MEDIUM

Number of Provinces | Disjoint Set

Intuition

Find the number of connected components, we have already did this now just we will be doing this using the Disjoint Set data structure

Eg.

0	1	0	0	0	0	0
1	0	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	1	0

There are 3 components

1->2->3

4->5

6->7

We can return 3 as it is the number of total ultimate parents

findUP(1) = 1 // BOSS
findUP(2) = 1
findUP(3) = 1
findUP(4) = 4 // BOSS
findUP(5) = 4
findUP(6) = 6 // BOSS
findUP(7) = 6

Approach

- Create a disjoint set of number of nodes
- Traverse for all components :
    - Insert edge for disjoint set
- Create a variable to count the number of provinces
- Traverse for all nodes :
    - Check if node parent is node itself :
        - Increment count
- Return count

Function Code

```
class DisjointSet
{
    // creating a parent and rank vector
    vector<int> parent;
    vector<int> rank;
    public:
    // creating a constructor
    DisjointSet(int n)
    {
        rank.resize(n,0);
        parent.resize(n);
        for(int i=0;i<n;i++)
        {
            parent[i]=i;
        }
    }
    // creating findParent function
    int findParent(int node)
    {
        if(parent[node] == node)
        {
            return node;
        }
        return parent[node] = findParent(parent[node]);
    }
    // creating union by rank
    void unionbyrank(int u,int v)
    {
        int pu = findParent(u);
        int pv = findParent(v);
        // same
        if(pu==pv)return;
        // same rank
        if(rank[pu]==rank[pv])
        {
            parent[pv] = pu;
        }
        else if(rank[pu]>rank[pv])
        {
            parent[pv] = pu;
            rank[pu]+=1;
        }
        else
        {
            parent[pu] = pv;
            rank[pv]+=1;
        }
    }
};
class Solution {
  public:
    int numProvinces(vector<vector<int>> adj, int n) {
        DisjointSet ds(n);
        // iterate for all components and make Disjoint set if there is a
path
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                if(adj[i][j]==1)
                {
                    ds.unionbyrank(i,j);
                }
            }
        }
        // creating a count variable to store the number of provinces
int count = 0;
        for(int i=0;i<n;i++)
        {
        // checking if parent is node itself ie.its the boss
            if(ds.findParent(i)==i)
            {
                count+=1;
            }
        }
        //return the count of provinces
        return count;

}
```

Time Complexity

O(N^2)