A lot of valuable hints for Programming Assignment 5:

The following figures and steps should help you to write your code for the insert function for Assignment 5. These steps assume that you already have good understanding of Trie data structure and you have understood the codes we have discussed in the class. It also assumed that you have read through the assignment description and you know what you need to do.

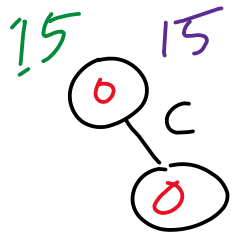
1. Make sure you have attended the Trie lecture (if not watch the recording)
2. Make sure you have attended the lecture where we have discussed the codes (if not, review that)
3. Review the lecture notes and the uploaded code
4. Now, read through the entire assignment description
5. Read the hints provided at the end of the assignment description
6. Now read and observe the following figures to understand how to design your insert function.

A node stores:

- freq; //similar to count
- sum_freq; //The sum of frequencies for which this string is a prefix of words in the dictionary, including itself.
- cur_max_freq; //current maximum frequency of any child node. It stores maximum sum_freq within this node's 26 children
- * next[26]; //node pointer for 26 children

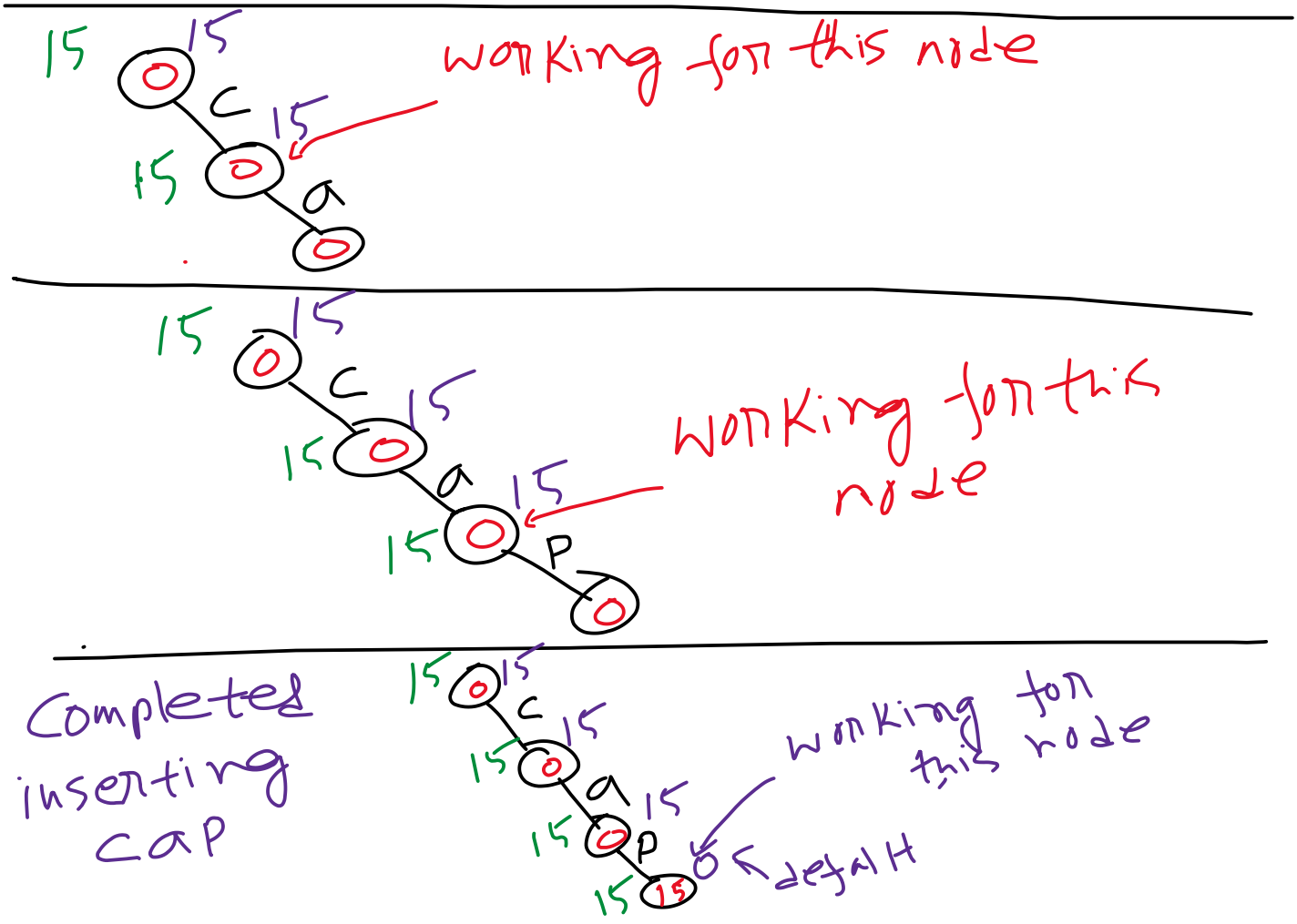
***(I will use the corresponding color shown above to represent those values in the figure):***

