721. Accounts Merge

Given a list accounts, 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 email that is common 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", "john00@mail.com"], ["John", "johnnybravo@
mail.com"], ["John", "johnsmith@mail.com", "john_newyork@mail.com"], ["Mary", "mary@m
ail.com"]]
Output: [["John", 'john00@mail.com', 'john_newyork@mail.com', 'johnsmith@mail.com'],
 ["John", "johnnybravo@mail.com"], ["Mary", "mary@mail.com"]]
Explanation:
The first and third John's are the same person as they have the common email "johnsmi
th@mail.com".
The second John and Mary are different people as none of their email addresses are us
ed 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 sti
ll be accepted.
```

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

        // union-find
        Map<String, String> owner = new HashMap<>();
        Map<String, String> parents = new HashMap<>();
        Map<String, TreeSet<String>> unions = new HashMap<>();

        for (List<String> a : accounts) {
```

```
            for (int i = 1; i < a.size(); i++) {
                parents.put(a.get(i), a.get(i));
                owner.put(a.get(i), a.get(0));
            }
        }

        for (List<String> a : accounts) {
            String p = find(a.get(1), parents);
            for (int i = 2; i < a.size(); i++) {
                parents.put(find(a.get(i), parents), p);
            }
        }

        for (List<String> a : accounts) {
            String p = find(a.get(1), parents);
            if (!unions.containsKey(p))
                unions.put(p, new TreeSet<>());
            for(int i = 1; i < a.size(); i++) {
                unions.get(p).add(a.get(i));
            }
        }

        List<List<String>> res = new ArrayList<>();
        for (String p : unions.keySet()) {
            List<String> emails = new ArrayList(unions.get(p));
            emails.add(0, owner.get(p));
            res.add(emails);
        }

        return res;
    }

    private String find(String s, Map<String, String> p) {
        return p.get(s) == s ? s : find(p.get(s), p);
    }
}
```