**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

Подготовила студентка

группы БВТ1901: Нкурикийе Хафидати

Проверил: Мелехин А.

Москва 2021

# 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.

using System;

using System.Collections.Generic;

namespace Project112

{

class Program

{

static int MySize = 20;

static void Main(string[] args)

{

int[,] Matrix = new int[MySize, MySize];

Random rand = new Random();

for (int i = 0; i < MySize; i++)

{

for (int j = 0; j < MySize; j++)

{

Matrix[i, j] = rand.Next(0, 5001);

}

}

Console.Write("Enter sort: " + "\r\n" + "1. ChoiceSort" + "\r\n" + "2. InsertSort" + "\r\n" + "3. ExchangeSort" + "\r\n" + "4. ShellaSort" + "\r\n" + "5. BuildInSort" + "\r\n" + "6. TournamenSort" + "\r\n" + "7. PyramidSort" + "\r\n" + "8. FastSort" + "\r\n");

int Sort = Convert.ToInt32(Console.ReadLine());

if (Sort == 1)

{

Sort1(Matrix);

}

else if (Sort == 2)

{

Sort2(Matrix);

}

else if (Sort == 3)

{

Sort3(Matrix);

}

else if (Sort == 4)

{

Sort4(Matrix);

}

else if (Sort == 5)

{

for (int i = 0; i < MySize; i++)

{

int[] MyArray = new int[MySize];

for (int j = 0; j < MySize; j++)

{

MyArray[j] = Matrix[i, j];

}

Array.Sort(MyArray);

for (int j = 0; j < MySize; j++)

{

Matrix[i, j] = MyArray[j];

}

}

}

else if (Sort == 6)

{

for (int i = 0; i < MySize; i++)

{

int[] MyArray = new int[MySize];

for (int j = 0; j < MySize; j++)

{

MyArray[j] = Matrix[i, j];

}

Sort6(ref MyArray);

for (int j = 0; j < MySize; j++)

{

Matrix[i, j] = MyArray[j];

}

}

}

else if (Sort == 7)

{

for (int i = 0; i < MySize; i++)

{

List<int> array = new List<int>(MySize);

for (int j = 0; j < MySize; j++)

{

array.Add(Matrix[i, j]);

}

for (int k = array.Count; k >= 0; k--)

{

Sort7(array, k);

}

while (array.Count > 0)

{

Console.Write(GetMax(array) + " ");

}

Console.Write("\r\n");

}

Console.ReadLine();

return;

}

else if (Sort == 8)

{

for (int k = 0; k < MySize; k++)

{

Sort8(Matrix, 0, MySize - 1, k);

}

}

else

{

Console.WriteLine("Error!");

}

for (int i = 0; i < MySize; i++)

{

for (int j = 0; j < MySize; j++)

{

Console.Write(Convert.ToString(Matrix[i, j] + " "));

}

Console.Write("\r\n");

}

Console.ReadLine();

}

static void Sort1(int[,] ListForSort) //**Сортировка выбором cложность О(n^2)**

{

for (int k = 0; k < MySize; k++)

{

int index = 0;

for (int i = 0; i < MySize - 1; i++)

{

index = i;

for (int j = i + 1; j < MySize; 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;

}

}

}

}

static void Sort2(int[,] ListForSort)//**Сортировка вставками cложность О(n^2)**

{

for (int k = 0; k < MySize; k++)

{

for (int i = 0; i < MySize; 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;

}

}

}

static void Sort3(int[,] ListForSort)//**Сортировка обменом cложность О(n^2)**

{

for (int k = 0; k < MySize; k++)

{

int count = MySize;

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--;

}

}

}

static void Sort4(int[,] ListForSort)//**Сортировка Шелла cложность О(n\*log^2(n))**

{

for (int k = 0; k < MySize; k++)

{

int step = MySize / 2;

while (step > 0)

{

for (int i = step; i < MySize; 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;

}

}

}

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[] Sort6(ref int[] arr) //Сортировка турнирная cложность О(n\*log^2(n))

{

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;

}

static void Sort7(List<int> arr, int curentIndex)//Сортировка пирамидальная cложность О(1)

{

int minIndex = curentIndex;

int leftIndex;

int rightIndex;

while (curentIndex < arr.Count)

{

leftIndex = 2 \* curentIndex + 1;

rightIndex = 2 \* curentIndex + 2;

if (leftIndex < arr.Count && arr[leftIndex] < arr[minIndex])

{

minIndex = leftIndex;

}

if (rightIndex < arr.Count && arr[rightIndex] < arr[minIndex])

{

minIndex = rightIndex;

}

if (minIndex == curentIndex)

{

break;

}

Swap(arr, curentIndex, minIndex);

curentIndex = minIndex;