## Insert: cap 15 (let's say 15 is count)

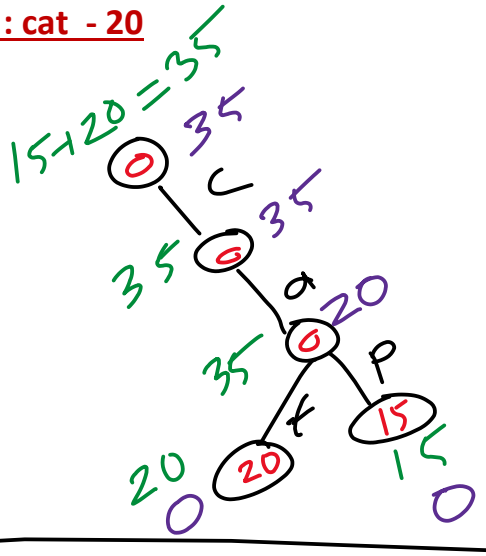



**Working for the first node (Most of them are also applicable for the other nodes too).**

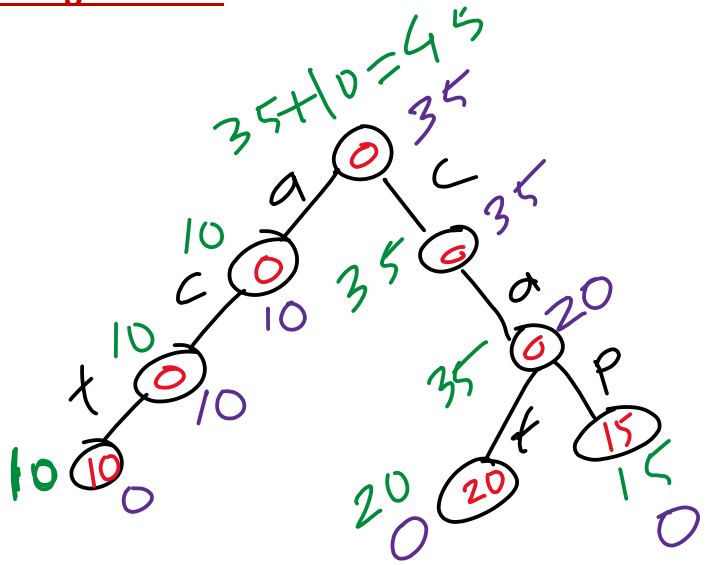
- freq is zero by default. It will remain 0 as we have not reached to complete word "cap" yet.
- Increase sum_freq based on count
- As child c is null, create a node and point that node from c.
- *Now you need to calculate cur_max_freq. But as we don't know the children's sum_freq yet, we calculate ahead what would be the sum_freq by adding count to the existing sum_freq of the child node in that path. So, it is 0+15 = 15. Based on the that, decide whether you need to update the cur_max_freq. As 15>0, update it to 15*
- Use the similar approaches for the following steps too!



