

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительная техника»

ОТЧЕТ

По лабораторной работе №2
«Оценка времени выполнения программ.»
По дисциплине «Л и ОА в ИЗ»

Выполнили: ст. гр. 22ВВ4
Жуков Илья
Чумаев Сабит

Приняли: Юрова О.В.
Акифьев И.В.

Цель работы:

Написать код программы, выполнив следующие задания:

Задание 1:

- 1.Вычислить порядок сложности программы (O-символику).
- 2.Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки `time.h` для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.
- 3.Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

Задание 2:

- 1.Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.
- 2.Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.
- 3.Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.
- 4.Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.
- 5.Оценить время работы стандартной функции `qsort`, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

Ход работы:

Описание кода программы по заданию 1:

В данной программе производится умножение двух случайно сгенерированных матриц различных размеров. Затем измеряется время выполнения этой операции для каждого размера матрицы из заданного списка.

```
static void Main()
{

//объявили массив matrixSizes, который содержит размеры матриц, для
которых будет выполняться умножение.
    int[] matrixSizes = { 100, 200, 400, 1000, 2000, 4000,
10000 };
```

```

//Перебираем данный массив
    foreach (int size in matrixSizes)
    {

//вызов метода GenerateRandomMatrix для генерации случайных матриц
(matrix1 и matrix2) заданного размера size x size.
        int[,] matrix1 = GenerateRandomMatrix(size, size);
        int[,] matrix2 = GenerateRandomMatrix(size, size);

//создание объекта класса Stopwatch для замера времени выполнения.
        Stopwatch stopwatch = new Stopwatch();

//запуск секундомера.
        stopwatch.Start();

//вызов метода MultiplyMatrices для умножения матриц matrix1 и matrix2
и сохранения результата в переменную resultMatrix.
        int[,] resultMatrix = MultiplyMatrices(matrix1, matrix2);

//остановка секундомера.
        stopwatch.Stop();

//сохранение прошедшего времени в переменную elapsedTime типа TimeSpan.
        TimeSpan elapsedTime = stopwatch.Elapsed;

//вывод сообщения о размере матрицы на консоль.
        Console.WriteLine($"Matrix size: {size}x{size}");

//вывод сообщения о времени выполнения на консоль.
        Console.WriteLine($"Elapsed                               Time:
{elapsedTime.TotalSeconds} seconds");
    }
}

//объявление метода GenerateRandomMatrix, который генерирует случайную
матрицу заданного размера rows x cols и возвращает ее.
    static int[,] GenerateRandomMatrix(int rows, int cols)
    {
        Random random = new Random();

//объявление двумерного массива matrix заданного размера rows x cols.
        int[,] matrix = new int[rows, cols];

//начало цикла for, который проходит по строкам матрицы.
        for (int i = 0; i < rows; i++)
        {

//начало цикла for, который проходит по столбцам матрицы.
            for (int j = 0; j < cols; j++)
            {

```

```

//генерация случайного числа от 0 до 99 и присваивание его элементу
матрицы с индексами i и j.
        matrix[i, j] = random.Next(100);
    }
}

return matrix;
}

//объявление метода MultiplyMatrices, который умножает две матрицы
matrix1 и matrix2 и возвращает полученную матрицу.
static int[,] MultiplyMatrices(int[,] matrix1, int[,] matrix2)
{
//получение количества строк в матрице
    int rows1 = matrix1.GetLength(0);

//получение количества столбцов в матрице
    int cols1 = matrix1.GetLength(1);

//получение количества столбцов в матрице
    int cols2 = matrix2.GetLength(1);

//объявление двумерного массива resultMatrix размером rows1 x cols2 для
сохранения результата умножения.
    int[,] resultMatrix = new int[rows1, cols2];

//начало цикла for, который проходит по строкам матрицы resultMatrix.
    for (int i = 0; i < rows1; i++)
    {

//начало цикла for, который проходит по столбцам матрицы resultMatrix.
        for (int j = 0; j < cols2; j++)
        {

//начало цикла for, который проходит по столбцам матрицы matrix1 или
строкам матрицы matrix2.
            for (int k = 0; k < cols1; k++)
            {

//Вычисление матрицы путем суммирования произведений соответствующих
элементов матриц matrix1 и matrix2.
                resultMatrix[i, j] += matrix1[i, k] * matrix2[k,
j];
            }
        }
    }

    Console.Write("-----\n");
}

