Welcome ☺

Agenda : Tries
Operations
1 ques^n

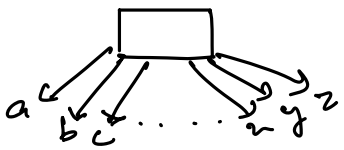
Tries / Prefix Tree.

⇒ Best used for strings, text

⇒ Tree like D.S that stores data from top to bottom.

Tries of character (a-z)

Autocomplete
Spelling checker



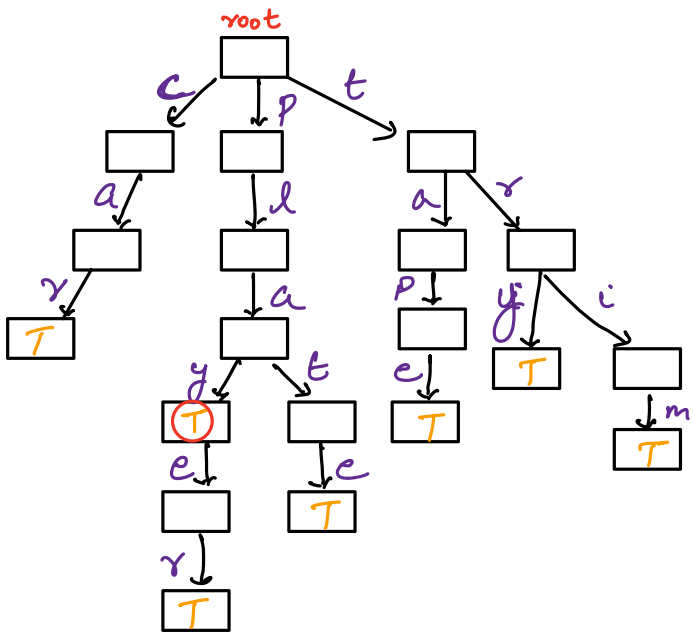

```
class Node {
    boolean isEnd
    Node child[26]
}
```

a ⇒ 0
b ⇒ 1
c ⇒ 2
.

⇒ Initially ∀i, child[i] == NULL

Visualize

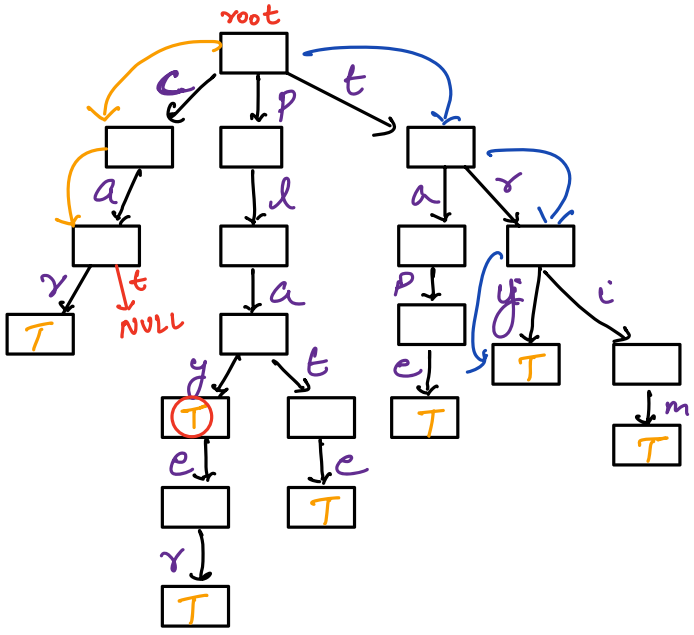
1) tape
2) try
3) trim
4) play
5) plate
6) car
7) players.



Operations

1) Searching.

1) tape
2) try
3) trim
4) play
5) plate
6) car
7) player.



search ("cat") ✗ ⟹ If all the characters are not traversed → not complete ⟹ word is not there.

search ("try")

search ("playe") ⟹ If all the characters are present / traversed, even then you need to check isEnd of the last travelled node. If true, then word is present.

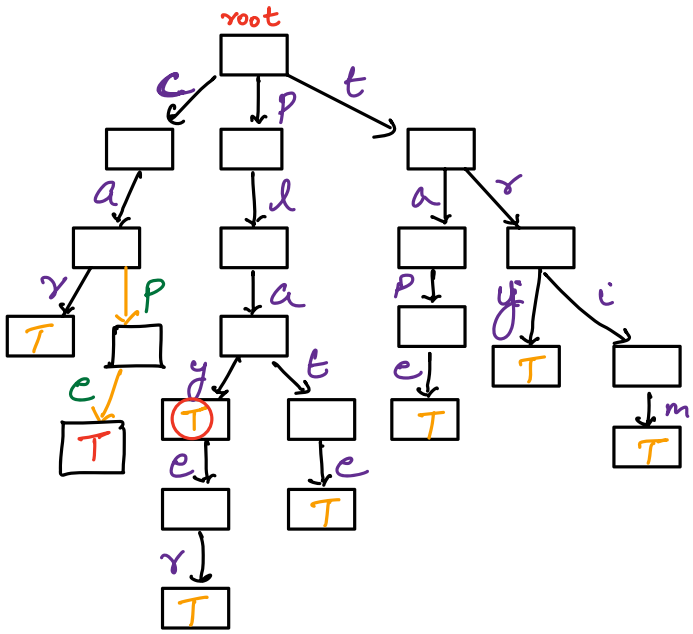
→ If all the characters are present / traversed, it is either present or it is a prefix of some other word.