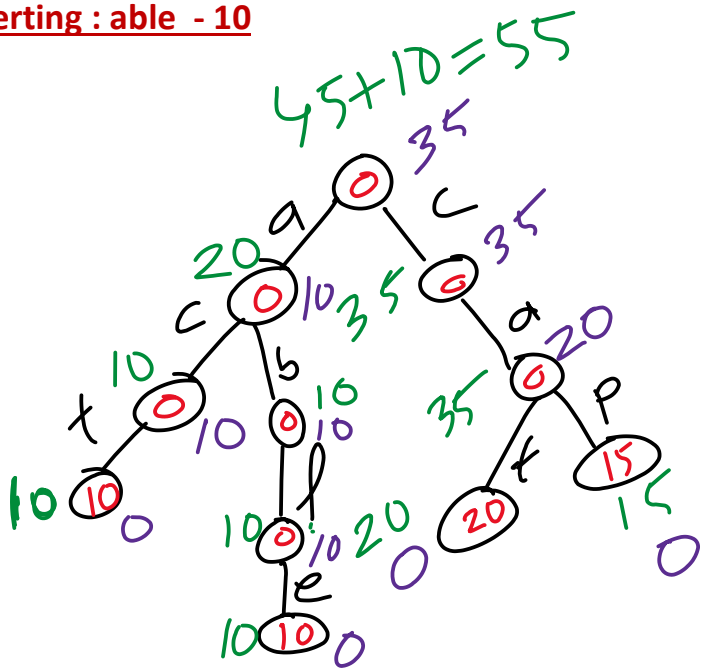
**After inserting : cat - 20**



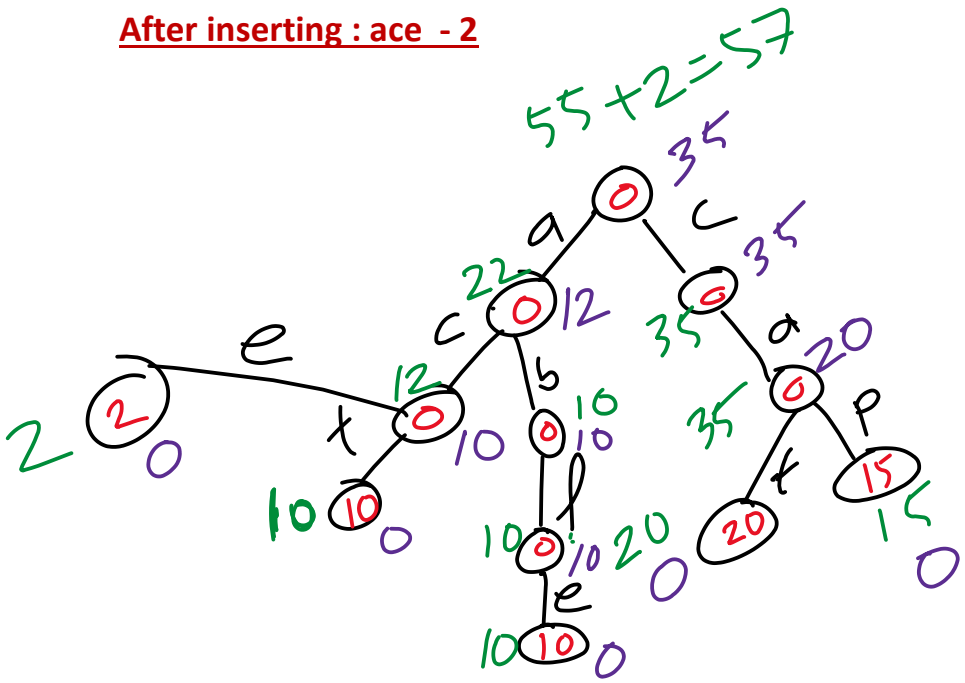
**After inserting: act - 10**



**After inserting : able - 10**



**After inserting : ace - 2**



## Example Queries:

## General steps:

*start traversing the path until you reach to the end of your search query. Then decide which children has the sum_freq similar the cur_max_freq of the current node and print those characters*

*Note that there can be situation where multiple children have similar sum_freq matching with the cur_max_freq of the current node. In that case all of those characters need to be printed.*

## Query: a

**Visit the node where a is pointing to.**

**From there it has two children and the node c is pointing to has the sum_freq similar to cur_max_freq.**

**So, the answer is c**

**Query: ac**

**After traversing, the cur_max freq  = 10**

**Matching children with sum_freq is pointing to t.**

**So, the answer is t.**