

DSA PRACTICE – 9

Name: Deepak S

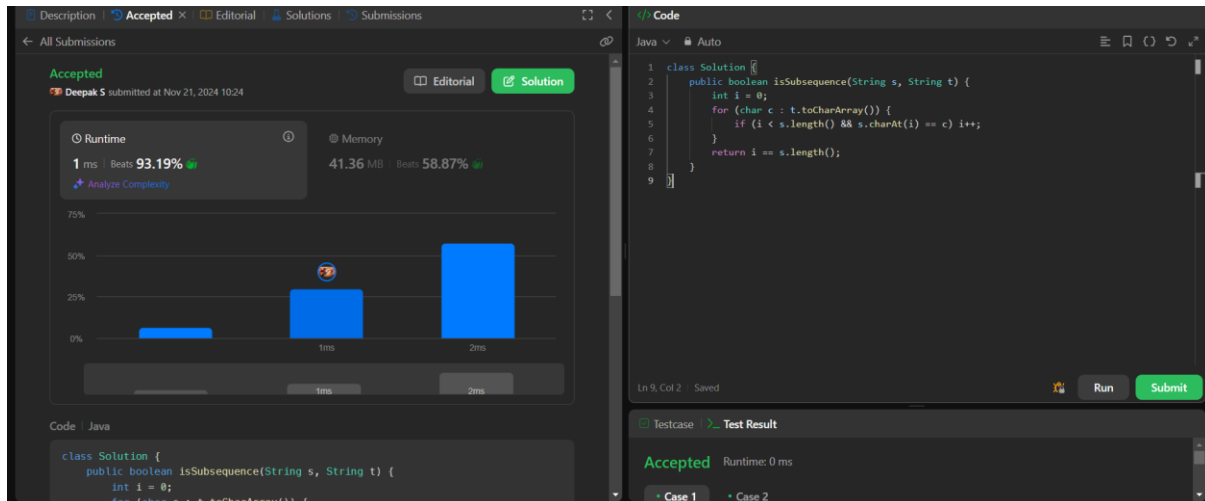
Register No: 22IT018

1.Is Subsequence

Code:

```
class Solution {
    public boolean isSubsequence(String s, String t) {
        int i = 0;
        for (char c : t.toCharArray()) {
            if (i < s.length() && s.charAt(i) == c) i++;
        }
        return i == s.length();
    }
}
```

Output:



2.Valid Palindrome

Code:

```
class Solution {
    public boolean isPalindrome(String s) {
        s = s.toLowerCase();
        int i = 0;
        int j = s.length()-1;
        while(i < j){
            if(!Character.isDigit(s.charAt(i)) &&
!Character.isLetter(s.charAt(i))){
                i++;
                continue;
            }
            if(!Character.isDigit(s.charAt(j)) &&
!Character.isLetter(s.charAt(j))){
                j--;
                continue;
            }
            if(!(s.charAt(i) == s.charAt(j))){
                return false;
            }
            i++;
            j--;
        }
        return true;
    }
}
```

Output:

