Experiment 5

Student Name:Myskina **UID: 22BCS10129**
Branch: CSE **Section/Group: 637-B**
Semester: 6th **Date of Performance:20/2/25**
Subject Name: Advanced Programming - 2 **Subject Code: 22CSH-351**

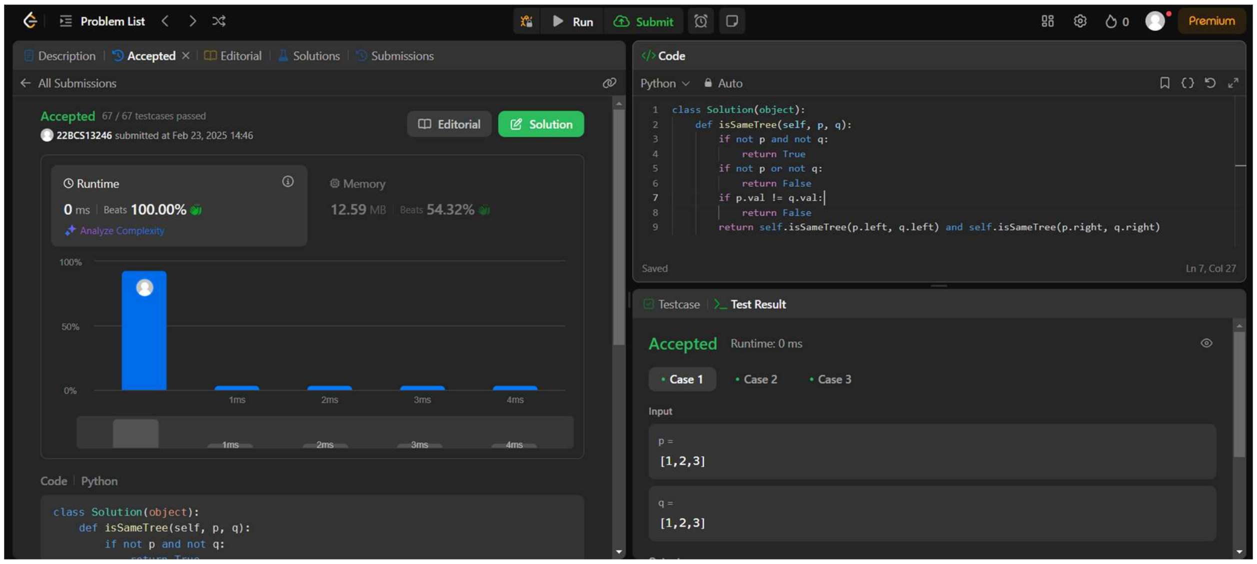
Ques 1:

Aim: Same Tree

Code:

```
class Solution(object):
    def isSameTree(self, p, q):
        if not p and not q:
            return True
        if not p or not q:
            return False
        if p.val != q.val:
            return False
        return self.isSameTree(p.left, q.left) and self.isSameTree(p.right, q.right)
```

Submission Screenshot:



Accepted 67 / 67 testcases passed
22BCS13246 submitted at Feb 23, 2025 14:46

Runtime 0 ms | Beats 100.00%
Memory 12.59 MB | Beats 54.32%

```
class Solution(object):
    def isSameTree(self, p, q):
        if not p and not q:
            return True
        if not p or not q:
            return False
        if p.val != q.val:
            return False
        return self.isSameTree(p.left, q.left) and self.isSameTree(p.right, q.right)
```

