IPAR 4.0 .NET C# ALAPOKON

7. forduló

SEMILAB

A kategória támogatója: Semilab Zrt.

Ismertető a feladathoz

A feladatlap több csatolmányt is tartalmaz, ezért a megoldását asztali gépen javasoljuk!

Fontos!

Fordulónként javasoljuk az összes részfeladat végigolvasását a kidolgozás megkezdése előtt, mivel a feladatok sokszor egymásra épülnek. Előfordul, hogy egy részfeladat nehézségét az input mérete adja, így érdemes hatékony megoldásokra törekedni.

Felhasznált idő: 40:00/40:00 **Elért pontszám:** 0/16

Indítás előtti csatolmányok

1. feladat 0/2 pont

Egy egyenes mentén mozgó mérőfej segítségével szeretnénk lemérni különböző mérési pontokat. A mérőfejnek van egy maximum sebessége, amely azt jelöli, hogy 1 időegység alatt mennyi pozíciót képes maximum mozogni. Minden mérési ponthoz tartozik egy pozíció és egy időegység. Egy mérési pontot akkor tekintünk lemértnek, ha a fej az adott időegységben a mérési pont pozícióján áll. A fej a 0. időpillanatban a 0 pozíción helyezkedik el, a pozíciók értékkészlete [-10000, 10000].

Az input formája:

maxSpeed

time_1 time_2 ... time_n

pos_1 pos_2 ... pos_n

Adjuk meg, hogy az 7_1.txt-ben szereplő pozíció-idő adatok alapján hány mérési pont érhető el fizikailag (egymástól függetlenül) a kezdőpozícióból a mérőfej számára a megadott maximális sebesség mellett.

Példa:

Input:

4

1 1 4 5 6 8

-4 2 11 12 25 -300

Output:

4

Az első 4 mérési pont elérhető, viszont az ötödik és hatodik nem, ezért a példára a válasz 4.

A 7_1_test.txt eredménye 940.

Segítség a beolvasáshoz (nem kötelező a használata):

```
public static List<Tuple<int, int>> ReadInput(string inputFile)
{
    var fileLines = File.ReadAllLines(inputFile);
    Speed = int.Parse(fileLines.First().Split(' ')[0]);
    fileLines = fileLines.Skip(1).ToArray();
    var times = fileLines[0].Split(' ').Select(int.Parse).ToList();
    var pos = fileLines[1].Split(' ').Select(int.Parse).ToList();
    //Item1=Time, Item2=Position
    return times.Select((t, i) => new Tuple<int, int>(t, pos[i])).ToList();
}
```

Válasz

A helyes válasz:

953

Magyarázat

```
//Robot sebessége
static int Speed { get; set; }
//Item1=Time, Item2=Position
static List<Tuple<int, int>> Nodes { get; set; }
public static int Solve(string inputFile)
{
    Nodes = ReadInput(inputFile);
    int sum = 0;
    for (int i = 0; i < Nodes.Count; i++)
    {
        if (Math.Abs(Nodes[i].Item2) <= Speed * Nodes[i].Item1)
        {
            sum++;
        }
    }
    return sum;
}

public static List<Tuple<int, int>> ReadInput(string inputFile)
{
    var fileLines = File.ReadAllLines(inputFile);
    Speed = int.Parse(fileLines.First().Split(' ')[0]);
    fileLines = fileLines.Skip(1).ToArray();
    var times = fileLines[0].Split(' ').Select(int.Parse).ToList();
    var pos = fileLines[1].Split(' ').Select(int.Parse).ToList();
    //Item1=Time, Item2=Position
```

```
    return times.Select((t, i) => new Tuple<int, int>(t, pos[i])).ToList();
}
```

2. feladat 0/7 pont

Adjuk meg, hogy a mérőfej maximum hány db mérési pontot tud lemérni egy menetben, a mérési pontokat egymás után fűzve a 7_2.txt inputra.

Példa

Input:

4

1 1 4 5 6 8

-4 2 11 12 25 -300

Output:

3

A legjobb sorozat (1,2), (4,11), (5,12) azaz 3 db a lemérhető helyek száma.

A 7_2_test.txt eredménye 93.

