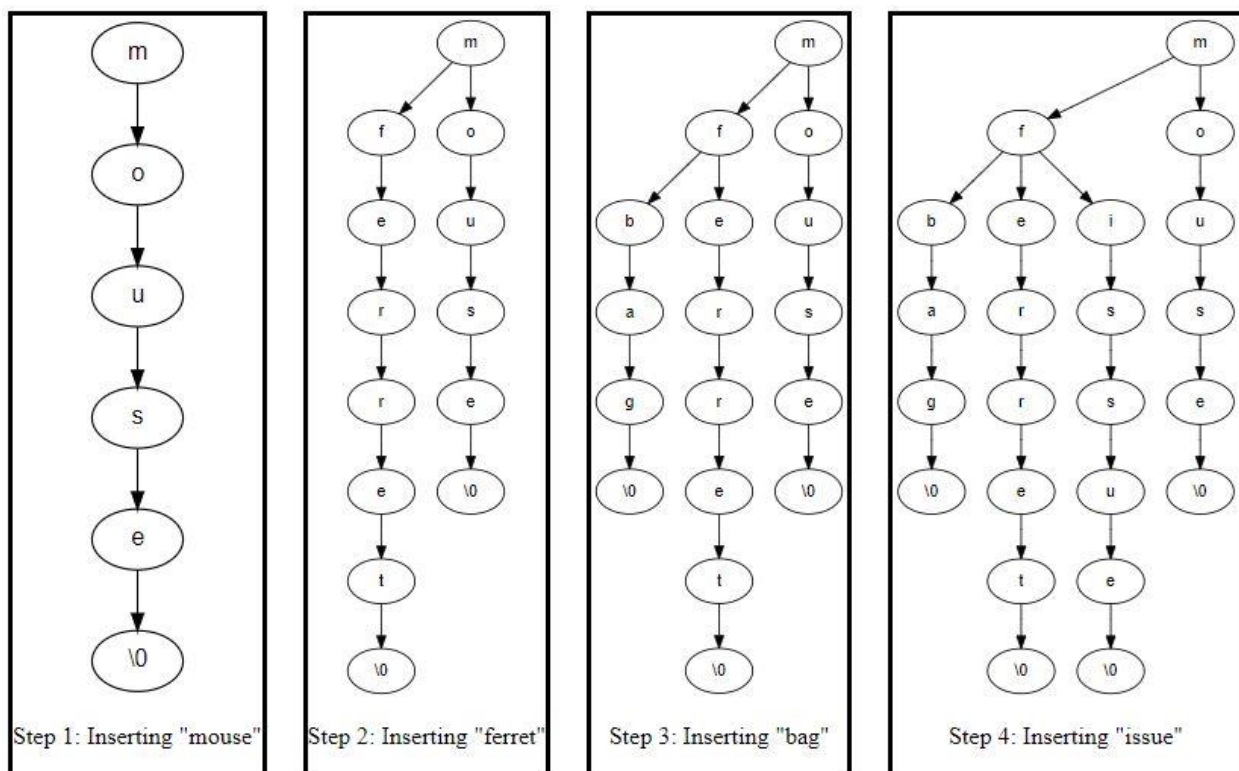


TERNARY TREE DEFINITION

A ternary tree is similar to a binary tree, except that nodes have three children instead of two. The tree is constructed in such a manner that the entire subtree anchored at the center child of a node contains all suffixes associated with the prefix that led to the current node. For example, if the prefix is “succ”, then the current node contains the second “c” of the prefix, and the subtree attached to the center child of the current node would contain the suffixes “eed”, “ess”, and “umb” to create the words “succeed”, “success”, and “succumb” (note: this list is not complete for the given example). The left subtree contains all suffixes of “suc” where the next letter is less than the current node, and the right subtree contains all suffixes of “suc” where the next letter is greater than the current node. See the next section for examples of how to build a ternary tree. This should help understand how to navigate it as well.

