Федеральное агентство связи

Государственное бюджетное образовательное учреждение высшего образование

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра «МКиИТ»

дисциплина «СиАОД»

Отчет по Лабораторной работе №1

Подготовил студент

группы БВТ1901: Балдова Татьяна

Проверил: Мелехин А.

Москва 2020

Задание 1

Реализовать заданный метод сортировки строк числовой матрицы в соответствии с индивидуальным заданием. Добавить реализацию быстрой сортировки (quicksort). Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки, используемой в выбранном языке программирования.

Сортировка выбором:

```
void ChoiceSort(int[,] ListForSort)
{
    for (int k = 0; k < MatrixSize; k++)
    {
        int index = 0;

        for (int i = 0; i < MatrixSize - 1; i++)
        {
            index = i;

            for (int j = i + 1; j < MatrixSize; j++)
            {
                if (ListForSort[k, j].CompareTo(ListForSort[k, index]) == -1)
                {
                    index = j;
                }
            }

            if (index != i)
            {
                int temp = ListForSort[k, i];
                ListForSort[k, i] = ListForSort[k, index];
                ListForSort[k, index] = temp;
            }
        }
    }
}
```

Сортировка вставками:

```
void InsertsSort(int[,] ListForSort)
{

    for (int k = 0; k < MatrixSize; k++)
    {
        for (int i = 0; i < MatrixSize; i++)
        {
            var temp = ListForSort[k, i];
```

```
            var j = i;
            while (j > 0 && temp.CompareTo(ListForSort[k, j - 1]) == -1)
            {
                ListForSort[k, j] = ListForSort[k, j - 1];
                j--;
            }
            ListForSort[k, j] = temp;
        }

    }
}
```

Сортировка обменом:

```
void ExchangeSort(int[,] ListForSort)
{

    for (int k = 0; k < MatrixSize; k++)
    {
        int count = MatrixSize;
        for (int j = 0; j < count; j++)
        {
            for (int i = 0; i < count - 1 - j; i++)
            {
                var a = ListForSort[k, i];
                var b = ListForSort[k, i + 1];
                if (a.CompareTo(b) == 1)
                {
                    int temp = ListForSort[k, i];
                    ListForSort[k, i] = ListForSort[k, i + 1];
                    ListForSort[k, i + 1] = temp;
                }

            }
            count--;
        }

    }
}
```

Сортировка Шелла:

```
void ShellaSort(int[,] ListForSort)
{

    for (int k = 0; k < MatrixSize; k++)
    {
        int step = MatrixSize / 2;

        while (step > 0)
        {
            for (int i = step; i < MatrixSize; i++)
            {
                int j = i;
                   while (j >= step && ListForSort[k, j -
                   step].CompareTo(ListForSort[k, j]) == 1)
                {
```

```
                        int temp = ListForSort[k, j - step];
                        ListForSort[k, j - step] = ListForSort[k, j];
                        ListForSort[k, j] = temp;
                        j -= step;
                    }
                }
                step /= 2;
            }
        }
    }
```

Быстрая сортировка:

```
void QuickSort(int[,] ListForSort, int left, int right, int k)
        {
            if (left >= right) { return; }
            else
            {
                var pivot = Sorting(ListForSort, left, right, k);
                QuickSort(ListForSort, left, pivot - 1, k);
                QuickSort(ListForSort, pivot + 1, right, k);
            }
        }

        int Sorting(int[,] ListForSort, int left, int right, int k)
        {
            var pointer = left;
            for (int i = left; i <= right; i++)
            {
                if (ListForSort[k, i].CompareTo(ListForSort[k, right]) == -1)
                {
                    int temp1 = ListForSort[k, i];
                    ListForSort[k, i] = ListForSort[k, pointer];
                    ListForSort[k, pointer] = temp1;
                    pointer++;
                }
            }

            int temp = ListForSort[k, right];
            ListForSort[k, right] = ListForSort[k, pointer];
            ListForSort[k, pointer] = temp;

            return pointer;
        }
```

Сортировка с помощью встроенной функции:

```
                for (int i = 0; i < MatrixSize; i++)
                {

                    int[] MyArray = new int[MatrixSize];
                    for (int j = 0; j < MatrixSize; j++)
                    {
                        MyArray[j]=FirstMatrix[i, j];
                    }

                    Array.Sort(MyArray);
```

```
            for (int j = 0; j < MatrixSize; j++)
            {
                FirstMatrix[i,j]=MyArray[j];
            }
        }
```

