Description | Editorial | Solutions | Submissions

2416. Sum of Prefix Scores of Strings

Solved

Hard | Topics | Companies | Hint

You are given an array `words` of size `n` consisting of **non-empty** strings.

We define the **score** of a string `word` as the **number** of strings `words[i]` such that `word` is a **prefix** of `words[i]`.

- For example, if `words = ["a", "ab", "abc", "cab"]`, then the score of `"ab"` is `2`, since `"ab"` is a prefix of both `"ab"` and `"abc"`.

Return *an array* `answer` *of size* `n` *where* `answer[i]` *is the* ***sum*** *of scores of every* ***non-empty*** *prefix of* `words[i]`.

Note that a string is considered as a prefix of itself.

Example 1:

```
Input: words = ["abc","ab","bc","b"]
Output: [5,4,3,2]
Explanation: The answer for each string is the following:
- "abc" has 3 prefixes: "a", "ab", and "abc".
- There are 2 strings with the prefix "a", 2 strings with the prefix "ab", and 1 string with the prefix "abc".
The total is answer[0] = 2 + 2 + 1 = 5.
- "ab" has 2 prefixes: "a" and "ab".
```

```cpp
class Node{
    public:
    Node* links[26];
    int precnt;
    public:
    Node(){
        for(int i=0;i<26;i++){
            links[i]=NULL;
        }
        precnt=0;
    }
    bool have(char ch){
        return links[ch-'a'] != NULL;
    }
    void put(char ch,Node* node){
        links[ch-'a']=node;
    }
    Node* get(char ch){
        return links[ch-'a'];
    }
};
class Trie{
    Node* root;
    public:
    Trie(){
        root=new Node();
    }
    void insert(string &s){
        Node* curr=root;
        for(int i=0;i<s.size();i++){
            if(!curr->have(s[i])){
                curr->put(s[i],new Node());
            }
            curr=curr->get(s[i]);
            curr->precnt++;
        }
    }

    int count(string &s){
         int cnt=0;
         string temp="";
         Node* curr=root;
         for(int i=0;i<s.size();i++){
            if(!curr->have(s[i])){
                return cnt;
            }
            curr=curr->get(s[i]);
            cnt+=curr->precnt;
         }
         return cnt;
    }
};

class Solution {
public:
    vector<int> sumPrefixScores(vector<string>& words) {
        Trie trie;
        for(auto &s:words){
            trie.insert(s);
        }
        vector<int>ans;
        for(auto &i:words){
            ans.push_back(trie.count(i));
        }
        return ans;
    }
```