Data Structure 2018
Lab 10

Mohammad Sadegh Najafi- Jonghyun Lee

May 17, 2018

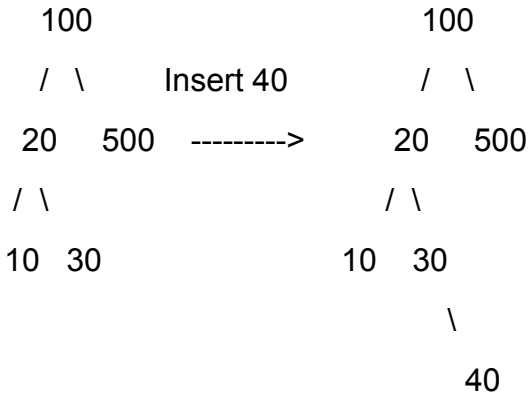
Binary Search Tree

Binary Search Tree, is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys lesser than the node’s key.
- The right subtree of a node contains only nodes with keys greater than the node’s key.
- The left and right subtree each must also be a binary search tree.
- There must be no duplicate nodes.

Insert

A new key is always inserted at leaf. We start searching a key from root till we hit a leaf node. Once a leaf node is found, the new node is added as a child of the leaf node.



Delete

When we delete a node, three possibilities arise.

1) *Node to be deleted is leaf:* Simply remove from the tree.

2) *Node to be deleted has only one child:* Copy the child to the node and delete the child

3) *Node to be deleted has two children:* Find inorder successor of the node. Copy contents of the inorder successor to the node and delete the inorder successor. Note that inorder predecessor can also be used.

Today’s Task 1

Your task to implement `BinarySearchTree` with insert and delete methods And use inorder traversal for printing the output.

The main method should be as same as the file which is posted in ETL.

Expected Output

```
Inorder traversal of the given tree
20 30 40 50 60 70 80
Delete 20
Inorder traversal of the modified tree
30 40 50 60 70 80
Delete 30
Inorder traversal of the modified tree
40 50 60 70 80
Delete 50
Inorder traversal of the modified tree
40 60 70 80
```

Today’s Task 2

Start/Finish Lab 09.

The solution is already uploaded, but try implementing it yourself.

Also, you cannot use Java’s PriorityQueue as in the solution. Please implement your own min heap.

Submission

Upload both your `BinarySearchTree.java` and `MYhuffman.java` at ETL.

You can submit your code until 9 pm.