Accepted

👤 Deepak S submitted at Nov 21, 2024 13:32

📖 Editorial

✍️ Solution

🕒 Runtime

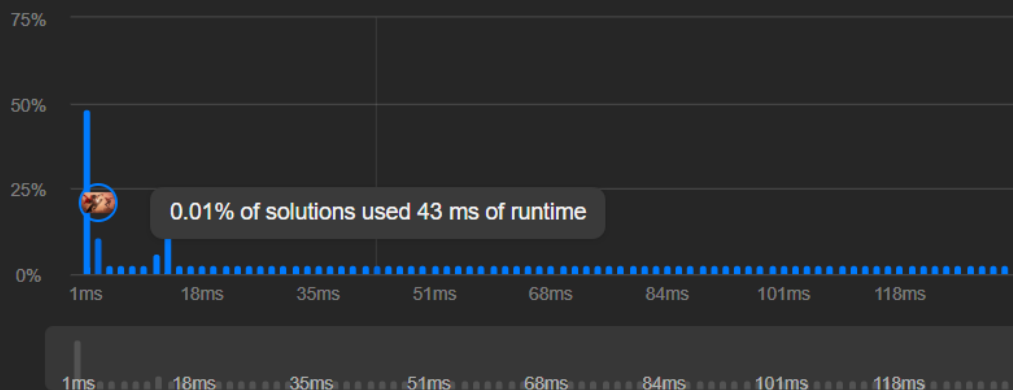


4 ms | Beats 51.59% 🌱

🔍 Analyze Complexity

💾 Memory

42.55 MB | Beats 95.05% 🌱

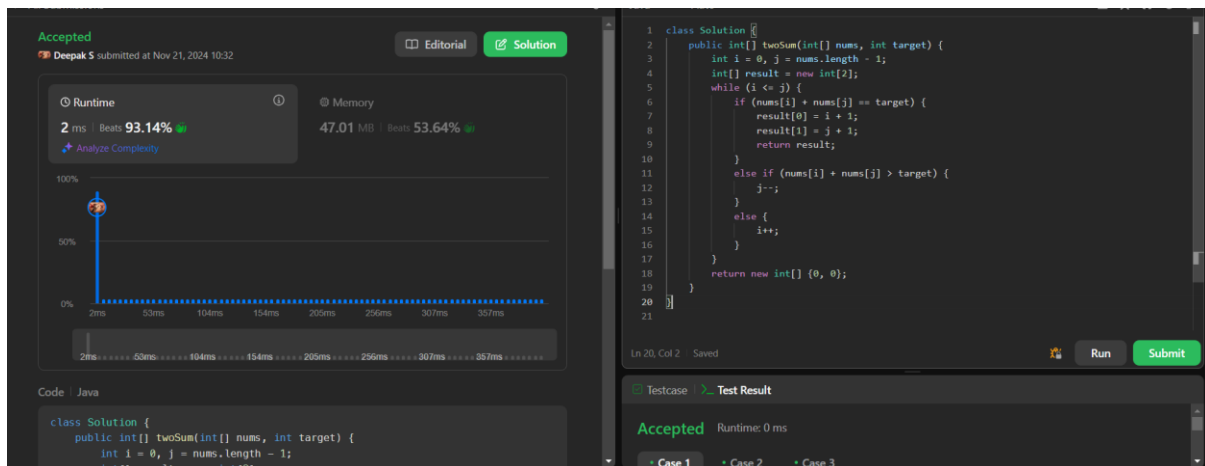


3.Two Sum II - Input Array Is Sorted

Code:

```
class Solution {
    public int[] twoSum(int[] nums, int target) {
        int i = 0, j = nums.length - 1;
        int[] result = new int[2];
        while (i <= j) {
            if (nums[i] + nums[j] == target) {
                result[0] = i + 1;
                result[1] = j + 1;
                return result;
            }
            else if (nums[i] + nums[j] > target) {
                j--;
            }
            else {
                i++;
            }
        }
        return new int[] {0, 0};
    }
}
```

Output:



4.Container With Most Water

Code:

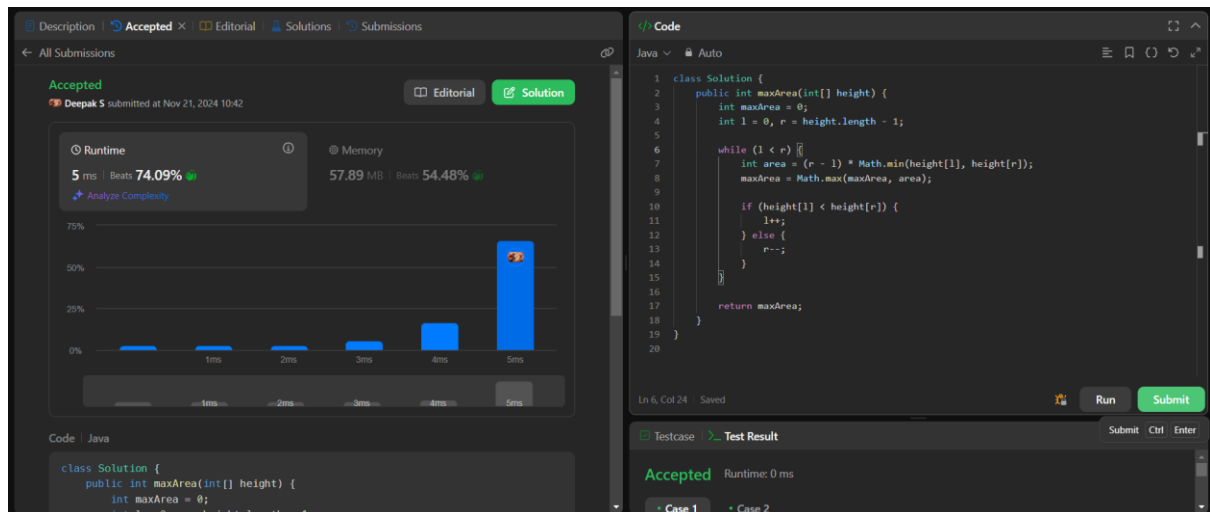
```
class Solution {
    public int maxArea(int[] height) {
        int maxArea = 0;
        int l = 0, r = height.length - 1;

        while (l < r) {
            int area = (r - l) * Math.min(height[l], height[r]);
            maxArea = Math.max(maxArea, area);

            if (height[l] < height[r]) {
                l++;
            } else {
                r--;
            }
        }

        return maxArea;
    }
}
```

Output:



