

Statement Purpose:

Purpose of this Lab is to familiarize the students with the use of Binary Search Trees data structure in writing simple Java programs. Another aim is to teach the students how to implement Binary Search Tree data structure using linked list. The students are given small tasks related to BST which they complete during the lab session under the supervision of the lab instructor. This helps them understand the concepts well which they learn in their lectures.

Activity Outcomes:

The students will learn how to write methods related to some actions performed on Binary Search Trees. They will understand how to write simple methods to

- print the nodes of BST in postorder form
- print the nodes of BST in descending order i.e. largest to smallest form
- count number of nodes which are divisible by 7
- modify all nodes by adding some specific value

Theory Review (10 Minutes):

Binary Search Tree

In computer science, **Binary Search Trees (BST)**, sometimes called **ordered** or **sorted binary trees**, are a particular type of data structures that store "items" (such as numbers, names etc.) in memory. They allow fast searching, addition and removal of items.

Binary search trees keep their keys in sorted order, so that lookup and other operations can use the principle of binary search. When looking for a key in a tree (or a place to insert a new key), they traverse the tree from root to leaf, making comparisons to keys stored in the nodes of the tree and deciding, based on the comparison, to continue searching in the left or right subtrees.

Applications of Binary Search Tree

In computer science, BST are most commonly used for

- Creating dictionary
- Making decision trees e.g. for sorting some numbers
- Creating Min-Heap or Max-Heap to be used in many algorithms



Commonly used operations on Binary Search Trees

Commonly used operations on BST are as follows:

1. insert

This process inserts the node in the BST in its proper location i.e. if it is less than its parent, it will be inserted on the left of its parent, otherwise it is inserted on the right of its parent.

2. remove

This process deletes any node from the BST.

3. search

This process returns true if any value is present in the tree otherwise it returns false. During this process, it compares the value to be searched with the parent. If it is found here, it returns true. If not found, it checks whether the value to be searched is less than or greater than the parent. If it is less than the parent, it recursively searches the left subtree otherwise it recursively searches the right subtree.

4. traverse

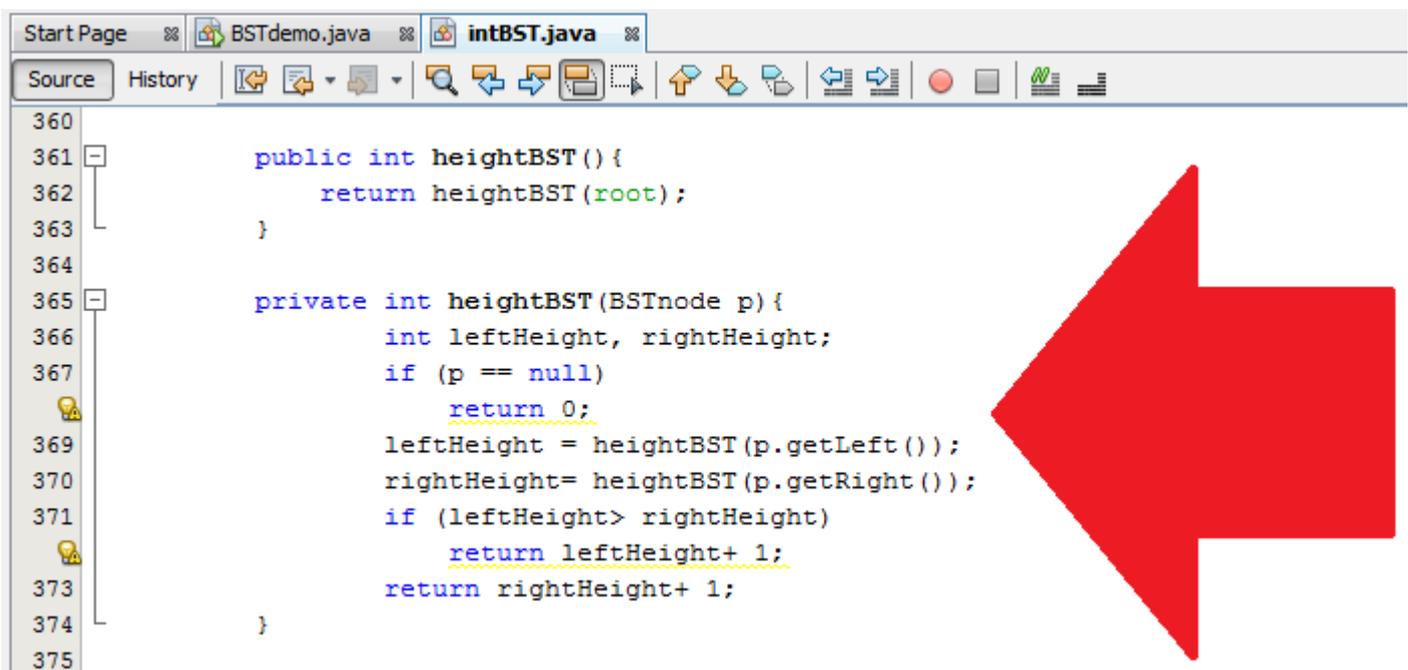
This process visits each and every node of the tree. It can traverse in pre-order, post-order or in-order.



