IPAR 4.0 C# .NET ALAPOKON

6. forduló

SEMILAB

A kategória támogatója: Semilab Zrt.

RENDELKEZÉSRE ÁLLÓ IDŐ:

60:00

Ismertető a feladathoz

A gyár további folymatok elvégzéséhez kéri a segítségünket, részletes leírások a feladatokban.

Felhasznált idő: 02:07/60:00 **Elért pontszám:** 0/20

1. feladat 0/5 pont

Az alábbi kódrészlet hányféle eredményt írhat ki a képernyőre?

```
static Random random=new Random();
2 references
private static void B(int i)
{
    Thread.Sleep(random.Next(100));
    Console.WriteLine($"B{i}");
}

1 reference
private static void A()
{
    Task.Run(()=>B(1));
    Thread.Sleep(random.Next(100));
    Console.WriteLine("A");
    Thread.Sleep(random.Next(100));
    Task.Run(() => B(2));
}

0 references
private static void Main(string[] args)
{
    Task.Run(() => A());
    Thread.Sleep(random.Next(100));
    Console.WriteLine("C");
    Console.ReadLine();
}
```

Válaszok

A helyes válasz:

12

26

Magyarázat

A kódrészletben egyedül az a megkötés látható, hogy B2 előtt mindenképp meghívódik A. Minden más variáció előfordulhat a párhuzamos futtatások miatt.

4 elemünk van: A, B1, B2, C. Ezeket 4! féle sorrendet alkothatnak, viszont ezeknek a fele olyan, amikor B2 megelőzi A-t, ezért felezni kell. Megoldás: 4! / 2 = 12.

Frissítés (2021.11.29): a kódsor egyik részének a működését másképp is lehet értelmezni, ezért a 26 is elfogadható megoldás (**Console.ReadLine()** sor)

2. feladat 0/15 pont

Egy mintán defekteket keresünk, és az analízisünk lefutása után egy bináris mátrixban tároltuk, hogy az adott pozícióban találtunk defektet (1) vagy sem (0). Két defekttel jelölt pozíciót összefüggő defektnek tekintünk, ha azok vízszintes vagy függőleges irány mentén szomszédosok (átló mentén nem tekintjük őket összefüggőnek!).

Számolja meg, hogy az **6_fordulo_2_feladat.txt**-ben található bináris mátrixban hány olyan összefüggő defekt található, amely legalább 4 egységnyi defektből tevődik össze! A fájlban egy sor a tömb egy sorát jelöli, azon belül ','-vel elválasztva

szerepelnek az értékek.

Teszteléshez a *6_fordulo_2_feladat_test.txt* eredménye: 150.

Válaszok

A helyes válasz:
160

Magyarázat

A mátrix tartalmát mélységi keresések segítségével bejárva meghatározzuk az egyes defekthalmazok nagyságát.

```
//Depth first search.
private static int DFS(int[,] matrix, bool[,] visited, int y, int x)
{
    visited[y, x] = true;
    int sum = 1;
    if (x > 0 && matrix[y, x - 1] == 1 && !visited[y, x - 1])
    {
        sum += DFS(matrix, visited, y, x - 1);
    }
    if (x < matrix.GetLength(1) - 1 && matrix[y, x + 1] == 1 && !visited[y, x + 1])
    {
        sum += DFS(matrix, visited, y, x + 1);
    }
    if (y > 0 && matrix[y - 1, x] == 1 && !visited[y - 1, x])
    {
        sum += DFS(matrix, visited, y - 1, x);
    }
    if (y < matrix.GetLength(0) - 1 && matrix[y + 1, x] == 1 && !visited[y + 1, x])
    {
        sum += DFS(matrix, visited, y + 1, x);
    }
    return sum;
}


private static int CountBlobs(string inputFile)
{
    const int MINIMUM_DEFECT_SIZE = 4;

    string[] rows = File.ReadAllLines(inputFile);
    int rowCount = rows.Length;
    int colCount = rows[0].Split(',').Length;
    bool[,] visited = new bool[rowCount, colCount];
    int[,] matrix = new int[rowCount, colCount];
    for (int y = 0; y < rows.Length; y++)
    {
        int[] row = rows[y].Split(',').Select(int.Parse).ToArray();
```

```
            for (int x = 0; x < colCount; x++)
            {
                matrix[y, x] = row[x];
            }
        }

        int defectSum = 0;
        for (int y = 0; y < matrix.GetLength(0); y++)
        {
            for (int x = 0; x < matrix.GetLength(1); x++)
            {
                if (matrix[y, x] == 1 && !visited[y, x])
                {
                    int size = DFS(matrix, visited, y, x);
                    if (size >= MINIMUM_DEFECT_SIZE)
                    {
                        defectSum++;
                    }
                }
            }
        }
        return defectSum;
    }
```