Question No. 1700 :-
https://leetcode.com/problems/number-of-students-unable-to-eat-lunch/description/
Solution Link :-
https://leetcode.com/problems/number-of-students-unable-to-eat-lunch/submissions/1403528108/
Description :-

Time Complexity :- O(n)

Counting count0 and count1 takes O(n), where n is the number of students.

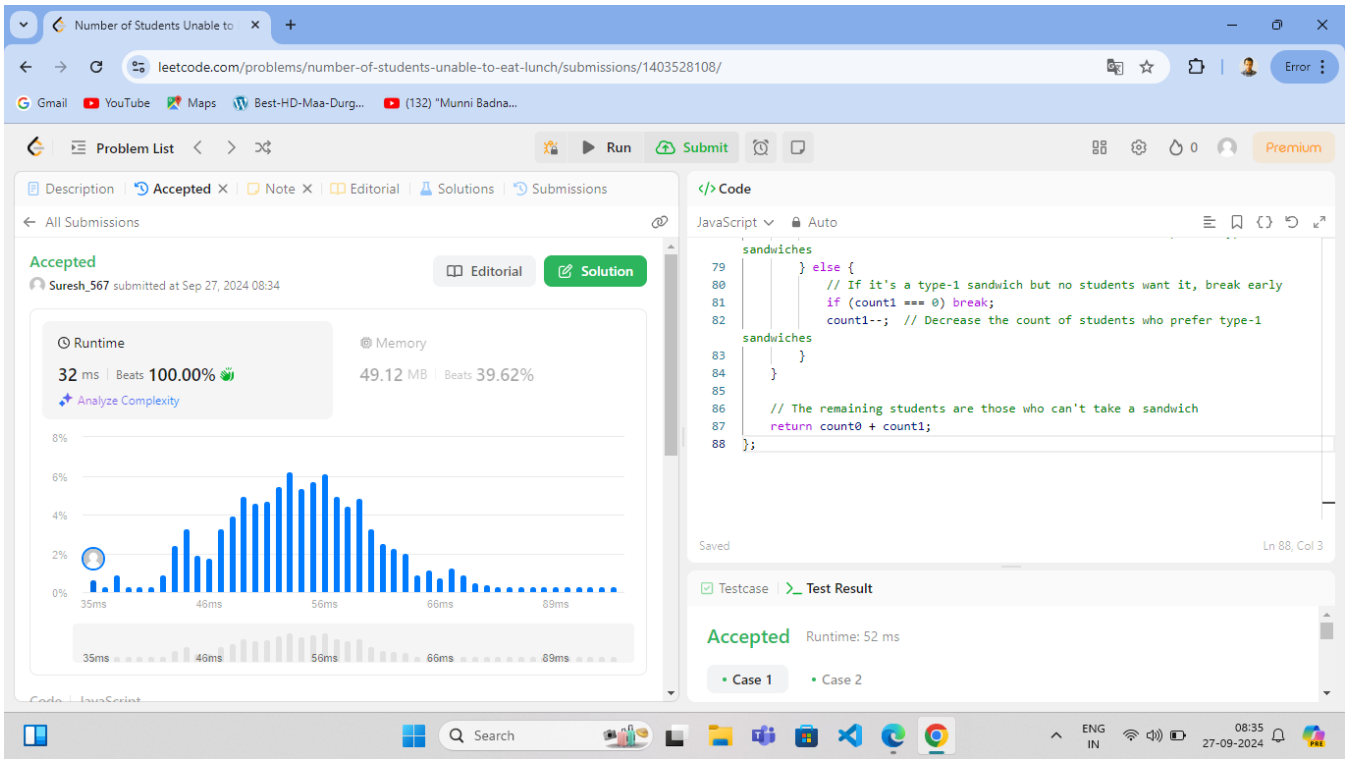
Iterating through the sandwiches takes O(n) in the worst case.

So, Overall time complexity is **O(n)**.

Space Complexity :- O(1)

Only a few constant variables (count0, count1) are used, so the space complexity is **O(1)**.

Screenshot :-



Question No. 111 :-

https://leetcode.com/problems/minimum-depth-of-binary-tree/description/

Solution Link :-

https://leetcode.com/problems/minimum-depth-of-binary-tree/submissions/1403542246/

Description :-

Time Complexity :- O(n)

Each node is processed exactly once in the BFS traversal. The number of nodes in the tree is n.

