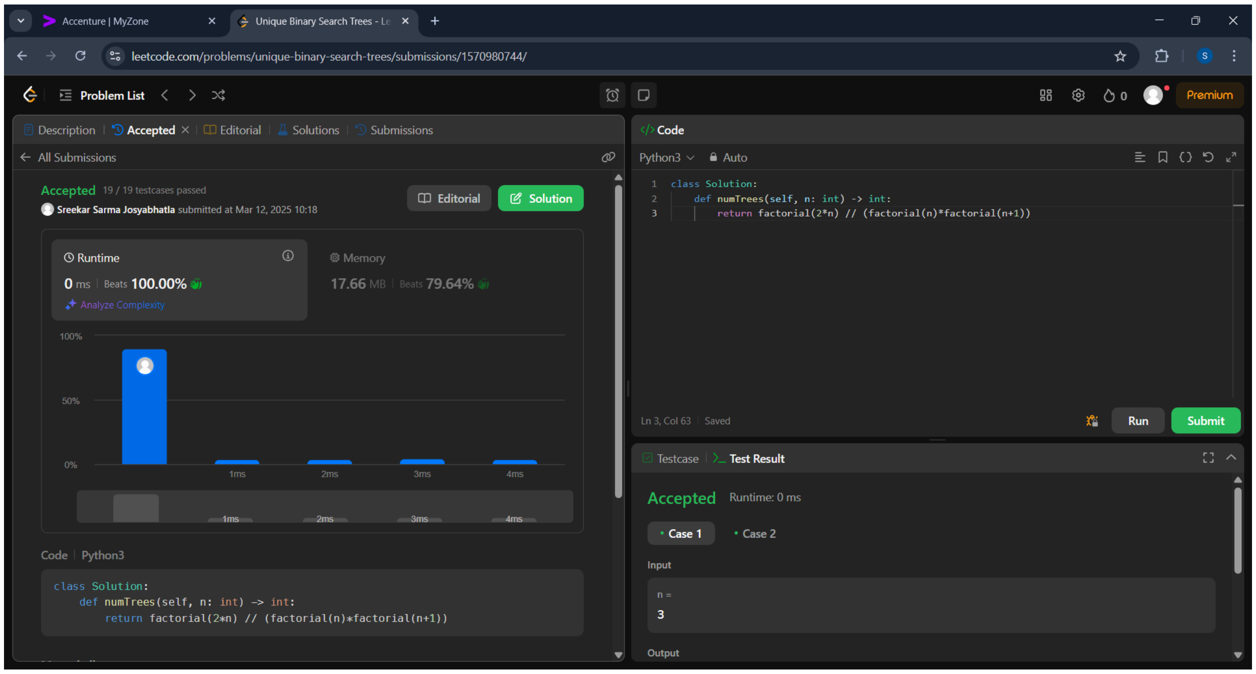
Assignment-4

J.Sreekar Sarma
VU21CSEN0300040

1. Unique Binary Search Trees

```
class Solution:
    def numTrees(self, n: int) -> int:
        return factorial(2*n) // (factorial(n)*factorial(n+1))
```

OUTPUT



Accenture | MyZone
Unique Binary Search Trees - Le
leetcode.com/problems/unique-binary-search-trees/submissions/1570980744/
Problem List
Description | Accepted | Editorial | Solutions | Submissions
All Submissions
Editorial
Solution
Analyze Complexity
Code
Python3
Auto
Run
Submit
Testcase | Test Result

Accepted 19 / 19 testcases passed
Sreekar Sarma Josyabhatla submitted at Mar 12, 2025 10:18

Runtime
0 ms | Beats 100.00%

Memory
17.66 MB | Beats 79.64%

Code | Python3

```
class Solution:
    def numTrees(self, n: int) -> int:
        return factorial(2*n) // (factorial(n)*factorial(n+1))
```

