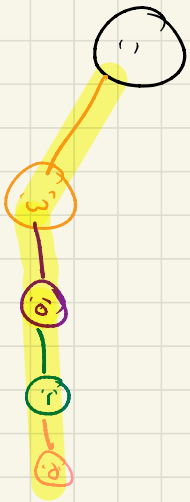
A2 Explanation : Tre Trees

Trie Trees

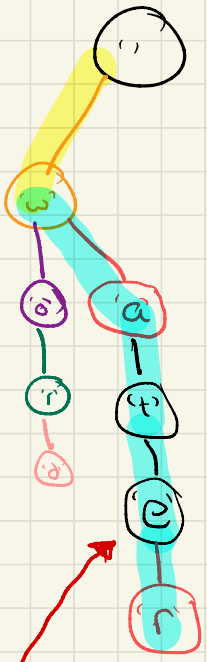
Insert("word")



1. Add "w" to subtrees, with value as a new TrieTree("w")
2. Add "o" to subtrees of w, make new TrieTree('o')
3. Add 'r' to subtrees of 'o', make new TrieTree('r')
4. Add 'd' to subtrees of 'r' make new TrieTree("d","word")

Trie Trees

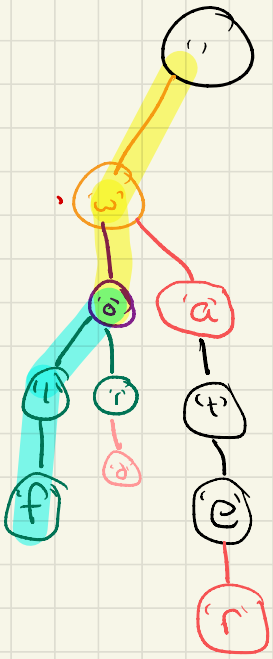
Insert("water")



1. Search for "w" in our children, retrieve that TrieTree
2. Add "a" to subtrees of w, make new TrieTree('a')
3/4. Add 't', 'e' to the chain
5. Add 'r' to subtrees of 'e' make new TrieTree("r", "water")

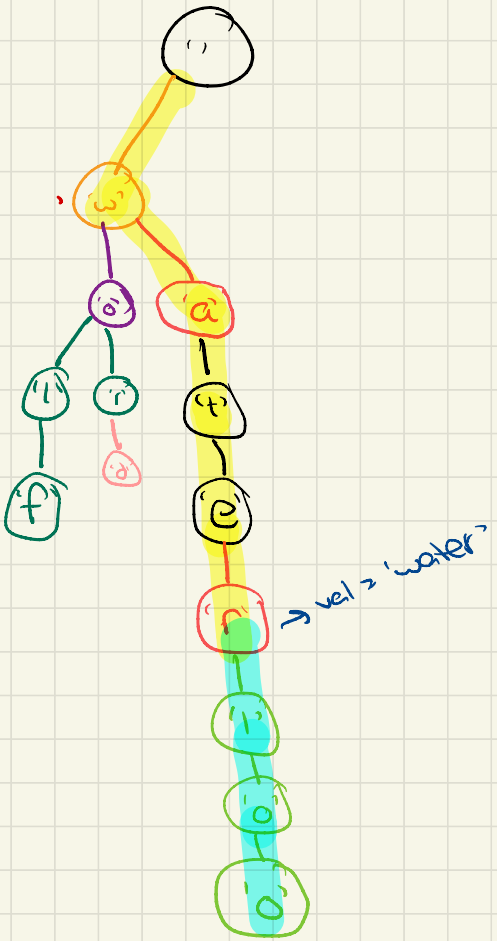
Trie Trees

Insert("wolf")



