

Leetcode Problem Number 1970

1970. Last Day Where You Can Still Cross

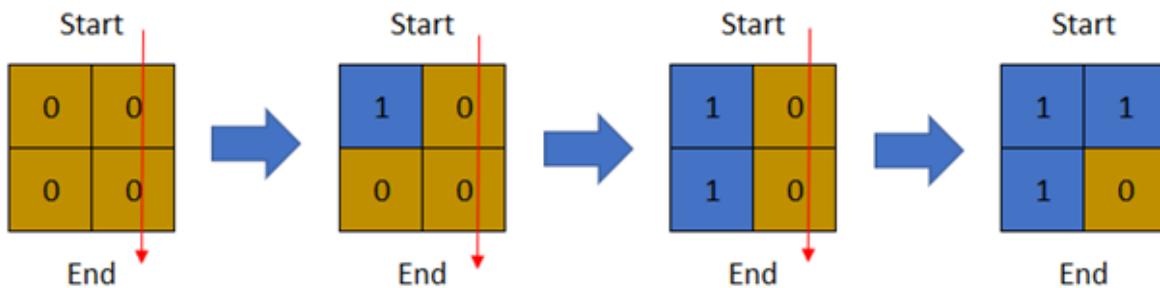
There is a **1-based** binary matrix where 0 represents land and 1 represents water. You are given integers row and col representing the number of rows and columns in the matrix, respectively.

Initially on day 0, the **entire** matrix is **land**. However, each day a new cell becomes flooded with **water**. You are given a **1-based** 2D array cells, where $\text{cells}[i] = [r_i, c_i]$ represents that on the i^{th} day, the cell on the r_i^{th} row and c_i^{th} column (**1-based** coordinates) will be covered with **water** (i.e., changed to 1).

You want to find the **last** day that it is possible to walk from the **top** to the **bottom** by only walking on land cells. You can start from **any** cell in the top row and end at **any** cell in the bottom row. You can only travel in the **four** cardinal directions (left, right, up, and down).

Return *the last day where it is possible to walk from the top to the bottom by only walking on land cells.*

Example 1:



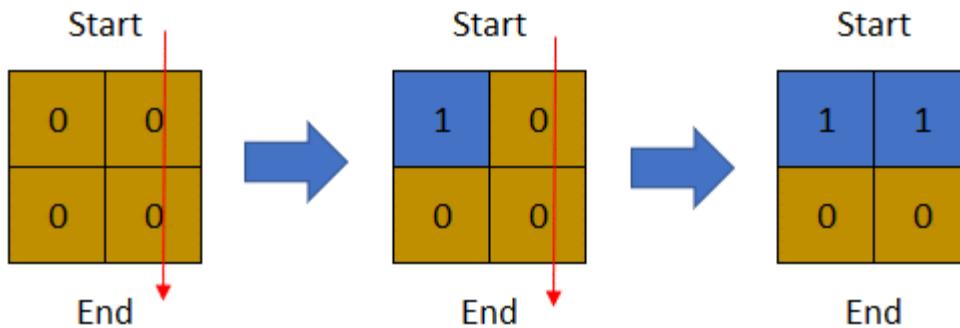
Input: row = 2, col = 2, cells = [[1,1],[2,1],[1,2],[2,2]]

Output: 2

Explanation: The above image depicts how the matrix changes each day starting from day 0.

The last day where it is possible to cross from top to bottom is on day 2.

Example 2:



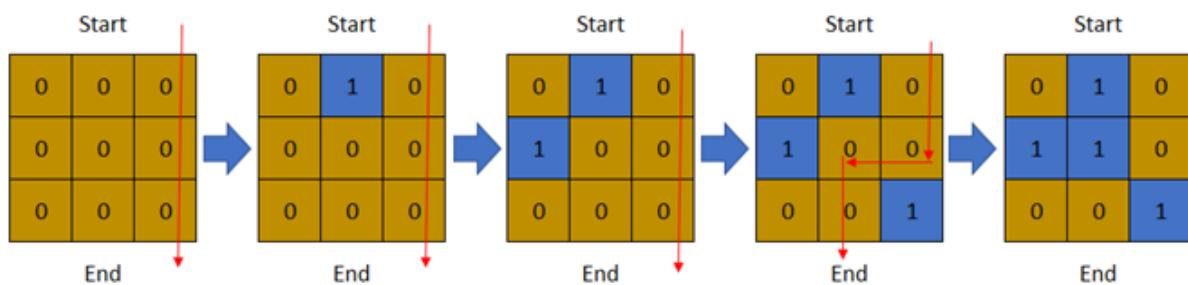
Input: row = 2, col = 2, cells = [[1,1],[1,2],[2,1],[2,2]]

Output: 1

Explanation: The above image depicts how the matrix changes each day starting from day 0.

The last day where it is possible to cross from top to bottom is on day 1.

Example 3:



Input: row = 3, col = 3, cells = [[1,2],[2,1],[3,3],[2,2],[1,1],[1,3],[2,3],[3,2],[3,1]]

Output: 3

Explanation: The above image depicts how the matrix changes each day starting from day 0.

