HARD

Merging Details | Accounts Merge

Intuition

We are given a lot of arrays in which we have names followed by email accounts, the questions states to merge the similar accounts.

Eg.

[["John","johnsmith@mail.com","john_newyork@mail.com"],
["John","johnsmith@mail.com","john00@mail.com"],
["Mary","mary@mail.com"],
["John","johnnybravo@mail.com"]]

This can be combined to form

[["John","john00@mail.com","john_newyork@mail.com", "johnsmith@mail.com"],
["Mary","mary@mail.com"],
["John","johnnybravo@mail.com"]]

This is how we have to merge the accounts, after merging the email IDs must be in sorted order.

We can break the problem into,

Every email is connected to the next email. And will be followed by connection to other emails to. Disjoint Set will give us option to solve these type of problems.

We will assign indexes to entire array, indexes represent the names

Initial configurations would be

Parent

0 1 2
0 1 2

We can create a map to mark the email belongs to particular index

0 - ["john00@mail.com"]
1 - ["john_newyork@mail.com", "johnsmith@mail.com"]

2 - ["mary@mail.com"]

So when we came across johnsmit@gmail.com again then we will see that it is already present and then we will perform its union

Eg.

0<-1

2

This is how we will be performing the merging.

Approach

- Calculate the number of accounts
- Create a disjoint set of n records
- Create an unordered map to mark the emails and nodes
- Traverse for all accounts :
    - Traverse through all emails :
        - Check if mail is not in map :
            - Insert the mail in map
        - Else :
            - Add the node with the map
- Create a vector of vector of strings to store ans
- Create a vector of vector of strings having merged mails having size of number of records
- Traverse through the map :
    - Find mail
    - Find parent of the mail
    - Insert the mail into the merged mail
- Traverse for all accounts :
    - Check if there is no merged mail :
        - Continue
    - Sort the merged mail list for the ith node
    - Extract the name under which merged mail needs to be added
    - In the answer vector insert the node mail with the merged mails
- Return the ans vector

Function Code

```
#include <vector>
#include <string>
#include <unordered_map>
```

```cpp
#include <algorithm>

class DisjointSet
{
public:
    std::vector<int> parent;
    std::vector<int> rank;

    DisjointSet(int n)
    {
        rank.resize(n, 0);
        parent.resize(n);
        for (int i = 0; i < n; i++)
        {
            parent[i] = i;
        }
    }

    int findParent(int node)
    {
        if (parent[node] == node)
        {
            return node;
        }
        return parent[node] = findParent(parent[node]);
    }

    void unionByRank(int u, int v)
    {
        int upu = findParent(u);
        int upv = findParent(v);
        if (upu == upv)
        {
            return;
        }
        if (rank[upu] < rank[upv])
        {
            parent[upu] = upv;
            rank[upv] += 1;
        }
        else if (rank[upu] == rank[upv])
        {
            parent[upv] = upu;
```

```
            rank[upu] += 1;
        }
        else
        {
            parent[upv] = upu;
            rank[upu] += 1;
        }
    }
};

class Solution {
public:
    std::vector<std::vector<std::string>>
accountsMerge(std::vector<std::vector<std::string>> &accounts) {
        int n = accounts.size();
        DisjointSet ds(n);
        std::unordered_map<std::string, int> mapMailNode;

        for (int i = 0; i < n; i++)
        {
            for (int j = 1; j < accounts[i].size(); j++)
            {
                std::string mail = accounts[i][j];
                if (mapMailNode.find(mail) == mapMailNode.end())
                {
                    mapMailNode[mail] = i;
                }
                else
                {
                    ds.unionByRank(i, mapMailNode[mail]);
                }
            }
        }

        std::vector<std::vector<std::string>> ans;
        std::vector<std::vector<std::string>> mergedMail(n);

        for (auto it : mapMailNode)
        {
            std::string mail = it.first;
            int node = ds.findParent(it.second);
            mergedMail[node].push_back(mail);
        }
```

```
        for (int i = 0; i < n; i++)
        {
            if (mergedMail[i].empty())
                continue;
            std::sort(mergedMail[i].begin(), mergedMail[i].end());
            std::vector<std::string> temp;
            temp.push_back(accounts[i][0]);
            for (auto it : mergedMail[i])
            {
                temp.push_back(it);
            }
            ans.push_back(temp);
        }

        return ans;
    }
};
```

Time Complexity

O(N*log(N))