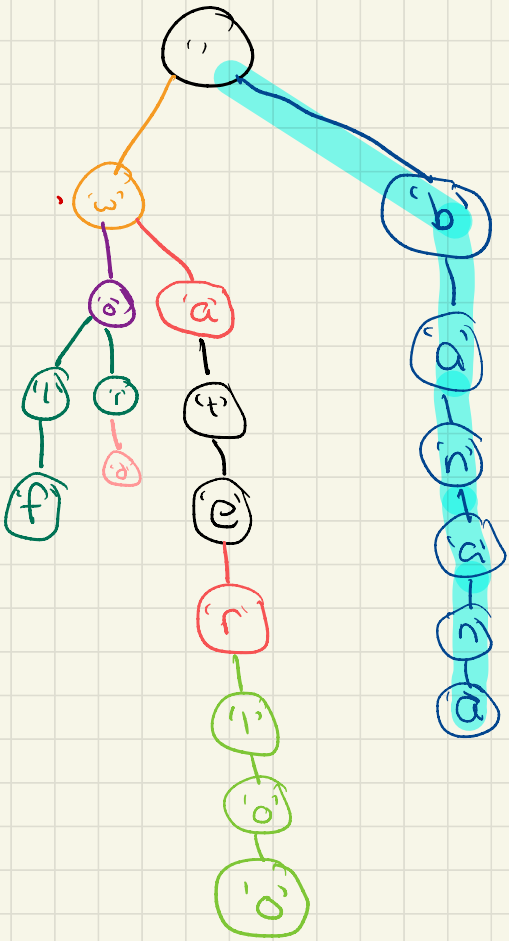
Trie Trees

Insert("waterloo")



Trie Trees

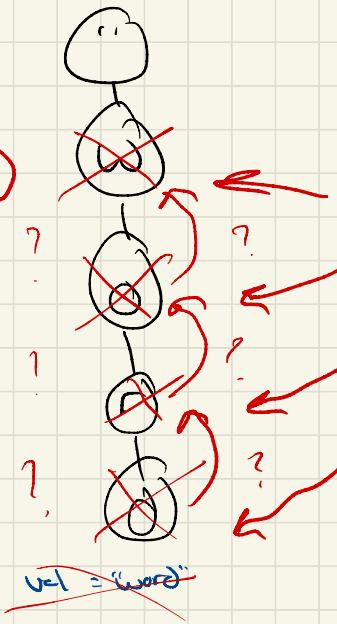
Insert("banana")



Deleting Items in Trie Trees

insert("word")

delItem("word")



Ex. 1

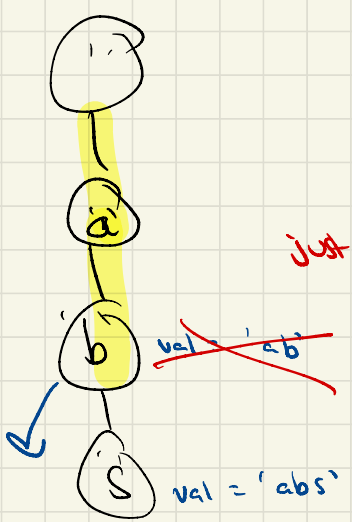
Delete nodes which have no other children.

Ex. 2

insert('ab')

insert('abs')

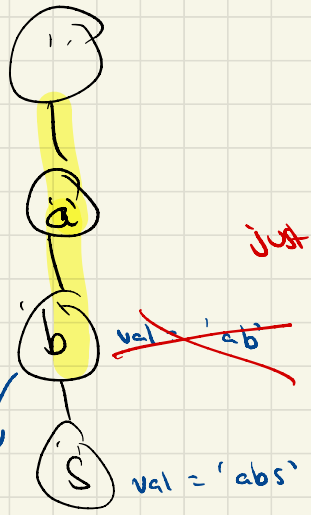
delItem('ab')



just delete the value 'ab'

don't delete b, because it has children

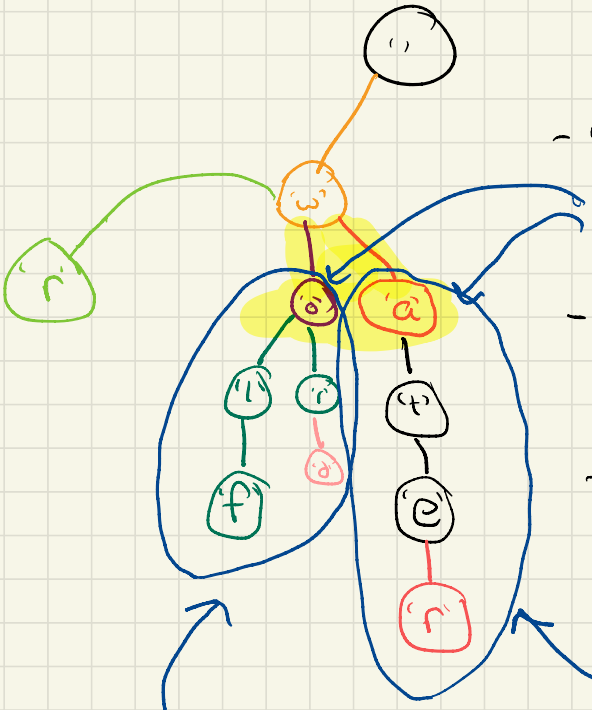
delitem('ab')



