Binary Search Tree

1 Data Structures Used

The Binary Search Tree (BST) is represented using structures containing left, right and parent pointers in addition to an integer data value. The tree itself has a pointer to the one of the nodes, denoting the root node. All the operations listed below are implemented as member functions of the BST class.

2 Processing

2.1 Insert Element

Here, the tree is traversed, starting at the root, to find a suitable leaf node where the new element can be attached. This traversal is done so that the Binary Search Tree property is satisfied, ie, left child is always less than the parent and the right child is always greater than or equal to the parent. The insertion algorithm takes a running time of O(h) where h is the height of the tree, since comparisons are made at each level, till a suitable leaf node is found for insertion.

2.2 Delete Element

Delete algorithm is implemented for three cases:

2.2.1 The node to be deleted is a leaf node

Here, the only action is delete the node and set the left/right pointer of it's parent to NULL

2.2.2 The node to be deleted has one child

Here, the left/right pointer of the parent is to be modified so that it points to the child of the node to be deleted

2.2.3 The node to be deleted has both children

In this case, we have to replace the node to be deleted with the maximum element in the left sub-tree of that node.

The first two cases take constant time for execution. In the third case, the finding the maximum element in the left sub-tree takes O(h) time. Also, on entry to the routine, a search is done in-order to locate the element to be deleted in the tree. This

also takes a running time of O(h). Hence, the worst case running time of the delete operation is O(h).

2.3 Search for an Element

Here, the tree is traversed till a node with the search key is found. If such a node is not found, an error is returned. Since this procedure makes one comparison at each level of the BST, it has a worst case running time of O(h), where h is the height of the BST. The worst case time occurs when the queried node is a leaf node of the longest sub-tree.

2.4 Find Predecessor

Here, the in-order predecessor of a specified node is returned. That is, if the node has a left sub-tree, the predecessor is the largest element in the left sub-tree. If it doesn't have a left sub-tree, the successor is the lowest ancestor, whose right child is also an ancestor of the selected node. Running time of this routine is O(h) since the search follows simple paths up the tree or down the tree, as the case may be.

2.5 Find Successor

Here, the in-order successor of a specified node is returned. That is, if the node has a right sub-tree, the successor is the lowest element in the right sub-tree. If it doesn't have a right sub-tree, the successor is the lowest ancestor, whose left child is also an ancestor of the selected node. Running time of this routine is O(h) since the search follows simple paths up the tree or down the tree, as the case may be.

2.6 Display In-order Traversal

The in-order traversal is done using a recursive algorithm, as listed below:

```
Starting from root of the tree (startNode=treeRoot),
inorderTreeWalk (startNode)
{
   print inorderTreeWalk (startNode->left)
   print startNode->value;
   print inorderTreeWalk (startNode->right)
}
```

Since the recursive function is called twice for each node in the tree, this procedure takes THETA(n) time to run.

2.7 Find Minimum

This function finds out the minimum element in the BST. This is done by following the left child of each node starting at the root node, till a leaf node is reached. Since this procedure makes one comparison at each level of the BST, it has a worst case running time of O(h), where h is the height of the BST.

2.8 Find Maximum

This function finds out the maximum element in the BST. This is done by following the right child of each node starting at the root node, till a leaf node is reached. Since this procedure makes one comparison at each level of the BST, it has a worst case running time of O(h), where h is the height of the BST.