DSA- Assignment-4

Questions List :-

1. Question was- *https://leetcode.com/problems/valid-parentheses/description/*

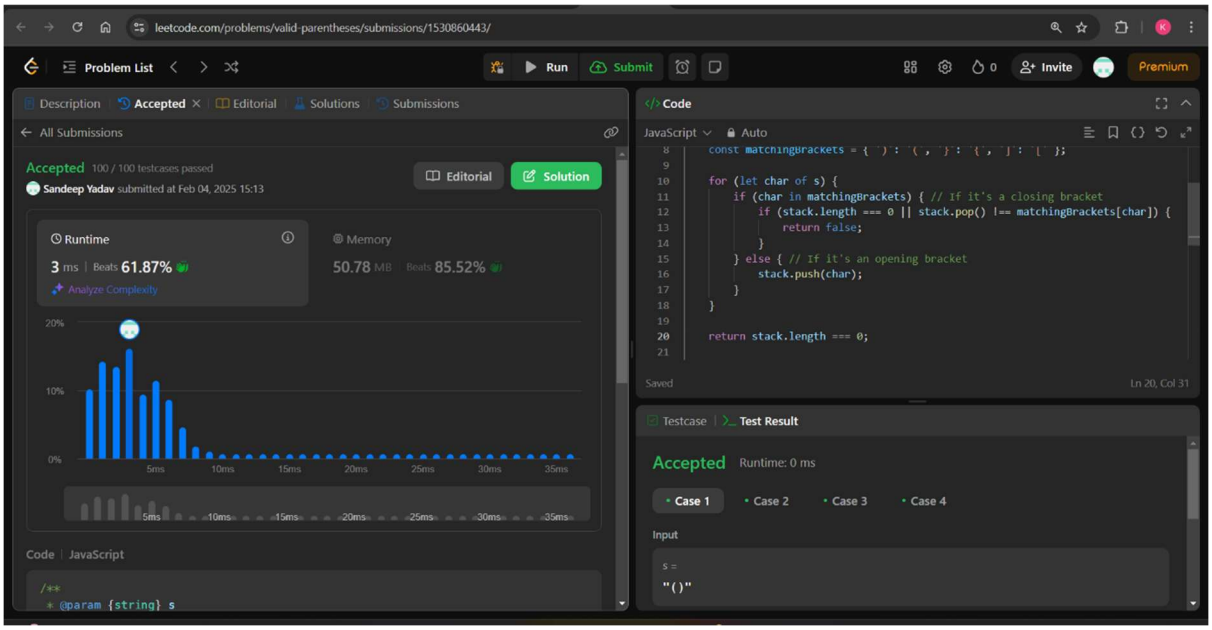
Solution Link - https://leetcode.com/problems/valid-parentheses/submissions/1530860443/

Description:

(i) Create an empty stack.
(ii) Create empty matching bracket collection.
(iii) Run loop for all s brackets.
(iv) Check **if** (stack.*length* **===** 0 **||** stack.pop() **!==** matchingBrackets[char]) return false.
(v) If it's an opening bracket then stack.push(char)

Time Complexity: O(n)

Space Complexity: O(n)



