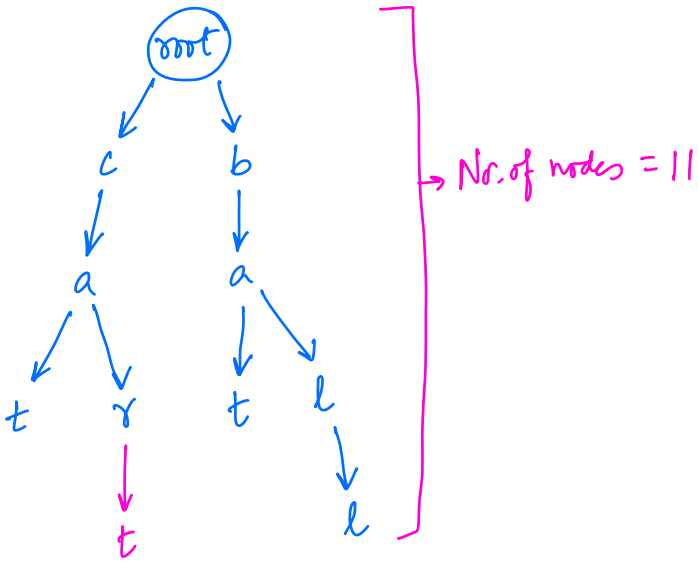
RETRIEVAL

Prefix Tree

cat
car
bat
ball
cart



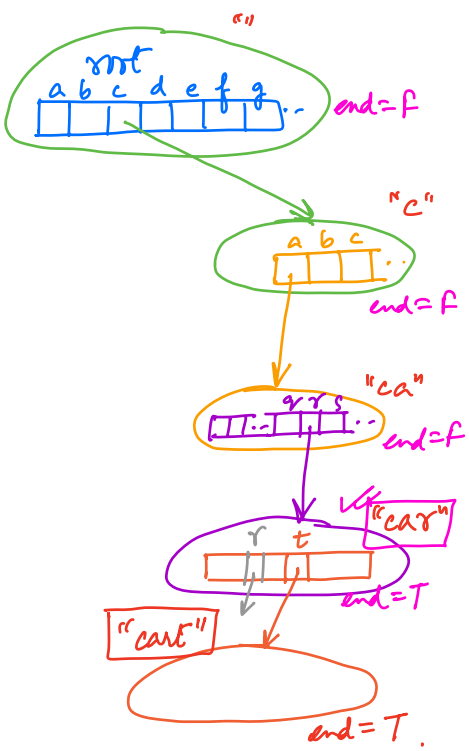
Applications :-

→ Spell checker

→ Auto-complete

```
class Node {
    Node children[];
    boolean isEndOfWord;
    Node() {
        children = new Node[26];
        isEndOfWord = false;
    }
}
```

