AlgoExpert   Quad Layout   C++   12px   Sublime   Monokai   00:00:0

Our Solution(s)   Run Code

Solution 1

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
using namespace std;

void traverseNode(int i, int j, vector<vector<int>> matrix,
                  vector<vector<int>> *visited, vector<int> *sizes);
vector<vector<int>> getUnvisitedNeighbors(int i, int j,
                                          vector<vector<int>> matrix,
                                          vector<vector<int>> visited);

// O(wh) time | O(wh) space
vector<int> riverSizes(vector<vector<int>> matrix) {
  vector<int> sizes = {};
  vector<vector<int>> visited(matrix.size(),
                              vector<int>(matrix[0].size(), false));
  for (int i = 0; i < matrix.size(); i++) {
    for (int j = 0; j < matrix[i].size(); j++) {
      if (visited[i][j]) {
        continue;
      }
      traverseNode(i, j, matrix, &visited, &sizes);
    }
  }
  return sizes;
}

void traverseNode(int i, int j, vector<vector<int>> matrix,
                  vector<vector<int>> *visited, vector<int> *sizes) {
  int currentRiverSize = 0;
  vector<vector<int>> nodesToExplore{{i, j}};
  while (nodesToExplore.size() != 0) {
    vector<int> currentNode = nodesToExplore.back();
    nodesToExplore.pop_back();
    i = currentNode[0];
    j = currentNode[1];
    if (visited->at(i)[j]) {
      continue;
    }
    visited->at(i)[j] = true;
    if (matrix[i][j] == 0) {
      continue;
    }
    currentRiverSize++;
    vector<vector<int>> unvisitedNeighbors =
        getUnvisitedNeighbors(i, j, matrix, *visited);
    for (vector<int> neighbor : unvisitedNeighbors) {
      nodesToExplore.push_back(neighbor);
    }
  }
  if (currentRiverSize > 0) {
    sizes->push_back(currentRiverSize);
  }
}

vector<vector<int>> getUnvisitedNeighbors(int i, int j,
                                          vector<vector<int>> matrix,
                                          vector<vector<int>> visited) {
  vector<vector<int>> unvisitedNeighbors{};
  if (i > 0 && !visited[i - 1][j]) {
    unvisitedNeighbors.push_back({i - 1, j});
  }
  if (i < matrix.size() - 1 && !visited[i + 1][j]) {
    unvisitedNeighbors.push_back({i + 1, j});
  }
  if (j > 0 && !visited[i][j - 1]) {
    unvisitedNeighbors.push_back({i, j - 1});
  }
  if (j < matrix[0].size() - 1 && !visited[i][j + 1]) {
    unvisitedNeighbors.push_back({i, j + 1});
  }
  return unvisitedNeighbors;
}
```

Your Solutions   Run Code

Solution 1   Solution 2   Solution 3

```cpp
#include <vector>
using namespace std;

vector<int> riverSizes(vector<vector<int>> matrix) {
  // Write your code here.
  return {};
}
```

Custom Output   Raw Output   Submit Code

Run or submit code when you're ready.