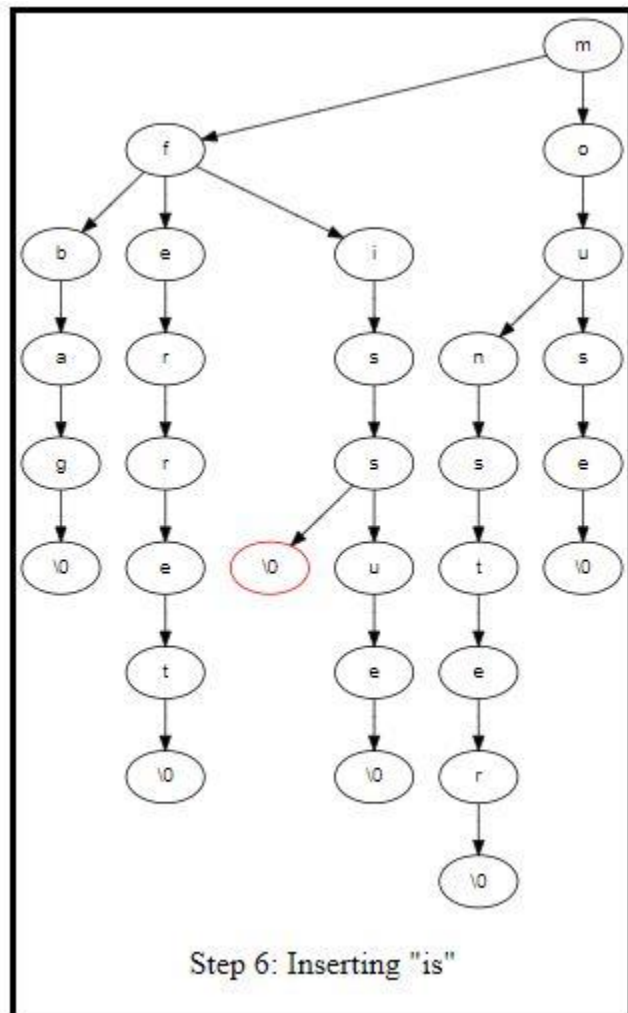
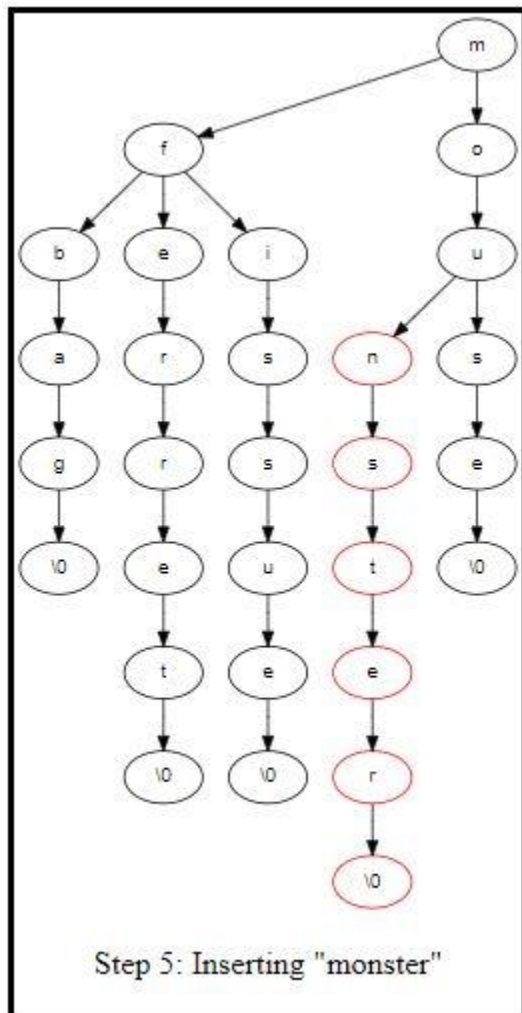
TERNARY TREE INSERTION

The series of steps below illustrate how a ternary tree is constructed by first inserting the word "mouse", then "ferret", then "bag", and finally inserting the word "issue".