```

```

        return resultMatrix;
    }

```

Порядок сложности перемножения матриц размера `NxN` составляет ` $O(N^3)$ `. Однако, в реальности время выполнения может отличаться из-за оптимизаций и особенностей компьютера.

sizeM	100	200	400	1000	2000	4000	10000
t	0,02	0,11	0,97	14,79	108,3	1090,55	30000



Описание кода программы по заданию 2:

В данной программе производится оценка времени работы каждого из 3 реализованных алгоритмов на различных массивах

```

//Сначала пользователю предлагается ввести размер массива
Console.Write("Введите размер массивов: ");
    int N = Convert.ToInt16(Console.ReadLine());
    Console.WriteLine("\nЗадание 1.(случайный набор значений
массива):\n");

    int[] mass = new int[N];

//Переменная "middleIndex" устанавливается как середина массива "mass"
    int middleIndex = N / 2;
    int value = 1;

    //int[] massTwo = new int[N];

```

```

//Создается экземпляр класса "Random" для генерации случайных чисел
Random r = new Random();

//Далее происходит заполнение массива
for (int i = 0; i < mass.Length; i++)
{
    mass[i] = r.Next(100);
}
//Создается экземпляр класса "Stopwatch" для измерения времени
выполнения сортировки
Stopwatch stopwatch = new Stopwatch();

Console.WriteLine("Изначальный массив: ");

for (int i = 0; i < mass.Length; i++)
{
    Console.Write($"{mass[i]} ");
}

//Создается копия массива "mass" под названием "shellSortedArray"
int[] shellSortedArray = new int[mass.Length];
Array.Copy(mass, shellSortedArray, mass.Length);

//Запускается секундомер
stopwatch.Start();

//Происходит сортировка массива "shellSortedArray" с помощью алгоритма
сортировки Шелла
shell(shellSortedArray);

//Останавливается секундомер
stopwatch.Stop();

//Выводится отсортированный массив "shellSortedArray" на консоль
Console.WriteLine("\n\nМассив после сортировки Шелла:");
for (int i = 0; i < shellSortedArray.Length; i++)
{
    Console.Write($"{shellSortedArray[i]} ");
}

Console.WriteLine($"{\n\nВремя вработы сортировки Шелла:
{stopwatch.Elapsed.TotalMilliseconds}");

//Вызывается функция "line", которая выводит разделительную линию на
консоль
line();

Console.WriteLine("Изначальный массив: ");

```

```

        for (int i = 0; i < mass.Length; i++)
        {
            Console.Write($"{mass[i]} ");
        }

//Создается копия массива "mass" под названием "qsSortedArray"
int[] qsSortedArray = new int[mass.Length];
Array.Copy(mass, qsSortedArray, mass.Length);

stopwatch.Start();

//Происходит сортировка массива "qsSortedArray" с помощью быстрой
сортировки
qs(qsSortedArray, 0, mass.Length - 1);
stopwatch.Stop();

Console.WriteLine("\n\nМассив после быстрой сортировки:");
for (int i = 0; i < qsSortedArray.Length; i++)
{
    Console.Write($"{qsSortedArray[i]} ");
}

Console.WriteLine($"{stopwatch.Elapsed.TotalMilliseconds}");

line();

Console.WriteLine("Изначальный массив:");

for (int i = 0; i < mass.Length; i++)
{
    Console.Write($"{mass[i]} ");
}

//Создается копия массива "mass" под названием "standartSortedArray"
int[] standartSortedArray = new int[mass.Length];
Array.Copy(mass, standartSortedArray, mass.Length);

stopwatch.Start();

//Происходит стандартная сортировка массива "standartSortedArray".
Array.Sort(standartSortedArray);
stopwatch.Stop();

Console.WriteLine("\n\nМассив после стандартной сортировки:");
for (int i = 0; i < standartSortedArray.Length; i++)
{
    Console.Write($"{standartSortedArray[i]} ");
}