Válasz

A helyes válasz:

102

Magyarázat

Algoritmus lépései:

1. Rendezzük a mérési pontokat úgy, hogy a legkésőbbi pont legyen az első. Ezáltal biztos nem lesz olyan, hogy egy kisebb indexű elem után egy nagyobb indexű elemet el tud érni a kar.
2. Végigmegyünk a tömbön, és minden elemhez kiszámoljuk a lehető leghosszabb mérési sorozatot, ami belőle indulna ki.

Van egy segédtömbünk, amelynek az indexei az útvonalhosszokat jelölik. Amikor kiszámoljuk egy elemhez a leghosszabb mérési sorozatát, akkor ezt az elemet beszúrjuk a hossz indexű helyére. Folyamatosan tároljuk, hogy mekkora az eddigi leghosszabb útvonal hossza.

2a. A rendezés miatt a leghosszabb útvonal kiszámításánál igaz lesz az, hogy az összes olyan pont már ki van számolva, amely az aktuális számolandó pontból elérhető.

2b. Az aktuális pontunk leghosszabb útvonalához elkezdünk iterálni a leghosszabb útvonalakat tartalmazó tömbünk elemein úgy, hogy először az eddigi leghoszabb útvonalat tartalmazó elemeket vizsgáljuk meg.

Az első olyan elemnél amely elérhető a vizsgált elemünkből vesszük a leghosszabb útvonalát, és hozzáadunk 1-et. Ez lesz a vizsgált elemünkből a leghosszabb elérhető útvonal.

```
public static int Solve(string inputFile)
{
    Nodes=ReadInput(inputFile);
    Nodes = Nodes.OrderByDescending(m => m.Item1).ToList();
    MaxCountNodeIndexes = new List<int>[Nodes.Count];
    for (int i = 0; i < MaxCountNodeIndexes.Length; i++)
    {
```

```
        MaxCountNodeIndexes[i] = new List<int>();
    }
    var max = 0;
    for (int i = 0; i < Nodes.Count; i++)
    {
        var current = SetNodeCount(i);
        if (Math.Abs(Nodes[i].Item2) <= Speed * Nodes[i].Item1)
        {
            if (current > max) max = current;
        }
    }
    return max;
}

//MaxCountNodeIndexes[i]: Azoknak a Node indexeknek a listája, ahonnan a maximális mérési sorozat pontosan i
static List<int>[] MaxCountNodeIndexes { get; set; }
//Az eddig ismert leghosszabb mérési sorozat nagysága
static int CurrentMaxCount { get; set; } = 0;
//Item1=Time, Item2=Position
static List<Tuple<int, int>> Nodes { get; set; }

public static List<Tuple<int, int>> ReadInput(string inputFile)
{
    var fileLines = File.ReadAllLines(inputFile);
    Speed = int.Parse(fileLines.First().Split(' ')[0]);
    fileLines = fileLines.Skip(1).ToArray();
    var times = fileLines[0].Split(' ').Select(int.Parse).ToList();
    var pos = fileLines[1].Split(' ').Select(int.Parse).ToList();
    //Item1=Time, Item2=Position
    return times.Select((t, i) => new Tuple<int, int>(t, pos[i])).ToList();
}

static int SetNodeCount(int index)
{
    int max = 0;
    for (int i = CurrentMaxCount; i > 0; i--)
    {
        bool hasNext = false;
        foreach (var itemIndex in MaxCountNodeIndexes[i])
        {
            if (Math.Abs(Nodes[index].Item2 - Nodes[itemIndex].Item2) <= (Nodes[itemIndex].Item1 - Nodes[inde
            {
                hasNext = true;
                max = i;
                break;
            }
        }
        if (hasNext)
        {
            break;
        }
    }
    MaxCountNodeIndexes[max + 1].Add(index);
    if (CurrentMaxCount < max + 1)
    {
        CurrentMaxCount = max + 1;
    }
    return max + 1;
}
```

3. feladat 0/7 pont

A 2. feladathoz hasonlóan adjuk meg, hogy maximum hány db helyet tudunk a mérőfejjel lemérni egy menetben, a mérési pontokat egymás után fűzve a 7_3.txt inputra.

Válasz

A helyes válasz:

1413

Magyarázat

A 2. feladatban megadott megoldás erre az inputra is belátható időn belül jó eredményt ad.