2556. Disconnect Path in a Binary Matrix by at Most One Flip

My Submissions (/contest/biweekly-contest-97/problems/disconnect-path-in-a-binary-matrix-by-at-most-one-flip/submissions/)

Back to Contest (/contest/biweekly-contest-97/)

User Accepted:	1309
User Tried:	4221
Total Accepted:	1440
Total Submissions:	11440
Difficulty:	Medium

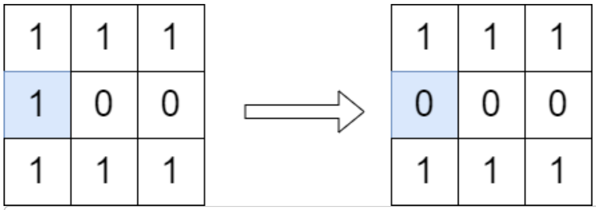
You are given a **0-indexed** `m x n` **binary** matrix `grid`. You can move from a cell `(row, col)` to any of the cells `(row + 1, col)` or `(row, col + 1)` that has the value `1`. The matrix is **disconnected** if there is no path from `(0, 0)` to `(m - 1, n - 1)`.

You can flip the value of **at most one** (possibly none) cell. You **cannot flip** the cells `(0, 0)` and `(m - 1, n - 1)`.

Return `true` *if it is possible to make the matrix disconnect or* `false` *otherwise*.

Note that flipping a cell changes its value from `0` to `1` or from `1` to `0`.

Example 1:

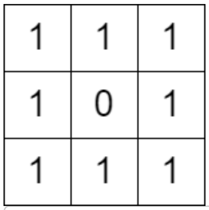


1	1	1
1	0	0
1	1	1

1	1	1
0	0	0
1	1	1

```
Input: grid = [[1,1,1],[1,0,0],[1,1,1]]
Output: true
Explanation: We can change the cell shown in the diagram above. There is no path from (0, 0) to (2, 2) in the resulting grid.
```

Example 2:



1	1	1
1	0	1
1	1	1

```
Input: grid = [[1,1,1],[1,0,1],[1,1,1]]
Output: false
Explanation: It is not possible to change at most one cell such that there is not path from (0, 0) to (2, 2).
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 1000`
- `1 <= m * n <=` 10^5
- `grid[i][j]` is either `0` or `1`.
- `grid[0][0] == grid[m - 1][n - 1] == 1`

Discuss (https://leetcode.com/problems/disconnect-path-in-a-binary-matrix-by-at-most-one-flip/discuss)

JavaScript

```
const isPossibleToCutPath = (g) => {
    let n = g.length, m = g[0].length, diagonals = Array(n + m - 1).fill(0);
    for (let i = 0; i < n; i++) {  // forward
        for (let j = 0; j < m; j++) {
            if (i == 0 && j == 0 || g[i][j] == 0) continue;
            if ((i == 0 || g[i - 1][j] == 0) && (j == 0 || g[i][j - 1] == 0)) g[i][j] = 0;
        }
    }
    for (let i = n - 1; i >= 0; i--) { // backward
        for (let j = m - 1; j >= 0; j--) {
            if (i == n - 1 && j == m - 1 || g[i][j] == 0) continue;
            if ((i == n - 1 || g[i + 1][j] == 0) && (j == m - 1 || g[i][j + 1] == 0)) g[i][j] = 0;
        }
    }
    // diagnonal count
    for (let i = 0; i < n; i++) {
        for (let j = 0; j < m; j++) diagonals[i + j] += g[i][j];
    }
    for (let i = 1; i < n + m - 2; i++) {
        if (diagonals[i] <= 1) return true;
    }
    return false;
};
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/894503785/)

More Details (/submissions/detail/894503785/)

Share your acceptance!

5

Run Code: Finished

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

United States (/region)