Ln 3, Col 63 | Saved

Accepted Runtime: 0 ms

Case 1 Case 2

Input

n =
3

Output

2. All Elements in Two Binary Search Trees

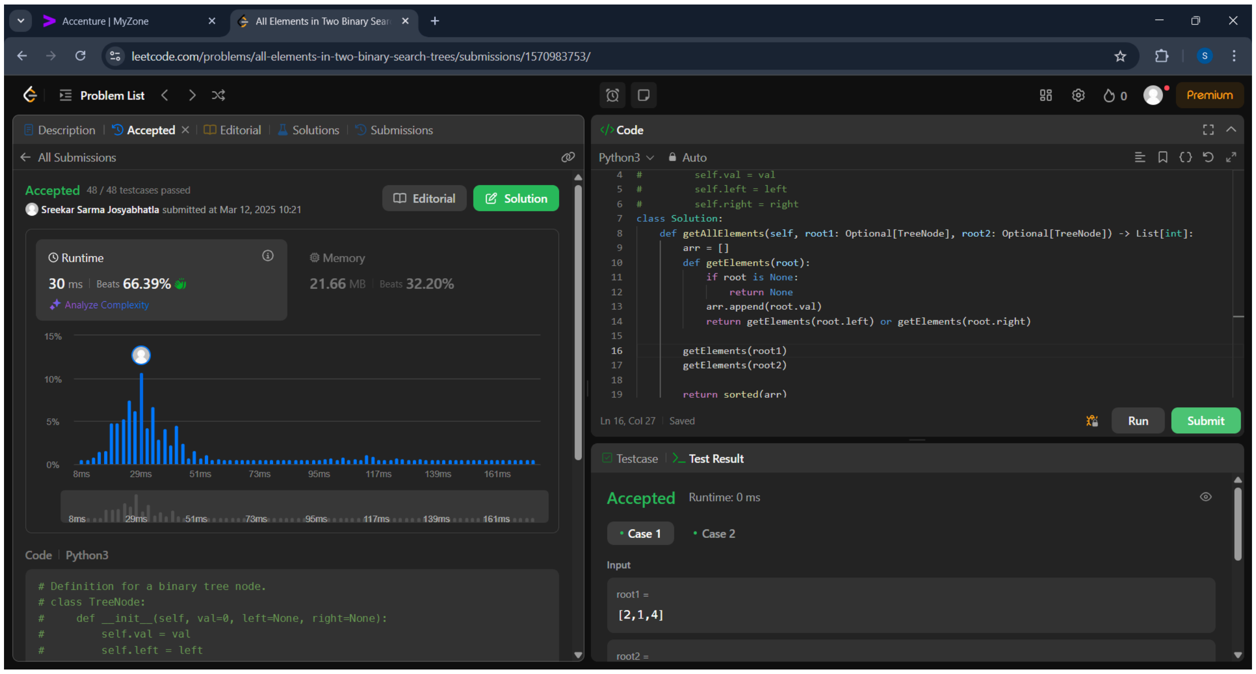
```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
          def    getAllElements(self,    root1:    Optional[TreeNode],    root2:
Optional[TreeNode]) -> List[int]:
        arr = []
        def getElements(root):
            if root is None:
                return None
            arr.append(root.val)
```

```python
            return getElements(root.left) or getElements(root.right)

        getElements(root1)
        getElements(root2)

        return sorted(arr)
```

OUTPUT



Accepted 48 / 48 testcases passed

Sreekar Sarma Josyabhatla submitted at Mar 12, 2025 10:21

Runtime 30 ms | Beats 66.39%

Memory 21.66 MB | Beats 32.20%

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def getAllElements(self, root1: Optional[TreeNode], root2: Optional[TreeNode]) -> List[int]:
        arr = []
        def getElements(root):
            if root is None:
                return None
            arr.append(root.val)
            return getElements(root.left) or getElements(root.right)

        getElements(root1)
        getElements(root2)

        return sorted(arr)
```

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

root1 =
[2,1,4]

root2 =