just delete the 'ab'

don't delete b, because it has children

Explaining self._children



Children of 'w'

- Store next characters as the keys 'a', 'o'
- The values of each key is just the TrieTree representing that character

Code-wise: (rooted at 'w')

self._children['a'] → TrieTree

self._children['o'] → TrieTree

Add new TrieTree

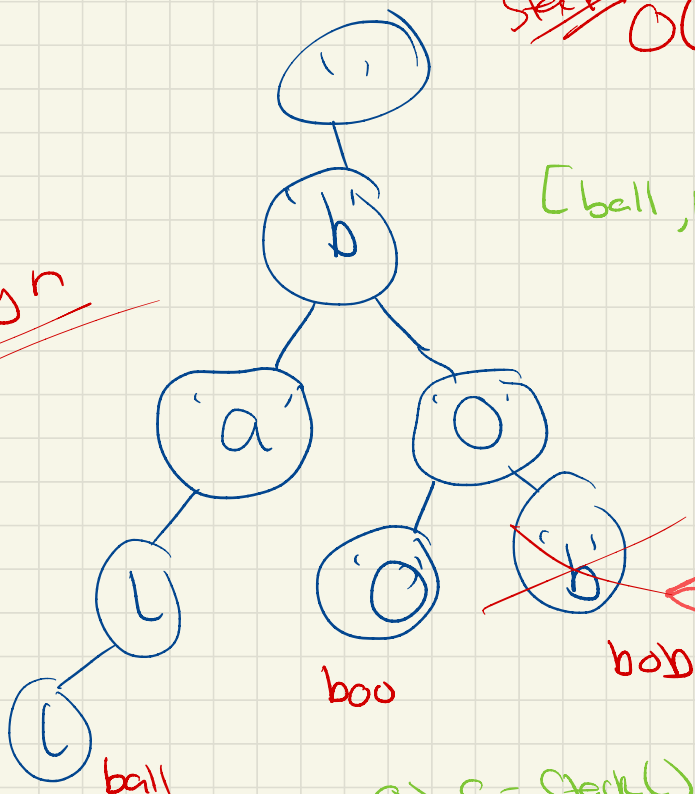
Let's say I want to add the word "write"

self._children['r'] = TrieTree('r')

Stack O(1)

Sorting
n log n

[ball, boo, bob]



