CSE214

Recitation 7: More on Binary Trees (TA Version)

1. Fill in the table below for a binary search tree:

|  |  |  |
| --- | --- | --- |
| Algorithm | Worst Case  Complexity | Average  Complexity |
| Search | **O(n)** | **O(logn)** |
| Insert | **O(n)** | **O(logn)** |
| Delete | **O(n)** | **O(logn)** |

Give an example of a BST for which these worst case times could occur.

Tree like a linked-list

1. Given the following definition of a Node, write a method that searches a binary search tree for a node with a given target key.

public class Node {

int key; Node left; Node right;

public Node (int key, Node left, Node right){ this.key = key;

this.left = left; this.right = right;

}

}

public Node search (Node root, int target){

if(root == null){return null;}

if(root.key == target){return root;}

if(target < root.key){

return search(root.left, target)

}

Return search(root.right, target);

}

1. Given the following definition of a Node, write a method that takes the root of two binary trees and determines if they are the same tree. Two trees are the same if they have the identical structure and the nodes have the same key.

public class Node {

int key; Node left; Node right;

public Node (int key, Node left, Node right){ this.key = key;

this.left = left; this.right = right;

}

}

public boolean isSameTree (Node tree1, Node tree2){

if(tree1 == null && tree2 == null){

return true;

}

if(tree1 == null && tree2 != null || tree2 == null && tree1 != null){

return false;

}

if(tree1.key != tree2.key){

return false;

}

return isSameTree(tree1.left, tree2.left), isSameTree(tree1.right, tree2.right);

}

1. Given the following definition of a Node, write a method that takes the root of a binary tree and determines if it is symmetrical. Note: A symmetric tree is a tree that is symmetric around its center

public class Node {

int key; Node left; Node right;

public Node (int key, Node left, Node right){ this.key = key;

this.left = left; this.right = right;

}

}

public boolean isSymmetric (Node root){

return isSameTree(root.left, root.right)

}

5a. Draw the binary search tree that is created if the following numbers are inserted in the tree in the given order: [10, 48, 22, 13, 90, 27, 21, 45, 12].

10

48



22 90



13 27

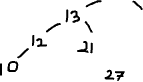


12 21 45



5b. Draw the balanced BST containing the same numbers in part (a)

Sort:[10,12,13,21,22,27,45,48,90], 22 middle so it’s root



1. If a node stores data up to 10 bytes and a reference requires 2 bytes, how much space would a full binary tree of height h take up? What about a full binary tree of height 2.

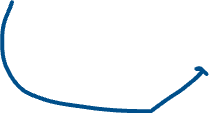
**Number of leaf nodes: 2^h Number of internal nodes: Total number of Nodes: Number of references: Space for nodes: 10\*(2^(h+1) - 1)**



**Space for references:**



**Total Space**



1. True or False
   1. Every binary tree is either complete or full

FALSE

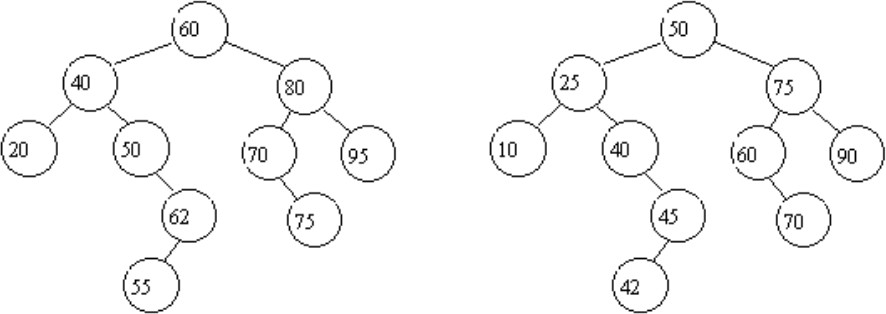
* 1. Every complete binary tree is also a full binary tree.

FALSE

* 1. The maximum number of binary trees that can be formed with 3 nodes is 5

6, SO FALSE. WOULD BE TRUE FOR BST

1. Consider the two following trees. Which of the following is true?



1. Both trees are binary search trees FALSE
2. Only the left tree is a binary search tree FALSE
3. Only the right tree is a binary search tree TRUE
4. Neither trees are binary search trees FALSE