DSA SERIES

- Learn Coding

Topic to be Covered today

Trie

14. Longest Common Prefix

```
struct TrieNode {
    TrieNode* links[26];
    bool isEnd;
    int cntChild;


    TrieNode() {
        for (int i = 0; i < 26; i++) links[i] = NULL;
        isEnd = false;
        cntChild = 0;
    }


    bool containsKey(char ch) {
        return links[ch - 'a'] != NULL;
    }


    TrieNode* get(char ch) {
        return links[ch - 'a'];
    }
    void put(char ch, TrieNode* node) {
        links[ch - 'a'] = node;
    }
};


class Trie {
private:
    TrieNode* root;


public:
    Trie() {
        root = new TrieNode();
    }


    void insert(string word) {
        TrieNode* node = root;
        for (char ch : word) {
            if (!node->containsKey(ch)) {
                node->put(ch, new TrieNode());
                node->cntChild++;
}
            node = node->get(ch);
        }
        node->isEnd = true;
    }


    string getLongestCommonPrefix(string firstWord) {
        string prefix = "";
        TrieNode* node = root;


        for (char ch : firstWord) {
            // if more than one child or word ended, stop
            if (node->cntChild > 1 || node->isEnd)
                break;
            prefix += ch;
            node = node->get(ch);
        }


        return prefix;
    }
```

```
};


class Solution {
public:
    string longestCommonPrefix(vector<string>& strs) {
         if (strs.empty()) return "";


    Trie trie;
    for (auto& word : strs)
        trie.insert(word);


    return trie.getLongestCommonPrefix(strs[0]);
    }
};
```

3043. Find the Length of the Longest Common Prefix

```
struct Node{
    Node* links[10];
};


class Solution {
public:


    Node* getNode(){
        Node* node = new Node();


        for(int i =0 ;i<10;i++){
            node->links[i]=NULL;
        }
        return node;
    }
```

```
void insert(int num , Node* root){
    Node* node = root;
    string numStr = to_string(num);


    for(char ch : numStr){
        int idx = ch -'0';
        if(!node->links[idx]){
            node->links[idx]=getNode();
        }
        node = node->links[idx];
    }
}


int search(int num,Node* root){
    Node* node = root;
    string numStr = to_string(num);
    int length = 0;


    for(char ch : numStr){
        int idx = ch - '0';
```

```
            if(node->links[idx]){
                length++;
                node = node->links[idx];
            }else{
                break;
            }
        }


        return length;
    }


    int longestCommonPrefix(vector<int>& arr1, vector<int>& arr2) {
        Node* root = getNode();
        int result = 0;


        for(int &num : arr1){
            insert(num,root);
        }
        for(int &num:arr2){
            result =  max(result,search(num,root));
        }


        return result;
    }
};
```

1268. Search Suggestions System

```
struct Node {
    Node* links[26];
    vector<string> suggestions;


    Node() {
        for (int i = 0; i < 26; i++) {
            links[i] = nullptr;
        }
    }
};


class Solution {
public:
    Node* root;
    Solution() { root = new Node(); }


    void insert(string word) {
        Node* node = root;
```

```
         for (char ch : word) {
                  int idx = ch - 'a';


                  if (!node->links[idx]) {
                      node->links[idx] = new Node();
                  }


                  node = node->links[idx];




                  if (node->suggestions.size() < 3) {
                      node->suggestions.push_back(word);
                  }
              }
          }


          vector<string> getSuggestions(string prefix){
              Node* node = root;
              vector<string> result;
```

```
for(char ch : prefix){
            int idx = ch - 'a';


            if(!node ->links[idx]){
                return {};
            }


            node = node->links[idx];
        }
        return node->suggestions;
    }


    vector<vector<string>> suggestedProducts(vector<string>& products,
                                             string searchWord) {
    sort(products.begin(),products.end());


    for(string &word : products){
        insert(word);
    }
```

```
vector<vector<string>> ans;
    string prefix = "";


    for(char ch : searchWord){
        prefix+=ch;


        ans.push_back(getSuggestions(prefix));
    }


    return ans;
}
};
```

648. Replace Words

```
class Solution {
public:
    struct Node {
        Node* links[26];
        bool isEnd;
    };


    Node* getNode() {
        Node* node = new Node();
        node->isEnd = false;


        for (int i = 0; i < 26; i++) {
            node->links[i] = NULL;
        }


        return node;
    }
```

```
Node* root;


void insert(string word) {
    Node* node = root;


    for (int i = 0; i < word.length(); i++) {
        int idx = word[i] - 'a';


        if (!node->links[idx]) {
            node->links[idx] = getNode();
        }
        node = node->links[idx];
    }
    node->isEnd = true;
}


string search(string word) {
    Node* node = root;
```

```
        for (int i = 0; i < word.length(); i++) {
            int idx = word[i] - 'a';


            if (!node->links[idx])
                return word;


            node = node->links[idx];


            if (node->isEnd) {
                return word.substr(0, i + 1);
            }
        }
        return word;
    }


    string replaceWords(vector<string>& dictionary, string sentence) {
        stringstream ss(sentence);
        string word;
        string result;
        root = getNode();
        for (string word : dictionary) {
            insert(word);
        }


        while (getline(ss, word, ' ')) {
            result += search(word) + " ";
        }

        result.pop_back();
        return result;
    }
};
```

Learn coding

THANK YOU