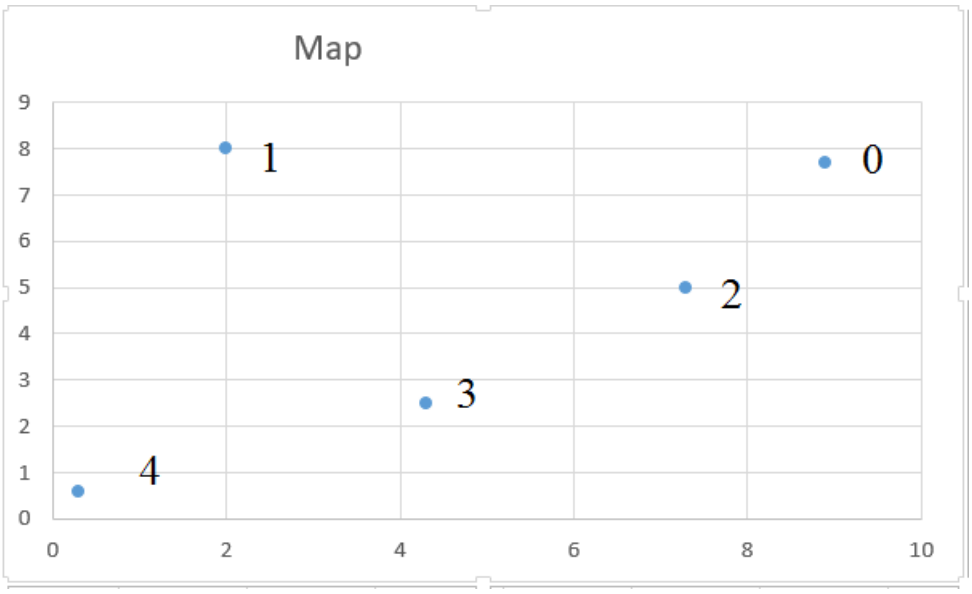
Polina Khan

0.4.10.2016

Map:



Matrix

	0	1	2	3	4
0	0				
1	6.9	0			
2	3.1D	6.1	0		
3	6.88	5.69	3.9	0	
4	11.15	3.14	8.27	4.42	0

Ways

[0, 1, 2, 3, 4, 0] - 36.32

[0, 1, 2, 4, 3, 0] - 32.57

[0, 1, 3, 2, 4, 0] - 35.91

[0, 1, 3, 4, 2, 0] - 32.269999999999996

[0, 1, 4, 2, 3, 0] - 33.54

[0, 1, 4, 3, 2, 0] -25.95

[0, 2, 1, 3, 4, 0] - 34.35

[0, 2, 1, 4, 3, 0] - 28.13

[0, 2, 3, 1, 4, 0] - 31.47

[0, 2, 3, 4, 1, 0] - 29.799999999999997

[0, 2, 4, 1, 3, 0] -31.57

[0, 2, 4, 3, 1, 0] - 28.42

[0, 3, 1, 2, 4, 0] - 38.09

[0, 3, 1, 4, 2, 0] -31.57

[0, 3, 2, 1, 4, 0] - 35.62

[0, 3, 2, 4, 1, 0] -33.54

[0, 3, 4, 1, 2, 0] -31.979999999999997

[0, 3, 4, 2, 1, 0] -36.419999999999995

[0, 4, 1, 2, 3, 0] -35.620000000000005

[0, 4, 1, 3, 2, 0] -31.470000000000002

[0, 4, 2, 1, 3, 0] -38.09

[0, 4, 2, 3, 1, 0] -35.910000000000004

[0, 4, 3, 1, 2, 0]- 30.5

[0, 4, 3, 2, 1, 0] -32.47

Mininal way :

[0, 1, 4, 3, 2, 0] -25.95

Source code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;

namespace SIIT_3
{
    public class Permutations
    {
```

```
private List<string> _premutations;
private string _str;
private void AddToList(char[] a, bool repeat = true)
{
    var bufer = new StringBuilder("");
    for (int i = 0; i < a.Count(); i++)
    {
        bufer.Append(a[i]);
    }
    if (repeat || !_premutations.Contains(bufer.ToString()))
    {
        _premutations.Add(bufer.ToString());
    }
}
private void RecPermutation(char[] a, int n, bool repeat = true)
{
    for (int i = 0; i < n;i++)
    {
        var tmp = a[n - 1];
        for (int j = n-1;j>0;j--)
        {
            a[j] = a[j - 1];
        }
        a[0] = tmp;
        if (i < n - 1) AddToList(a, repeat);
        if (n > 0) RecPermutation(a, n - 1, repeat);
    }
}
public Permutations()
{
    _str = "";
}
```

