Problem 721: Accounts Merge

Problem Information

Difficulty: Medium
Acceptance Rate: 60.51%
Paid Only: No
Tags: Array, Hash Table, String, Depth-First Search, Breadth-First Search, Union Find, Sorting

Problem Description

Given a list of `accounts` where each element `accounts[i]` is a list of strings, where the first element `accounts[i][0]` is a name, and the rest of the elements are **emails** representing emails of the account.

Now, we would like to merge these accounts. Two accounts definitely belong to the same person if there is some common email to both accounts. Note that even if two accounts have the same name, they may belong to different people as people could have the same name. A person can have any number of accounts initially, but all of their accounts definitely have the same name.

After merging the accounts, return the accounts in the following format: the first element of each account is the name, and the rest of the elements are emails **in sorted order**. The accounts themselves can be returned in **any order**.

Example 1:

Input: accounts = [["John","johnsmith@mail.com","john_newyork@mail.com"],["John","johnsmith@mail.com","john00@mail.com"],["Mary","mary@mail.com"],["John","johnnybravo@mail.com"]] **Output:** [["John","john00@mail.com","john_newyork@mail.com","johnsmith@mail.com"],["Mary","mary@mail.com"],["John","johnnybravo@mail.com"]] **Explanation:** The first and second John's are the same person as they have the common email "johnsmith@mail.com". The third John and Mary are different people as none of their email addresses are used by other accounts. We could return these lists in any order, for example the answer [['Mary', 'mary@mail.com'], ['John', 'johnnybravo@mail.com'], ['John', 'john00@mail.com', 'john_newyork@mail.com', 'johnsmith@mail.com']] would still be accepted.

Example 2:

Input: accounts = [["Gabe","Gabe0@m.co","Gabe3@m.co","Gabe1@m.co"],["Kevin","Kevin3@m.co","Kevin5@m.co","Kevin0@m.co"],["Ethan","Ethan5@m.co","Ethan4@m.co","Ethan0@m.co"],["Hanzo","Hanzo3@m.co","Hanzo1@m.co","Hanzo0@m.co"],["Fern","Fern5@m.co","Fern1@m.co","Fern0@m.co"]] **Output:** [["Ethan","Ethan0@m.co","Ethan4@m.co","Ethan5@m.co"],["Gabe","Gabe0@m.co","Gabe1@m.co","Gabe3@m.co"],["Hanzo","Hanzo0@m.co","Hanzo1@m.co","Hanzo3@m.co"],["Kevin","Kevin0@m.co","Kevin3@m.co","Kevin5@m.co"],["Fern","Fern0@m.co","Fern1@m.co","Fern5@m.co"]]

Constraints:

* `1 <= accounts.length <= 1000` * `2 <= accounts[i].length <= 10` * `1 <= accounts[i][j].length <= 30` * `accounts[i][0]` consists of English letters. * `accounts[i][j] (for j > 0)` is a valid email.

Code Snippets

C++:

```
class Solution {
public:
vector<vector<string>> accountsMerge(vector<vector<string>>& accounts) {

}
};
```

Java:

```
class Solution {
public List<List<String>> accountsMerge(List<List<String>> accounts) {

}
}
```

Python3:

```
class Solution:
def accountsMerge(self, accounts: List[List[str]]) -> List[List[str]]:
```