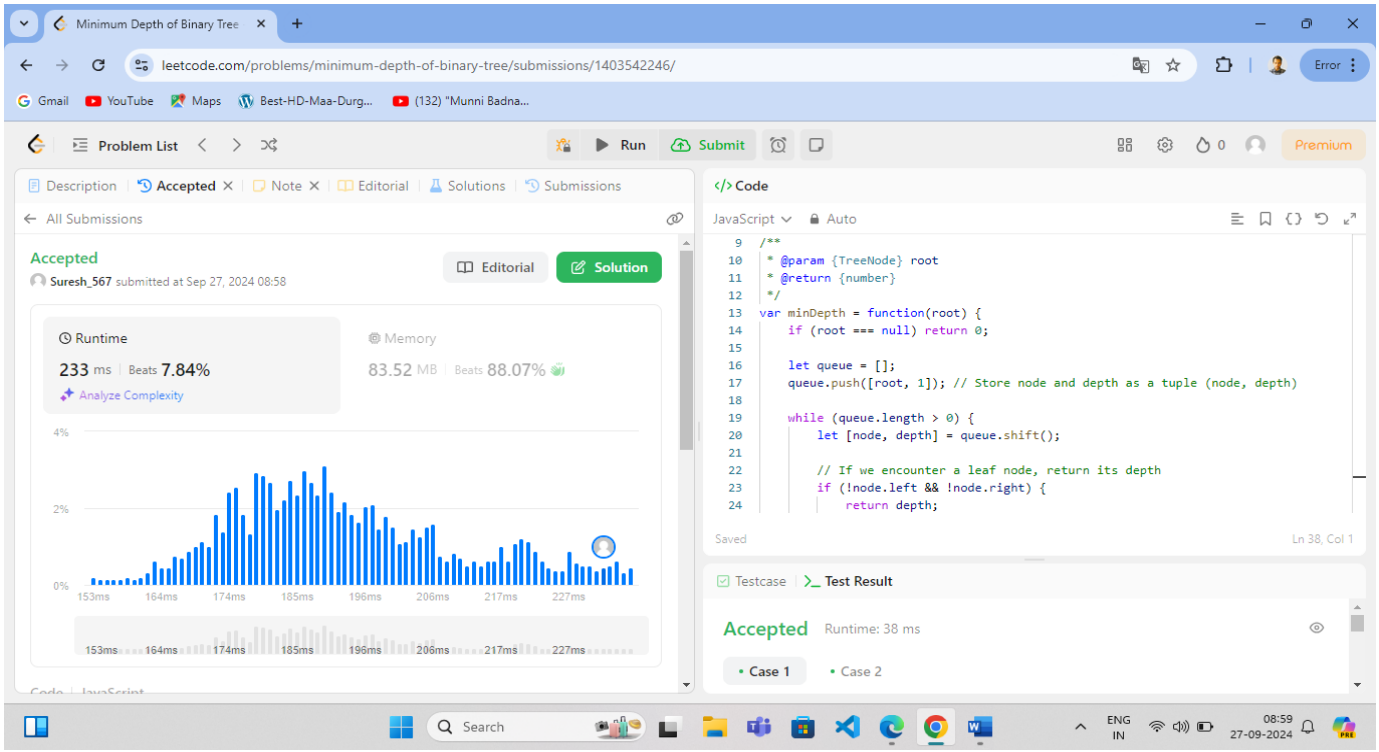
Therefore, the time complexity is **O(n)**.

Space Complexity :- O(n)

The space complexity is dominated by the queue. In the worst case, the queue can store up to one level of nodes at a time. In a full binary tree, the maximum number of nodes at any level is O(n) (the last level).

Therefore, the space complexity is **O(n)** in the worst case.

Screenshot :-



Question No. 145 :-

https://leetcode.com/problems/binary-tree-postorder-traversal/description/

Solution Link :-

https://leetcode.com/problems/binary-tree-postorder-traversal/submissions/1403548394/

Description :-

Time Complexity :- O(n)

The code visits each node exactly once and performs constant time operations (like pushing/popping from the stack) for each node.

