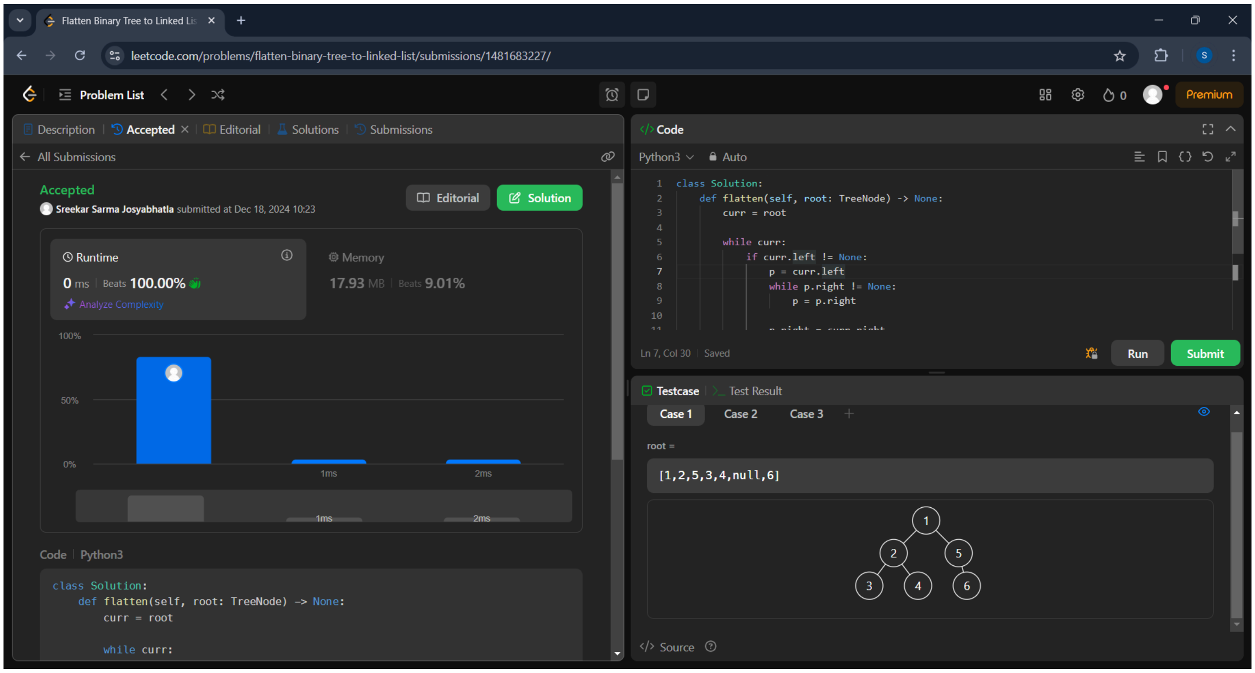
Task-2

J.Sreekar Sarma
VU21CSEN0300040

1. Flatten Binary Tree to Linked List

```
class Solution:
    def flatten(self, root: TreeNode) -> None:
        curr = root

        while curr:
            if curr.left != None:
                p = curr.left
                while p.right != None:
                    p = p.right

                p.right = curr.right

                curr.right = curr.left
                curr.left = None

            curr = curr.right
```

OUTPUT



Accepted
Sreekar Sarma Josyabhatla submitted at Dec 18, 2024 10:23

Runtime
0 ms | Beats 100.00%

Memory
17.93 MB | Beats 9.01%

Code | Python3

```
class Solution:
    def flatten(self, root: TreeNode) -> None:
        curr = root

        while curr:
```

```
class Solution:
    def flatten(self, root: TreeNode) -> None:
        curr = root

        while curr:
            if curr.left != None:
                p = curr.left
                while p.right != None:
                    p = p.right

                p.right = curr.right
```

Ln 7, Col 30 | Saved

root =
[1,2,5,3,4,null,6]