Practice Activity with Lab Instructor: (10 Minutes)

BST Implementation using linked list:

1. Create a project with the name BSTProject
2. Create a package with the name TreesPackage within this project
3. Copy BSTnode class in this package
4. Copy intBST class in this package and add the following method in this class



```
360
361     public int heightBST() {
362         return heightBST(root);
363     }
364
365     private int heightBST(BSTnode p) {
366         int leftHeight, rightHeight;
367         if (p == null)
368             return 0;
369         leftHeight = heightBST(p.getLeft());
370         rightHeight = heightBST(p.getRight());
371         if (leftHeight > rightHeight)
372             return leftHeight + 1;
373         return rightHeight + 1;
374     }
375
```

5. Copy BSTdemo class in this package and add the following lines of code in showMenu() method of this main class.

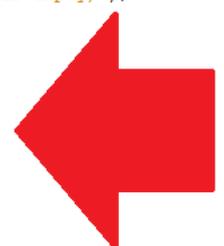


```
Start Page  BSTdemo.java  intBST.java
Source  History  [Icons]
138
139     public static void showMenu() {
140         System.out.println("|-----|");
141         System.out.println("|----- Binary Search Tree Menu -----|");
142         System.out.println("|-----|");
143         System.out.println("| 1. Insert an item into the tree |");
144         System.out.println("| 2. Delete an item from the tree |");
145         System.out.println("| 3. Search for an item in the tree |");
146         System.out.println("| 4. Find the parent of some node |");
147         System.out.println("| 5. Print the sum of all data values |");
148         System.out.println("| 6. Print an inorder traversal of the tree |");
149         System.out.println("| 7. Print height of the tree |");
150         System.out.println("| 8. Quit |");
151         System.out.println("|-----|");
152         System.out.println();
153         System.out.print("> Please enter your choice: ");
154     }
```



6. Add the following lines of code in the main() method of this main class

```
Start Page  BSTdemo.java  intBST.java
Source  History  [Icons]
114         else if (choice == 7) {
115             if (myTree.isEmpty()) {
116                 System.out.println("> Error: cannot print height (the tree is empty)");
117                 System.out.println();
118             }
119             else {
120                 System.out.println(">Height of tree is: "+myTree.heightBST());
121                 System.out.print("> ");
122                 System.out.println();
123                 System.out.println();
124             }
125         }
126         else if (choice == 8) {
127             System.out.println("> Goodbye!");
128             System.out.println();
129         }
130         else {
131             System.out.println("> Wrong selection. Try again.");
132             System.out.println();
133         }
134     } while (choice != 8);
135
136
```



7. When you run this program, it will display the following menu on the screen:



```
Output - BSTProject (run) ✖
run:
-----|
----- Binary Search Tree Menu -----|
-----|
| 1. Insert an item into the tree |
| 2. Delete an item from the tree |
| 3. Search for an item in the tree |
| 4. Find the parent of some node |
| 5. Print the sum of all data values |
| 6. Print an inorder traversal of the tree |
| 7. Print height of the tree |
| 8. Quit |
-----|

> Please enter your choice:
```

8. After insertingsome elements e.g. 50, 30, 60, 25, 40, 55, 58, 35, 45, 44 and 43 in the BST, if you select option 6, it will display the following output:

```
Output - BSTProject (run) ✖
-----|
----- Binary Search Tree Menu -----|
-----|
| 1. Insert an item into the tree |
| 2. Delete an item from the tree |
| 3. Search for an item in the tree |
| 4. Find the parent of some node |
| 5. Print the sum of all data values |
| 6. Print an inorder traversal of the tree |
| 7. Print height of the tree |
| 8. Quit |
-----|

> Please enter your choice: 6
>Inorder Traversal of nodes:
> 25, 30, 35, 40, 43, 44, 45, 50, 55, 58, 60,
```

9. Now if you select option 7, it will display the following output:



```
Output - BSTProject (run) %
-----
Binary Search Tree Menu
-----
1. Insert an item into the tree
2. Delete an item from the tree
3. Search for an item in the tree
4. Find the parent of some node
5. Print the sum of all data values
6. Print an inorder traversal of the tree
7. Print height of the tree
8. Quit
-----
> Please enter your choice: 7
> Height of tree is: 6
>
```

LAB EXERCISES: (60 Minutes)

1. Write a method postorder() in intBST class which prints all nodes of the BST in postorder. Call this method from main class.