Сортировка турнирная:

```
        static int[] heapify(ref int[] arr,int n,int k)
        {
            int m = k;
            int left = 2 * k;
            int right = 2 * k + 1;
            if(left<n && arr[m] < arr[left])
            {
                m = left;
            }
            if(right< n && arr[m] < arr[right])
            {
                m = right;
            }
            if (m != k)
            {
                int temp = arr[k];
                arr[k] = arr[m];
                arr[m] = temp;
                heapify(ref arr, n, m);
            }
            return arr;
        }

        static int[] TornSort(ref int[] arr)
        {
            for(int i = arr.Length / 2; i > -1; i--)
            {
                heapify(ref arr, arr.Length, i);
            }
            for(int i = arr.Length -1; i > -1; i--)
            {
                if (arr[0] > arr[i])
                {
                    int temp = arr[0];
                    arr[0] = arr[i];
                    arr[i] = temp;
                    heapify(ref arr, i, 0);
                }
            }
            return arr;
        }
```

Сортировка пирамидальная:

```
using System;
using System.Collections;
using System.Collections.Generic;

namespace Лабораторная_работа__1
{
    class Heap : IEnumerable
    {
        private List<int> items = new List<int>();
        public int Count => items.Count;
```

```
        public Heap() { }

        public Heap(List<int> items)
        {
            this.items.AddRange(items);
            for (int i = Count; i >= 0; i--)
            {
                Sort(i);
            }
        }

        public int GetMax()
        {
            var result = items[0];
            items[0] = items[Count - 1];
            items.RemoveAt(Count - 1);
            Sort(0);
            return result;
        }

        private void Sort(int curentIndex)
        {
            int minIndex = curentIndex;
            int leftIndex;
            int rightIndex;

            while (curentIndex < Count)
            {
                leftIndex = 2 * curentIndex + 1;
                rightIndex = 2 * curentIndex + 2;

                if (leftIndex < Count && items[leftIndex] <
items[minIndex])
                {
                    minIndex = leftIndex;
                }

                if (rightIndex < Count && items[rightIndex] <
items[minIndex])
                {
                    minIndex = rightIndex;
                }

                if (minIndex == curentIndex)
                {
                    break;
                }

                Swap(curentIndex, minIndex);
                curentIndex = minIndex;
            }
        }

        private void Swap(int currentIndex, int parentIndex)
        {
            int temp = items[currentIndex];
            items[currentIndex] = items[parentIndex];
            items[parentIndex] = temp;
        }


        public IEnumerator GetEnumerator()
        {
            while (Count > 0)
            {
```

```
                        yield return GetMax();
                    }
                }
            }
        }
```

Генерация матрицы:

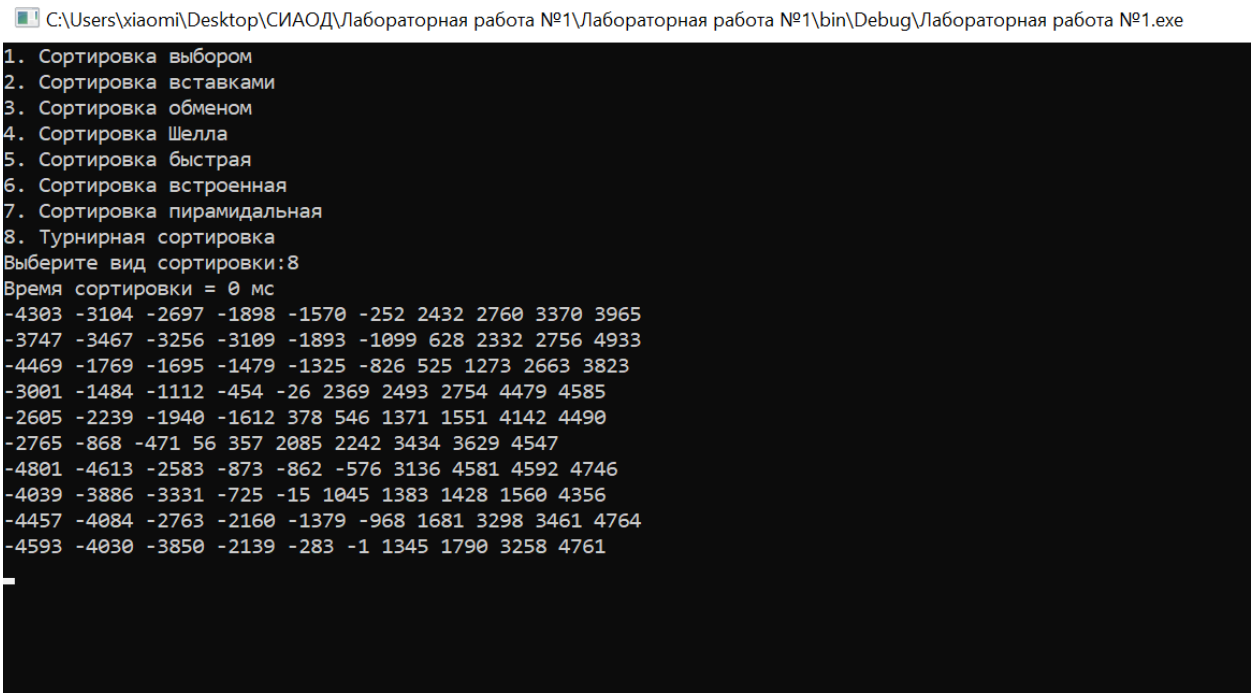
```
int[,] CreateMatrix()
{
    int[,] Matrix = new int[MatrixSize, MatrixSize];
    Random rand = new Random();
    for (int i = 0; i < MatrixSize; i++)
    {
        for (int j = 0; j < MatrixSize; j++)
        {
            Matrix[i, j] = rand.Next(-5000, 5001);
        }
    }
    return Matrix;
}
```