Accepted Runtime: 0 ms

Input

```
p =
[1,2,3]

q =
[1,2,3]
```

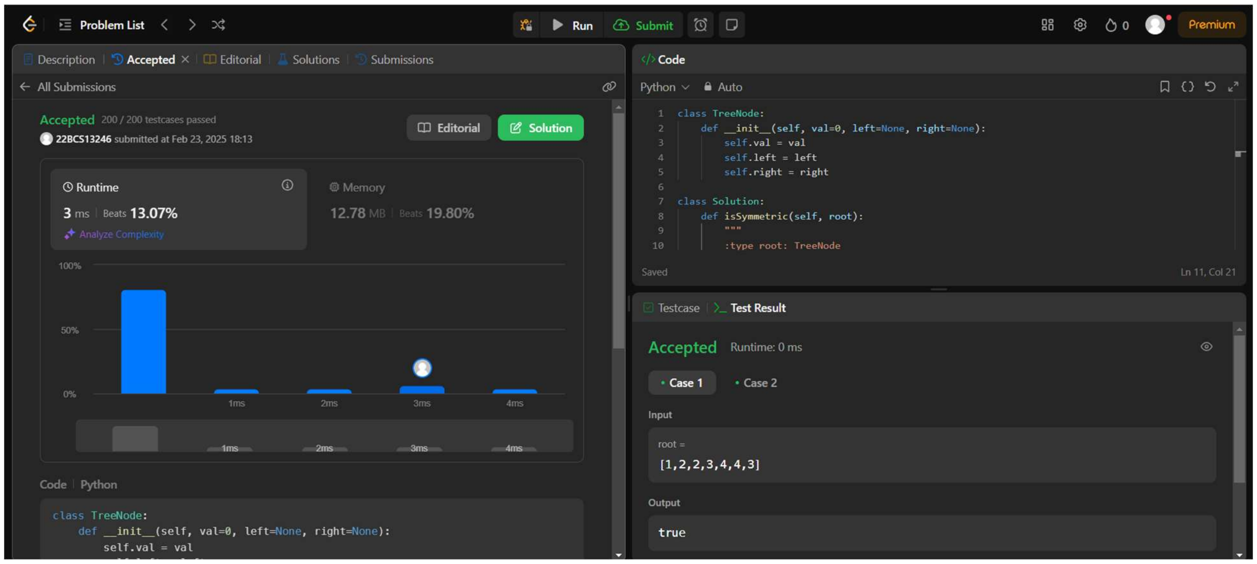
Ques 2:

Aim: Symmetric Tree

Code:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def isSymmetric(self, root):
        if not root:
            return True
        return self.isMirror(root.left, root.right)
    def isMirror(self, t1, t2):
        if not t1 and not t2:
            return True
        if not t1 or not t2:
            return False
        if t1.val != t2.val:
            return False
        return self.isMirror(t1.left, t2.right) and self.isMirror(t1.right, t2.left)
```

Submission Screenshot:



Accepted 200 / 200 testcases passed

22BCS13246 submitted at Feb 23, 2025 18:13

Runtime: 3 ms | Beats 13.07%

Memory: 12.78 MB | Beats 19.80%

Code | Python

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
```

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def isSymmetric(self, root):
        """
        :type root: TreeNode
```

Saved   Ln 11, Col 21

Accepted Runtime: 0 ms

Input

```
root =
[1,2,2,3,4,4,3]
```

Output

```
true
```

Ques 3:

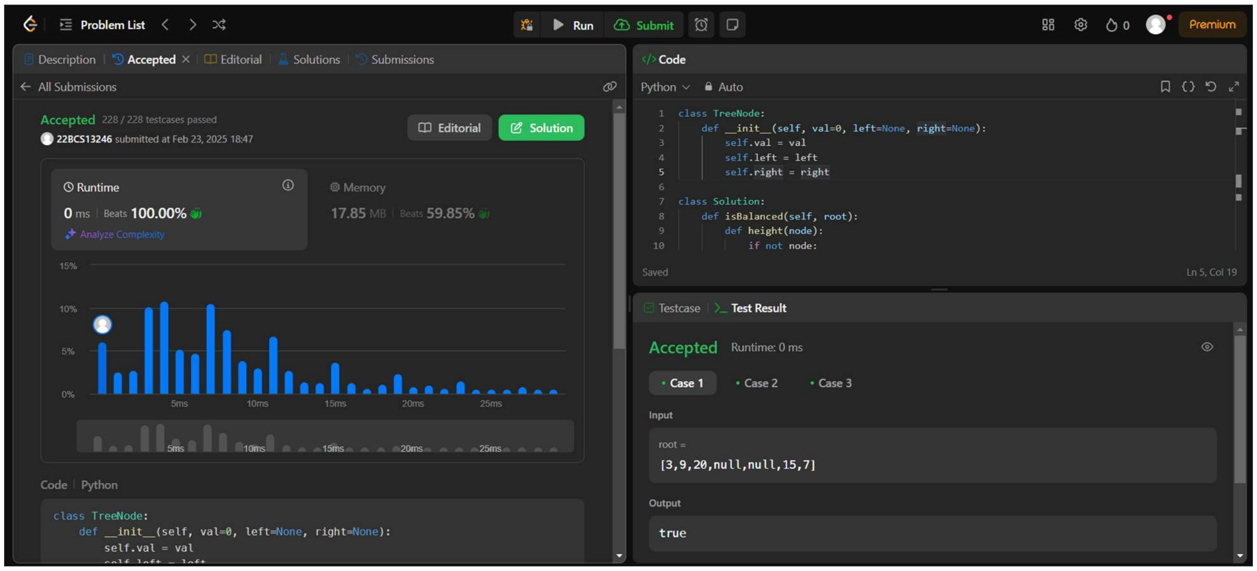
Aim: Balanced Binary Tree

Code:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def isBalanced(self, root):
        def height(node):
            if not node:
                return 0
            left, right = height(node.left), height(node.right)
            if abs(left - right) > 1 or left == -1 or right == -1:
                return -1
            return max(left, right) + 1

        return height(root) != -1
```