Sample Run:

If we insert the nodes 50, 30, 63, 25, 42, 55, 58, 35, 45, 44 and 43 and choose the option 8, it will display the following output:



```
Output - BSTProject (run) ✖
-----
Binary Search Tree Menu
-----
1. Insert an item into the tree
2. Delete an item from the tree
3. Search for an item in the tree
4. Find the parent of some node
5. Print the sum of all data values
6. Print an inorder traversal of the tree
7. Print height of the tree
8. Print postorder traversal of the tree
9. Print nodes in descending order
10. Count nodes which are divisible by 7
11. Modify all nodes by adding some value
12. Quit

> Please enter your choice: 8
>Postorder traversal of nodes is:
> 25, 35, 43, 44, 45, 42, 30, 58, 55, 63, 50,
```

2. Write a method `descendingorder()` in `intBST` class which prints all nodes of the BST in descending order i.e. largest to smallest form. Call this method from main class.

Sample Run:

If we insert the nodes 50, 30, 63, 25, 42, 55, 58, 35, 45, 44 and 43 and choose the option 9, it will display the following output:



```
Output - BSTProject (run) ✖
-----
Binary Search Tree Menu
-----
1. Insert an item into the tree
2. Delete an item from the tree
3. Search for an item in the tree
4. Find the parent of some node
5. Print the sum of all data values
6. Print an inorder traversal of the tree
7. Print height of the tree
8. Print postorder traversal of the tree
9. Print nodes in descending order
10. Count nodes which are divisible by 7
11. Modify all nodes by adding some value
12. Quit

> Please enter your choice: 9
>Nodes of tree in descending order . :
> 63, 58, 55, 50, 45, 44, 43, 42, 35, 30, 25,
```

3. Write a method `count7()` in `intBST` class which counts and returns the number of nodes which are divisible by 7. Call this method from main class.

Sample Run:

If we insert the nodes 50, 30, 63, 25, 42, 55, 58, 35, 45, 44 and 43 and choose the option 10, it will display the following output:



```
Output - BSTProject (run) ✖
|-----|
|----- Binary Search Tree Menu -----|
|-----|
| 1. Insert an item into the tree |
| 2. Delete an item from the tree |
| 3. Search for an item in the tree |
| 4. Find the parent of some node |
| 5. Print the sum of all data values |
| 6. Print an inorder traversal of the tree |
| 7. Print height of the tree |
| 8. Print postorder traversal of the tree |
| 9. Print nodes in descending order |
| 10. Count nodes which are divisible by 7 |
| 11. Modify all nodes by adding some value |
| 12. Quit |
|-----|
> Please enter your choice: 10
>Number of nodes divisible by 7 is: 3
>
```

4. Write a method `modifyAllNodes()` in `intBST` class which modifies all nodes of BST by adding some specific value in them. Call this method from main class.

Sample Run:

If we insert the nodes 50, 30, 63, 25, 42, 55, 58, 35, 45, 44 and 43 and choose the option 11, it will display the following output:



```

Output - BSTProject (run) ❏
-----
Binary Search Tree Menu
-----
1. Insert an item into the tree
2. Delete an item from the tree
3. Search for an item in the tree
4. Find the parent of some node
5. Print the sum of all data values
6. Print an inorder traversal of the tree
7. Print height of the tree
8. Print postorder traversal of the tree
9. Print nodes in descending order
10. Count nodes which are divisible by 7
11. Modify all nodes by adding some value
12. Quit

> Please enter your choice: 11
What value you want to add in all nodes?
100
>All nodes of the tree have been successfully modified
>

```

Now if you choose option 8, it will display modified nodes in postorder as:

```

Output - BSTProject (run) ❏
-----
Binary Search Tree Menu
-----
1. Insert an item into the tree
2. Delete an item from the tree
3. Search for an item in the tree
4. Find the parent of some node
5. Print the sum of all data values
6. Print an inorder traversal of the tree
7. Print height of the tree
8. Print postorder traversal of the tree
9. Print nodes in descending order
10. Count nodes which are divisible by 7
11. Modify all nodes by adding some value
12. Quit

> Please enter your choice: 8
>Postorder traversal of nodes is:
> 125, 135, 143, 144, 145, 142, 130, 158, 155, 163, 150,

```

