MICROSOFT PREPARATION

DAY : 07

Maximal Network Rank :

Problem Link : https://leetcode.com/problems/maximal-network-rank/

Test Cases Passed : 83 / 83

Time Used : 12.24

Difficulty Level : MEDIUM

Approach Used :

- Create an adjacency list of the graph
- Make a vector that contains the outdegree of elements
- Initialize maxEle with -1
- Traverse for all elements in vector :
    - Initialize sum with -1
    - Traverse for other pair of elements in vector :
        - Check if the elements are adjacent :
            - Calculate sum as outdegree1+outdegree2-1
        - Else :
            - Calculate sum as outdegree1+outdegree2
    - Check if sum is greater than maxEle :
        - Update maxEle with sum
- Return maxEle

Solution :

```
int maximalNetworkRank(int n, vector<vector<int>>& roads) {
        /*

        Approach coming to my mind is that for every node we can calculate the
number of out degrees as the roads are bi-directional.

        And if we are finding for a pair only then we can eliminate that particular
```

```
pair once

       eg.

       0 - 2
       1 - 3

       2+3 = 5

       0 has adjacent 1 hence 5-1 = 4


       1 - 3
       2 - 1
       3+1 = 4
       4-1 = 3

       */

       // creating an adjacency list

       vector<vector<int>> adj(n);

       for(int i=0;i<roads.size();i++)
       {
           adj[roads[i][0]].push_back(roads[i][1]);
           adj[roads[i][1]].push_back(roads[i][0]);
       }

       // vector containing outdegree
       vector<int> outdegree(n,0);

       for(int i=0;i<n;i++)
       {
           outdegree[i] = adj[i].size();
       }
       int maxEle = -1;

       for(int i=0;i<n;i++)
       {
           for(int j=i+1;j<n;j++)
           {
               int res = -1;
               auto it = find(adj[i].begin(),adj[i].end(),j);
               if(it!=adj[i].end())
               {
                   res = outdegree[i]+outdegree[j]-1;
```

```
            }
            else
            {
                res = outdegree[i]+outdegree[j];
            }
            if(res>maxEle)
            {
                maxEle = res;
            }
        }
    }
    return maxEle;
}
/*
    0----------1--------2-----------3
                        |
                        4


    5----------6
    |
    7
*/
```