Submission Screenshot:



Accepted 228 / 228 testcases passed
22BCS13246 submitted at Feb 23, 2025 18:47

Runtime
0 ms | Beats 100.00%
Analyze Complexity

Memory
17.85 MB | Beats 59.85%

Code | Python

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
```

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def isBalanced(self, root):
        def height(node):
            if not node:
```

Saved   Ln 5, Col 19

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

root =
```
[3,9,20,null,null,15,7]
```

Output
```
true
```

Ques 4:

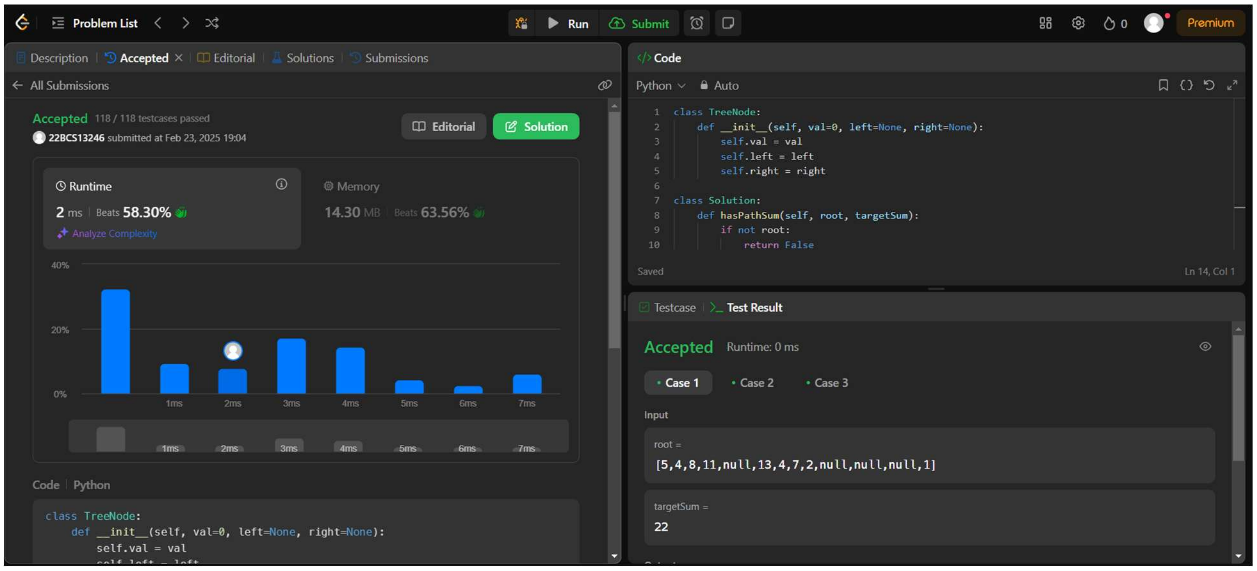
Aim: Path Sum

Code:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def hasPathSum(self, root, targetSum):
        if not root:
            return False
        if not root.left and not root.right and root.val == targetSum:
            return True
        return self.hasPathSum(root.left, targetSum - root.val) or self.hasPathSum(root.right,
targetSum - root.val)
```

Submission Screenshot:



Accepted 118 / 118 testcases passed
22BCS13246 submitted at Feb 23, 2025 19:04

Runtime
2 ms | Beats 58.30%

Memory
14.30 MB | Beats 63.56%

Code | Python

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
```