2. Question was- https://leetcode.com/problems/next-greater-element-i/description/

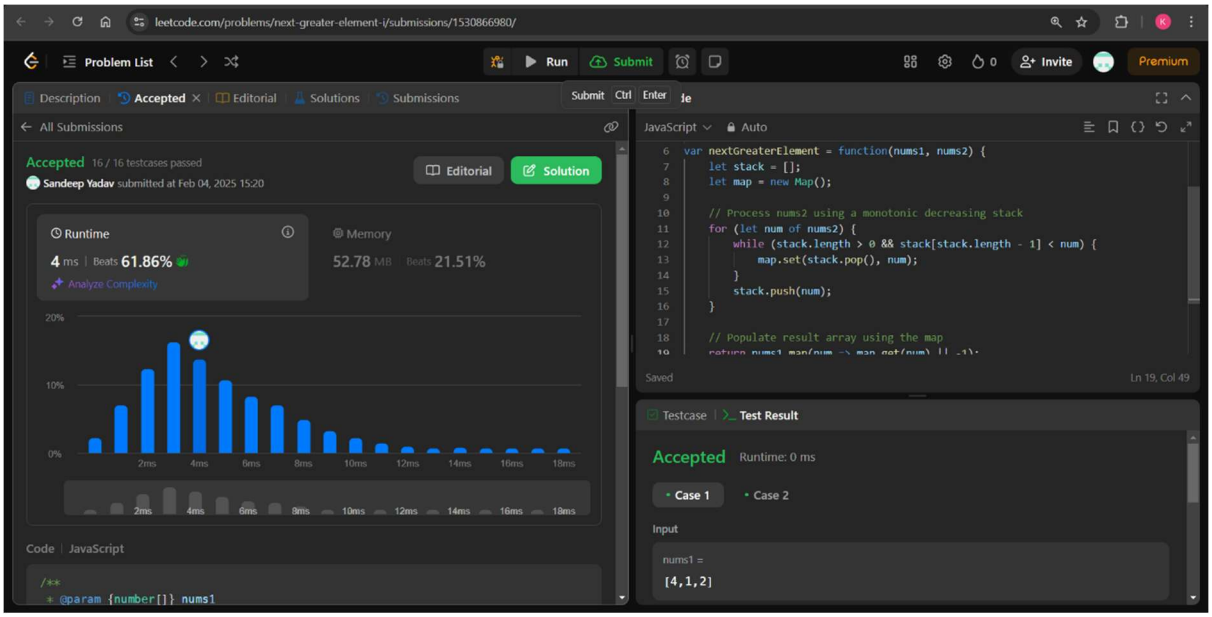
Solution Link- https://leetcode.com/problems/next-greater-element-i/submissions/1530866980/

Description:

(i) Create an empty stack and map.
(ii) Process num2 using a monotonic decreasing stack.
(iii) Define condition using while loop.
(iv) Populate result array using the map.

Time Complexity: O(n+m), where N is the length of nums2 and M is the length of nums1

Space Complexity: O(n) extra space for HashMap and Stack



3. Question was- https://leetcode.com/problems/remove-all-adjacent-duplicates-in-string/description/

Solution Link- https://leetcode.com/problems/remove-all-adjacent-duplicates-in-string/submissions/1558739045/

Description:

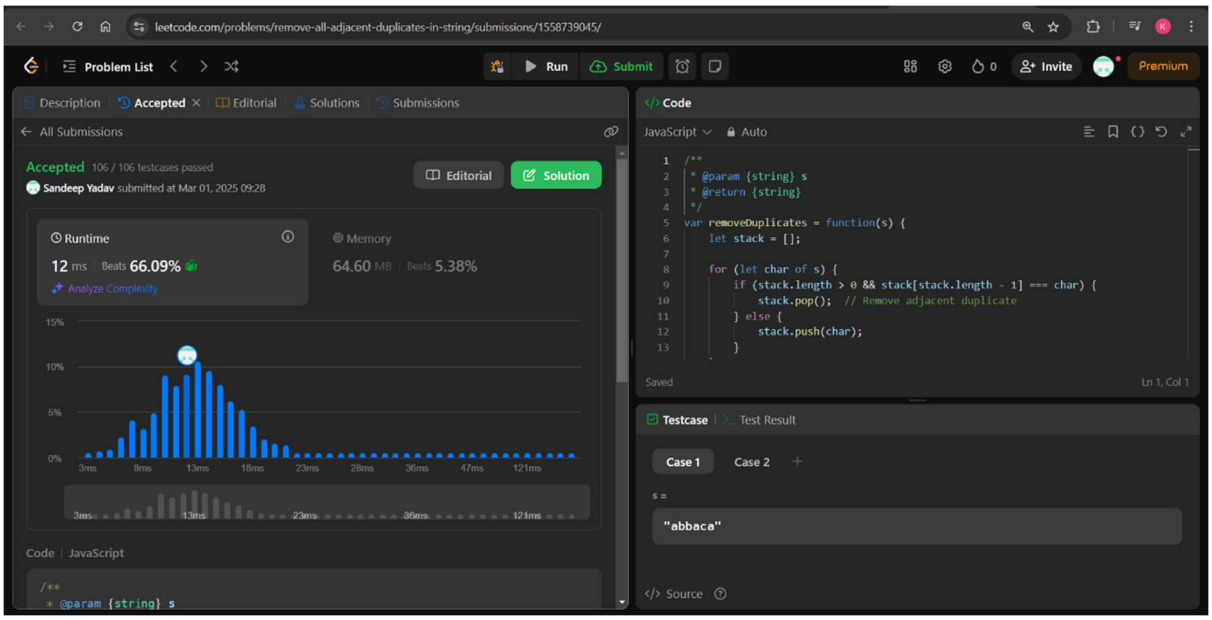
(i) Run loop for string s.
(ii) Check if stack.length > 0 && stack[stack.length - 1] === char)
Remove adjacent duplicate.