```

```

        Console.WriteLine($"\\n\\nВремя работы стандартной сортировки:
{stopwatch.Elapsed.TotalMilliseconds}");

        line();
        stop();

        Console.WriteLine("Задание 2.(убывающая последовательность
чисел в массиве):\\n");
        Console.WriteLine("Изначальный массив:");

//Отсортировал массив пузырьковым алгоритмом, чтобы получить убывающую
последовательность
        BubbleSortDecreasing(mass);

        for (int i = 0; i < mass.Length; i++)
        {
            Console.Write($"{mass[i]} ");
        }

        int[] shellSortedArray2 = new int[mass.Length];
        Array.Copy(mass, shellSortedArray2, mass.Length);

        Stopwatch stopwatch2 = new Stopwatch();

        stopwatch2.Start();

        shell(shellSortedArray2);

        stopwatch2.Stop();

        Console.WriteLine("\\n\\nМассив после сортировки Шелла: ");

        for (int i = 0; i < shellSortedArray2.Length; i++)
        {
            Console.Write($"{shellSortedArray2[i]} ");
        }
        Console.WriteLine($"\\n\\nВремя вработы сортировки Шелла:
{stopwatch2.Elapsed.TotalMilliseconds}");

        line();

        Console.WriteLine("Изначальный массив:");

        for (int i = 0; i < mass.Length; i++)
        {
            Console.Write($"{mass[i]} ");
        }

        int[] qsSortedArray2 = new int[mass.Length];

```



```

        Array.Copy(mass, qsSortedArray2, mass.Length);

        stopwatch2.Start();

        qs(qsSortedArray2, 0, mass.Length - 1);
        stopwatch2.Stop();

        Console.WriteLine("\n\nМассив после быстрой сортировки:");
        for (int i = 0; i < qsSortedArray2.Length; i++)
        {
            Console.Write($"{qsSortedArray2[i]} ");
        }

        Console.WriteLine($"
\n\nВремя работы быстрой сортировки:
{stopwatch2.Elapsed.TotalMilliseconds}");

        line();

        Console.WriteLine("Изначальный массив:");

        for (int i = 0; i < mass.Length; i++)
        {
            Console.Write($"{mass[i]} ");
        }

        int[] standartSortedArray2 = new int[mass.Length];
        Array.Copy(mass, standartSortedArray2, mass.Length);

        stopwatch2.Start();
        Array.Sort(standartSortedArray2);
        stopwatch2.Stop();

        Console.WriteLine("\n\nМассив после стандартной сортировки:");
        for (int i = 0; i < standartSortedArray2.Length; i++)
        {
            Console.Write($"{standartSortedArray2[i]} ");
        }

        Console.WriteLine($"
\n\nВремя работы стандартной сортировки:
{stopwatch2.Elapsed.TotalMilliseconds}");

        line();
        stop();

        Console.WriteLine("Задание 3.(возрастающая последовательность
чисел в массиве):\n");
        Console.WriteLine("Изначальный массив:");
        //Отсортировал массив пузырьковым алгоритмом, чтобы получить
возрастающую последовательность

```

```

BubbleSortIncreasing(mass);

for (int i = 0; i < mass.Length; i++)
{
    Console.Write($"{mass[i]} ");
}

int[] shellSortedArray3 = new int[mass.Length];
Array.Copy(mass, shellSortedArray3, mass.Length);

Stopwatch stopwatch3 = new Stopwatch();

stopwatch3.Start();

shell(shellSortedArray3);

stopwatch3.Stop();

Console.WriteLine("\n\nМассив после сортировки Шелла: ");

for (int i = 0; i < shellSortedArray3.Length; i++)
{
    Console.Write($"{shellSortedArray3[i]} ");
}
Console.WriteLine($"
\n\nВремя вработы сортировки Шелла:
{stopwatch3.Elapsed.TotalMilliseconds}");

line();

Console.WriteLine("Изначальный массив:");

for (int i = 0; i < mass.Length; i++)
{
    Console.Write($"{mass[i]} ");
}

int[] qsSortedArray3 = new int[mass.Length];
Array.Copy(mass, qsSortedArray3, mass.Length);

stopwatch3.Start();

qs(qsSortedArray3, 0, mass.Length - 1);
stopwatch3.Stop();

Console.WriteLine("\n\nМассив после быстрой сортировки:");
for (int i = 0; i < qsSortedArray3.Length; i++)
{
    Console.Write($"{qsSortedArray3[i]} ");
}

