Министерство науки и высшего образования РФ
ФГАОУ ВПО
Национальный исследовательский технологический университет «МИСиС»

Институт Информационных технологий и компьютерных наук (ИТКН)

Кафедра Инфокоммуникационных технологий (ИКТ)

Отчет по лабораторной работе №4
по дисциплине «Технологии программирования»
на тему «Оценка эффективности алгоритмов»

Выполнил:
студент группы БИСТ-22-1

Меленцов Д.А.

Проверил:
Карпищук А.В.

Москва, 2023

Цель работы: ознакомление с основами анализа эффективности алгоритмов.

Индивидуальное задание:

Задача A: Яблочный сад содержит n рядов, в каждом ряду растет n яблонь, на каждой яблоне зреет n яблок разного диаметра. Яблоки диаметром до 7 см помещаются в малые ящики, диаметром от 7 до 10 см – в средние, более 10 см – в большие. Вместимость всех ящиков одинаковая – 100 яблок. Определить необходимое количество ящиков для сбора урожая со всего сада.

Задача B: Коллекционер старинных монет хранит свою коллекцию в хранилище, состоящем из n боксов, в каждом боксе размещено 500 монет различного достоинства. Найти в хранилище все монеты, номинал которых можно представить суммой номиналов двух других монет из того же бокса.

Алгоритм A я реализовал с помощью класса Random, тройного вложенного цикла и условной конструкции из трёх операторов if. Алгоритм B я реализовал также с помощью класса Random, ещё я использовал список из списков целочисленных чисел, для поиска монет подходящих под условие исползовался тройной вложенный цикл.

Работа алгоритма A представлена на рисунке 1. Инициализируется объект класса Random и три переменные для хранения количества яблок разных размеров. Далее тройной цикл, зависящий от n. С помощью Random.Next генерируется переменная appleDiameter в нижнем цикле, а затем конструкцией операторов if определяется принадлежность диаметра к одному из трёх размеров, переменная подходящего размера увеличивается на 1. После работы цикла в переменную boxes присваивается значение суммы целочисленных делений трёх переменных для разных размеров. И в конце проверка каждой

переменной, если её значение меньше 100 и не равно 0, то добавляется 1 к boxes. В конце возвращается значение переменной boxes.

Работа алгоритма B представлена на рисунке 2. Этот алгоритм имеет метод CreateFullBox, которая инициализирует список из 500 элементов, представляющих случайные номиналы монет. Алгоритм создает хранилище, состоящее из n таких списков. Далее, алгоритм ищет монеты с номиналом, равным удвоенному номиналу каждой из 4 доступных монет. Каждый раз, когда такая монета найдена, значение переменной coins увеличивается на 1. В конце алгоритма возвращается общее количество найденных монет.

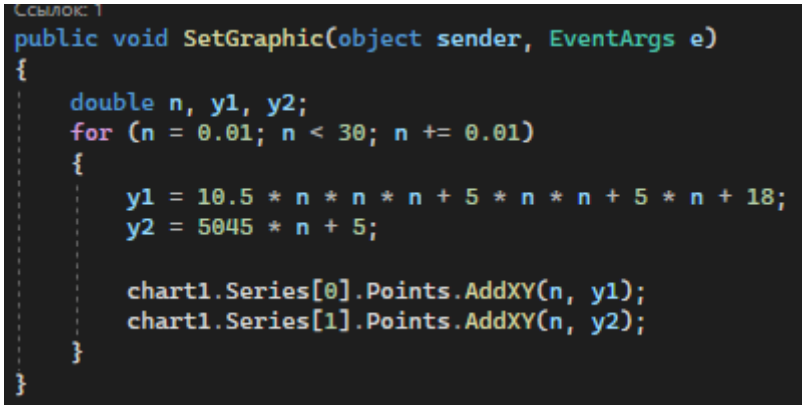
Временная сложность алгоритмов. Для алгоритма A функция временной сложности 10.5 * n^3 + 5 * n^2 + 5 * n + 18. Для алгоритма B 5045 * n + 5. Реализация графика зависимости времени выполнения функции от входных данных в windowsforms представлена на рисунке 3, а сам график зависимости этих функций от значения n представлен на рисунке 4.

```
Ссылок: 0
public int AppleBoxes(int n)
{
    Random random = new Random();
    int smallApples = 0, middleApples = 0, bigApples = 0;


    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            for (int k = 0; k < n; k++)
            {
                int appleDiameter = random.Next(5, 12);
                if (appleDiameter < 7)
                {
                    smallApples++;
                }
                else if (appleDiameter >= 7 && appleDiameter < 10)
                {
                    middleApples++;
                }
                else if (appleDiameter >= 10)
                {
                    bigApples++;
                }
            }

    int boxes = smallApples / 100 + middleApples / 100 + bigApples / 100;

    if (smallApples < 100 && smallApples != 0) boxes++;
    if (middleApples < 100 && middleApples != 0) boxes++;
    if (bigApples < 100 && bigApples != 0) boxes++;

    return boxes;
}
```