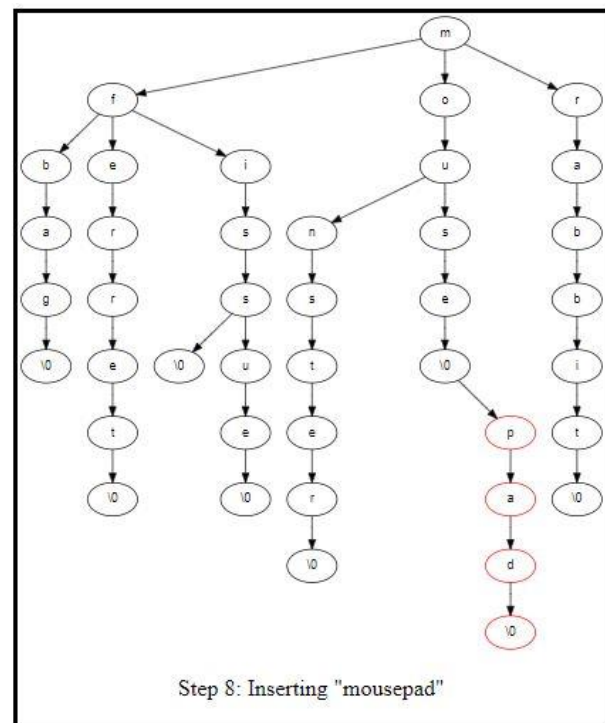
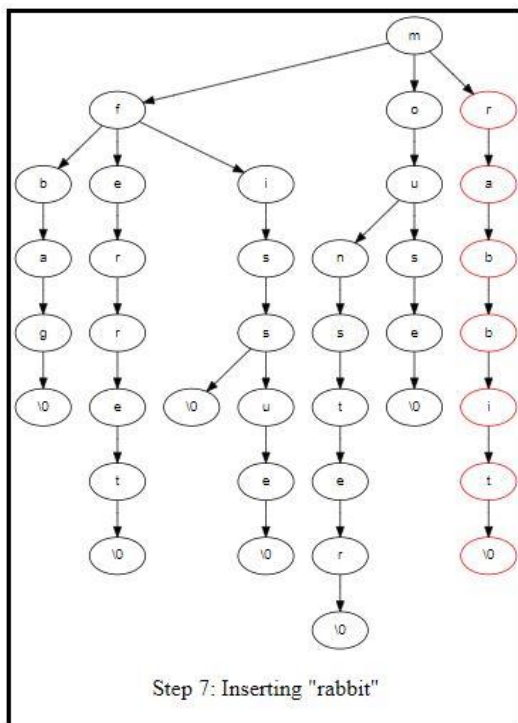
Note that in this and the remaining figures, the '\0' character used to mark the end of a string.

The next series of steps illustrates how the ternary tree grows by inserting the words "monster" and "is". Take special note as to how these words are inserted when some of the letters already exist in the tree. For example, in step 5, the 'm' and 'o' of mouse are also used in monster. Be sure to notice the branch for the 'n' is from the 'u' of mouse because the down branch from the 'o' must be followed to include 'o' in the word.

