CSCI 115 Lab

Week 10- Binary Search Tree

- Professor: Dr. Matin Pirouz
  Email: mpirouz@csufresno.edu

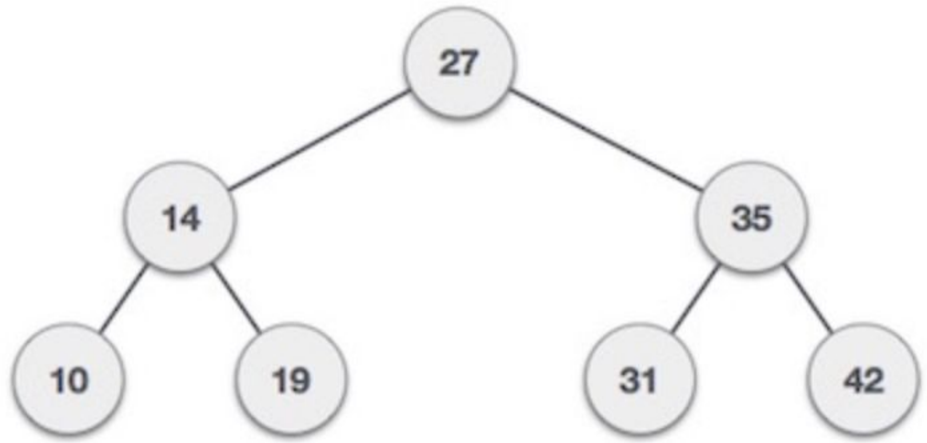
- TA: Shreeja Miyyar
  Email:shreejarao12@mail.fresnostate.edu

Table of Contents

Binary Search Tree

- A Binary Search Tree is a tree which has the following properties.
    - The value of all the nodes in the left subtree is less than the value of its node.
    - The value of all the nodes in the right subtree is greater than or equal to the value of its node.

- BST is a collection of nodes arranged in a way so that the properties of the BST is always maintained.



BST operations

- Search
    - Given a value to search, this operation returns the node if found.
- Insert
    - Give a value to insert, this operation inserts it to the leaf node.
- Find maximum value
    - Return the node which has the maximum value.
- Tree traversal - O(n) n is the size of the tree
    - Post order: left, right, root
    - Pre order: Root, left, right
    - In order: Left, root, right

Running time of basic operations on binary search trees

- On average:$\Theta(\lg n)$ The expected height of the tree is lgn
- In the worst case:$\Theta(n)$ The tree is a linear chain of n nodes

Search operation algorithm

- Start from the root
- If the element to be searched is less than the root, search recursively in the left subtree.
- If the element to be searched is greater than the root, search recursively in the right subtree.

```
bool FindNode(root, data){
  if root = NULL or data of root =data)
      then return root

   // If data is greater than root's data
   if (data of root < data)
      return FindNode(right of root, data);

   // If data is smaller than root's data
   return FindNode(left of root, data);
```

Insert operation algorithm

A new element is always inserted to the leaf of the tree.

- Start from the root
- If the element to be inserted is less than the root, recurse the left subtree.
- If the element to be inserted is greater than the root, recurse the right subtree.
- After reaching the end;
    - Insert the element at the left (if value is less than current node value), Else; Insert the element at the right side.

```
InsertNode(root, data)
    if  root = NULL
        create new node with data
        assign the new node as root
        return root
     // Traverse to the correct place and insert the node
     if (data < data in root)
       left of root = InsertNode(left of root, data)
     else
       right of rot = InsertNode(right of root, data)
     return root
```

Post order traversal algorithm

PrintTree(root)

- If root does not exist, return.
- Traverse to the left subtree (Recursively call PrintTree(root->left))
- Traverse to the right subtree (Recursively call PrintTree(root->right))
- Print the value of the root of the tree

Find maximum node algorithm

- Start from the root
- Traverse the right subtree, until the last right most node is reached.
- Return the node.

```
Largest(root)
    while (right of root !=NULL)
        root=right of root;
    return root.data;
```

Lab Assignment

Hints:

- Use the header file as a reference for writing functions
- Refer the algorithms in this slide for different BST operations.

Coding guidelines

- In the main function provide input elements to be inserted.
- Create a separate function for each operation required for the assignment.

Questions?