**Name:\_\_\_\_\_\_\_\_\_\_\_\_Shuqing Ye\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_UCI NET ID:\_\_\_\_\_\_\_shuqiny2\_\_\_\_\_\_\_\_**

|  |  |
| --- | --- |
| Test cases (including the edge cases):  Input:[[1,2,4],[0,3],[0,3,4],[1,2],[1,2],[6],[5,7],[6],[]], 9  Output: 3  Input:[[]], 0  Output: 0  Input:[[], [], [], [], []], 4  Output: 4  Input: [[1,2], [0,2], [0,1]], 3  Output: 1 | time complexity:  O(N + E) // E is the number of edges  space complexity:  O(N) + call stack |

|  |  |
| --- | --- |
| public int countConnectComponents (List<List<Integer>> adjList, int N) {  int cnt = 0;  boolean[] visited = new boolean[N];  for (int i = 0; i < N; i++) {  if (!visited[i]) {  cnt++;  dfs(adjList, visited, i);  }  }  return cnt;  }  // use depth-first-search to travel all the nodes that have a path to u  void dfs(List<List<Integer>> adjList, boolean[] visited, int u) {  visited[u] = true;  for (int v : adjList.get(u)) {  if (!visited[v])  dfs(adjList, visited, v);  }  } |  |