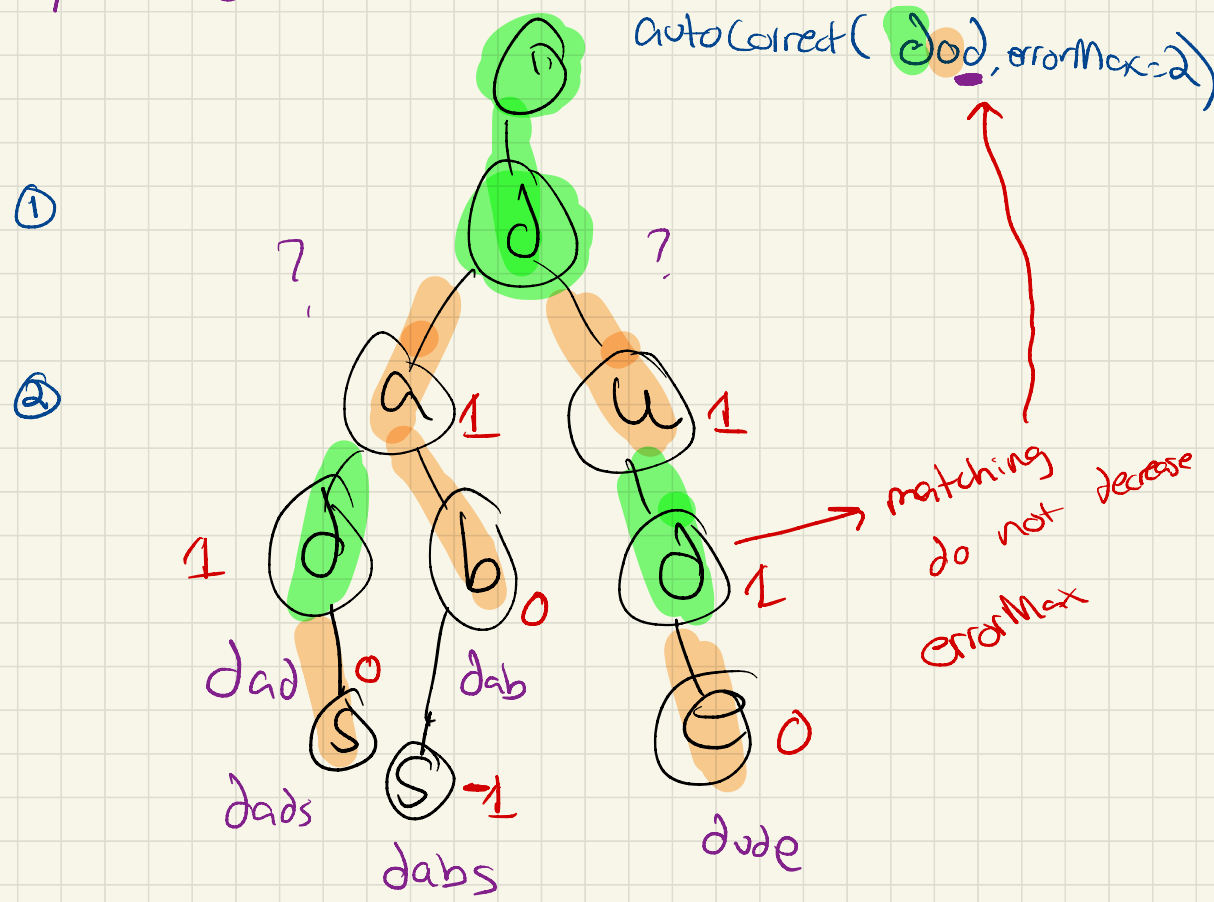
delitem('bob')

0) S = Stack()
1) Search for b, S.push(b)
2) Search for o, S.push(o)
3) Search for b, S.push(b)
4) delete value 'bob' from
Start popping
5) parent = S.pop()
6) Check if parent has children
- If it has children, return (you're done)

7) toDelete = parent.char
8) parent = S.pop()
9) delete toDelete from parent.children

Repeat 6-9 while not S.is_empty()

Auto Correct



[dad, dads, dab, dude]

