AlgoExpert Quad Layout Java 12px Sublime Monokai 00:00:0

Our Solution(s) Run Code

Solution 1

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

import java.util.*;

class Program {
  // O(wh) time | O(wh) space
  public static List<Integer> riverSizes(int[][] matrix) {
    List<Integer> sizes = new ArrayList<Integer>();
    boolean[][] visited = new boolean[matrix.length][matrix[0].length];
    for (int i = 0; i < matrix.length; i++) {
      for (int j = 0; j < matrix[0].length; j++) {
        if (visited[i][j]) {
          continue;
        }
        traverseNode(i, j, matrix, visited, sizes);
      }
    }
    return sizes;
  }

  public static void traverseNode(
      int i, int j, int[][] matrix, boolean[][] visited, List<Integer> sizes) {
    int currentRiverSize = 0;
    List<Integer[]> nodesToExplore = new ArrayList<Integer[]>();
    nodesToExplore.add(new Integer[] {i, j});
    while (!nodesToExplore.isEmpty()) {
      Integer[] currentNode = nodesToExplore.get(nodesToExplore.size() - 1);
      nodesToExplore.remove(nodesToExplore.size() - 1);
      i = currentNode[0];
      j = currentNode[1];
      if (visited[i][j]) {
        continue;
      }
      visited[i][j] = true;
      if (matrix[i][j] == 0) {
        continue;
      }
      currentRiverSize++;
      List<Integer[]> unvisitedNeighbors = getUnvisitedNeighbors(i, j, matrix, vi
      for (Integer[] neighbor : unvisitedNeighbors) {
        nodesToExplore.add(neighbor);
      }
    }
    if (currentRiverSize > 0) {
      sizes.add(currentRiverSize);
    }
  }

  public static List<Integer[]> getUnvisitedNeighbors(
      int i, int j, int[][] matrix, boolean[][] visited) {
    List<Integer[]> unvisitedNeighbors = new ArrayList<Integer[]>();
    if (i > 0 && !visited[i - 1][j]) {
      unvisitedNeighbors.add(new Integer[] {i - 1, j});
    }
    if (i < matrix.length - 1 && !visited[i + 1][j]) {
      unvisitedNeighbors.add(new Integer[] {i + 1, j});
    }
    if (j > 0 && !visited[i][j - 1]) {
      unvisitedNeighbors.add(new Integer[] {i, j - 1});
    }
    if (j < matrix[0].length - 1 && !visited[i][j + 1]) {
      unvisitedNeighbors.add(new Integer[] {i, j + 1});
    }
    return unvisitedNeighbors;
  }
}
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```java
import java.util.*;

class Program {
  public static List<Integer> riverSizes(int[][] matrix) {
    // Write your code here.
    return null;
  }
}
```

Custom Output Raw Output Submit Code

Run or submit code when you're ready.