Our Solution(s) Run Code

Solution 1

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

// O(wh) time | O(wh) space
function riverSizes(matrix) {
  const sizes = [];
  const visited = matrix.map(row => row.map(value => false));
  for (let i = 0; i < matrix.length; i++) {
    for (let j = 0; j < matrix[i].length; j++) {
      if (visited[i][j]) continue;
      traverseNode(i, j, matrix, visited, sizes);
    }
  }
  return sizes;
}

function traverseNode(i, j, matrix, visited, sizes) {
  let currentRiverSize = 0;
  const nodesToExplore = [[i, j]];
  while (nodesToExplore.length) {
    const currentNode = nodesToExplore.pop();
    i = currentNode[0];
    j = currentNode[1];
    if (visited[i][j]) continue;
    visited[i][j] = true;
    if (matrix[i][j] === 0) continue;
    currentRiverSize++;
    const unvisitedNeighbors = getUnvisitedNeighbors(i, j, matrix, visited);
    for (const neighbor of unvisitedNeighbors) {
      nodesToExplore.push(neighbor);
    }
  }
  if (currentRiverSize > 0) sizes.push(currentRiverSize);
}

function getUnvisitedNeighbors(i, j, matrix, visited) {
  const unvisitedNeighbors = [];
  if (i > 0 && !visited[i - 1][j]) unvisitedNeighbors.push([i - 1, j]);
  if (i < matrix.length - 1 && !visited[i + 1][j]) unvisitedNeighbors.push([i + 1
  if (j > 0 && !visited[i][j - 1]) unvisitedNeighbors.push([i, j - 1]);
  if (j < matrix[0].length - 1 && !visited[i][j + 1]) unvisitedNeighbors.push([i,
  return unvisitedNeighbors;
}

exports.riverSizes = riverSizes;

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```
function riverSizes(matrix) {
  // Write your code here.
}

// Do not edit the line below.
exports.riverSizes = riverSizes;

```

Custom Output Raw Output Submit Code

Run or submit code when you're ready.