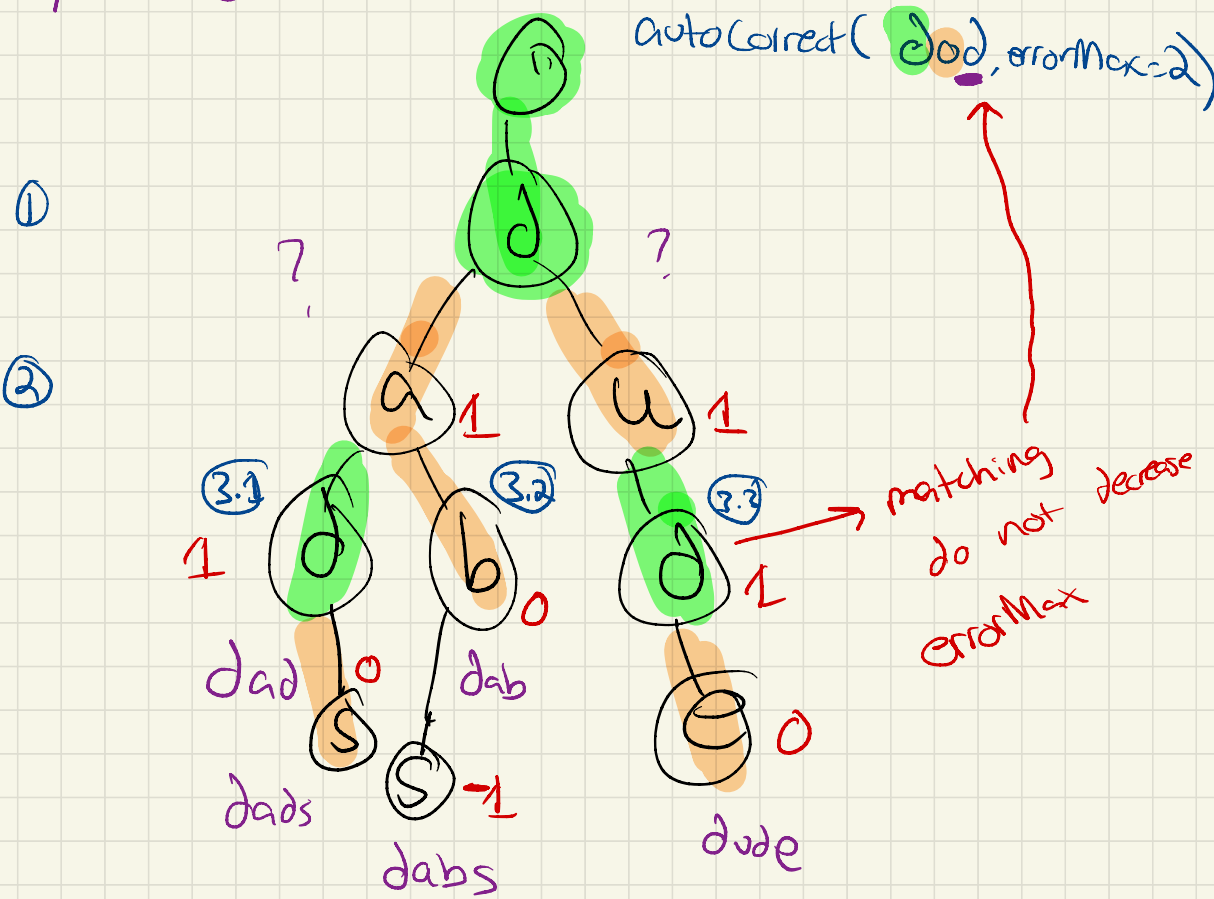
Steps

① Look for the first character 'd', Find it!

② Try looking for the second character 'o', fail to find it :(

Now instead, we recurse on the children and reduce the errors we can make by 1.

Auto Correct



[dad, dads, dab, dude]

