IPAR 4.0 .NET C# ALAPOKON

3. forduló

SEMILAB

A kategória támogatója: Semilab Zrt.

Ismertető a feladathoz

A 3.forduló feladatait a hosszú hétvége miatt kivételesen szerda (11.02.) éjfélig tudod megoldani!

Érdemes ebben a fordulóban is játszanod, mert a következő forduló kezdetekor, 11.03-án 18 órától kiosztjuk az 1.-2.-3. fordulóban megszerzett badgeket!

A verseny közben az alábbi teljesítményeket díjazzuk:

- fordulógyőztes
- átlagnál jobb időeredmény
- átlag feletti pontszám
- hibátlan forduló

Szeretnénk rá felhívni figyelmedet, hogy az egyszer megkapott badge-eket nem vonjuk vissza, akkor sem, ha esetleg az adott fordulóban a visszajelzések alapján változások vannak.

Jó játékot!

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 0/3 pont

Egy képi analízis során négyzetrácsra osztott mintáink egyes celláit elemezve próbálunk hibákat azonosítani. Alapvetően két cellatípust különböztetünk meg: 0-s típusút és 1-es típusút. Az input fájl a négyzetrácsra osztott mintát egy 2D-s tömb formájában tartalmazza.

Adja meg a minták szélén (legszélső oszlopokban/sorokban) lévő 0-s típusú cellák számát!

Példa:

Input:

111111110

101111111

111100111

111011101

111111101

Output:

2

A 3_1_test.txt-re adott kimenet:58

Adjuk meg a 3_1.txt-re adott kimenetet!

Segítség a beolvasáshoz (nem kötelező a használata):

```
static List<List<int>> ReadInput(string inputFile)
{
    return File.ReadLines(inputFile)
    .Select(s => s.ToCharArray()
        .Select(c => (int)c - '0')
        .ToList())
    .ToList();
}
```

Válasz

A helyes válasz:

312

Magyarázat

Végigmegyünk a mátrix elemein és ha egy '0' elemet találunk a mátrix szélén (első v. utolsó sorban/oszlopban), akkor növeljük a számlálót.

```
public static int Solve(string inputFile)
{
    var matrix = ReadInput(inputFile);
    var count = 0;
    for (int i = 0; i < matrix.Count; i++)
    {
        for (int j = 0; j < matrix[i].Count; j++)
        {
            if (matrix[i][j] == 0 && (i == 0 || i == matrix.Count - 1 || j == 0 || j == matrix[i].Count - 1))
            {
                count++;
            }
        }
    }
    return count;
}
```

2. feladat 0/9 pont

Az előző feladat folytatásaként a hibákat úgy definiáljuk, hogy azon cellák minősülnek hibásnak, amelyek 0-ás típusú cellák (egy, vagy akár több szomszédos cella, összefüggő régiót alkotva), melyeket teljesen körbezárnak 1-es típusú cellák. A körbezárás azt takarja, hogy a régióval szomszédos összes pozíción 1-es típusú cella található. A minták szélén lévő cellákat nem tekintjük hibáknak (hiszen nem tudják őket teljesen körbezárni 1-es típusú cellák). Két cellát szomszédosnak tekintünk, ha egymás alatt/felett vagy mellett helyezkednek el, illetve bármely átló mentén, csúcson keresztül érintkeznek.

Példa:

Input:

111111110

1**0**1111111

1111**00**111

111**0**11101

111111101

Output:

4

Az alábbi 0-s cellák nem hibásak (nincsenek körbezárva 1-es cellákkal):

111

101

110

A 3_2_test.txt-re adott kimenet: 25089

Adjuk meg a 3_2.txt-re adott kimenetet!

Válasz

A helyes válasz:

149174

Magyarázat

Végigmegyünk a mátrix szélein, és 0-ás helyeken egy mélységi bejárást indítunk, amely a 0-ákat átalakítja -1 értékre, illetve az 1es értékeknél megáll.

Az így kapott mátrixban igaz lesz az, hogy ahol 0 szerepel az biztosan körbe van véve 1-es értékkel. Az eredményünk a 0-ák száma ebben az új mátrixban.

```
static List<List<int>> Matrix { get; set; }
public static int Solve(string inputFile)
{
    Matrix = ReadInput(inputFile);
    for (int i = 0; i < Matrix.Count; i++)
    {
        for (int j = 0; j < Matrix[i].Count; j++)
        {
            if ((i == 0 || i == Matrix.Count - 1 || j == 0 || j == Matrix[i].Count - 1) && Matrix[i][j] == 0)
            {
                Flood(i, j);
            }
        }
```

```
    }
    var sum = Matrix.SelectMany(v => v).Count(v => v == 0);
    return sum;
}


static void Flood(int y, int x)
{
    if (Matrix[y][x] == -1) return;
    if (Matrix[y][x] == 1) return;


    Matrix[y][x] = -1;
    for (int i = -1; i <= 1; i++)
    {
        for (int j = -1; j <= 1; j++)
        {
            if (i == 0 && j == 0) continue;
            if (y + i < 0 || y + i >= Matrix.Count) continue;
            if (x + j < 0 || x + j >= Matrix[0].Count) continue;
            Flood(y + i, x + j);
        }
    }
}
```