Question: _

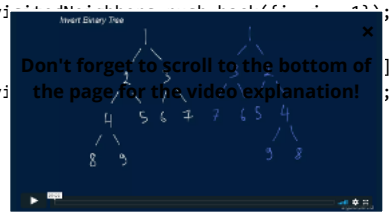
Input: Your Solution Our Solution Run Code

```
  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]) {
```

```
  unvi
}
if (j                                  ][j + 1]) {
  unvi                                         ;
}
  .
```



Don't forget to scroll to the bottom of the page for the video explanation!

Video Explanation

Go to Conceptual Overview

Go to Code Walkthrough

Questions List (/questions)