Code

```
boolean search ( root , word )
{
    temp = root
    for ( i → 0 to word.length - 1 )
    {
        ch = word [i]
        if ( temp.child [ ch - 'a' ] == NULL
                return false
        else
                temp = temp.child [ ch - 'a' ]
    }
    return temp.isEnd
}
```

T.C → O(word length)
S.C → O(1)

2) Insertion



1) tape
2) try
3) trim
4) play
5) plate
6) car
7) player.

insert ("cape")

Code

```
void insert( root , word)
{ temp = root
  for( i -> 0 to word.length - 1 )
  {
     ch = word[i]
     if( temp.child[ ch - 'a' ] == NULL
           temp.child[ ch - 'a' ] = new Node()
     temp = temp.child[ch-'a']
  }
  temp.isEnd = true
}
```

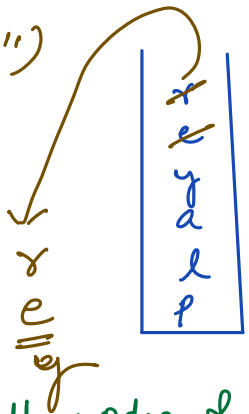
T.C (word length)

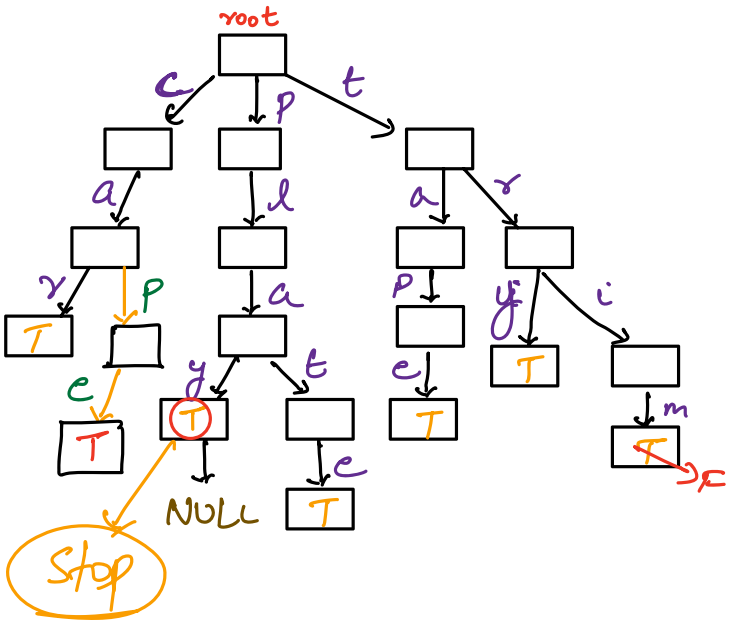
Deletion

delete("trim")

delete("play")

delete("player")





1) Travel all nodes & insert in stack. If not present, you cannot delete.

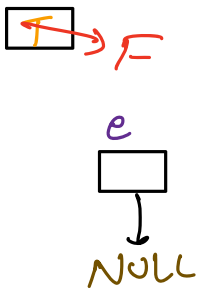
2) Update isEnd to False for last char.

3) Pop element from stack.

if leaf node

&

isEnd == false



$\forall i$ child[i] == NULL

only then delete it

T.C → O(word length)

S.C → O(word length)

1) pop from stack

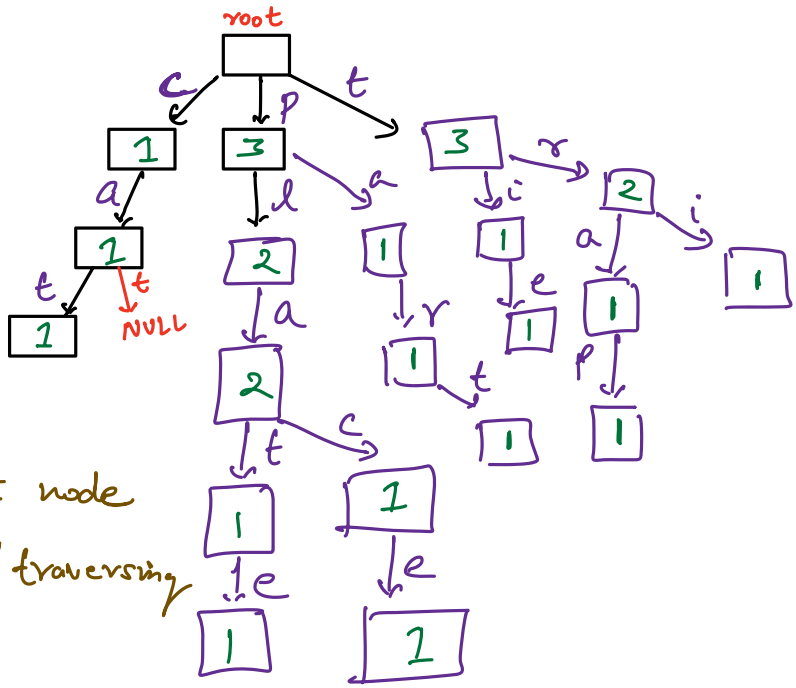
2) temp.child[ ] = NULL

Q Given a list of words, find shortest prefix of each word that uniquely defines the word. [no word is prefix of another word)

eg: [tri, trap, plate, cat, part, place, tie]

⇒ tri, tra, plat, c, pa, plac, ti

```
class Node {
    int freq
    Node child[26]
}
```



1) #nodes → store # times that node is travelled while inserting / traversing

2) Travel the word till the a node with freq = 1 is reached.

insert & search

T.C → O(2 * N * length)

S.C → O(N * length)