Lab 7: Binary Search Tree

Data Structures and Algorithms

8 April, 2025

NOTE: This lab can be attempted in both C and C++, but the candidate MUST follow the criterion and the instructions specified for their language below. The `Node` class/struct should have the attributes/pointers for the data, the left child, the right child, and the parent. NO OTHER ATTRIBUTE IS ALLOWED FOR BOTH C AND C++

C++ Instructions

You are required to make a C++ Binary Search Tree class `BSTree` with datatype integer. The class should have only one attribute `root` which must be initialized with a `Node` class/struct pointer. The class constructor takes in a single integer parameter and initializes the root with the said integer as its data.

The class should have the following methods that can be called upon a `BSTree` object:

- Methods for printing InOrder, and PostOrder traversal of the tree. These methods have no parameters, returns nothing.
- Two separate methods for returning the Successor and Predecessor of a node.
- A Search method. Searches if a node is present in the tree with an input integer key. Return false or -1 if not found. Return true or the key if found.
- An Insert method. Inserts a new value in the tree.
- A Deletion method. Deletes the node with an integer key from the tree.

You can choose whether to write an iterative or recursive application of the methods above. In the main function, all of the above methods will be tested appropriately.

C Instructions

Implement a Binary Search Tree in C, using self implemented struct `Node`. Define the functions given below in the code, which will be tested later in the main function.

- A function to create a tree node. Takes in an integer parameter and returns `Node*` with data as the said parameter and all other pointers NULL.
- Functions for printing InOrder, and PostOrder traversal of the tree.
- A Search Function. Searches if a node is present in the tree with an input integer key. Also takes the input of tree root. Return false or -1 if not found. Return true or the key if found.
- Two separate function for returning the Successor and Predecessor of a node.
- An Insert function. Inserts a new value in the tree. Also takes the input of tree root.
- A Deletion function. Deletes the node with an integer key from the tree. Also takes the input of tree root.

You can choose whether to write an iterative or recursive application of the methods above.

Evaluation

In the main function, you are required to execute the following steps using the functions/methods you defined above.

```
- Create a Tree with root = 5
- Insert in Tree value = 2
- Insert in Tree value = 3
- Print InOrder traversal of Tree
- Print PostOrder traversal of Tree
- Delete from Tree value = 3
- Search the Tree for key = 3
- Insert in Tree value = 4
- Insert in Tree value = 6
- Insert in Tree value = 9
- Insert in Tree value = 7
- Print InOrder traversal of Tree
- Print PostOrder traversal of Tree
- Delete from Tree value = 6
- Delete from Tree value = 5
- Search in Tree value = 9
- Print InOrder traversal of Tree
- Print PostOrder traversal of Tree
```
