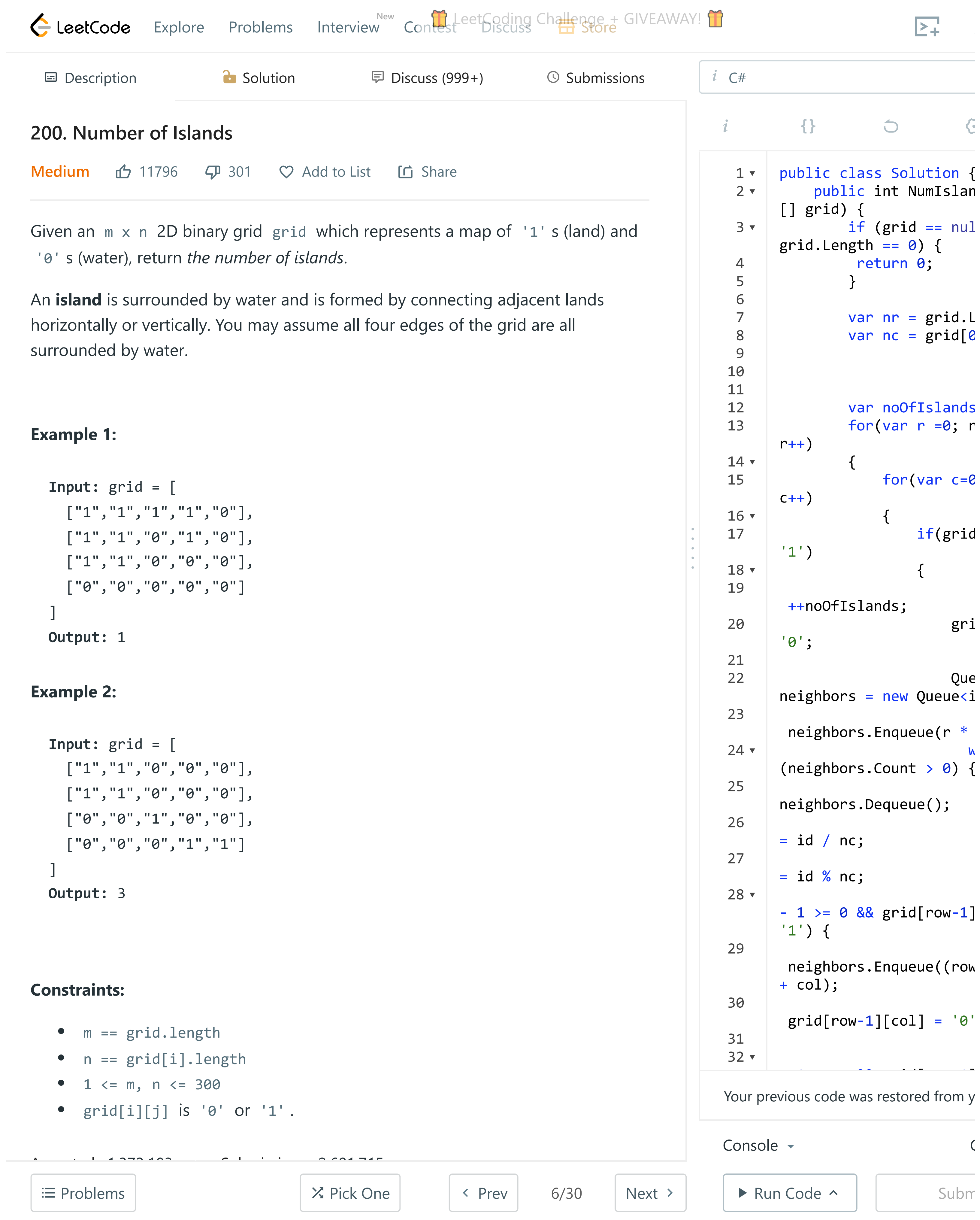

200. Number of Islands

Medium 11796 301 Add to List Share

Given an `m x n` 2D binary grid `grid` which represents a map of `'1'` s (land) and `'0'` s (water), return *the number of islands*.

An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:

```
Input: grid = [
  ["1","1","1","1","0"],
  ["1","1","0","1","0"],
  ["1","1","0","0","0"],
  ["0","0","0","0","0"]
]
Output: 1
```

Example 2:

```
Input: grid = [
  ["1","1","0","0","0"],
  ["1","1","0","0","0"],
  ["0","0","1","0","0"],
  ["0","0","0","1","1"]
]
Output: 3
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 300`
- `grid[i][j]` is `'0'` or `'1'` .

Accepted 1,372,103 Submissions 2,601,715

```
public class Solution {
    public int NumIslan
[] grid) {
        if (grid == nul
grid.Length == 0) {
         return 0;
        }

        var nr = grid.L
        var nc = grid[0



        var noOfIslands
        for(var r =0; r
r++)
        {
            for(var c=0
c++)
            {
                if(grid
'1')
                {

 ++noOfIslands;
                        gri
'0';


                        Que
neighbors = new Queue<i

 neighbors.Enqueue(r *
                          w
(neighbors.Count > 0) {

neighbors.Dequeue();

= id / nc;

= id % nc;

- 1 >= 0 && grid[row-1]
'1') {

 neighbors.Enqueue((row
+ col);

 grid[row-1][col] = '0'
```

Your previous code was restored from y