2. Trapping Rain Water

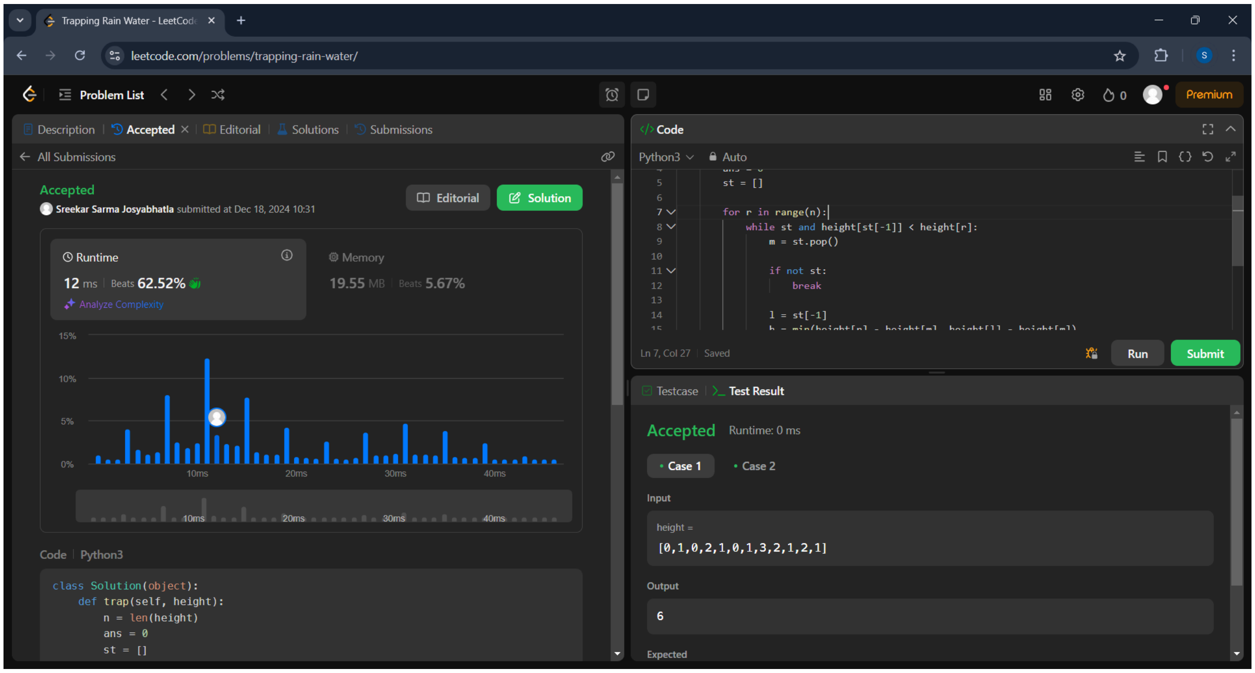
```
class Solution(object):
    def trap(self, height):
        n = len(height)
        ans = 0
        st = []
```

Task-2

J.Sreekar Sarma
VU21CSEN0300040

```
        for r in range(n):
            while st and height[st[-1]] < height[r]:
                m = st.pop()

                if not st:
                    break

                l = st[-1]
                h = min(height[r] - height[m], height[l] -
height[m])
                w = r - l - 1

                ans += h * w
            st.append(r)
        return ans
```

OUTPUT



Accepted

Sreekar Sarma Josyabhatla submitted at Dec 18, 2024 10:31

Runtime: 12 ms | Beats 62.52%

Memory: 19.55 MB | Beats 5.67%

Code | Python3

```
class Solution(object):
    def trap(self, height):
        n = len(height)
        ans = 0
        st = []
```

```
        st = []

        for r in range(n):
            while st and height[st[-1]] < height[r]:
                m = st.pop()

                if not st:
                    break

                l = st[-1]
                h = min(height[r] - height[m], height[l] - height[m])
```

Ln 7, Col 27 | Saved

Accepted Runtime: 0 ms

Input

```
height =
[0,1,0,2,1,0,1,3,2,1,2,1]
```

Output

```
6
```

Expected