C++ (ENGLISH)

3. forduló

Google

A kategória támogatója: Google

Ismertető a feladatlaphoz

Kérjük, hogy a feladatlap indítása előtt mindenképp olvasd el az alábbi útmutatót:

> Amennyiben olyan kategóriában játszol, ahol van csatolmány, de hibába ütközöl a letöltésnél, ott valószínűleg a vírusirtó korlátoz, annak ideiglenes kikapcsolása megoldhatja a problémát. (Körülbelül minden 3000. letöltésnél fordul ez elő.)

> Helyezéseket a 4. forduló után mutatunk, százalékos formában: adott kategóriában a TOP 20-40-60%-hoz tartozol.

> A feltűnően rövid idő alatt megoldott feladatlapok kizárást vonnak maguk után, bármilyen más gyanús esetben fenntartjuk a jogot a forduló érvénytelenítésére!

> A feladatot javasoljuk NEM mobilon/tableten megoldani!

This round consists of algorithmic problems. Before the timer starts, please prepare your favorite C++ IDE (C++ version 17 or above is recommended). The input will be provided in a downloadable include file (less than 100KB), and you will also find a solution code template.

You are allowed to use any content that you own or find on the internet that was published before the start of the round. However, any other form of assistance is prohibited.

Good luck!

Indítás utáni csatolmányok

1. feladat 10 pont

Wall Demolisher Robot

You have a robot in a rectangular grid. Each field in the grid can be empty or wall. The robot can move horizontally and vertically on the grid. It always starts from the top left corner and needs to reach the bottom right corner. Start and end points are always empty. You realize that sometimes there is no way for the robot to reach its destination, so you add a wall demolition tool to it.

The input for all tasks consists of 10 test cases. Each of them has:

The number of **rows** and **cols** of the grid, and

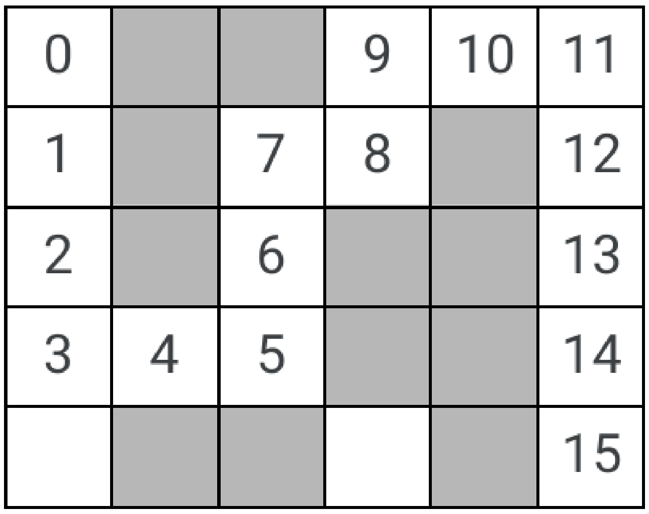
A 2D integer array **field** consisting of **rows** rows and **cols** columns with values of 0 or 1, where 0 means free space and 1 means wall/concrete.

The index of the top left corner is (0,0) and the bottom right corner is (rows-1,cols-1). You don't have to check the input for correctness. You get the input in a C++ file, wall_demolish_inputs.cpp.inc. For each task, you have to provide a space-separated list of integers, the answers for each test case (see the input style and a suggested template code below, you only have to implement the solution logic).

Constraints:

3 < **rows** < 60
3 < **cols** < 60
field[i][j] is 0 or 1

For the first run, you can't demolish walls. For each test case, you have to provide the minimum number of moves required to arrive in the opposite corner, or -1 if there is no possible way for the robot to reach the end.



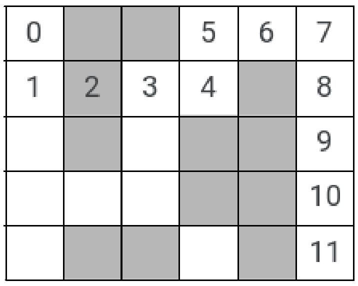
For the sample case pictured, the answer is 15. The answer for the sample inputs (test_cases below) is:

15 -1 -1

Válaszok

2. feladat 10 pont

For the second run, the robot is able to demolish a single wall piece in the way. For each test case, you have to provide the minimum number of moves required to arrive in the opposite corner, or -1 if there isn't a way with only one demolition.



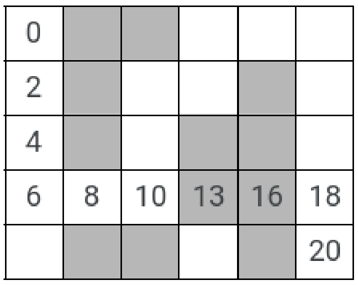
For the sample case pictured, the answer is 11. The answer for the sample inputs (test_cases below) is:

11 5 -1

Válasz

3. feladat 10 pont

For the third run, you want the robot to choose the lowest cost: a step costs 2 and a step with wall demolition costs 3. For each test case, you have to provide the minimum cost required to arrive in the opposite corner.



For the sample case pictured, the answer is 20. The answer for the sample inputs (test_cases below) is:

20 11 12

Common code snippets

The input file will be similar to this sample input file:

```cpp
#include <vector>
struct TestCase {
  size_t rows;
  size_t cols;
  std::vector<std::vector<int>> field;
};

std::vector<TestCase> test_cases = {
  /* sample test cases */
  {5,
   6,
   {{0, 1, 1, 0, 0, 0},
    {0, 1, 0, 0, 1, 0},
    {0, 1, 0, 0, 1, 0},
    {0, 0, 0, 1, 1, 0},
    {0, 1, 1, 0, 1, 0}}
  },
  {3,
   4,
   {{0, 0, 1, 0},
    {1, 1, 1, 0},
    {1, 0, 0, 0}}
   },
  {3,
   4,
   {{0, 1, 1, 0},
    {0, 1, 1, 0},
    {0, 1, 1, 0}}
   }
};
```

You can use the wall_demolish_inputs.cpp.inc file like:

```cpp
#include <iostream>
#include <vector>
#include "wall_demolish_inputs.cpp.inc"

// full copy of the test case so it can be modified locally
int task1(TestCase test_case) {
  // TODO: Task 1 code here
  return -1;
}

int task2(TestCase test_case) {
  // TODO: Task 2 code here
  return -1;
}

int task3(TestCase test_case) {
  // TODO: Task 3 code here
  return -1;
}

int main()
{
  std::cout << "Solution for Task 1:" << std::endl;
  for (TestCase &test_case : test_cases) {
    std::cout << task1(test_case) << " ";
  }
  std::cout << std::endl;
  std::cout << "Solution for Task 2:" << std::endl;
  for (TestCase &test_case : test_cases) {
    std::cout << task2(test_case) << " ";
  }
  std::cout << std::endl;
  std::cout << "Solution for Task 3:" << std::endl;
  for (TestCase &test_case : test_cases) {
    std::cout << task3(test_case) << " ";
  }
  std::cout << std::endl;
  return 0;
}
```

Válaszok

Megoldások beküldése