```
    public Permutations(string str)
    {
        _str = str;
    }
    public string PermutationStr
    {
        get
        {
            return _str;
        }
        set
        {
            _str = value;
        }
    }
    public List<string> GetList(bool repeat = true)
    {
        _premutations = new List<string> { _str };
        RecPermutation(_str.ToArray(), _str.Length, repeat);
        return _premutations;
    }
    public List<string> SortList(bool repeat = true)
    {
        GetList(repeat).Sort();
        return _premutations;
    }
}
class Program
{
    static void Main(string[] args)
    {
        List<double> historymin = new List<double>();
```

```
string all = "0123456789";
string cities = "";
string start = "0";
Random rnd = new Random();
double d;
List<double> x = new List<double>();
List<double> y = new List<double>();
List<double> dist = new List<double>();
int numbers = 6;
for (int i = 0; i < numbers; i++)
{
   cities += all[i];
}
cities = cities.Replace(start, "");
for (int i = 0; i < numbers; i++)
{
   d = rnd.Next(0, 1000);
   d = d / 100;
   x.Add(d);
   d = rnd.Next(0, 1000);
   d = d / 100;
   y.Add(d);
}
for (int i = 0; i < x.Count; i++)
{
   Console.Write((i + 1).ToString() + ':' + '(' + x[i] + ';' + y[i] + ')' + '\n');
}
double[,] matrix = new double[numbers, numbers];
for (int i = 0; i < numbers; i++)
{
   for (int j = 0; j < numbers; j++)
   {
```

```
        if (i == j)
        {
            matrix[i, j] = 0;
        }
        else
        {
            double ras = 0;
            ras = Math.Sqrt(Math.Pow(x[j] - x[i], 2) + Math.Pow(y[j] - y[i], 2));
            matrix[i, j] = ras;
        }
    }
}
for (int i = 0; i < numbers; i++)
{
    for (int j = 0; j < numbers; j++)
    {
        Console.Write(matrix[i, j].ToString() + '\t');
    }
    Console.Write('\n');
}
var per = new Permutations(cities);
var list = per.SortList(false);
string[] tmp = list.ToArray();
for (int i = 0; i < tmp.Length; i++)
{
    string temp = "";
    temp += start;
    temp += tmp[i];
    temp += start;
    double dis = 0;
    for (int j = 0; j < temp.Length - 1; j++)
    {
```

```
            char t = temp[j];
            char t1 = temp[j + 1];
            int w = all.IndexOf(t);
            int w1 = all.IndexOf(t1);
            dis += matrix[w, w1];
        }
        Console.WriteLine(temp + ':' + dis.ToString() + '\n');
        dist.Add(dis);
    }
    dist.Sort();
    for (int i = 0; i < dist.Count; i++)
    {
        double dis = 0;
        string temp = "";
        temp += start;
        temp += tmp[i];
        temp += start;
        for (int j = 0; j < temp.Length - 1; j++)
        {
            char t = temp[j];
            char t1 = temp[j + 1];
            int w = all.IndexOf(t);
            int w1 = all.IndexOf(t1);
            dis += matrix[w, w1];
        }
        if (dist[0] == dis)
            Console.WriteLine("Minimum:" + temp + ':' + dist[0].ToString());
    }
    Console.ReadKey();
    historymin.Add(dist[0]);
    double www = dist[0];
    dist.Clear();
```

```
            for (int i = 0; i < 100; i++)
            {
                double temp = rnd.Next(2500);
                temp = temp / 10000;
                www = www - temp;
                historymin.Add(www);
                Console.WriteLine(www);
            }
            www = historymin[historymin.Count -1];
            for (int i = 0; i < 10; i++)
            {
                historymin.Add(www);
            }
            string answer = "";
            for (int i = 0; i < historymin.Count; i++)
            {
                answer += '(' + (i + 1).ToString() + ';' + historymin[i].ToString() +')'+'\n';
            }
            File.WriteAllText("Graphic.txt",answer);
        }
    }
}
```

Answer: