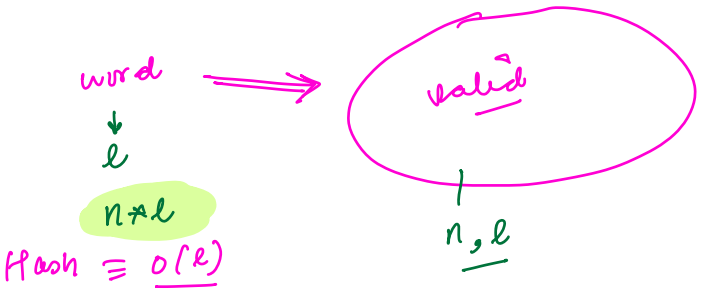
spelling



autocomplete

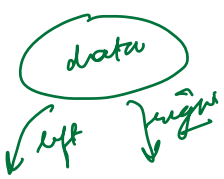
play

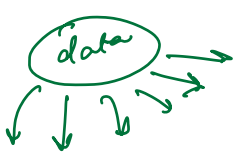
playground
play
player
⋮

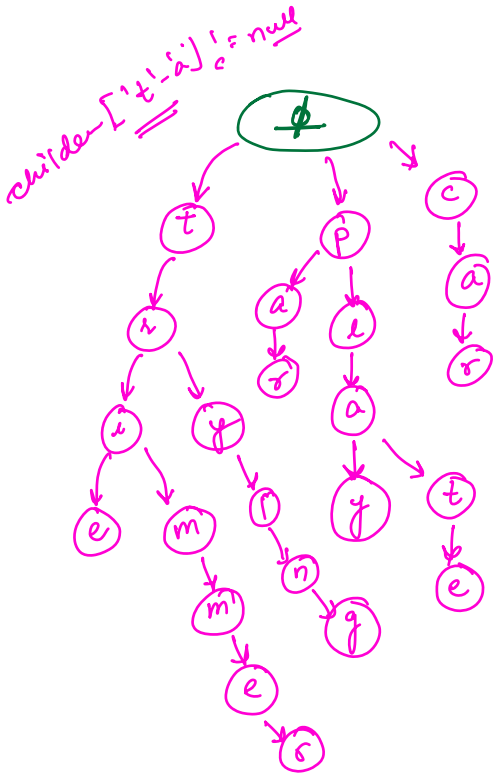
Trie : tree based DS which stores the info from top to bottom

↳ prefix-tree

Trie	Try	Trim	play
plate	car	par	trimmer
trying	pla		








```
class Node{
    char data;
    Node children[26];
    bool isEnd;
}
```

'a' → 0
'b' → 1
'c' → 2
'd' → 3
⋮

ch - 'a'

search (trap)
(tri)
pla

Traverse the string & search char by char in the trie

if a character's node doesn't exist
↓
word is invalid

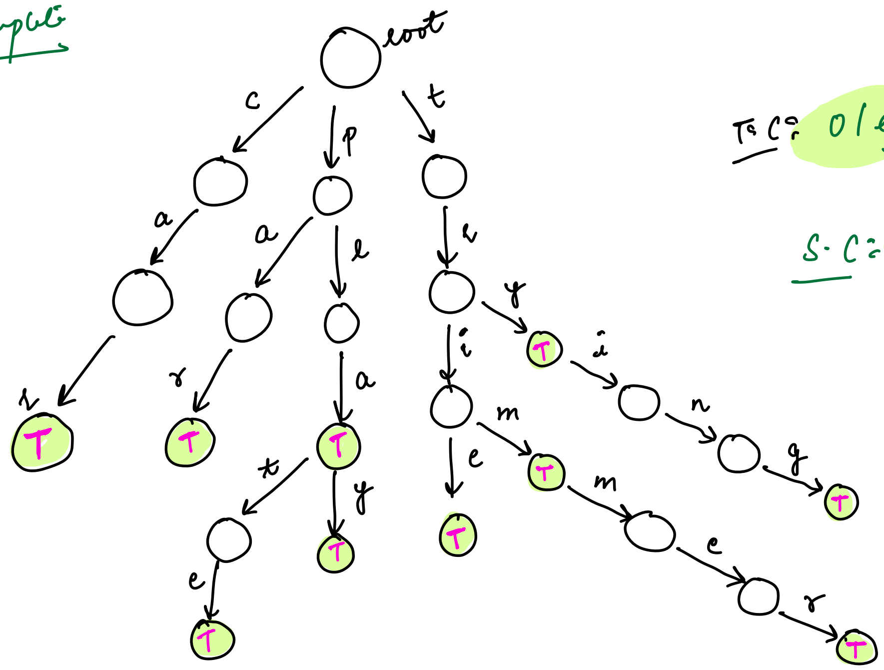
find all characters

last char node's isEnd = true
↓
valid

isEnd = false
↓
invalid

autocomplete

trie



T.C: O(e)

S.C: n * l * 26

If a word can be present multiple times?

```
Node {
    char data;
    // bool isEnd;   (struck out)
    Node children[26];
}
```

int freq; → how many words end at a particular node (freq>0);

```
void insert ( Node root, string word)
{
        curr = root;
        l = word.length();

        for( i=0;   i<l; i++)
        {
                int index = word[i] - 'a';
            if ( curr.children[index] == null)
              {
              curr.children[index] = new Node( word[i]);
              }
            curr = curr.children[index];
        }
       curr.freq++;          //    curr.isEnd = true;
}
```

```
bool search ( Node root, string word)
{
        curr = root;
        l = word.length();

        for( i=0;   i<l; i++)
        {
                int index = word[i] - 'a';
            if ( curr.children[index] == null)
             {
                return false;
             }
            curr = curr.children[index];
        }
        if ( cur.freq >0) return true;        //isEnd
            else false;
}
```

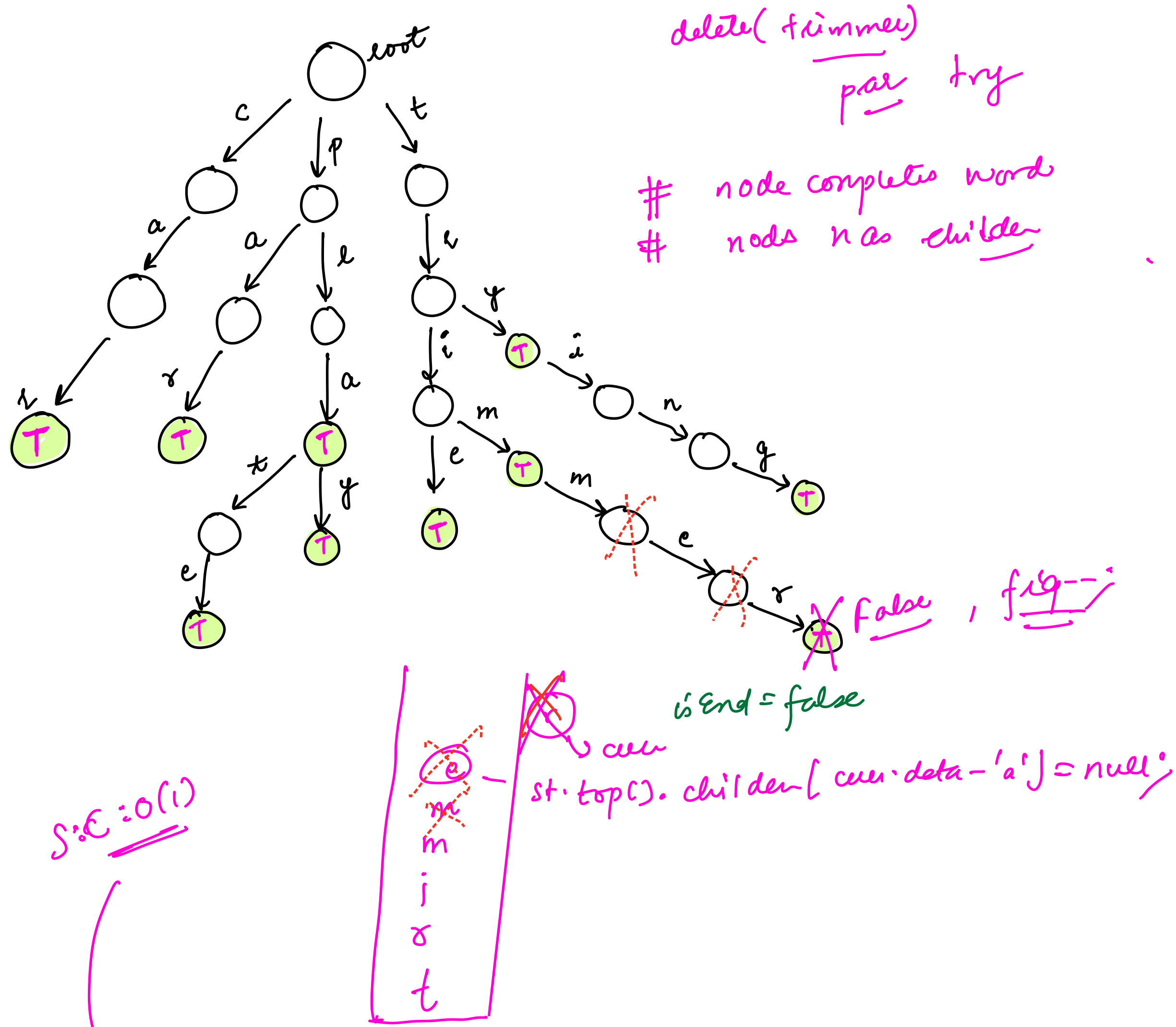
deletion

lastnode = root ~~t~~ ~~r~~ ~~i~~ m

delete(trimmer)

par try

node completes word
node has children



S.C : O(1)

get the last node where you can't delete at all.

shortest unique prefix

Q Find shortest unique prefix to represent each word.

Note: Assume that no word is prefix of another
In other words, the representation is always possible.

tri	trap	plate	cat	pat	place	tie

tri tra plat c pa plac ti

"_" → how many words is which it is prefix

tc: creation O(n*l)
for 1 query = O(l)
for T.C. O(n*l)