Binary Search Tree

Practice Programs - 2

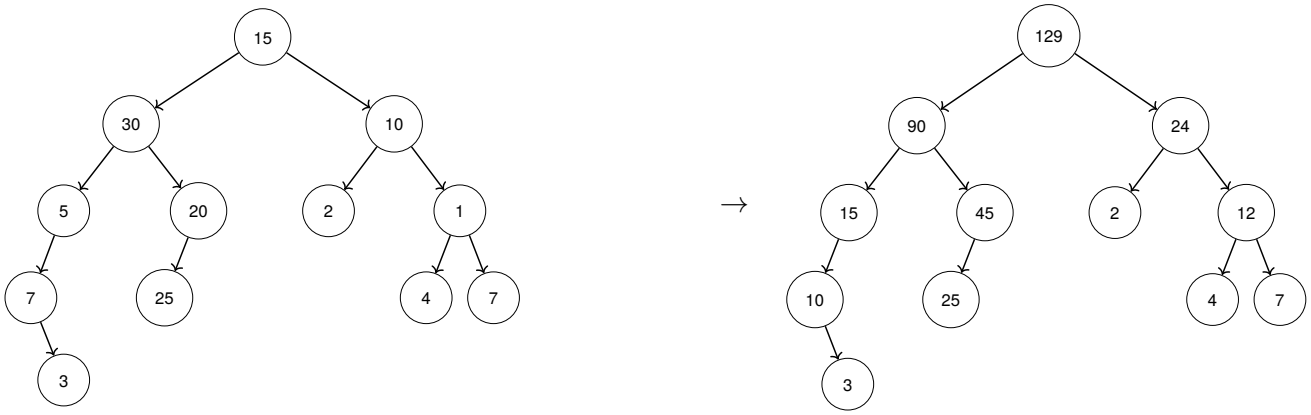
---

1. Given the root of a binary tree **T**, check whether the tree is a binary search tree or not.
2. Given the root of a binary search tree **T** having integers as the information part, and two integers `k1`, `k2` as input, write a function to get all the elements in the range from `k1` to `k2` in tree **T**.
3. Given the root of two binary search trees **T1** and **T2**, check whether the two trees contain same information or not. The structure of **T1** and **T2** can be same or different.
4. A binary tree is balanced, if for every node in the tree, the heights of the left and right children differ by at most 1. In other words, if we define the balancing factor of any binary tree as

   ```
   balancing_factor = height(left_subtree) - height(right_subtree)
   ```

   then the value of `balancing_factor` for each node of a balanced binary tree would be either -1, 0 or 1.

   Given the root of a binary tree **T**, write a function to check whether it is height balanced or not.
5. Given the root of a binary search tree **T** and two integers `a`, `b`, remove all elements which are outside the range `[a, b]`.
   Challenge problem: Removing the elements outside the range, modify the existing binary search tree containing all the elements inside the range.
6. Given the root of a binary search tree **T**, write a program to make another tree that would contain the values which are the sum of the present value in **T**, the values in the left subtree and the values in the right subtree. Following is a sample input tree (in left) and the modified tree (in right).
7. Given the root of a binary tree **T**, make a binary search tree with the values stored in the binary tree.
   Hints: You may employ some traversal techniques to get a sequence from a binary tree; the sequence can then be used to form the binary search tree.
8. Given the root of a binary search tree **T**, make a sorted doubly linked list with the nodes in the input binary search tree.
   Hints: Think of the traversal technique that gives a sorted sequence from a binary search tree.



9. Given the root of a binary search tree **T**, output the sum of the values stored in the nodes of the binary search tree.

10. Given the root of a binary search tree **T** and an integer **m** as input, modify **T** such that all elements in **T** having value <= **m** is incremented by 5 and all elements in **T** having value > **m** is decremented by 5.

11. Given the root of a binary search tree **T** and an integer **m** as input, find the mth largest element in the binary search tree.

12. Given this file, make a binary search tree with where the values to be stored is the length of the word and the count (i.e. the number of words having the same length.
    The following can be the structure type type:

```
struct btNode
{
    int length;
    int freq;
    struct btNode *left, *right;
};
```

- **General information**: For all the programs, you may consider taking the binary search tree as input from the user using the code of binary search tree that has already been shared.