length of word = L ✓
# correct words = N } Spelling checker ✓ ✓

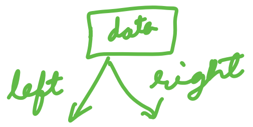
Tra _ _ _ Auto complete ✓ (Oracle Interview * 2 + Hike) ✓

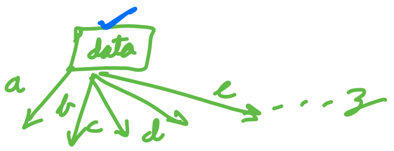
Data Compression → 10^6 char → 1000 char ✓

playground → playg

Trie → Tree like DS that stores data from top to bottom.

Types → 1) Trie of characters ✓ (lower case alphabets) ✓
2) Trie of bits





Eg → trie, try, trim, play, plate, car, par, trimmer, trying, pla ✓

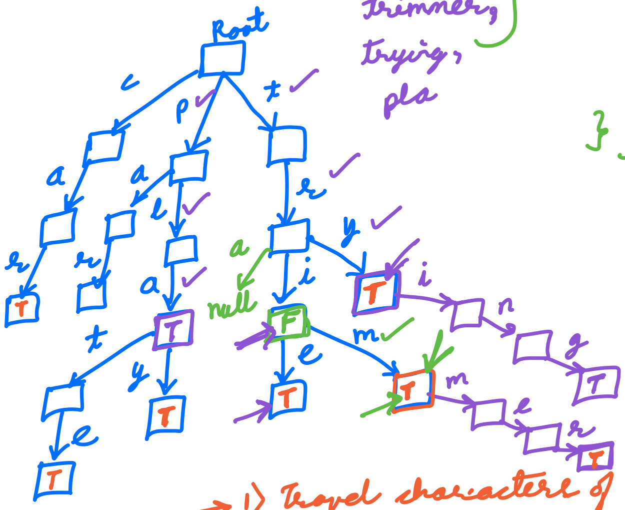
```
class node {
    node children[]; ✓
    boolean isEnd;
    node() {
        children = new node[26]; ✓
        isEnd = false;
} }
```

∀i children[i] = null

index 0 — 25

a, b, c, d -- z
0 1 2 --- — 25

(char – 'a') ✓



Search → 1) Travel characters of word from top to bottom in trie.
2) If at any point, char is not present ⇒ word was not part of trie. ✓ ✓

Eg → search(trap) X

3) If all characters are travelled ⇒ word is part of input X
⇒ word is a valid prefix. ✓ ←

Eg → Search(tri
e
m

Q→ How to know that word is complete? ←

using isEnd data in the node. ✓

search(trim) ✓

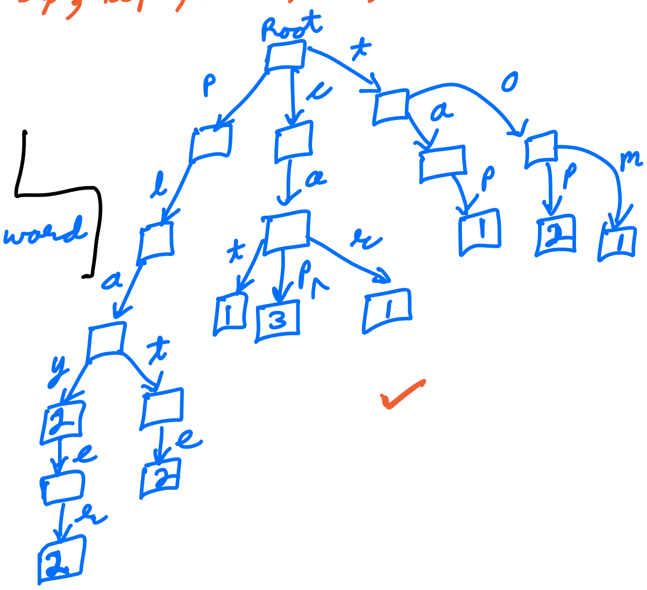
TC of insert = O(L)
TC of search = O(L)
$\Big\}$ ≃ O(1) in practice

Q→ What if same word is present multiple times in i/p?

isEnd → freq

```
class node {
    node children[];
    int freq;
    node () {
        children = new node[26];
        freq = 0; // 0 ⇒ incomplete word
    }
}
```

I/p → play, player, plate, player, cap, car, cat, cap, cap, play, tap, top, tom, top, plate.



