

Федеральное агентство связи

**Государственное бюджетное образовательное учреждение высшего
образования**

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.

<https://github.com/hafidhati/->

[https://github.com/hafidhati/-
/blob/main/%D0%BE%D1%82%D1%87%D0%B5%D1%82%20%D0%BF%D0%
BE%20%D0%B9%20lab.docx](https://github.com/hafidhati/-/blob/main/%D0%BE%D1%82%D1%87%D0%B5%D1%82%20%D0%BF%D0%BE%20%D0%B9%20lab.docx)

```
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 сложность  $O(n^2)$ 
{
    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 сложность $O(n^2)$**

```

{

    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 сложность  $O(n^2)$ 
```

```
{  
  
    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 сложность  $O(n \cdot \log^2(n))$ 
```

```
{
```

```
    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 сложность  $O(n \cdot \log^2(n))$ 
{
    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 сложность $O(1)$

```

{
    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**
СЛОЖНОСТЬ $O(n \cdot \log(n))$

```
{  
    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:

```

C:\Users\79777\source\repos\Project112\bin\Debug\netcoreapp3.1\Project112.exe
1. ChoiceSort
2. InsertSort
3. ExchangeSort
4. ShellaSort
5. BuildInSort
6. TournamenSort
7. PyramidSort
8. FastSort
6
499 1156 1289 1330 1800 1919 1932 2671 2716 2843 3022 3317 3916 3916 4039 4187 4370 4416 4441 4627
580 803 808 1037 1367 1456 1464 2094 2198 2331 2354 2469 2950 3190 3955 3964 4175 4521 4597 4897
502 629 948 996 1121 1468 1853 2129 2153 2369 2906 3044 3336 3607 3808 4178 4290 4388 4482 4710
71 174 748 1393 1536 1679 1717 2087 2222 2295 3427 3452 3720 4336 4382 4518 4701 4721 4742 4963
191 334 414 481 592 802 839 862 1098 1519 1909 2228 3081 3461 3572 4185 4311 4344 4786 4858
75 99 200 822 945 1158 1433 1492 2087 2129 2202 2297 3571 3636 4366 4458 4592 4631 4647 4909
727 787 978 1502 1750 2031 2253 2494 2650 2777 3265 3372 3435 3570 3743 3903 4177 4239 4318 4650
276 315 328 417 550 744 930 1436 1846 2257 3425 3843 4246 4425 4554 4635 4657 4719 4891 4931
189 201 218 646 1339 1421 1561 1570 1941 2557 2574 2600 2769 2846 2941 2972 3058 4098 4374 4829
110 145 662 1213 1611 1689 1954 2393 2619 2690 2778 2785 2821 2848 3361 3471 4520 4592 4893 4900
62 727 1177 1247 1367 1645 1672 1694 2422 2469 2719 2772 2779 2826 3045 3100 3435 3856 4488 4815
227 280 669 748 986 1112 1193 1311 1549 1654 2511 2877 3194 3201 3381 3957 3983 4709 4753 4929
133 261 924 1474 1611 1644 2050 2139 2241 2285 2367 2478 3011 3255 3398 3526 3993 4134 4918 4967
241 577 655 707 716 761 1086 1110 1496 1919 2077 2451 3703 3822 3827 4117 4304 4352 4392 4523
217 702 952 1368 1415 1614 2371 2593 2939 3109 3231 3338 3545 3644 3847 4027 4193 4195 4389 4984
207 526 1144 1288 1383 1541 2084 2121 2260 2294 2439 2683 2716 2938 3014 3120 3437 3810 4571 4979
80 282 463 878 988 1082 1372 1746 1795 1860 1912 2066 2379 2688 2856 4086 4216 4289 4408 4510
73 110 383 410 412 424 621 1119 1242 1350 1942 3006 3015 3357 3726 3733 3820 4284 4584 4899
348 431 913 920 1347 1450 1600 2167 2822 3265 3852 3928 3995 4289 4399 4475 4529 4745 4915 4917
390 729 947 1124 1273 1435 1483 1635 1757 1799 2026 2329 2397 3366 3963 4075 4127 4318 4746 4858

```

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