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 САМОСТОЯТЕЛЬНОЙ РАБОТЕ №4

по дисциплине
«Структуры и алгоритмы обработки данных»

Тема: Определение эффективного алгоритма сортировки на основе эмпирического и асимптотического методов анализа

Выполнил студент группы ИКБО-15-23 Гольд Д.В.

Принял старший преподаватель Скворцова Л.А.

Москва 2024

Оглавление

1. ЗАДАНИЕ 1

1.1 Код ускоренной сортировки по методу варианта

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

using namespace std;

void accumulate(int elementsArray[], int size, int i, int& seematching, int&
elementsmovement) {
    int morelargest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    seematching++;

    if (left < size && elementsArray[left] > elementsArray[morelargest])
        morelargest = left;

    if (right < size && elementsArray[right] > elementsArray[morelargest])
        morelargest = right;

    if (morelargest != i) {
        elementsmovement++;

        swap(elementsArray[i], elementsArray[morelargest]);

        accumulate(elementsArray, size, morelargest, seematching, elementsmovement);
    }
}

void accumulateSorted(int elementsArray[], int size) {
    int seematching = 0;
    int elementsmovement = 0;

    for (int i = size / 2 - 1; i >= 0; i--)
        accumulate(elementsArray, size, i, seematching, elementsmovement);

    for (int i = size - 1; i > 0; i--) {
        swap(elementsArray[0], elementsArray[i]);
        elementsmovement++;

        accumulate(elementsArray, i, 0, seematching, elementsmovement);
    }

    cout << "number of seematching: " << seematching << endl;
    cout << "number of swaps: " << elementsmovement << endl;
}

int main() {
    srand(time(NULL));

    int size;
    cout << "enter size array: ";
    cin >> size;

    int* elementsArray = new int[size];

    for (int i = 0; i <size; i++)
        elementsArray[i] = rand() % 100;
```

```
    accumulateSorted(elementsArray,size);

    cout << "sorted array: \n";
    for (int i = 0; i < size; i++)
        cout << elementsArray[i] << " ";
    cout << endl;

    delete[] elementsArray;

    return 0;
}
```

//Предусловие: size – кол-во чисел в массиве, elementsArray[] – массив чисел, mid – кол-во цифр в числе

//Постусловие: x/y
accumulateSorted(elementsArray[], size)

Номер строки инструкции алгоритма	Инструкция (оператор) алгоритма	Количество выполнений инструкции
1	For i ← size / 2 - 1; i >= 0 do	size/2
2	accumulate(elementsArray, size);	size * Log(size)
3	od	
4	For i ← size / 2 - 1; i >= 0 do	size /2
5	swap(elementsArray [0], elementsArray [i]);	size /2
6	accumulate(elementsArray, size);	size * Log(size)
7	od	

Ёмкость алгоритма: size

Таблица 1 – сводная таблица результатов

size	T(size), мс	$T_п$(size)=C+M
100	1 620	1408
1000	24 135	21 438
10000	41 147	261 058
100000	525 849	3 374 620

1.2. Код быстрой сортировки по методу варианта

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

using namespace std;

int generateRandomNumber(int minValue, int maxValue) {
    return rand() % (maxValue - minValue + 1) + minValue;
}

void merge(int array[], int left, int mid, int right, int& compareCount, int&
moveCount) {
    int leftSize = mid - left + 1;
    int rightSize = right - mid;

    int* leftArray = new int[leftSize];
    int* rightArray = new int[rightSize];

    for (int i = 0; i < leftSize; i++) {
        *(leftArray + i) = array[left + i];
        moveCount++;
    }
    for (int j = 0; j < rightSize; j++) {
        *(rightArray + j) = array[mid + 1 + j];
        moveCount++;
    }

    int i = 0, j = 0, k = left;
    while (i < leftSize && j < rightSize) {
        if (*(leftArray + i) <= *(rightArray + j)) {
            array[k] = *(leftArray + i);
            i++;
        }
        else {
            array[k] = *(rightArray + j);
            j++;
        }
        compareCount++;
        moveCount++;
        k++;
    }
```

```
    while (i < leftSize) {
        array[k] = *(leftArray + i);
        i++;
        k++;
        moveCount++;
    }

    while (j < rightSize) {
        array[k] = *(rightArray + j);
        j++;
        k++;
        moveCount++;
    }

    delete[] leftArray;
    delete[] rightArray;
}

void mergeSort(int array[], int left, int right, int& compareCount, int& moveCount) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        mergeSort(array, left, mid, compareCount, moveCount);
        mergeSort(array, mid + 1, right, compareCount, moveCount);

        merge(array, left, mid, right, compareCount, moveCount);
    }
}

int main() {
    int size;

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

    int* array = new int[size];
    srand(time(nullptr));

    cout << "initial array: \n";
    for (int i = 0; i < size; i++) {
        array[i] = generateRandomNumber(1, 100);
        cout << array[i] << " ";
    }
    cout << "\n";

    int compareCount = 0;
    int moveCount = 0;

    mergeSort(array, 0, size - 1, compareCount, moveCount);

    cout << "sorted array: \n";
    for (int i = 0; i < size; i++) {
        cout << array[i] << " ";
    }
    cout << "\n";

    cout << "number of comparisons: " << compareCount << endl;
    cout << "number of moves: " << moveCount << endl;

    delete[] array;

    return 0;
}
```

