**Shortest Unique prefix for every word**

**Hard**Accuracy: 78.75% Submissions: 5258 Points: 8

Given an array of words, find all shortest unique prefixes to represent each word in the given array. Assume that no word is prefix of another.

**Example 1:**

**Input:**

N = 4

arr[] = {"zebra", "dog", "duck", "dove"}

**Output:** z dog du dov

**Explanation:**

z => zebra

dog => dog

duck => du

dove => dov

**Example 2:**

**Input:**

N = 3

arr[] = {"geeksgeeks", "geeksquiz",

"geeksforgeeks"};

**Output:** geeksg geeksq geeksf

**Explanation:**

geeksgeeks => geeksg

geeksquiz => geeksq

geeksforgeeks => geeksf

**Your task:**

You don't have to read input or print anything. Your task is to complete the function **findPrefixes()** which takes the array of strings and it's size N as input and returns a list of shortest unique prefix for each word

**Expected Time Complexity:** O(N\*length of each word)

**Expected Auxiliary Space:**O(N\*length of each word)

**Constraints:**

1 ≤ N, Length of each word ≤ 1000

class TrieNode {

    public:

        TrieNode\* children[26];

        bool isLeaf;

        int count;

        TrieNode() {

            isLeaf=false;

            count=0;

            for (int i=0; i<26; i++) {

                this->children[i]=NULL;

            }

        }

};

class Solution

{

    public:

    void insert(TrieNode\* root, string s) {

        TrieNode\* p=root;

        for (int i=0; i<s.length(); i++) {

            int index=s[i]-'a';

            if (p->children[index]) {

                p=p->children[index];

            }

            else {

                TrieNode\* newNode=new TrieNode();

                p->children[index]=newNode;

                p->count++;

                p=newNode;

            }

        }

        p->isLeaf=true;

    }

    vector<string> findPrefixes(string arr[], int n) {

        //code here

        TrieNode\* root=new TrieNode();

        for (int i=0; i<n; i++) {

            insert(root, arr[i]);

        }

        vector<string> ans;

        for (int i=0; i<n; i++) {

            string str=arr[i];

            string prefix="";

            TrieNode\* p=root;

            int loc=-1;

            for (int i=0; i<str.length(); i++) {

                int index = str[i]-'a';

                if (p->children[index]->count > 1) loc=i;

                p=p->children[index];

            }

            for (int i=0; i<loc+2; i++) prefix+=str[i];

            ans.push\_back(prefix);

        }

        return ans;

    }

};