January 2024, CSE 106

Assignment 4

Binary Search Tree

A **Binary Search Tree (BST)** is a binary tree data structure that has the following properties:

1. The left subtree (if it exists) of a node contains only nodes with keys lesser than the node's key.
2. The right subtree (if it exists) of a node contains only nodes with keys greater than the node's key.
3. The left and right subtrees must also be a binary search tree.

We assume that all the keys will be unique.

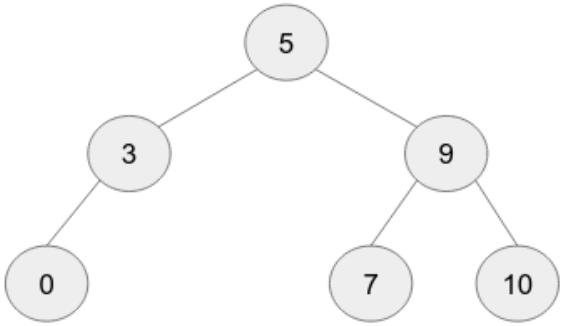

Figure 1: A sample binary search tree (BST)

For this assignment, you will implement a Binary Search Tree. Our desired BST has the following four operations,

1. **Insert:** It inserts a key in the BST.

2. **Delete:** It deletes the key in the BST if it exists in the BST. Remember, deletion in the BST can be divided into three cases.
    a. **The node to be deleted is the leaf:** Simply remove it from the tree.
    b. **The node to be deleted only has one child:** Copy the child to the node and delete the child.
    c. **The node to be deleted has two children:** Find the node's in-order successor. Copy contents of the inorder successor to the node and delete the inorder successor.

3. **Find:** Searches whether a key is present in our BST.

4. **Traversal:** Traverses the tree as specified. Traversal can be of three types.
    a. In-order
    b. Post-order
    c. Post-order

Your implementation should maintain the optimal time complexity of each operation. Roughly all operations (except traversal) should take `O(logn)` time. This means if you have `n` keys in the BST, your operation (insert/delete/find) should take around `log(n)` comparisons on average.

Input

- Take input from a file named `input.txt`.

- Each line in the input will specify one of the following operations:
    1. Insert (I) followed by an integer denoting the value of the node to be inserted

    2. Delete (D) followed by an integer denoting the value of the node to be deleted

    3. Find (F) followed by an integer denoting the value of the node to be searched for

    4. Traversal (T) followed by the type of traversal: In-order (1), Pre-order (2), or Post-order (3)

- You will generate output on a file named `output.txt`.

Output

After,

1. **Insert:** The tree's current state after insertion will be printed in the parentheses format shown below. For example, the parentheses format for the tree in Figure 1 will be written as,
   `5 (3 (0, _), 9 (7, 10))`
   For each sub-tree, first, the value of the root is shown, followed by its left and right sub-tree in a parenthesis.

2. **Delete:** Same as insert.

3. **Find:** Print “Found“ or “Not Found“

4. **Traversal:** Print the keys in a specified mode separated by space.
   An in-order traversal in Figure 1 would be:

```
0 3 5 7 9 10
```
A pre-order traversal in Figure 1 would be:
```
5 3 0 9 7 10
```

Please refer to the sample test cases to better understand the input-output format.

Sample Inputs	Sample Outputs
I 20	20
D 20	_
I 20	20
I 25	20 (_, 25)
I 30	20 (_, 25 (_, 30))
I 15	20 (15, 25 (_, 30))
I 22	20 (15, 25 (22, 30))
I 5	20 (15 (5, _), 25 (22, 30))
I 10	20 (15 (5 (_, 10), _), 25 (22, 30))
I 3	20 (15 (5 (3, 10), _), 25 (22, 30))
I 6	20 (15 (5 (3, 10 (6, _)), _), 25 (22, 30))
I 27	20 (15 (5 (3, 10 (6, _)), _), 25 (22, 30 (27, _)))
F 30	Found
F 33	Not Found
D 25	20 (15 (5 (3, 10 (6, _)), _), 27 (22, 30))
D 30	20 (15 (5 (3, 10 (6, _)), _), 27 (22, _))
D 20	22 (15 (5 (3, 10 (6, _)), _), 27)
D 10	22 (15 (5 (3, 6), _), 27)
T 1	3 5 6 15 22 27
T 2	22 15 5 3 6 27
T 3	3 6 5 15 27 22

Note

- Your task is to complete the implementations of the functions in `bst.h`. You only need to edit this file by replacing the comments "`//write your codes here`" with your codes.
- You are provided a file `main.cpp` to evaluate your implementation. Do not edit any line of code in this file. You can enter your inputs in the `input.txt`, and all the outputs will be generated in `output.txt`. The sample input is already given in `input.txt`.

Submission

1. Create a directory with your 7-digit student ID as its name.
2. Put all the source files (`.cpp/.hpp/.c/.h` files) only into the directory created above.

3. Zip the directory (compress in `.zip` format. Any other format like `.rar`, `.7z`, etc. are not acceptable).
4. Upload the `.zip` file in Moodle.
5. Submission Deadline: November 03, 23:59 PM
6. **DO NOT COPY solutions from anywhere (your friends, seniors, the internet, etc.). Any form of plagiarism (irrespective of source or destination) will be penalized severely.**

Mark Distribution

Task Details	Marks
Insert	25
Delete	30
Find	20
Traversal	25
Total	100