```

```

        Console.WriteLine($"\\n\\nВремя работы быстрой сортировки:
{stopwatch3.Elapsed.TotalMilliseconds}");

        line();

        Console.WriteLine("Изначальный массив:");

        for (int i = 0; i < mass.Length; i++)
        {
            Console.Write($"{mass[i]} ");
        }

        int[] standartSortedArray3 = new int[mass.Length];
        Array.Copy(mass, standartSortedArray3, mass.Length);

        stopwatch3.Start();
        Array.Sort(standartSortedArray3);
        stopwatch3.Stop();

        Console.WriteLine("\\n\\nМассив после стандартной сортировки:");
        for (int i = 0; i < standartSortedArray3.Length; i++)
        {
            Console.Write($"{standartSortedArray3[i]} ");
        }

        Console.WriteLine($"\\n\\nВремя работы стандартной сортировки:
{stopwatch3.Elapsed.TotalMilliseconds}");

        line();
        stop();

        Console.WriteLine("Задание 4. (первая половина эл. - возрастает,
вторая половина эл. - убывает):\\n");

        for (int i = 0; i < middleIndex; i++)
        {
            mass[i] = value;
            value++;
        }

        value -= 2;
        for (int i = middleIndex; i < N; i++)
        {
            mass[i] = value;
            value--;
        }

        Console.WriteLine("Изначальный массив:");

        for (int i = 0; i < mass.Length; i++)

```

```

    {
        Console.Write($"{mass[i]} ");
    }

    int[] shellSortedArray4 = new int[mass.Length];
    Array.Copy(mass, shellSortedArray4, mass.Length);

    Stopwatch stopwatch4 = new Stopwatch();

    stopwatch4.Start();

    shell(shellSortedArray4);

    stopwatch4.Stop();

    Console.WriteLine("\n\nМассив после сортировки Шелла: ");

    for (int i = 0; i < shellSortedArray4.Length; i++)
    {
        Console.Write($"{shellSortedArray4[i]} ");
    }
    Console.WriteLine("\n\nВремя вработы сортировки Шелла:
{stopwatch4.Elapsed.TotalMilliseconds}");

    line();

    Console.WriteLine("Изначальный массив:");

    for (int i = 0; i < mass.Length; i++)
    {
        Console.Write($"{mass[i]} ");
    }

    int[] qsSortedArray4 = new int[mass.Length];
    Array.Copy(mass, qsSortedArray4, mass.Length);

    stopwatch4.Start();

    qs(qsSortedArray4, 0, mass.Length - 1);
    stopwatch4.Stop();

    Console.WriteLine("\n\nМассив после быстрой сортировки:");
    for (int i = 0; i < qsSortedArray4.Length; i++)
    {
        Console.Write($"{qsSortedArray4[i]} ");
    }

    Console.WriteLine("\n\nВремя работы быстрой сортировки:
{stopwatch4.Elapsed.TotalMilliseconds}");

```

```

        line();

        Console.WriteLine("Изначальный массив:");

        for (int i = 0; i < mass.Length; i++)
        {
            Console.Write($"{mass[i]} ");
        }

        int[] standartSortedArray4 = new int[mass.Length];
        Array.Copy(mass, standartSortedArray4, mass.Length);

        stopwatch4.Start();
        Array.Sort(standartSortedArray4);
        stopwatch4.Stop();

        Console.WriteLine("\n\nМассив после стандартной сортировки:");
        for (int i = 0; i < standartSortedArray4.Length; i++)
        {
            Console.Write($"{standartSortedArray4[i]} ");
        }

        Console.WriteLine($"{"\n\nВремя работы стандартной сортировки:"}
        {stopwatch4.Elapsed.TotalMilliseconds}");

        line();
        stop();
    }

//Алгоритм Шелла
public static void shell(int[] Array)
{
    int j;
    int step = Array.Length / 2;
    while (step > 0)
    {
        for (int i = 0; i < (Array.Length - step); i++)
        {
            j = i;
            while ((j >= 0) && (Array[j] > Array[j + step]))
            {
                int tmp = Array[j];
                Array[j] = Array[j + step];
                Array[j + step] = tmp;
                j -= step;
            }
        }
        step = step / 2;
    }
}

