Top-20 Training Program
(Tree Problems)

Apply the problem solving techniques discussed in class to solve the following problems.

Problem1: Tree Similarity

Write an efficient function that returns true if the given two binary trees are similar, and false otherwise. Two binary trees are similar if they are both empty, or both nonempty and have similar left and right subtrees. What are the time and space complexities of your solution?
Function Prototype:

```
boolean isSimilar(TreeNode t1, TreeNode t2)
```

Problem2: Bottom-up Lever Order

Write an efficient function that takes a binary tree as input and displays the elements of tree level by level, but from last level to first. What are the time and space complexities of your solution?
Function prototype:

```
void bottomUpLevel(TreeNode t)
```

Input:

```
        3
       / \
      4   7
     / \  / \
    5  1 6   8
```

Output: 5 1 6 8 4 7 3

Problem3: Zig-Zag Traversal

Write an efficient function to display given binary tree in zig-zag order. What are the time and space complexities of your solution?
Function prototype:

```
void printTree(TreeNode t)
```

Input:

```
        3
       /   \
      4     7
     / \   / \
    5   1 6    8
```

Output: 3 7 4 5 1 6 8

Copyright © Algorithmica
www.algorithmica.co.in
Ph: +91-9246582537

Top-20 Training Program (Tree Problems)

Problem4: Maximum Level Sum

Given a binary tree, write an efficient function that returns the maximum level sum. If tree is empty return false. Assume that all values in binary tree are positive integers. What are the space and time complexities of your solution?

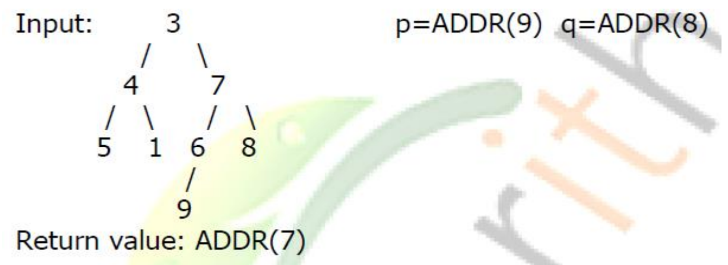
Function prototype:

    int MaxLevelSum(TreeNode t)

Input:
```
          3
         / \
        4   7
       / \ / \
      5  1 1  2
```
Output: 11

Problem5: Least Common Ancestor(LCA)

Least Common Ancestor(LCA) in a tree is defined as the first node that comes common for both the given nodes while travelling towards the root. Write an efficient function "TreeNode FindLca(TreeNode t, TreeNode p, TreeNode q)" to find the least common ancestor of nodes p and q in a binary tree t. You are not allowed to modify the structure of tree node. What are the time and space complexities of your solution? Assume that p and q points to valid nodes in a given binary tree.



```
Input:       3                    p=ADDR(9)  q=ADDR(8)
           /   \
          4     7
         / \   / \
        5   1 6   8
             /
            9
Return value: ADDR(7)
```

Problem6: Depth of tree

Given a tree of n elements as a parent array A i.e., A[i] contains the parent of ith element, write an efficient function to find the depth of tree. The value of -1 in an array indicates there is no parent for that indexed element. What are the time and space complexities of your solution?

Function Prototype:

    int FindDepth(int a[], int n)

Input: 2 5 1 2 0 -1 1 5 [parent of 0 is 2, parent of 1 is 5, etc.,]

Output: 4

Copyright © Algorithmica
www.algorithmica.co.in
Ph: +91-9246582537