Therefore, the time complexity is **O(n)**, where n is the number of nodes in the tree.

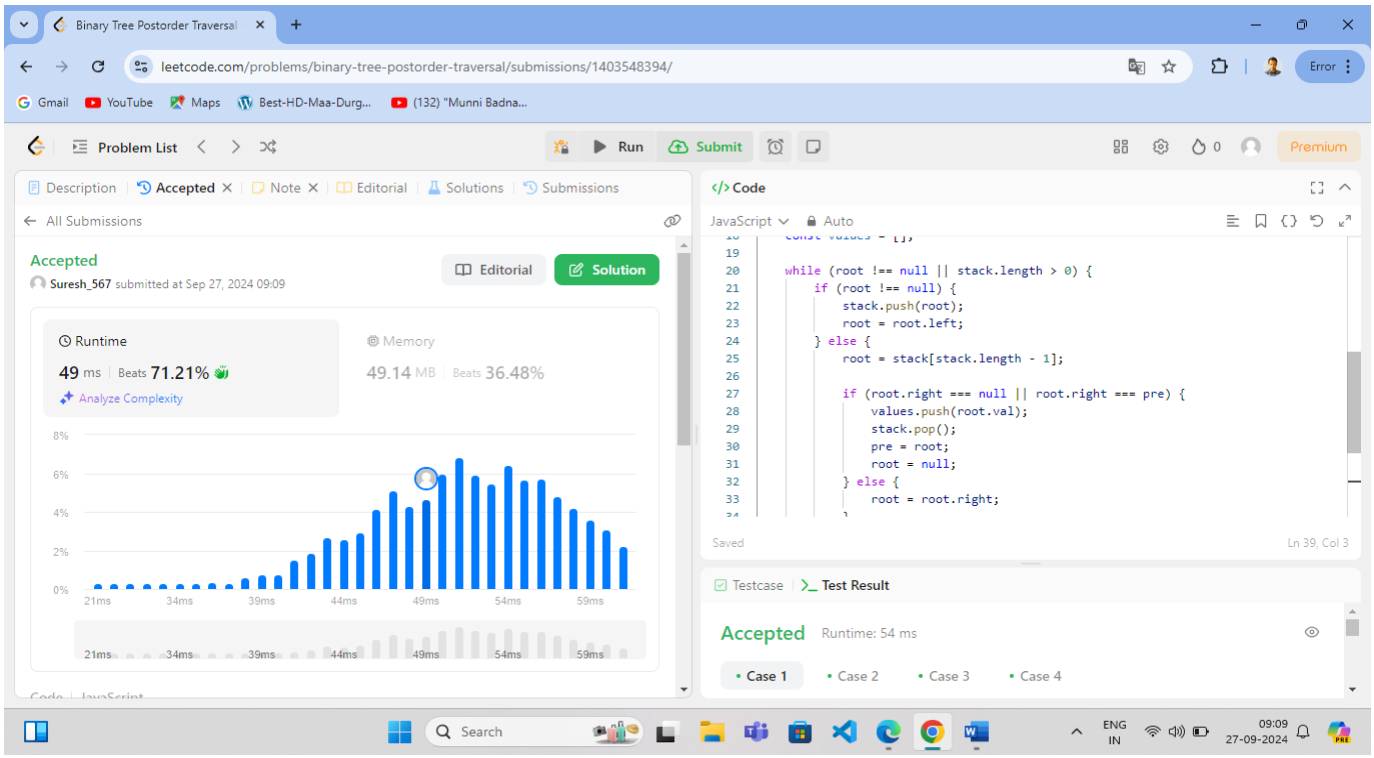
Space Complexity :- O(n)

In the worst case (for an unbalanced tree), the stack may contain all the nodes along the longest path from root to a leaf, which can be **O(n)**.

Additionally, the output array values will contain all n node values.

Therefore, the space complexity is **O(n)**.

Screenshot :-



Question No. 144 :-

https://leetcode.com/problems/binary-tree-preorder-traversal/description/

Solution Link :-

https://leetcode.com/problems/binary-tree-preorder-traversal/submissions/1403553776/

Description :-

Time Complexity :- O(n)

Each node in the tree is visited exactly once.