car
cart

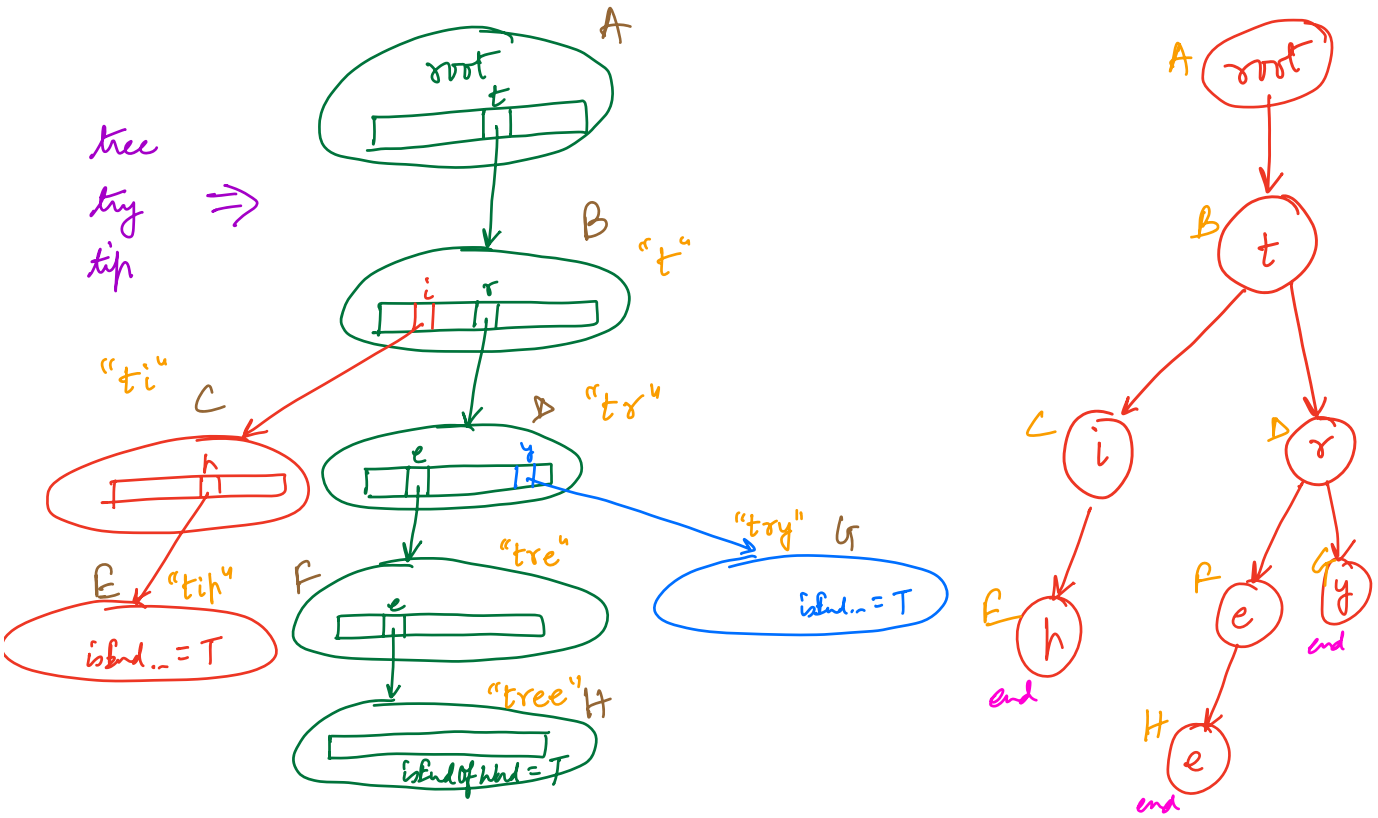


cart ✓
car ✓
ca ✗

Search :-

Start at root and for each character of string, traverse down the trie

- The character is not found in the Trie → Doesn't Exist
- All the characters are found → Check isEndOfWord flag
  - T → Exists
  - F → Doesn't Exist

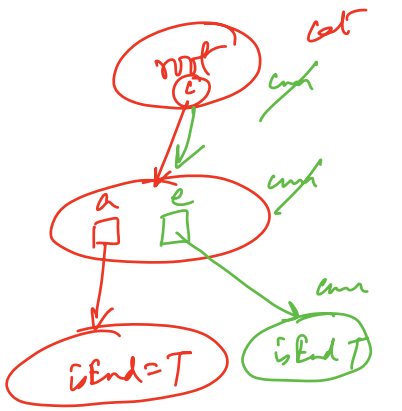


Insert a word in Trie

```
void insert( char word[], int len, Node root){
    Node curr = root;
    for (int i=0; i<len; i++){
        int idx = word[i]-'a';
        if ( curr.children[idx] == null)
            curr.children[idx] = new Node();
        curr = curr.children[idx];
    }
    curr.isEndOfWord = true;
}
```

→ length of word.

O(len) T.C.



Search a word in Trie

```
boolean search (char word[], int len, Node root){
    Node curr = root;
    for (int i=0; i<len; i++){
        int idx = word[i]-'a';
        if ( curr.children[idx] == null)
            return false;
        curr = curr.children[idx];
    }
    return curr.isEndOfWord;
}
```

→ length of word.

O(len) T.C.

Delete a word from Trie (We don't maintain frequency of words)

word doesn't exist → Do nothing

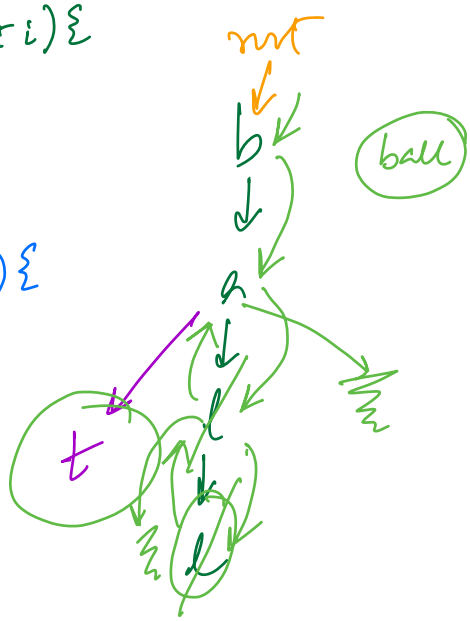
word only exists as a prefix of another word → Do nothing