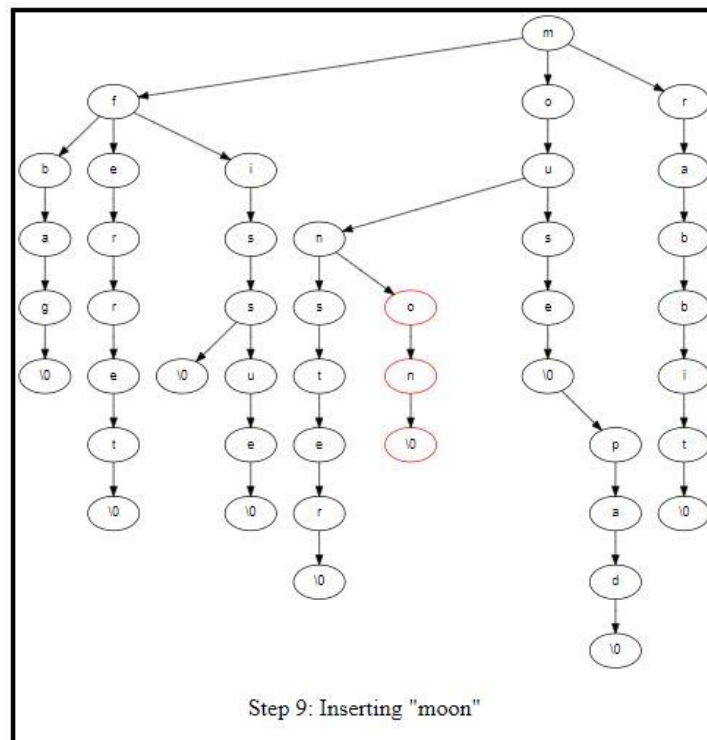


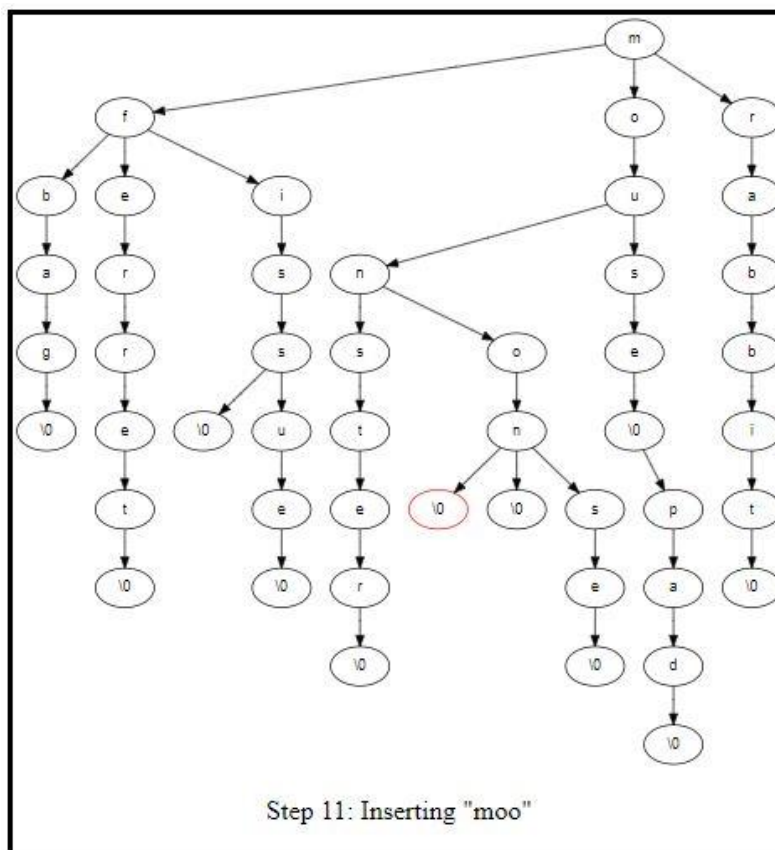
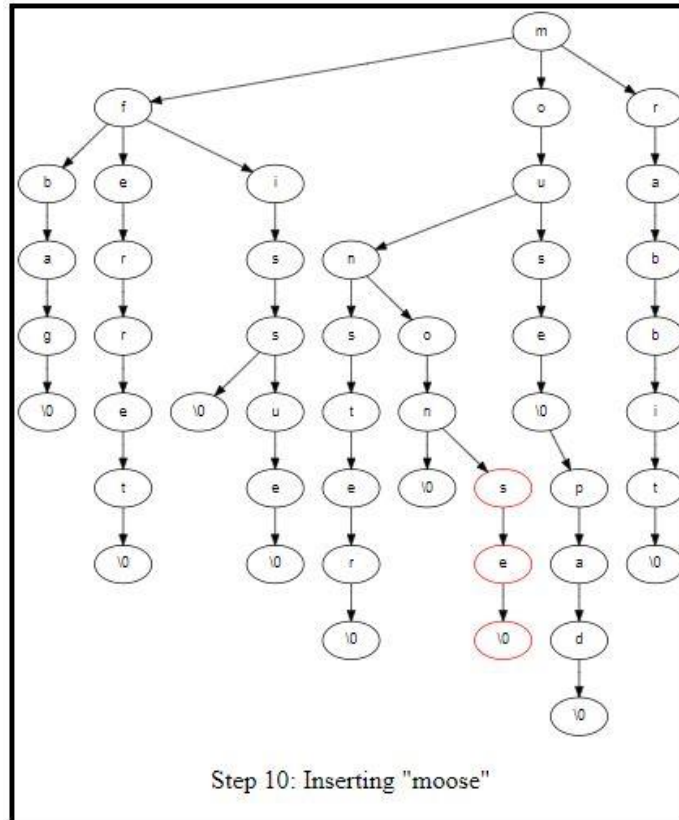
The word "is" shows the case where the necessary letters are already in the tree so the null character is added to the left of the second 's' in issue. Again note carefully the down branch from the first 's' must be followed, so the null character is left of the second 's'.

Steps 7 and 8 show the insertion of "rabbit" and "mousepad". The 'p' of mousepad is to the right of the '\0' at the end of "mouse".



Finally, "moon", "moose", and "moo" further illustrate the insertion process.





TERNARY TREE SEARCH

A search is a node-by-node traversal of the fully populated tree in conjunction with a character-by-character traversal of the search string. This is done by comparing the current character of the search string with the character at the current node in the tree. Similar to a typical tree search, if less, the search goes left; if greater, the search goes right. In both of these cases, the current character in the search string does not move forward. If the characters match, the search moves down in the tree and also moves forward one character in the search string.

If the end of the search string is reached at the same time a node containing the null character is reached, the search is successful. If the end of the search string is reached on a node that does not contain the null character or if a null tree node is reached before the end of the search string, the search fails.

To search for the string "festive" in the tree shown above, start with a current node equal to the root of the tree:

1. Compare the first letter in "festive" to the character in the current node ('m'); move the current node left if 'f' is less than 'm', and continue searching for "festive" (do not consume the 'f' in "festive").
2. Compare 'f' to the character in the current node ('f'); they match, so consume the 'f' in "festive", move the current node down, and continue searching for "estive".
3. Compare 'e' to the character in the current node ('e'); they match, so consume the 'e' in "estive", move the current node down, and continue searching for "stive".
4. Compare 's' to the character in the current node ('r'); it is greater, so move the current node right, and continue searching for "stive" (do not consume the 's' in "stive").
5. Because the 'r' had no right child, the current node is now a null node (of the tree, not a null character), so the search fails.