Therefore, the time complexity is **O(n)**, where n is the number of nodes in the tree.

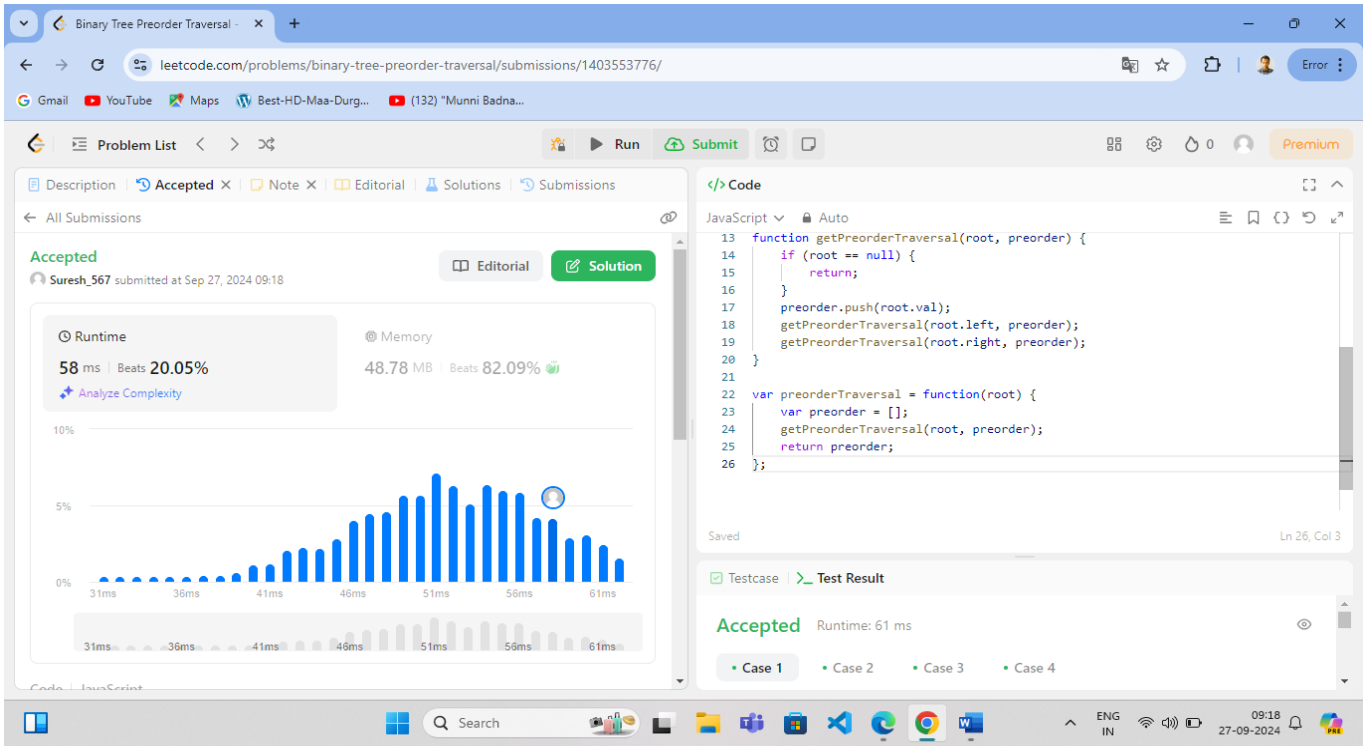
Space Complexity :- O(n)

In the worst case (for an unbalanced tree), the recursion depth will be **O(n)**, and in the best case (for a balanced tree), it will be **O(log n)**.

The preorder array stores all n nodes, so the space complexity for storage is **O(n)**.

Therefore, the worst-case space complexity is **O(n)**.

Screenshot :-



Question No. 94:-

https://leetcode.com/problems/binary-tree-inorder-traversal/description/

Solution Link :-

https://leetcode.com/problems/binary-tree-inorder-traversal/submissions/1403564561/

Description :-

Time Complexity :- O(n)

We visit each node exactly once, so the time complexity is **O(n)**.

Space Complexity :- O(n)

The space complexity is **O(n)** in the worst case due to the stack storing nodes. In the case of a highly unbalanced tree (like a linked list), the stack can grow to size n.

Screenshot :-