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def hasPathSum(self, root, targetSum):
        if not root:
            return False
```

Ln 14, Col 1

Accepted Runtime: 0 ms

Input

root =
[5,4,8,11,null,13,4,7,2,null,null,null,1]

targetSum =
22

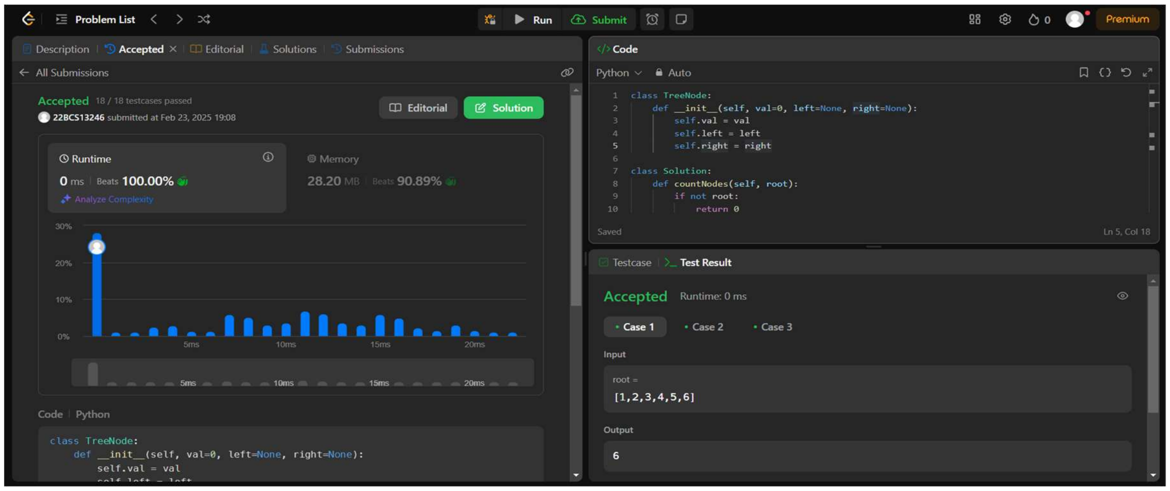
Ques 5:

Aim: Count Complete Tree Nodes

Code:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def countNodes(self, root):
        if not root:
            return 0
        lh, rh = self.getHeight(root.left), self.getHeight(root.right)
        if lh == rh:
            return (1 << lh) + self.countNodes(root.right)
        else:
            return (1 << rh) + self.countNodes(root.left)
    def getHeight(self, node):
        h = 0
    while node:
            h += 1
            node = node.left
        return h
```

Submission Screenshot:



Accepted 18 / 18 testcases passed
22BCS13246 submitted at Feb 23, 2025 19:08

Runtime
0 ms | Beats 100.00%

Memory
28.20 MB | Beats 90.89%

Code | Python

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
```

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def countNodes(self, root):
        if not root:
            return 0
```

Saved Ln 5, Col 18

Accepted Runtime: 0 ms

Input
root =
```
[1,2,3,4,5,6]
```

Output
```
6
```

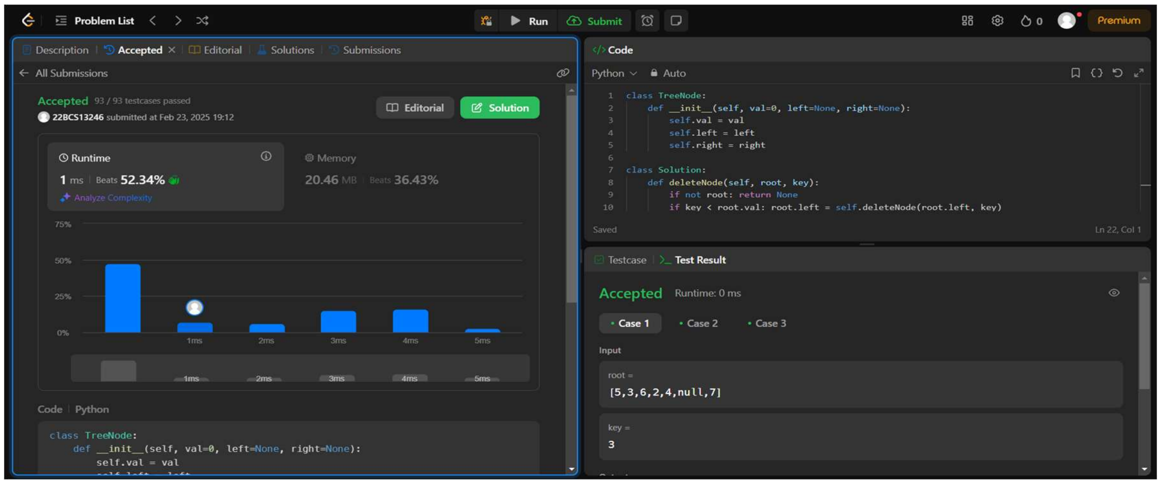
Ques 6:

Aim: Delete node in a BST

Code:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def deleteNode(self, root, key):
        if not root: return None
        if key < root.val: root.left = self.deleteNode(root.left, key)
        elif key > root.val: root.right = self.deleteNode(root.right, key)
        else:
            if not root.left: return root.right
            if not root.right: return root.left
            minNode = self.getMin(root.right)
            root.val, root.right = minNode.val, self.deleteNode(root.right, minNode.val)
        return root
    def getMin(self, node):
        while node.left: node = node.left
        return node
```

Submission Screenshot:



Accepted 93 / 93 testcases passed
22BCS13246 submitted at Feb 23, 2025 19:12

Runtime
1 ms | Beats 52.34%

Memory
20.46 MB | Beats 36.43%

Code | Python

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
```

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def deleteNode(self, root, key):
        if not root: return None
        if key < root.val: root.left = self.deleteNode(root.left, key)
```

Accepted Runtime: 0 ms

Input

root =
[5,3,6,2,4,null,7]

key =
3

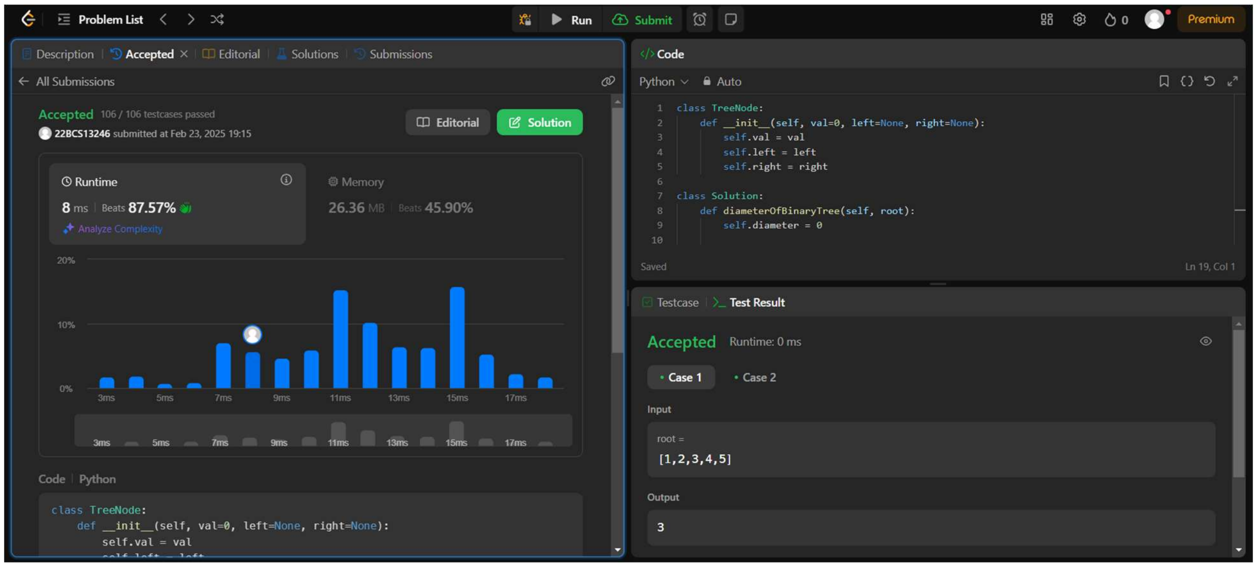
Ques 7:

Aim: Diameter of Binary Tree

Code:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def diameterOfBinaryTree(self, root):
        self.diameter = 0
        def depth(node):
            if not node: return 0
            left, right = depth(node.left), depth(node.right)
            self.diameter = max(self.diameter, left + right)
            return 1 + max(left, right)
        depth(root)
        return self.diameter
```

Submission Screenshot:



Accepted 106 / 106 testcases passed
22BCS13246 submitted at Feb 23, 2025 19:15

Runtime
8 ms | Beats 87.57%

Memory
26.36 MB | Beats 45.90%

Code | Python

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
```

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def diameterOfBinaryTree(self, root):
        self.diameter = 0

```

Accepted Runtime: 0 ms

Input

root =
[1,2,3,4,5]

Output

3