Trie

Prefix Tree

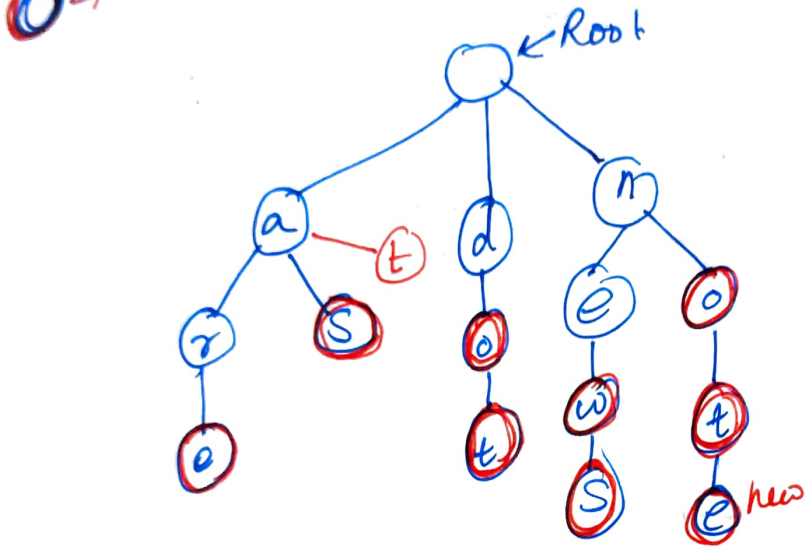
"a - z"

- add()
- search()
- delete()
- startsWith()

O(1)

Hashmap → Known to be O(1) but it's diff for diff types Hence not HashMap

→ Terminator



→ adding

for storing "at"

- Go to a
- check if t is child
- else add it

→ for searching

"not e"

- Go to each letter node & check if child has next
- Final check for terminator

Ex "ne" will return false.

→ start with

"do"

- Just search for child & no need to check for terminator

→ Delete

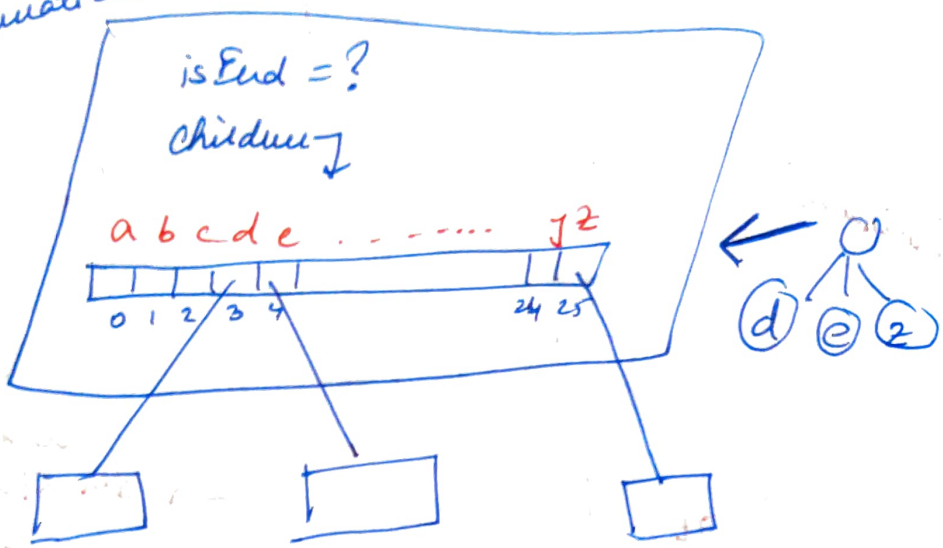
not e

- make the last node as non-terminating

OR

- make the previous to last node point to NULL

Trie Node
Structural
Information




```
Class TrieNode{
    TrieNode [] children = new TrieNode[26];
    Boolean isEnd = false;
}

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

    public void insert(String word){
        TrieNode node = root;
        for(int i=0; i<word.length(); i++){
            char c = word.charAt(i);
            if(node.children[c-'a'] == null){
                node.children[c-'a'] = new TrieNode();
            }
            node = node.children[c-'a'];
        }
        node.isEnd = true;
    }
```

word ends IMP

```
public boolean search (String word){
    TriNode node = root;
    for (int i=0; i<word.length; i++){
        char c = word.charAt(i);
        if (node.children[c-'a'] == null)
            return false
        node = node.children[c-'a'];
    }
    return node.isEnd; // Check if word end then IMP
}

public boolean startsWith (String prefix) {
    TriNode node = root;
    for (int i=0; i<prefix.length(); i++){
        char c = prefix.charAt(c);
        if (node.children[c-'a'] == null)
            return false;
        node = node.Children[c-'a'];
    }
    return true;
}
}
```