```

//Алгоритм быстрой сортировки

```
public static void qs(int[] items, int left, int right)
{
    int i, j;
    int x, y;

    i = left;
    j = right;

    // Выбор компаранда
    x = items[(left + right) / 2];

    do
    {
        while (items[i] < x && i < right)
        {
            i++;
        }
        while (x < items[j] && j > left)
        {
            j--;
        }

        if (i <= j)
        {
            y = items[i];
            items[i] = items[j];
            items[j] = y;
            i++;
            j--;
        }
    } while (i <= j);

    if (left < j)
    {
        qs(items, left, j);
    }
    if (i < right)
    {
        qs(items, i, right);
    }
}

static void stop()
{
    Console.ReadLine();
}

static void line()
{
    Console.WriteLine("-----
-");
```

```

    }
    //Алгоритм пузырька, выводящий убывающую последовательность
    static int[] BubbleSortDecreasing(int[] mass)
    {
        int temp;
        for (int i = 0; i < mass.Length; i++)
        {
            for (int j = 0; j < mass.Length; j++)
            {
                if (mass[i] > mass[j])
                {
                    temp = mass[i];
                    mass[i] = mass[j];
                    mass[j] = temp;
                }
            }
        }
        return mass;
    }
    //Алгоритм пузырька, выводящий возрастающую последовательность

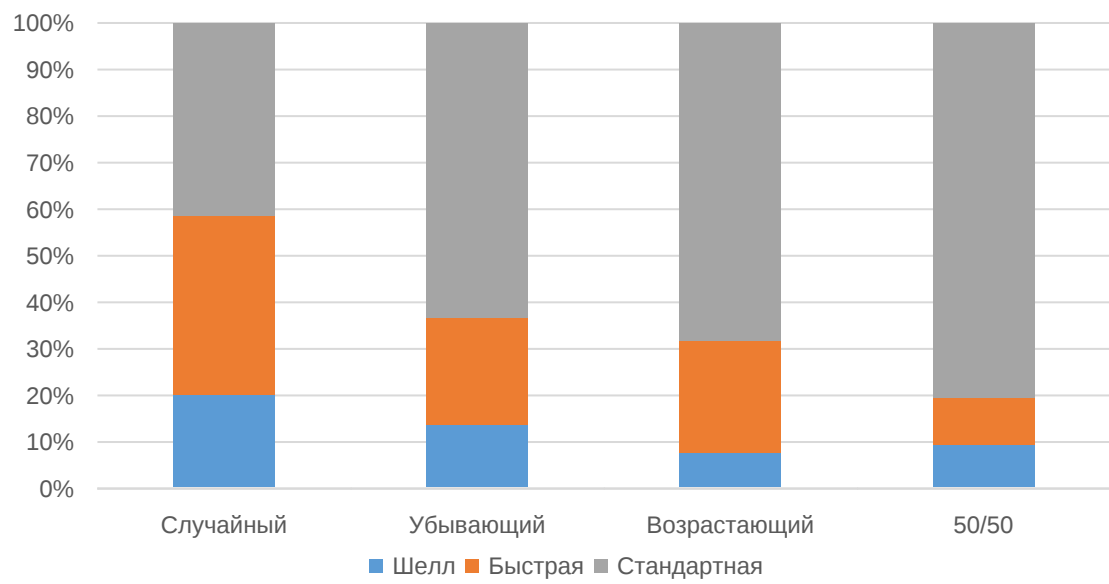
    static int[] BubbleSortIncreasing(int[] mass)
    {
        int temp;
        for (int i = 0; i < mass.Length; i++)
        {
            for (int j = 0; j < mass.Length; j++)
            {
                if (mass[i] < mass[j])
                {
                    temp = mass[j];
                    mass[j] = mass[i];
                    mass[i] = temp;
                }
            }
        }
        return mass;
    }
}