Steps

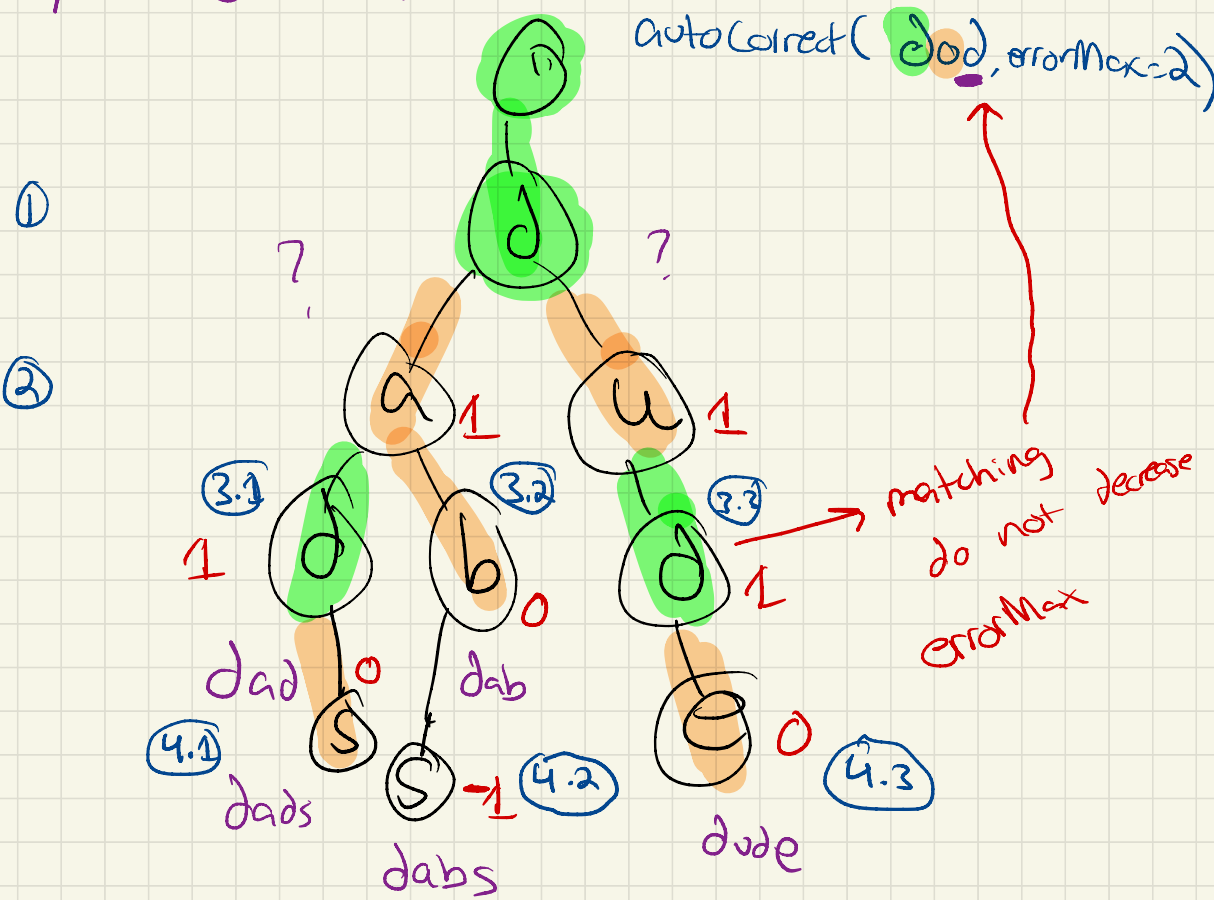
③ Look for the third (and last) character, 'd'.

[3.1] Find it, add that value into our list

[3.2] We don't find it, reduce errorMax, add value into our list

[3.3] Find it, don't reduce errorMax, keep recursing

Auto Correct



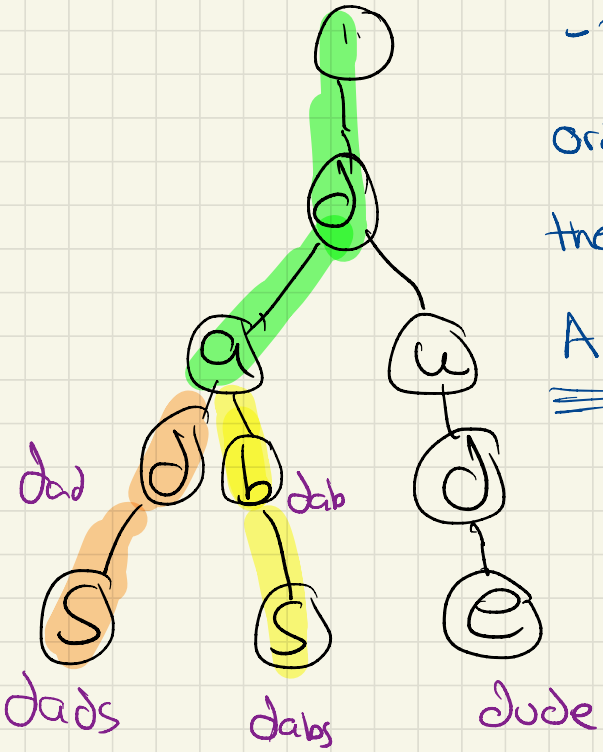
[dad, dads, dab, dude]

Steps

④ No more letters to traverse through, however in some cases we still have some errors to play with.

[4.1] Find an 's' with value 'dads', add it into our list,
[4.2] Find an 's' with value 'dabs', however no more errors
[4.3] Find an 'e' with value 'dude', add it into our list.

Autocomplete



- Traverse in sorted order by using the constant ALPHABET

autocomplete(da, N=10)

[dab, dabs, dad, dads]

autocomplete (da, N=2)

[dab, dabs]

- only return the first 2 found in sorted order.