Номер	Инструкция	Кол-во выполнений инструкции
1	leftSize ← mid	size
2	rightSize ← right – mid	size
3	mid = left + (right - left) / 2;	size
4	for i ← 0 < leftSize	size + 1
5	leftSize[i] = array[leftArray + i]	size + 1
6	for j ← 0 < rightSize	size*(size - 1)/2
7	rightSize[j] = array[mid + leftArray + j]	size*(size - 1)/2
8	od od	
9	if (left < right)	size+1
10	Sort(array, left, mid)	size+1
11	Sort(array, mid + leftArray, right);	size+1
12	merge(array, leftArray, mid, right);	size+1
13	endif	

Ёмкость алгоритма: 2*size

Таблица 2 – сводная таблица результатов

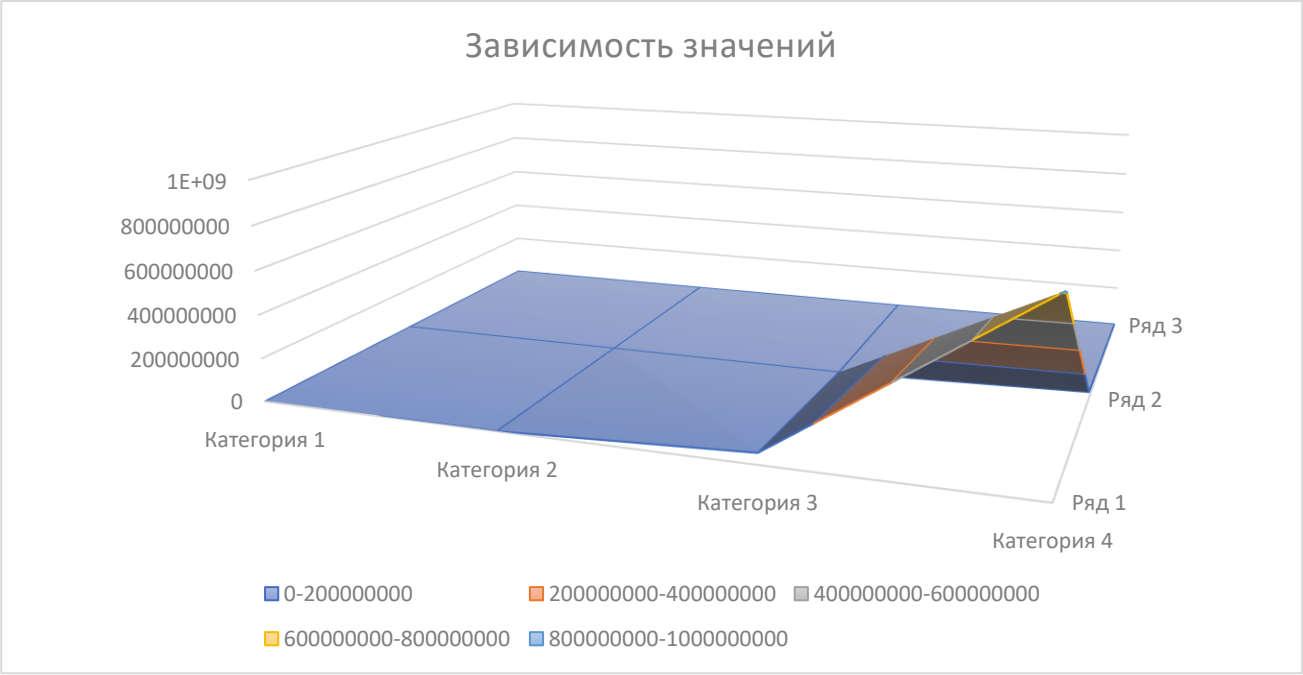
size	**T(size), мс**	**$T_п$(size)=C+M**
100	747	2475
1000	9478	31487
10000	142742	410692
100000	1864674	5248932

1.3. Код простой сортировки по методу варианта

Таблица 3 – сводная таблица результатов

size	T(size)	$T_{п(size)}=C+M$
100	7411	9495
1000	784971	949864
10000	78405737	94616876
100000	7864515951	9485082707

1.4. График зависимости от таблиц 1-3



1.5. Сводная таблица результатов при применении быстрой и ускоренной сортировки к массиву

Исправить все что снизу и наверху

Таблица 4 – сводная таблица результатов по убыванию

size	**T(size)1**	**T(size)2**
100	4842	2071
1000	425785	35639
10000	42826085	441830
100000	4794500582	5870925

Таблица 5 – сводная таблица результатов по возрастанию

size	**T(size)1**	**T(size)2**
100	4842	2071
1000	425785	35639
10000	42826085	441830
100000	4794500582	5870925

2. ЗАДАНИЕ 2

2.1. Асимптотический анализ сложности алгоритмов
Из материалов предыдущей практической работы:

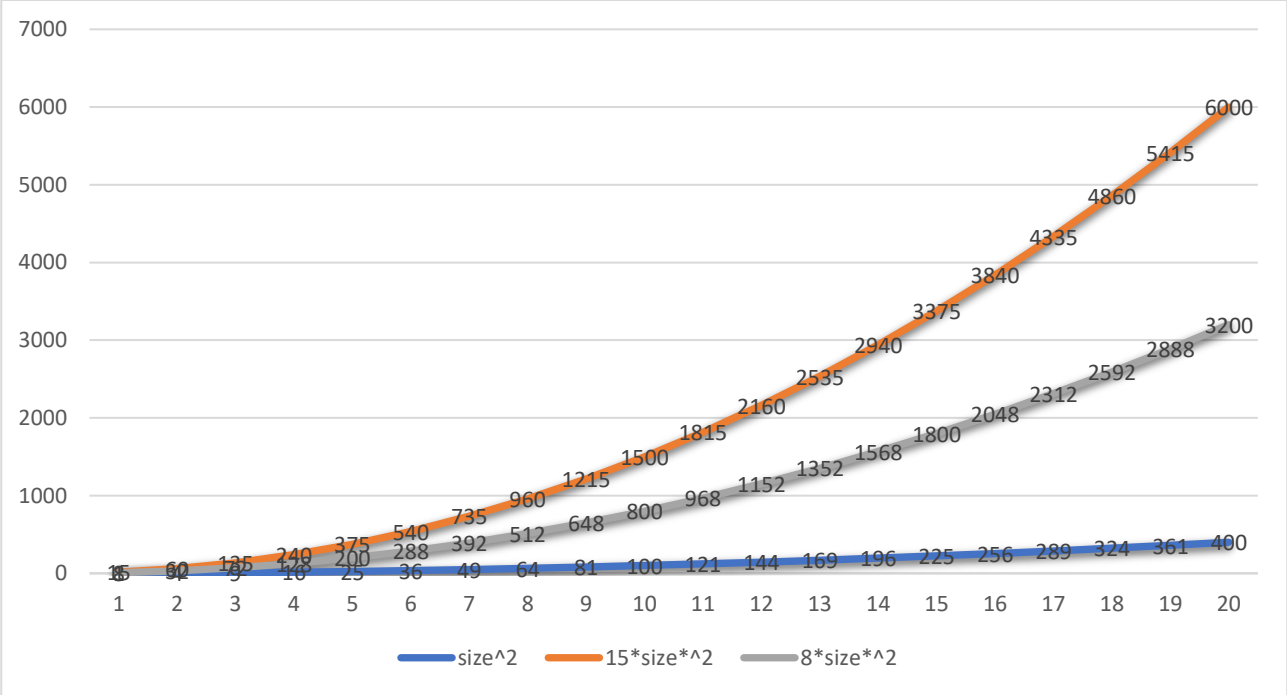
$T_{луч}$(size) = 1 + size + 2*size*(size - 1)/2 = size**2 + 1

На основе определений соответствующих нотаций получаем асимптотическую оценку вычислительной сложности простого алгоритма сортировки:
- в O - нотации (оценка сверху) для анализа худшего случая;
- в Ω - нотации (оценка снизу) для анализа лучшего случая

O для простой сортировки:
size**2 + 8*size + 6
size**2 ≤ 1*size**2; size ≥ 0
8*size ≤ 8*size**2 ; size ≥ 0
6 ≤ 6*size**2 ; size ≥ 1
N0 = 1
C = 15
Ω для простой сортировки:
size**2 + 8*size + 6
size**2 ≥ 1*size**2; size ≥ 0
8*size ≥ 0*size**2; size ≥ 0
6 ≥ 0*size**2 ; size ≥ 1
N0 = 1
C = 1
θ для простой сортировки:
1*size**2 ≤ 8*size**2 ≤ 15*size**2 ; size ≥ 1

2.2. График роста

Графическое представление функции роста и полученных асимптотических оценок сверху и снизу.



3.ВЫВОД

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

4.ЛИТЕРАТУРА

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. — 1 электрон. опт. диск (CD-ROM)
2. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. – СПб: Питер, 2017. – 288 с.
3. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 406 с.
4. Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск, 2-е изд. – М.: ООО «И.Д. Вильямс», 2018. – 832 с.
5. Седжвик Р. Фундаментальные алгоритмы на C++. Анализ/Структуры данных/Сортировка/Поиск. – К.: Издательство «Диасофт», 2001. – 688 с.
6. AlgoList – алгоритмы, методы, исходники [Электронный ресурс]. URL: http://algolist.manual.ru/ (дата обращения 15.03.2022).
7. Алгоритмы – всё об алгоритмах / Хабр [Электронный ресурс]. URL: https://habr.com/ru/hub/algorithms/ (дата обращения 15.03.2022).
8. НОУ ИНТУИТ | Технопарк Mail.ru Group: Алгоритмы и структуры данных [Электронный ресурс]. URL: https://intuit.ru/studies/courses/3496/738/info (дата обращения 15.03.2022).