IPAR 4.0 .NET C# ALAPOKON

4. forduló

SEMILAB

A kategória támogatója: Semilab Zrt.

Ismertető a feladathoz

A 4. forduló után elérhetőek lesznek a helyezések %-os formában: azaz kiderül, hogy a kategóriában a versenyzők TOP 20% - 40% -60% -ához tartozol-e!

Szeretnénk rá felhívni figyelmedet, hogy a játék nem Forma-1-es verseny! Ha a gyorsaságod miatt kilököd a rendesen haladó versenyzőket, kizárást vonhat maga után!

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:** 0/12

Indítás előtti csatolmányok

1. feladat 0/4 pont

Egy új mérőgép került a gyárba, amely képes egyszerre több wafert is megmérni. A mérőgéphez csoportosan érkeznek a waferek tárolókban, úgynevezett foupokban. Minden egyes fouphoz meg van határozva, hogy hány db wafer van benne. A mérőgép fogadja az érkező foupokat mindaddig, amíg a waferkapacitása engedi, és akkor kezd el mérni, amikor a soron következő foup összes wafere már nem férne el a gépben (ezen foup waferei csak a következő mérésben fognak sorrakerülni). A foupok egymás után sorakoznak, a darabszámuk véges. A lemért waferek visszakerülnek a saját foupukba, majd azonnal visszakerülnek a sor végére újabb mérésre várakozva.

A következő szabályok érvényesek:

- Egy foup wafereit csak egyszerre mérhetjük, tehát lesz olyan, amikor a mérőgép nem teljes kihasználtság mellett dolgozik (amikor a következő foup összes wafere már nem férne bele a mérőgépbe, csak egy része).

- A foupok sorrendben jönnek, két foup mérésének a sorrendje nem felcserélhető.

Input tartalma:

[machine capacity] [measurement count]

foup1_count, foup2_count, ... foupK_count

A machine_capacity jelöli, hogy hány db wafer fér el a gépben, illetve a measurement_count jelöli, hogy összesen hányszor mér a gépünk.

Adjuk meg, hogy összesen hány db wafert mértünk meg!

Példa:

Input:

10 3

1 3 8

Ekkor a mérések: (1,3) (8,1) (3), azaz 1+3+8+1+3=16 wafert mértünk le.

A 4_1_test.txt-re adott kimenet: 98483

Adjuk meg a 4_1.txt-re adott kimenetet!

Válasz

A helyes válasz:

98545

Magyarázat

A feladat ekkora inputra brute force megoldással megoldható. A végleges nagy inputokra is működő megoldást lásd a 2. feladat megoldásánál.

2. feladat 0/8 pont

Az előző feladathoz hasonlóan adjuk meg a lemért waferek számát, de egy sokkal nagyobb inputra:

A 4_2_test.txt-re adott kimenet: 996885619831922

Adjuk meg a 4_2.txt-re adott kimenetet!

Válasz

A helyes válasz:

996653333332237

Magyarázat

```
public static long Solve(string inputFile)
{
    var lines = File.ReadAllLines(inputFile);
    string[] inputs = lines[0].Split(' ');
    long Capacity = long.Parse(inputs[0]);
    long MaxMeasurementCount = long.Parse(inputs[1]);
```

```
    var foups = lines[1].Split(' ').Select(i => new Foup { WaferCount = int.Parse(i) }).ToList();
    long sumWafer = foups.Sum(g => g.WaferCount);
    if (Capacity > sumWafer)
    {
        return sumWafer * MaxMeasurementCount;
    }


    long measuredWaferCount = 0;
    var currentFoupIndex = 0;
    long currentMeasurementNumber = 1;
    while (currentMeasurementNumber <= MaxMeasurementCount)
    {
        //Kezdődött-e már a mérőgép feltöltése ezzel a fouppal
        if (foups[currentFoupIndex].StartRound.HasValue)
        {
            //Hány mérés volt aközött, hogy ez a foup volt az első a sorban a mérés megkezdése előtt. Ez a so
            var intervalLength = currentMeasurementNumber - foups[currentFoupIndex].StartRound.Value;


            //Összesen hány db ilyen sorozatot tudunk lemérni
            long intervalMultiply = (MaxMeasurementCount - currentMeasurementNumber) / intervalLength;


            //Hány darab wafert mérünk le az összes sorozat alatt.
            measuredWaferCount += intervalMultiply * (measuredWaferCount - foups[currentFoupIndex].MeasuredWa


            //Léptetjük a mérések számát a sorozatok számával
            currentMeasurementNumber = currentMeasurementNumber + intervalMultiply * intervalLength;
        }
        else
        {
            //Amikor a mérőgép feltöltése ezzel a fouppal kezdődik, eltároljuk az aktuális mérés számát, ille
            foups[currentFoupIndex].StartRound = currentMeasurementNumber;
            foups[currentFoupIndex].MeasuredWafersUntilFirstStart = measuredWaferCount;
        }


        //Bepakoljuk a mérőgépbe a foupokat ameddig lehet.
        long currentMeasurementWaferCount = 0;
        while(currentMeasurementWaferCount+ foups[currentFoupIndex].WaferCount <= Capacity)
        {
            currentMeasurementWaferCount += foups[currentFoupIndex].WaferCount;
            currentFoupIndex = (currentFoupIndex + 1) % foups.Count;
        }
        measuredWaferCount += currentMeasurementWaferCount;


        currentMeasurementNumber++;
    }


    return measuredWaferCount;
}

class Foup
{
    public long WaferCount { get; set; }
    public long? StartRound { get; set; }
    public long MeasuredWafersUntilFirstStart { get; set; }
}
```

Megjelenés

Világos