Data Structures Lab Fall 2023

Lab 10

Binary Search Trees

Task # 1

For this task, you need to implement a Binary Search Tree (BST). You will have to write the following Classes and functions.

1 The basic node class will have left and right pointers alongside T data.

```
template <typename T> class
binaryTreeNode   {   T   data;
binaryTreeNode<T>*       llink;
binaryTreeNode<T>* rlink;
    binaryTreeNode() : data(T()), llink(nullptr), rlink(nullptr) {}
};
```

2 The Tree class will consist of root.

```
template    <typename    T>
class bSearchTreeType {
   binaryTreeNode<T>* root;
}
```

3 The Tree class will consist of the following functions as well

a. ***void insert(T insertItem)***

You will insert the appropriate item in left or right subtree. If value is smaller than root , then value will be in left subtree, otherwise it will be in right subtree

b. ***void search (T serachItem)***

This function will search the item, in left or right subtree according to the value

c. ***void inoderTraversal(Node <T> * root)***

This function will use root to perform inorder traversal same as we did in BinaryTree implementation

*You have to write the same for pre and post-order traversals

d. ***void inoderTraversal()***

This function will just call the above function with root.

*You have to write the same for pre and post-order traversals

e. ***void deleteNode(int value)*** **Pointer Initialization:**

Initialize current and trailCurrent pointers to traverse the tree. current moves through the tree, while trailCurrent tracks the parent node of current.

Check if the Tree is Empty:

Verify if the tree is empty by checking if root is NULL. If the tree is empty, indicate that deletion cannot occur on an empty tree.

Search for the Node to Delete:

Loop through the tree until the current pointer becomes NULL or the node to delete (deleteItem) is found.If the node's data at current matches the deleteItem, set found to indicate the item is found. Move current to the left or right child based on whether deleteItem is smaller or larger than the current node's data.

Handle Node Found or Not Found:

If the deleteItem is not found (current becomes NULL), indicate that the item to be deleted is not in the tree
.If the deleteItem is found (found is true): Determine the type of node found (root or non-root) and initiate the deletion process by calling deleteFromTree.

f. void deleteFromTree(Node <T> & treenode)*

Pointer Initialization:

Initialize current, trailCurrent, and temp pointers of type binaryTreeNode<T> to navigate and delete nodes in the tree.

Check for Node Existence:

Verify if the node to be deleted (p) is NULL.

If p is NULL, display an error message stating that the node to be deleted is NULL.

Check Different Node Deletion Scenarios:

Examine multiple scenarios based on the structure of the node to be deleted:

If the node is a leaf node (has no children), delete it by setting it to NULL and freeing the allocated memory.

If the node has only a left child, update pointers to bypass the node and delete it.

If the node has only the right child, adjust pointers accordingly and delete the node.

If the node has both left and right children, find the rightmost node in the left subtree, copy its data to the current node, adjust pointers, and delete the rightmost node.

Task # 2

(You can attempt this only when the first question is completely executed)

You are given a binary search tree (BST), and your task is to write a program that counts the number of leaf nodes in the tree.

Your program should include the following:

1. Input:

- A binary search tree represented by a collection of nodes. Each node contains an integer value, and the tree is constructed following the BST properties.

2. Output:

An integer representing the count of leaf nodes in the given BST.

int countLeafNodes(Node* root); This function takes the root node of the BST as input and returns an integer count representing the number of leaf nodes in the tree.

int countleaf(); This function will call countLeafNodes(root) and return count variables that will count the number of leaf nodes.

Example: The BST should look like:

```
         50

       /     \

   30         70

  /   \        /    \

20      40    60    80
```

There are 4 leaf nodes: 20, 40, 60, and 80 so, count should be 4.

6. Constraints

- The input tree is a valid binary search tree.
- The integer values in the tree are distinct.

The tree may be empty (no nodes), in which case the count of leaf nodes is zero.

Home task:

You can practice more questions from the website:

https://www.geeksforgeeks.org/top-50-tree-coding-problems-for-interviews/