Find The Number Of Islands

solved by	Senan
Platform	GeeksForGeeks
difficulty	Medium
tags	BFS
language	C++
solved on	@06/10/2024
link	https://www.geeksforgeeks.org/problems/find-the-number-of-islands/1
Completion	☑

Intuition

The problem of counting islands in a grid can be approached using Breadth-First Search (BFS). Each island is a connected component of `1`s, and BFS can explore the entire connected region starting from any unvisited `1`. Every time BFS starts from a new `1`, it indicates a new island.

Approach

1. Iterate through the grid and initiate a BFS from each unvisited `1`.
2. Mark all connected `1`s as visited using BFS, which explores the grid in 8 possible directions (up, down, left, right, and diagonals).
3. Each BFS initiation counts as finding a new island, so increment the island count every time a BFS starts.

Complexity

Time Complexity:

- Traversing all the cells in the grid takes O(*n * m*), where **n** is the number of rows and **m** is the number of columns in the grid.
- BFS explores each cell at most once, so the time complexity of BFS is also O(*n * m*).

Thus, the overall time complexity is *O(n * m)*.

Space Complexity:

- The space complexity is O(*n * m*)due to the `visited` array and the queue used in BFS.

Code

```
class Solution {
  public:
    void bfs(int i, int j, vector<vector<char>> &grid, vector<vector<int>> &visited, i
nt n, int m) {
        queue<pair<int,int>> q;
        q.push({i, j});
        visited[i][j] = 1;

        while(!q.empty()) {
            int u = q.front().first;
```

```
            int v = q.front().second;
            q.pop();

            for(int r = -1; r < 2; r++) {
                for(int s = -1; s < 2; s++) {
                    int x = u + r;
                    int y = v + s;

                    if(x >= 0 && x < n && y >= 0 && y < m && grid[x][y] == '1' && !vis
ited[x][y]) {
                        q.push({x, y});
                        visited[x][y] = 1;
                    }
                }
            }
        }
    }

    int numIslands(vector<vector<char>>& grid) {
        int n = grid.size();
        int m = grid[0].size();
        int island = 0;

        vector<vector<int>> visited(n, vector<int>(m, 0));

        for(int i = 0; i < n; i++) {
            for(int j = 0; j < m; j++) {
                if(grid[i][j] == '1' && !visited[i][j]) {
                    island++;
                    bfs(i, j, grid, visited, n, m);
                }
            }
        }

        return island;
    }
};
```