**Description**: This C++ article involves insertion of nodes to a Binary tree

**Link to source code**: https://github.com/pyskmr/D4datastructures/tree/master/binary%20tree

First of all, what is binary tree?

A binary tree is a tree in which each node can have at most two nodes below it and the node(s) connected to the node is/are called child of the above node whereas the above node is called parent to the child nodes. Childs (not children) sharing a common parent are called siblings.

Parent

Right child

Left child

CONDITIONS IN BINARY TREE

Usually the left child of a binary tree node is always lesser than the parent and right child is larger than the parent. So, the conclusions are

* The left subtree of any node always contains nodes with less data/value and the right subtree of any node always contains nodes bigger than the parent.
* The leftmost node contains the least element and rightmost contains the largest element.

**INSERTING A NODE IN BINARY TREE**

In this article we’re going to see how to insert a node in a binary tree.

Some cases related to insert a node in a binary tree are following

* Case 1: what if there is no tree or the node which is about to get inserted is the very first node? In this case we will initial the new node and assign its address to the root of the tree (since the starting of tree is called root and a single node in a tree is by default its root)
* Case 2: if there is a tree, that is a root exist, then we have to compare the values of the root to the new node value let’s say temp node.
  + 1. Subcase 1: if the temp node value is greater than the node being compared then move to the right subtree by pointing to the right pointer of the current node being compared.
    2. Subcase 2: if the temp value is smaller or equal to the node being compared then move to the left subtree by pointing to the left of current node being compared.

Keep repeating this until the pointer points to the NULL value but also do track of the node last compared just before hitting NULL.

At last do one more check to the last visited node, if the value is more than point the right pointer to the temp node if not then point the left one.

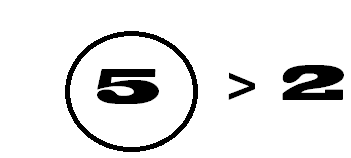
Example

1. 5 is inserted since no tree was there (root ==NULL) before so 5 will be the root

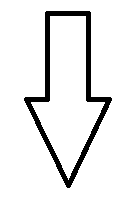


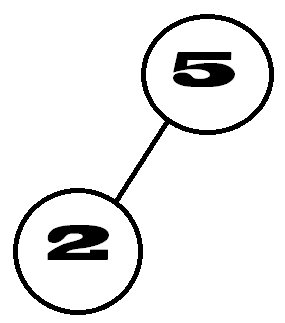
Case 1 is being satisfied

1. 2 is inserted

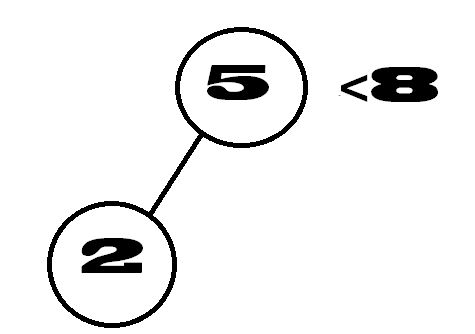


CASE 2 satisfied and then its subcase 2  
so we move to the left subtree, and since we found NULL there, we add it to the last node visited which is root node in this case

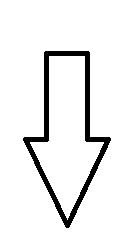


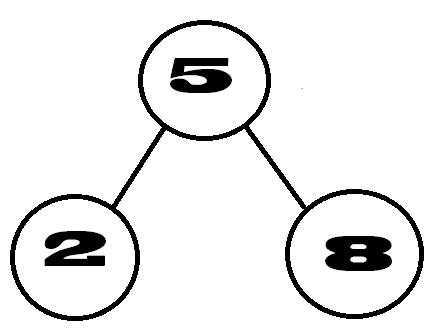


1. 8 is inserted

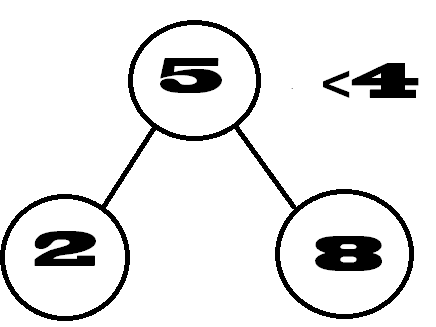


CASE 2 satisfied and then its subcase 1  
so we move to the right subtree, and since we found NULL there, we add it to the last node visited which is root node in this case



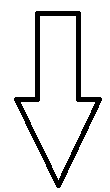


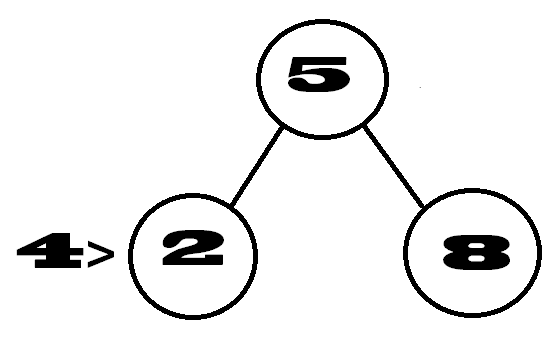
1. 4 is inserted



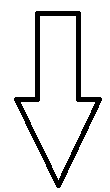
CASE 2 satisfied and then its subcase 2

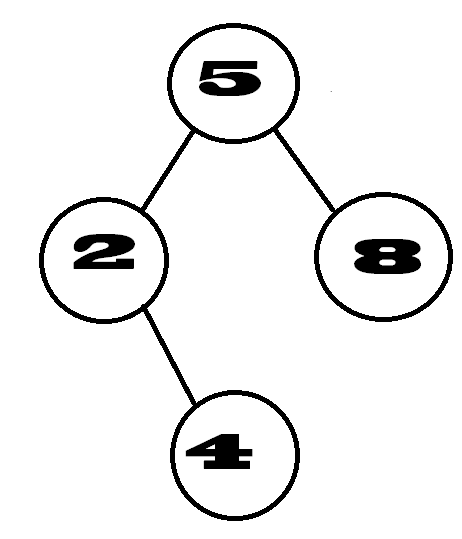
so, we will move to the left subtree.





Here we found that the subcase 1 is true so we will go to the left subtree of the visited node





And since we found NULL to the right subtree, we will add the new node in the last child visited which was 2 in this case.

And in the same way new nodes will be added to the tree.

**ANALYSIS OF INSERTING A NODE IN BINARY TREE**

* The best case complexity is O(1) and it will be when there is no tree.
* Worst case complexity is O(n) and it will be when all nodes are being inserted into the same branch and this will happen when we insert node in increasing or decreasing fashion.
* The average case complexity will be O(lg2(n)) since we have to traverse the height of tree to insert.