Рисунок 1 - реализация алгоритма А

```
Ссылок: 0
internal class Storage
{
    Ссылок: 1
    private List<int> CreateFullBox(ref int[] denominations)
    {
        Random random = new Random();
        List<int> box = new List<int>(500);
        for (int j = 0; j < 500; j++)
        {
            box[j] = random.Next(denominations.Length);
        }
        return box;
    }

    Ссылок: 0
    public int Coins(int n)
    {
        int[] denominations = {1, 2, 5, 10};
        List<List<int>> storage = new List<List<int>>(n);

        //Формирование хранилища
        for (int i = 0; i < n; i++)
        {
            storage[i] = CreateFullBox(ref denominations);
        }

        //Поиск монет с номиналом
        int coins = 0;
        for (int i = 0; i < n; i++)
        {
            List<int> box = new List<int>(500);
            box = storage[i];
            for (int j = 0; j < 500; j++)
            {
                for (int k = 0; k < 4; k++)
                {
                    if (box[j] == 2 * denominations[k])
                    {
                        coins++;
                        break;
                    }
                }
            }
        }

        return coins;
    }
}
```

Рисунок 2 - реализация алгоритма B



```
Ссылок: 1
public void SetGraphic(object sender, EventArgs e)
{
    double n, y1, y2;
    for (n = 0.01; n < 30; n += 0.01)
    {
        y1 = 10.5 * n * n * n + 5 * n * n + 5 * n + 18;
        y2 = 5045 * n + 5;

        chart1.Series[0].Points.AddXY(n, y1);
        chart1.Series[1].Points.AddXY(n, y2);
    }
}
```

Рисунок 3 - реализация графика зависимости времени выполнения функции от входных данных

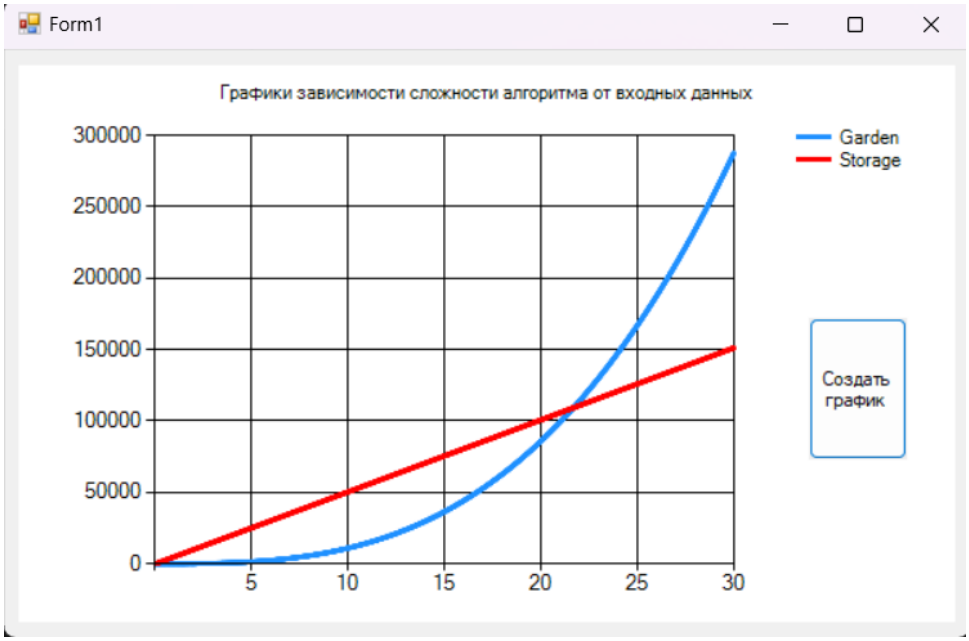


Рисунок 4 - график график зависимости функций от значения n

В ходе выполнения данной работы было проведено ознакомление с основами анализа эффективности алгоритмов. В результате выполнения работы были решены задачи A и B, а также проведен анализ эффективности алгоритмов, примененных при их решении. Были выявлены оптимальные подходы к решению данных задач и оценены временные затраты при использовании этих подходов. А также построен график зависимости обоих алгоритмов от входных данных.