}

}

static void Swap(List<int> arr, int currentIndex, int parentIndex)

{

int temp = arr[currentIndex];

arr[currentIndex] = arr[parentIndex];

arr[parentIndex] = temp;

}

static int GetMax(List<int> arr)

{

var result = arr[0];

arr[0] = arr[arr.Count - 1];

arr.RemoveAt(arr.Count - 1);

arr.Sort();

arr.Reverse();

return result;

}

static void Sort8(int[,] ListForSort, int left, int right, int k) //**Сортировка быстрая cложность** **О(n\*Log(n))**

{

if (left >= right) { return; }

else

{

var pivot = FastSorting(ListForSort, left, right, k);

Sort8(ListForSort, left, pivot - 1, k);

Sort8(ListForSort, pivot + 1, right, k);

}

}

static int FastSorting(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;

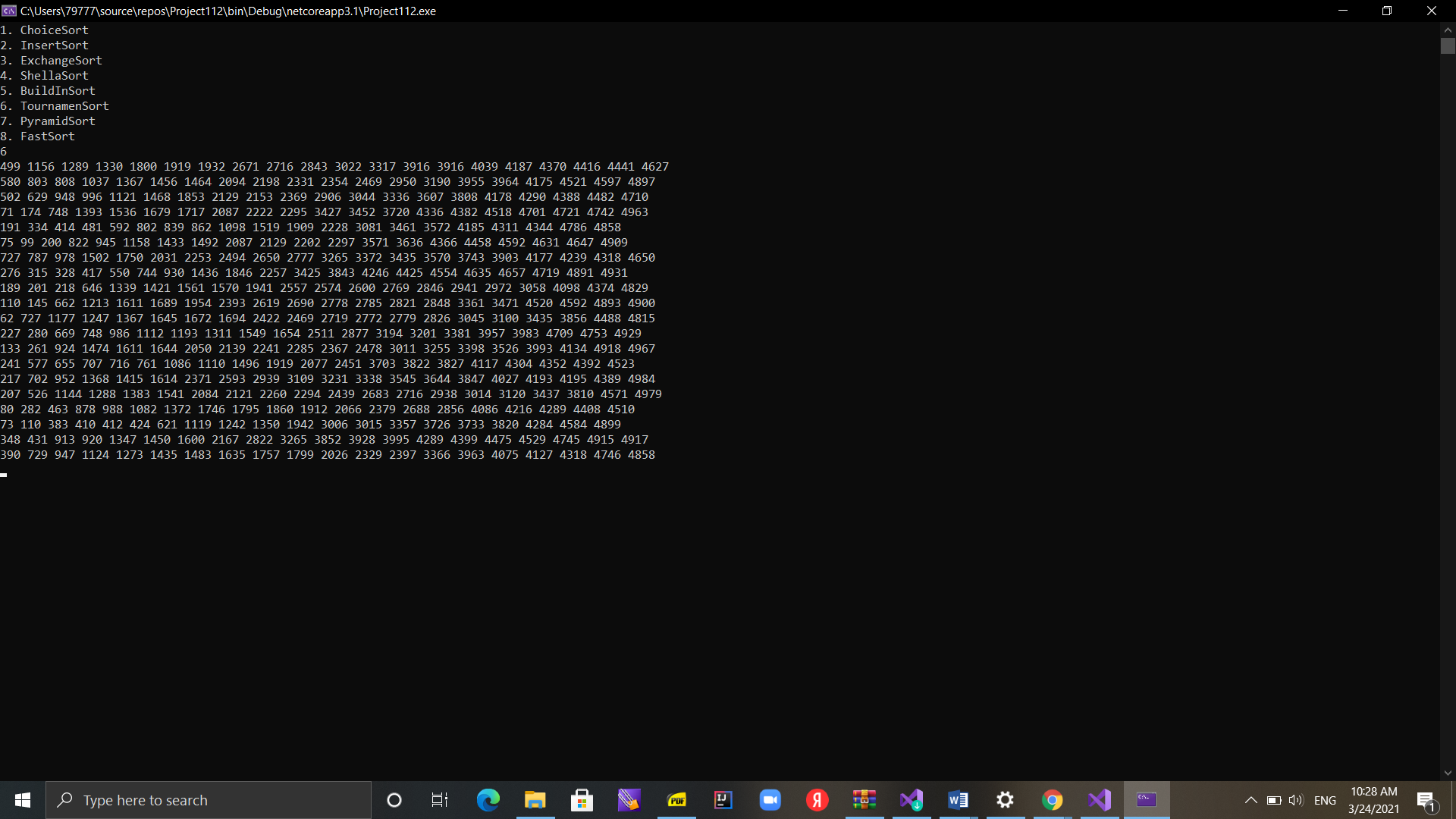
return pointer;

}

}

}

**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