✓

plate

$SC < O(\Sigma \text{ length of string})$

$TC \Rightarrow O(\underbrace{L}_{\text{length}} * \underbrace{N}_{\text{count of words}})$ ✓

```
void insert (root, word) {
    cur = root;
    l = word.length();
    for ( i=0 ; i < l ; i++) {          TC=O(length)
        c = word[i];
        idx = c - 'a';   // a→0, b→1, c→2 .. z→25
        if(cur.children[idx] == null){
                cur.children[idx] = new node();
        }
        cur = cur.children[idx];
    }
    cur.freq++; ✓
}
```

Deletion

delete(trimmer) ✓   search(trimmer) ✗

t r i m m _ _ _ ⇒ valid prefix

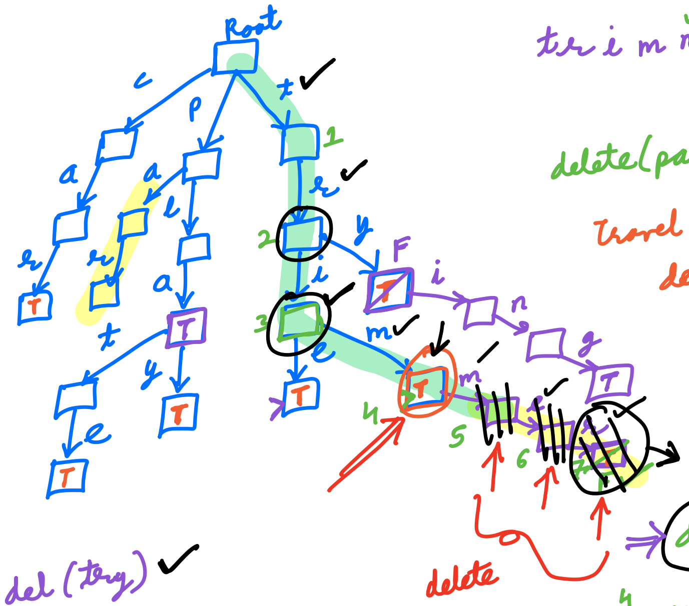
travel all characters

delete(par)

Travel the nodes in rev order ✓

delete till → current node completes a word. ✓

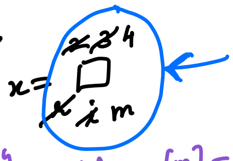
→ current node also has other children ✓



del(try) ✓

$TC = O(L)$

$SC = O(L) \Rightarrow O(1)$ ✓

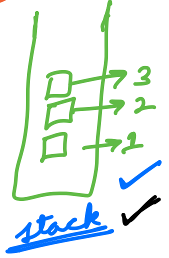


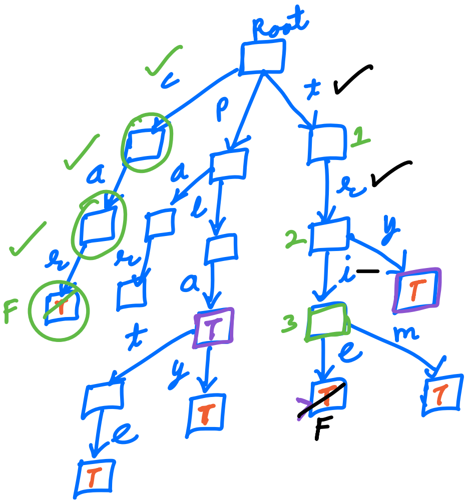
$\underset{m}{\overset{4}{\square}}$.children[m] = null

del(trimmer)

$\overset{4}{\square}$.children[m] = null

last = $\underset{m}{\overset{4}{\square}}$





del(car)

x = $\overset{root}{\square}$ ✓

$\overset{root}{\square}$.children[c] = null ;

del(trie)   x = $\overset{i}{\square}$.children[e] = null

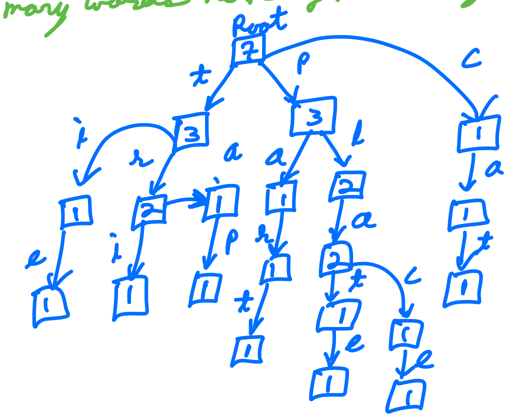
Q→ Data Compression

Find prefix of each word that uniquely defines the word.
(No word is prefix of another word)

Eg→ [tri, trap, plate, cat, part, place, tie]

O/P→ [tri, tra, plat, c, pa, plac, ti] ← ✓

How many words have given string as prefix? → =1 ✓
>1 ✗



TC = O(L * N)
SC < O(L*N)