(iii) Else stack.push(char).

(iv) Convert stack to string using stack.join().

Time Complexity: O(n) .

Space Complexity: *O(n)* //Uses a stack for efficient duplicate removal.



4. Question was- https://leetcode.com/problems/trapping-rain-water/description/

Solution Link– https://leetcode.com/problems/trapping-rain-water/submissions/1558734466/

Description:

(i) Define Left & Right Pointer.

(ii) Run while loop till Left<Right.

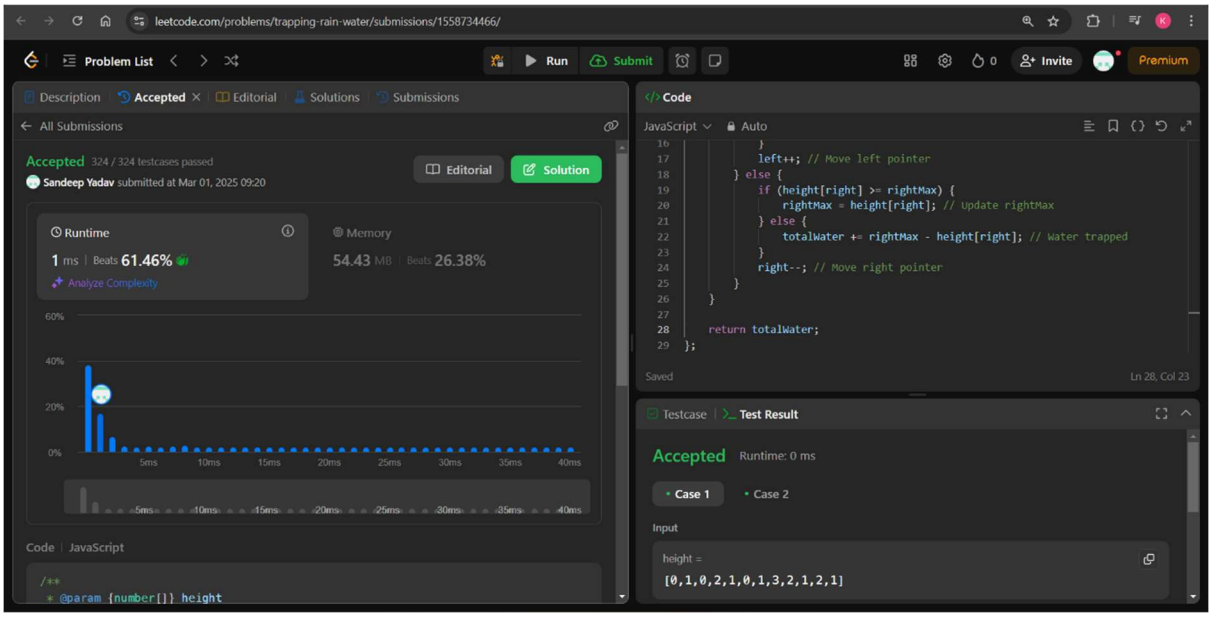
(iii) Check if (height[left] **>=** leftMax)
leftMax = height[left];
else totalwater += leftMax – height[left].
Then move Left Pointer.

