# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Spring 2022** |
|  | | | | | |
| Points |  | Document author: | **MingkuanPang** | | |
|  | Author’s email: | **Yafking20 @mail.fresnostate.edu. email** | | |
| Laboratory number: | **08 – Binary Search Tree** | | |
|  | | | | | |

**1. Statement of Objectives**

This lab asks for implementation of Binary Search Tree with its insert, find node, largest key, and post-order traversal operations.

**2. Experimental Procedure**

**BST ()**

A picture containing diagram

Description automatically generated

Bst () is the default constructor of Binary Search Tree. It sets root to NULL when the data structure is initialized.

**printTree()**

Text

Description automatically generated

printTree() print out the elements in the tree with post-order. It travels through the left subtree of the root until reach the leaf and print its value, then travels through the right subtree of the root until reach the leaf and print its value. Finally, print the root value.

**Insert Node ()**

Text

Description automatically generated

First, it checks if the node is NULL, if it is then it means there is an available position for insertion, therefore, assigns it to a new node with the parameter data. If it is not NULL, then compares its value to the parameter data. If the parameter data is greater than its value then continue checking its right child, otherwise continue checking its left child. Continue this process until the current node is NULL (available position).

**Find Node ()**

Text

Description automatically generated

First check if the node is NULL, if it already finished travelling the whole tree and there is not such node in the tree, therefore, return false. If the node is not NULL, then checks if its value is equal to the data, if it is that means we found the node, then print the data. If is not equal to the data, then compares its value to the data. If it is less than the data then keep checking its right child, otherwise keep checking its left child.

**Largest ()**

Text

Description automatically generated

First, checks if it has right child, if it does not than return its value, otherwise keep checking its right child until the current node does not have right child (means there is not node in the tree that has value greater than the current node).

**3. Analysis**

**Main function**

Text

Description automatically generated

The main function asks user to input a list of numbers and insert every number into a Binary Search Tree. Then it will display the Binary Search Tree in post-order traversal way. Then it will print out the largest number in the tree. In the last, it will ask user to input a number for searching if the number is in the tree. If it is not in the tree, then the program will prompt “Number not found.”. If the number is int the tree, then it will print out the number. In the end, it will ask user if continues the program, input letter ‘n’ for not, and other any letters for yes.

**Output**

Text

Description automatically generated

**4. Encountered Problems**

To be honest, I did not encounter any problems while doing this lab.

**5. Conclusions**

From this experiment, I went over the implementation of Binary Search Tree. It provides a more efficient way to search than other data structures. This laid a solid foundation for me to learn Red Black Trees. Deepened my understanding of binary search trees.

**6. References**

I did not use any reference in this lab.