```

Таблица оценки времени работы алгоритмов:

	Случайный	Убывающий	Возрастающий	50/50
Шелл	0,2762	0,0013	0,0007	0,0016
Быстрая	0,5244	0,0022	0,0022	0,0017
Стандартная	0,5667	0,0060	0,0062	0,0135

Время работы каждого из алгоритмов



Результат работы программы 1:

```
C:\windows\system32\cmd.exe

-----
Matrix size: 100x100
Elapsed Time: 0,0169607 seconds
-----
Matrix size: 200x200
Elapsed Time: 0,1141118 seconds
-----
Matrix size: 400x400
Elapsed Time: 0,9690363 seconds
-----
Matrix size: 1000x1000
Elapsed Time: 14,7881942 seconds
-----
Matrix size: 2000x2000
Elapsed Time: 108,2983861 seconds
-----
Matrix size: 4000x4000
Elapsed Time: 1090,5546619 seconds
-----
```

Результат работы программы 2:

```
C:\windows\system32\cmd.exe

Введите размер массивов: 10

Задание 1.(случайный набор значений массива):

Изначальный массив:
95 9 52 90 30 10 56 80 87 61
Массив после сортировки Шелла:
9 10 30 52 56 61 80 87 90 95
Время работы сортировки Шелла: 0,2762
-----
Изначальный массив:
95 9 52 90 30 10 56 80 87 61
Массив после быстрой сортировки:
9 10 30 52 56 61 80 87 90 95
Время работы быстрой сортировки: 0,5244
-----
Изначальный массив:
95 9 52 90 30 10 56 80 87 61
Массив после стандартной сортировки:
9 10 30 52 56 61 80 87 90 95
Время работы быстрой сортировки: 0,5667
-----

Задание 2.(убывающая последовательность чисел в массиве):

Изначальный массив:
95 90 87 80 61 56 52 30 10 9
Массив после сортировки Шелла:
9 10 30 52 56 61 80 87 90 95
Время работы сортировки Шелла: 0,0013
-----
Изначальный массив:
95 90 87 80 61 56 52 30 10 9
Массив после быстрой сортировки:
9 10 30 52 56 61 80 87 90 95
Время работы быстрой сортировки: 0,0022
-----
Изначальный массив:
95 90 87 80 61 56 52 30 10 9
```

```
C:\windows\system32\cmd.exe
Массив после стандартной сортировки:
0 10 30 52 56 61 80 87 90 95
Время работы стандартной сортировки: 0,006
-----

Задание 3.(возрастающая последовательность чисел в массиве):
Изначальный массив:
0 10 30 52 56 61 80 87 90 95
Массив после сортировки Шелла:
0 10 30 52 56 61 80 87 90 95
Время работы сортировки Шелла: 0,0007
-----
Изначальный массив:
0 10 30 52 56 61 80 87 90 95
Массив после быстрой сортировки:
0 10 30 52 56 61 80 87 90 95
Время работы быстрой сортировки: 0,0022
-----
Изначальный массив:
0 10 30 52 56 61 80 87 90 95
Массив после стандартной сортировки:
0 10 30 52 56 61 80 87 90 95
Время работы стандартной сортировки: 0,0062
-----

Задание 4.(первая половина эл. - возрастает, вторая половина эл. - убывает):
Изначальный массив:
1 2 3 4 5 4 3 2 1 0
Массив после сортировки Шелла:
0 1 1 2 2 3 3 4 4 5
Время работы сортировки Шелла: 0,0016
-----
Изначальный массив:
1 2 3 4 5 4 3 2 1 0
Массив после быстрой сортировки:
0 1 1 2 2 3 3 4 4 5
Время работы быстрой сортировки: 0,0037
-----
Изначальный массив:
1 2 3 4 5 4 3 2 1 0
Массив после стандартной сортировки:
0 1 1 2 2 3 3 4 4 5
Время работы стандартной сортировки: 0,0135
-----
```

Вывод: написали код и выполнили поставленные задачи.