

DATA STRUCTURES & ALGORITHMS

#10

Tree basics

|  |
| --- |
| Student Name: Muhammad Hassam Bin Zahid |
| Roll Number: 22SP – 019 - SE Section: SE – A |
| Work submitted on: |

|  |  |  |  |
| --- | --- | --- | --- |
| **Maximum Marks** | **Performance** | **Viva** | **Total** |
| **Marks Obtained** |  |  |  |
| **Remarks (if any)** |  | | |
|  | | | |
| **Experiment evaluated by** | | | |
| Instructor Name: | | | |
| Signature: | | | |

|  |
| --- |
| Tree Related Tasks |

Question: Write a function to calculate the height of a binary tree.

Example:

Input:

5

/

3 7

/

2 4

Output: 2

**Java Code:**

int calculateHeight(Node node) //A function to calculate height of a tree

{

if (node == null) //Checks if The Current Node is empty or not

{

return 0;

}

int leftHeight = calculateHeight(node.left); //calculates the height of both left and right side of the tree

int rightHeight = calculateHeight(node.right);

if(leftHeight > rightHeight) //Compares both the heights and will

{ return the greatest height

return leftHeight;

}

else

{

return rightHeight;

}

}

Question: Given a binary tree, write a function to check if it is a binary search tree (BST).

Example:

Input:

4

/

2 5

/

1 3

Output: True

**Java Code:**

boolean isBSTUtil(Node node, int min, int max)

{

if(node == null) //Will check if the current node is empty or not

{

return true;

}

if(node.data <= min || node.data >= max) //To check if the Binary

{ Tree is BST or not

return false;

}

return isBSTUtil(node.left, min, node.data) && isBSTUtil(node.right, node.data, max); //This will recursively run the function.

}

Question: Write a function to find the maximum element in a binary tree.

Example:

Input:

5

/

3 7

/

2 4

Output: 7

**Java Code:**

int findMax(Node node)

{

if(node == null) //Checks if the tree is empty or not

{

return 0;

}

int max = node.data; //Initializes the variable ‘max’

int lMax = findMax(node.left); //recursively finds the left and right

int rMax = findMax(node.right); max.

if(max < lMax) //Checks if left max is greater or not

{

max = lMax; //Replaces if it’s true

}

if(max < rMax) //Checks if right max is greater or not

{

max = rMax; //Replaces if it’s true

}

return max; //Returns the largest Value

}

Question: Write a function to check if two binary trees are identical (have the same structure and values).

Example:

Input:

Tree 1:

5

/

3 7

/

2 4

Tree 2:

5

/

3 7

/

2 4

Output: True

**Java Code:**

public Boolean isSame(Node node , Node exNode)

{

if(node == null && exNode == null) //If both are null, They are identical

{

return true;

}

if(node == null || exNode == null) //If only one is null, They are

{ not identical

return false;

}

if(node.data != exNode.data) //If both values are not same, They

{ are not identical

return false;

}

//Recursively Check both Left and Right Subtress

boolean leftSame = isSame(node.left , exNode.left);

boolean rightSame = isSame(node.right , exNode.right);

return leftSame && rightSame; //Returns True if both trees are same

}

Question: Given a binary tree, write a function to check if it is a balanced tree (the heights of the two subtrees of any node differ by at most one).

Example:

Input:

3

/

9 20

/

15 7

Output: True

**Java Code:**

boolean isBalanced(Node node)

{

int lh;

int rh;

if (node == null) //If tree is empty then return true.

{

return true;

}

lh = height(node.left); //Gets the height of left and right sub trees

rh = height(node.right);

if (Math.abs(lh - rh) <= 1 && isBalanced(node.left) && isBalanced(node.right))

{

return true; //If the above Statement is true, then the tree must

} be balanced

return false; //Otherwise the tree is not balanced.

}