(iv) Else **if** (height[right] **>=** rightMax)
rightMax **=** height[right]**;** *// Update rightMax*
else

totalWater += rightMax - height[right]; *// Water trapped*

Time Complexity: O(n + k) n is the numbers of nodes and k is the cycle length

Space Complexity: O(1) no extra space



5. Question was- https://leetcode.com/problems/largest-rectangle-in-histogram/description/

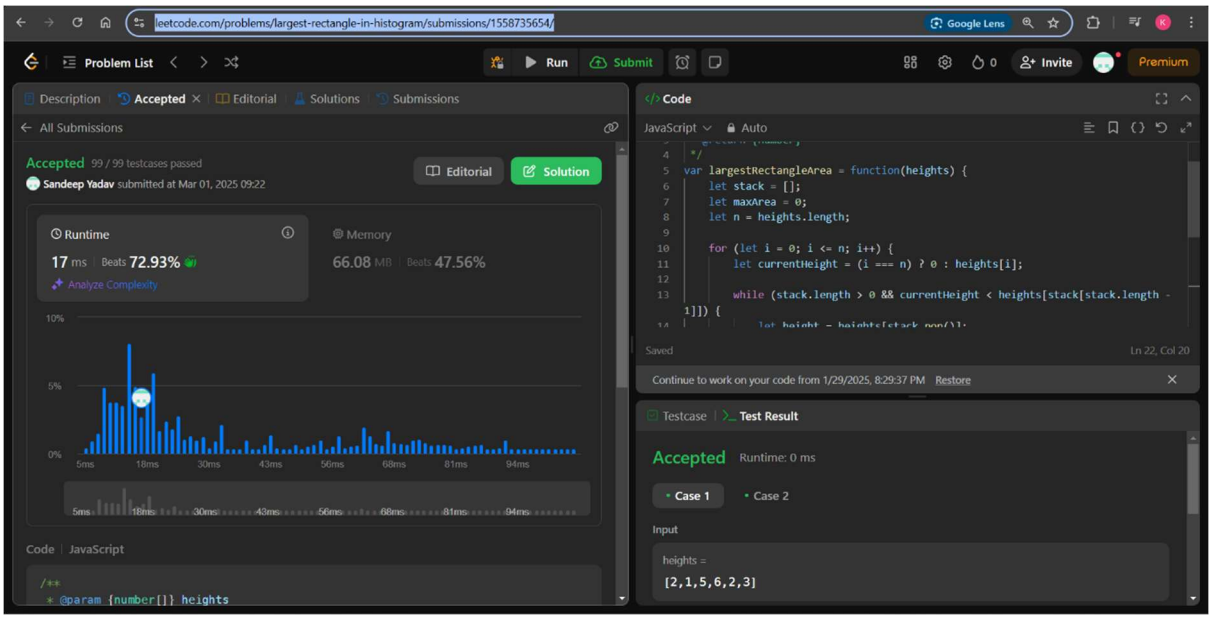
Solution Link - https://leetcode.com/problems/largest-rectangle-in-histogram/submissions/1558735654/

Description:

(i) Create empty stack and maxArea = 0.
(ii) Run a loop till n.
(iii) Define currentHeight= (i===n) ? 0: heights[i].
(iv) Run while loop till stack.length > 0 && currentHeight < heights[stack.length-1]] .
maxArea = Math.max(maxArea, height * width).
(v) Stack.push(i).

Time Complexity: O(L), single traversal using two pointers. // L is length of List

Space Complexity: O(1) no use extra space



Key Features :-

- Solutions are less time taking.

Difficulties :-

- Facing Difficulties in Code Optimization.
- Try to make code efficient and use better approach to solve them.

GitHub Repository Link : -

https://github.com/SandyBhai03/Internshala-Assignments/blob/main/Assignment-Course5/DSA-2/Assignment-4/app.js