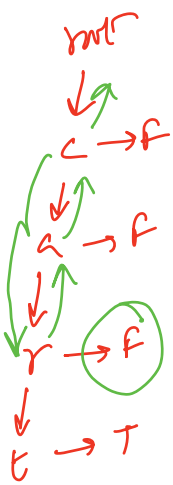
word exists and it is also a prefix of another word → Mark the flag as False (No deletion of nodes)

otherwise :) delete all the nodes from the bottom that are not shared with any other word.

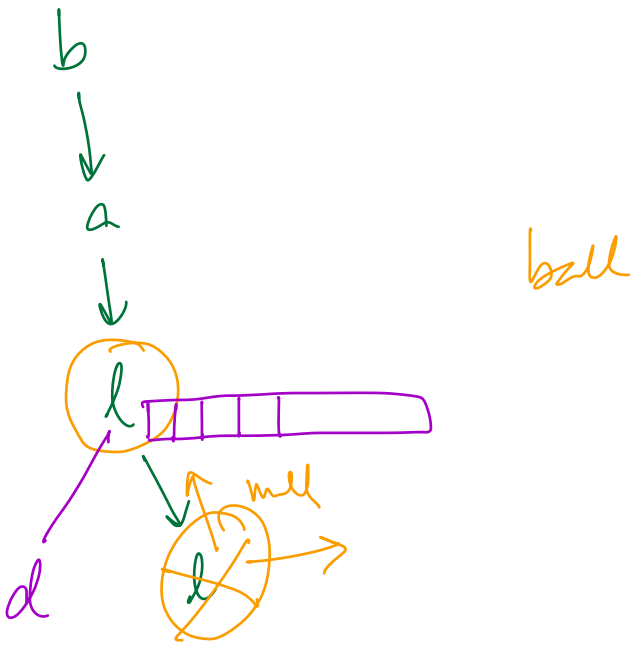
```
Node delete (Node curr, char word[], int len, int i){
    if(curr == null)
        return curr;
    if (i == len){
        if (hasChild(curr) || !curr.isEndOfWord){
            curr.isEndOfWord = false;
            return curr;
        }
        else{
            return null;
        }
    }
    curr.children[word[i]-'a'] = delete(curr.children[word[i]-'a'], word, len, i+1);
    if( curr.children[word[i]-'a'] == null){
        if(!hasChild(curr) && !curr.isEndOfWord){
            return null;
        }
    }
    return curr;
}

delete(root, word, word.length, 0);
```





```
boolean hasChild (Node curr){
    for (int i=0; i<26; i++){
        if (curr.children[i] != null)
            return true;
    }
    return false;
}
```

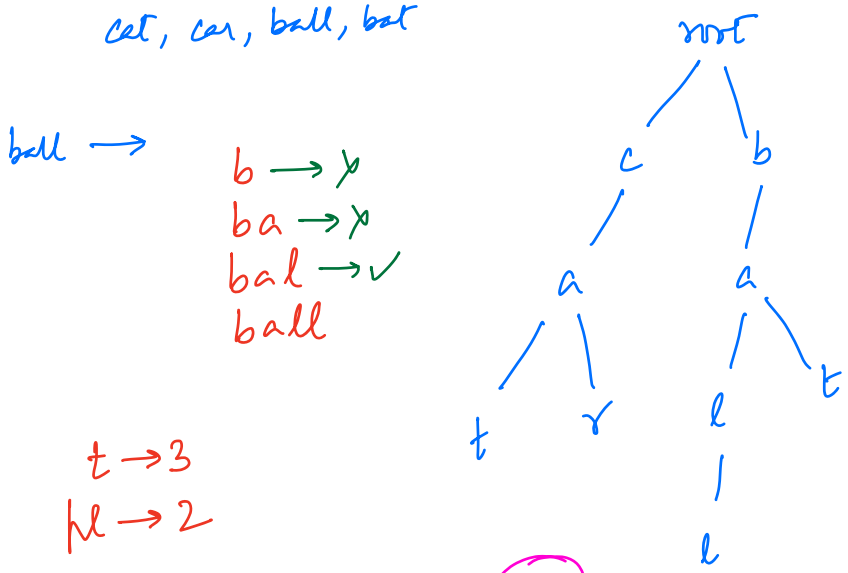


bzll

[Break till 10:39 PM]

Q) Find the shortest unique prefix of a given word from a set of words. Each word is inserted exactly once. Assume that the given word exists in the set of words.

