100000	999_800_001	7485043342	7485082703
1000000		-	-

3.4 Графики

Сравнение двух алгоритмов:

Худший случай:



Лучший случай:



4.ВЫВОД

Были актуализированы знания и приобретены практические умения по эмпирическому определению вычислительной сложности алгоритмов

5. СПИСОК ИСТОЧНИКОВ

1. Скворцова Л.А. Структуры и алгоритмы обработки данных. Часть 1: линейные структуры данных в алгоритмах [Электронный ресурс]: Практикум / Скворцова Л.А., Гусев К.В., Филатов А.С. — М.: МИРЭА – Российский технологический университет, 2023.