Add and Search Word

add ("bad")
add ("dad")
add ("mad")
search ("pad") → false
search ("bad") → true

→ Simple trie

search (".ad") → true
search ("b..") → true

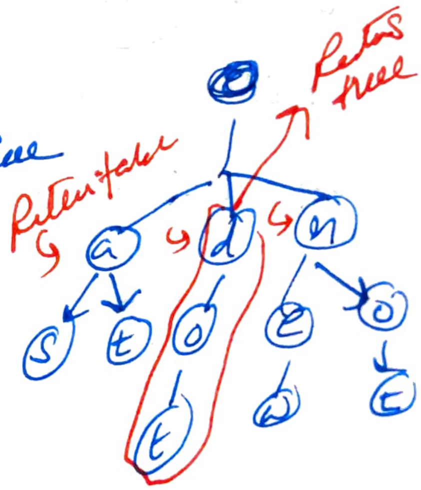
→ Dot variation

add → Simple insert

. → not ✓ true dot ✓ true

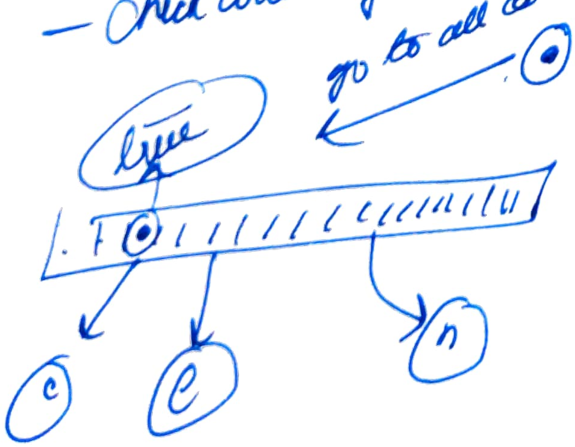
(.)dot → true

matched with any node child



- check code on github :)

go to all chi

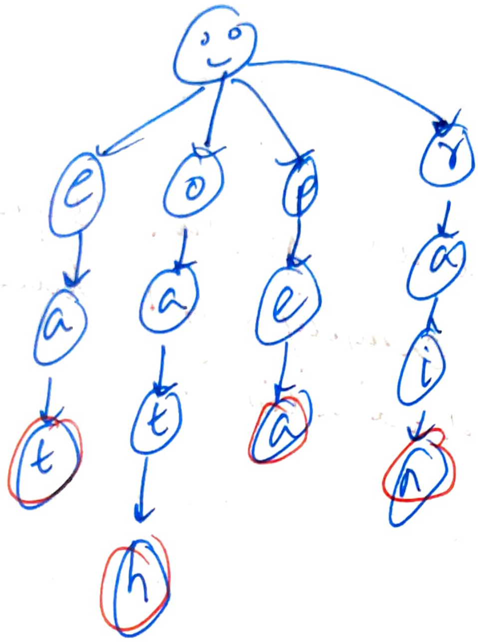


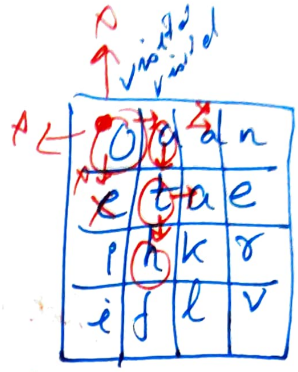
for handling dot

Word Search II

Using trie

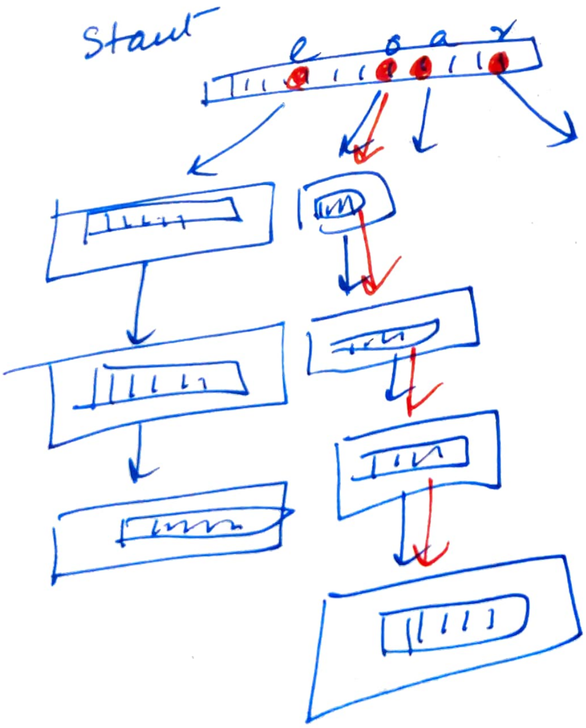
["oath", "pea", "eat", "rain"]

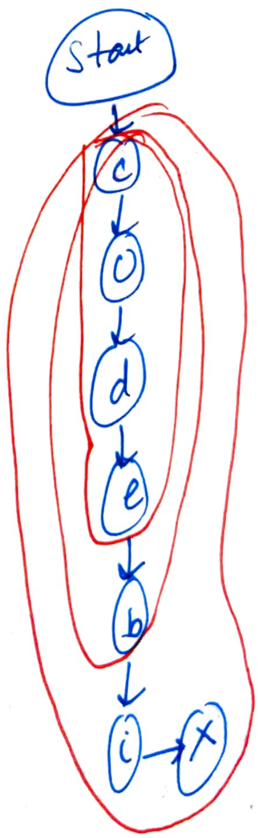




Why trie ?

["code"
"codeb"
"codebi"
"codebix"]





All in one

Hence

LLTC