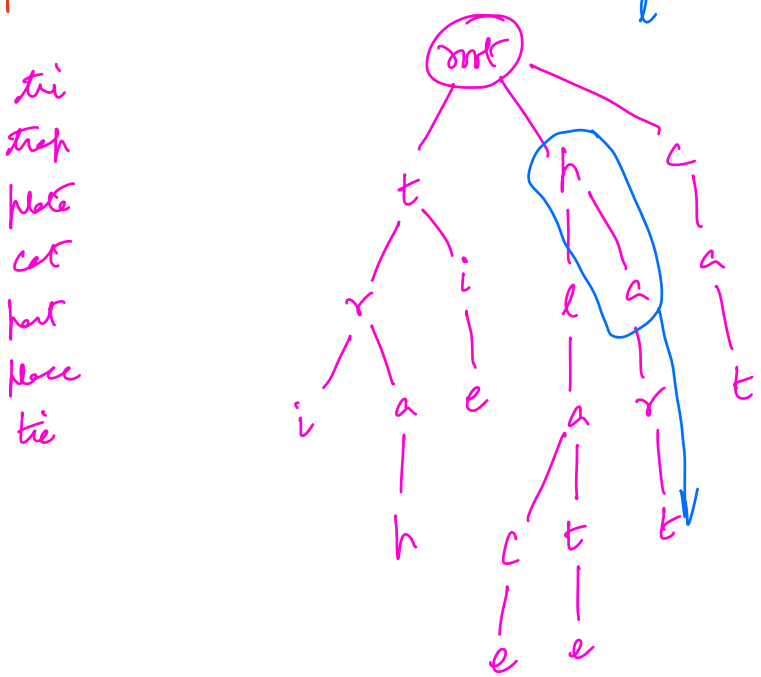
cat, car, ball, bat



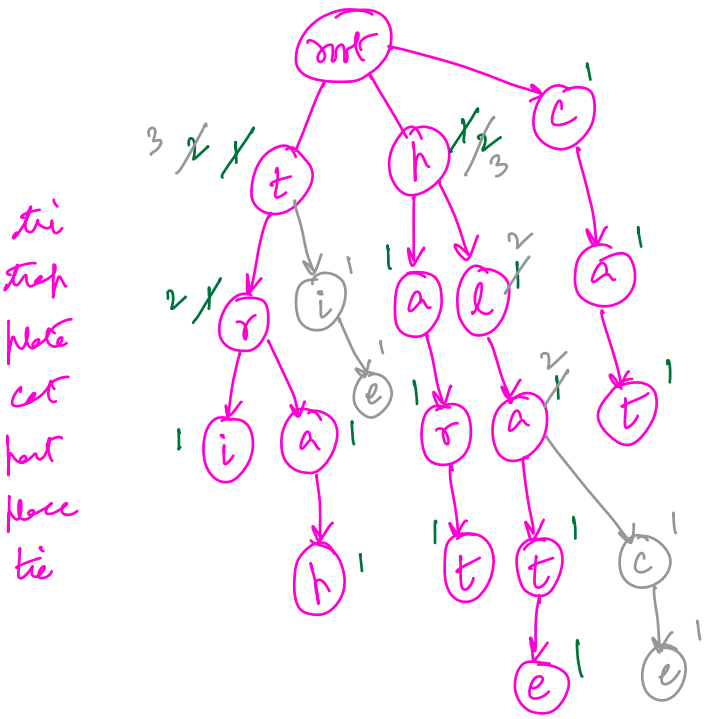
ball →

b → ✗
ba → ✗
bal → ✓
ball

t → 3
ll → 2



tri
trap
plate
cat
part
place
tie



trie stop plate cat pat place tie


```
class Node {
    Node children[];
    int count;
    Node() {
        children = new Node[26];
        count = 0;
    }
}
void insert(char word[], int len, Node root) {
    Node curr = root;
    for (int i=0; i<len; i++) {
        int idx = word[i] - 'a';
        if (curr.children[idx] == null)
            curr.children[idx] = new Node();
        curr = curr.children[idx];
        curr.count += 1;
    }
```

```
}
String findUniqueShortestPrefix(Node root, char word[], int len){
    Node curr = root;
    for(int i=0; i<len; i++){
        curr = curr.children[word[i]-'a'];
        if(curr.count==1)
            return word.substring(0,i+1);   // [0,i] closed.
    }
    return "";
}
```