Выбор сортировки и демонстрация результата:

```
            /Выбор сортировки
            Console.Write("1. Сортировка выбором" + "\r\n" + "2. Сортировка вставками" +
"\r\n" + "3. Сортировка обменом" + "\r\n" + "4. Сортировка Шелла" + "\r\n" + "5.
Сортировка быстрая" + "\r\n" + "6. Сортировка встроенная" + "\r\n" + "7. Сортировка
пирамидальная" + "\r\n" + "8. Турнирная сортировка" + "\r\n" + "Выберите вид
сортировки:");

            //Генерация матрицы
            int[,] FirstMatrix = CreateMatrix();

            //Осуществление выбранной сортировки
            switch (Convert.ToInt32(Console.ReadLine()))
            {
                case 1:

                    ChoiceSort(FirstMatrix);
                    break;
                case 2:
                    InsertsSort(FirstMatrix);
                    break;
                case 3:
                    ExchangeSort(FirstMatrix);
                    break;
                case 4:
                    ShellaSort(FirstMatrix);
                    break;
                case 5:
                    for (int k = 0; k < MatrixSize; k++)
                    {
                        QuickSort(FirstMatrix, 0, MatrixSize - 1, k);
                    }
                    break;
                case 6:
```

```
            for (int i = 0; i < MatrixSize; i++)
            {
                timer.Start();
                int[] MyArray = new int[MatrixSize];
                for (int j = 0; j < MatrixSize; j++)
                {
                    MyArray[j] = FirstMatrix[i, j];
                }
                Array.Sort(MyArray);
                for (int j = 0; j < MatrixSize; j++)
                {
                    FirstMatrix[i, j] = MyArray[j];
                }
            }
            break;
        case 8:
            for (int i = 0; i < MatrixSize; i++)
            {

                int[] MyArray = new int[MatrixSize];
                for (int j = 0; j < MatrixSize; j++)
                {
                    MyArray[j] = FirstMatrix[i, j];
                }
                TornSort(ref MyArray);
                for (int j = 0; j < MatrixSize; j++)
                {
                    FirstMatrix[i, j] = MyArray[j];
                }
            }
            break;
        case 7:
            for (int i = 0; i < MatrixSize; i++)
            {
                List<int> array = new List<int>(MatrixSize);
                for (int j = 0; j < MatrixSize; j++)
                {
                    array.Add(FirstMatrix[i, j]);
                }
                Heap MyHeap = new Heap(array);
                foreach (int a in MyHeap)
                {
                    Console.Write(a + " ");
                }
                Console.Write("\r\n");

            }
```

Пример отображения результата:



Вывод: реализовала заданный метод сортировки строк числовой матрицы в соответствии с индивидуальным заданием. Добавила реализацию быстрой сортировки (quicksort). Оценила время работы каждого алгоритма сортировки и сравнила его со временем стандартной функции сортировки, используемой в выбранном языке программирования.