The last day where it is possible to cross from top to bottom is on day 3.

Constraints:

- $2 \leq \text{row}, \text{col} \leq 2 * 10^4$
- $4 \leq \text{row} * \text{col} \leq 2 * 10^4$
- $\text{cells.length} == \text{row} * \text{col}$
- $1 \leq r_i \leq \text{row}$

- $1 \leq c_i \leq \text{col}$
- All the values of cells are **unique**.

Explanation

Problem का मतलब (Hindi Explanation)

आपको एक **binary matrix** दी गई है:

- 0 = जमीन (Land)
- 1 = पानी (Water)

लेकिन शुरुआत में:

- Day 0 पर पूरी matrix जमीन होती है (सब 0)
-

◆ रोज क्या होता है?

- हर दिन एक cell पानी बन जाता है।
- यह जानकारी cells array में दी गई है।
- $\text{cells}[i] = [r_i, c_i]$ का मतलब:
 - i-th दिन, row r_i और column c_i वाला cell पानी (1) बन जाएगा।

Indexing 1-based है (row और column दोनों 1 से शुरू होते हैं)।

◆ आपका लक्ष्य क्या है?

आपको यह पता करना है:

👉 आखिरी कौन सा दिन (last day) ऐसा था
जब आप ऊपर वाली row (top row) से
नीचे वाली row (bottom row) तक
सिर्फ जमीन (0) पर चलते हुए पहुँच सकते थे।

◆ चलने के नियम

- आप किसी भी cell से start कर सकते हैं जो top row में हो
 - आप किसी भी cell पर end कर सकते हैं जो bottom row में हो
 - आप केवल जमीन (0) पर चल सकते हैं
 - आप सिर्फ इन दिशाओं में चल सकते हैं:
 - ऊपर
 - नीचे
 - बायें
 - दायें
 - तिरछा चलना allowed नहीं है
-

◆ Output में क्या देना है?

- एक integer return करना है
 - जो बताए कि कौन सा आखिरी दिन था जब ऊपर से नीचे जाना possible था
-

◆ Example 1 का मतलब

row = 2, col = 2

cells = [[1,1],[2,1],[1,2],[2,2]]

- Day 0: सब जमीन → रास्ता मौजूद
- Day 1: (1,1) पानी
- Day 2: (2,1) पानी
- अभी भी रास्ता possible
- Day 3: रास्ता टूट जाता है

👉 इसलिए उत्तर = 2

◆ Simple लाइन में पूरा सवाल

"हर दिन कुछ cells पानी बनते जाते हैं।
बताओ वो आखिरी दिन कौन सा था,
जब ऊपर की row से नीचे की row तक
सिर्फ जमीन पर चलते हुए पहुँचना possible था।"

Code

Java

```
class Solution {  
  
    public int latestDayToCross(int row, int col, int[][] cells) {  
  
        DSU dsu = new DSU(row * col + 2);  
  
        int[][] grid = new int[row][col];  
  
        int[][] dirs = {{ 0, 1 }, { 0, -1 }, { 1, 0 }, { -1, 0 } };  
  
        for (int i = cells.length - 1; i >= 0; i--) {  
  
            int r = cells[i][0] - 1;  
  
            grid[r][cells[i][1]] = 1;  
  
            for (int[] dir : dirs) {  
                int nr = r + dir[0];  
                int nc = cells[i][1] + dir[1];  
  
                if (nr < 0 || nr >= row || nc < 0 || nc >= col || grid[nr][nc] == 0) {  
                    continue;  
                }  
  
                dsu.union(r * col + 1, nr * col + nc + 1);  
            }  
        }  
  
        return dsu.find(0) == dsu.find(row * col);  
    }  
}
```

```

int c = cells[i][1] - 1;
grid[r][c] = 1;

int id1 = r * col + c + 1;

for (int[] d : dirs) {
    int nr = r + d[0];
    int nc = c + d[1];
    if (nr >= 0 && nr < row && nc >= 0 && nc < col && grid[nr][nc] == 1)
        dsu.union(id1, nr * col + nc + 1);
}

if (r == 0)
    dsu.union(0, id1);

if (r == row - 1)
    dsu.union(row * col + 1, id1);

if (dsu.find(0) == dsu.find(row * col + 1))
    return i;
}

return -1;
}

class DSU {
    int[] root;
    int[] size;

    DSU(int n) {
        root = new int[n];
        size = new int[n];
        for (int i = 0; i < n; i++)
            root[i] = i;
        Arrays.fill(size, 1);
    }
}

```

```
int find(int x) {
    if (root[x] != x)
        root[x] = find(root[x]);
    return root[x];
}

void union(int x, int y) {
    int rx = find(x);
    int ry = find(y);

    if (rx == ry)
        return;

    if (size[rx] > size[ry]) {
        int tmp = rx;
        rx = ry;
        ry = tmp;
    }

    root[rx] = ry;
    size[ry] += size[rx];
}
```