Question: https://leetcode.com/problems/number-of-provinces/

Lets simplify the question!

The question is about a graph annd to check if it is disconnnected or not? If disconnnected then return the number of sub graphs.

So first we create a adjacency list, and travel the graph through DFS.

Since we nneed to check number of sub graphs considering the fact that the graph is disconnnected we start DFs from each node, if it is not visited, so the number off time we perform DFS is the annswer.

Code:

class Solution {

public int findCircleNum(int[][] isConnected) {

List<List<Integer>> adjList = new ArrayList<>();

int count=0;

int vis[] = new int[isConnected.length];

Arrays.fill(vis, 0);

for(int i=0; i<isConnected.length; i++){

List<Integer> adjNodes = new ArrayList<>();

for(int j=0; j<isConnected[i].length; j++){

if(isConnected[i][j]==1 && i!=j) adjNodes.add(j);

}

adjList.add(adjNodes);

}

// System.out.println(adjList);

for(int i=0; i<isConnected.length; i++){

if(vis[i]!=1){

count++;

dfs(i, adjList, vis);

}

}

return count;

}

public void dfs(int node, List<List<Integer>> adjList, int[] vis){

vis[node]=1;

if(adjList.get(node).size()==0) return;

for(int n : adjList.get(node)){

if(vis[n]!=1) dfs(n, adjList, vis);

}

return;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>