5.3Sum

Code:

```
import java.util.*;

class Solution {
    public List<List<Integer>> threeSum(int[] nums) {
        int target = 0;
        Arrays.sort(nums);
        Set<List<Integer>> s = new HashSet<>();
        List<List<Integer>> output = new ArrayList<>();

        for (int i = 0; i < nums.length; i++) {
            int j = i + 1;
            int k = nums.length - 1;

            while (j < k) {
                int sum = nums[i] + nums[j] + nums[k];
                if (sum == target) {
                    s.add(Arrays.asList(nums[i], nums[j], nums[k]));
                    j++;
                    k--;
                } else if (sum < target) {
                    j++;
                } else {
                    k--;
                }
            }
        }

        output.addAll(s);
        return output;
    }
}
```

Output:

The screenshot displays a code submission interface for the 5.3Sum problem. On the left, a performance summary shows the solution is 'Accepted' with a runtime of 832 ms (beats 13.72%) and a memory usage of 52.55 MB (beats 18.03%). Below this is a runtime distribution chart. The main area shows the Java code for the Solution class, which implements the threeSum method. The code sorts the input array and uses a two-pointer technique to find all unique triplets that sum to zero. On the right, a 'Testcase' section shows the result 'Accepted' with a runtime of 0 ms. The bottom of the interface includes buttons for 'Run' and 'Submit'.

Accepted
Deepak S submitted at Nov 21, 2024 11:07

Editorial Solution

Runtime: 832 ms Beats 13.72%
Memory: 52.55 MB Beats 18.03%

9ms 194ms 378ms 563ms 748ms 933ms 1118ms 1303ms

```
import java.util.*;

class Solution {
    public List<List<Integer>> threeSum(int[] nums) {
        int target = 0;
        Arrays.sort(nums);
        Set<List<Integer>> s = new HashSet<>();
        List<List<Integer>> output = new ArrayList<>();

        for (int i = 0; i < nums.length; i++) {
            int j = i + 1;
            int k = nums.length - 1;

            while (j < k) {
                int sum = nums[i] + nums[j] + nums[k];
                if (sum == target) {
                    s.add(Arrays.asList(nums[i], nums[j], nums[k]));
                    j++;
                    k--;
                } else if (sum < target) {
                    j++;
                } else {
                    k--;
                }
            }
        }

        output.addAll(s);
        return output;
    }
}
```

Ln 32, Col 1 Saved Run Submit

Testcase Test Result

Accepted Runtime: 0 ms

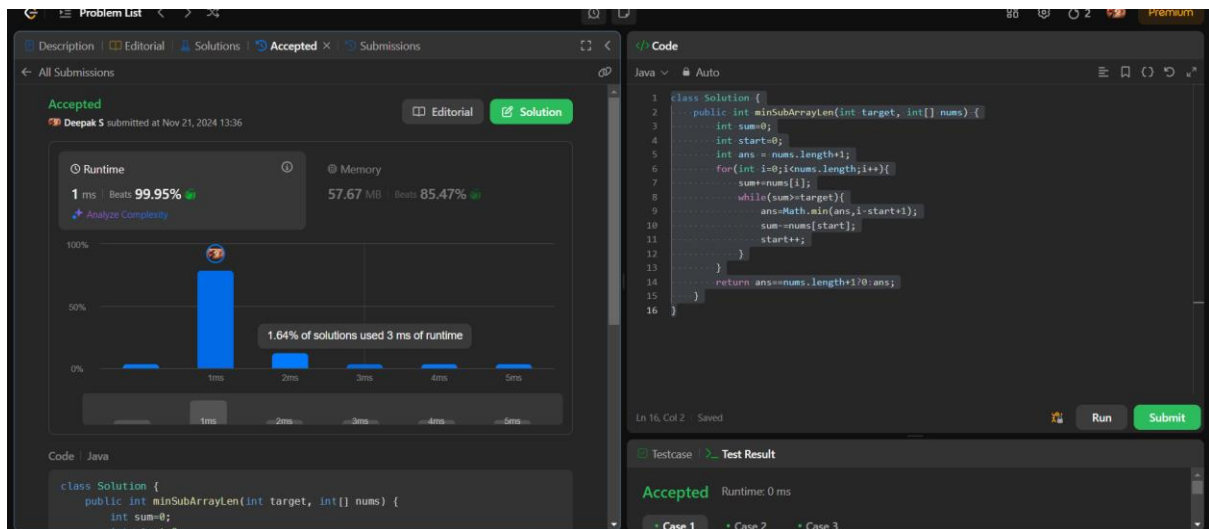
Case 1 Case 2 Case 3

6. Minimum Size Subarray Sum

Code:

```
class Solution {
    public int minSubArrayLen(int target, int[] nums) {
        int sum=0;
        int start=0;
        int ans = nums.length+1;
        for(int i=0;i<nums.length;i++){
            sum+=nums[i];
            while(sum>=target){
                ans=Math.min(ans,i-start+1);
                sum-=nums[start];
                start++;
            }
        }
        return ans==nums.length+1?0:ans;
    }
}
```

Output:



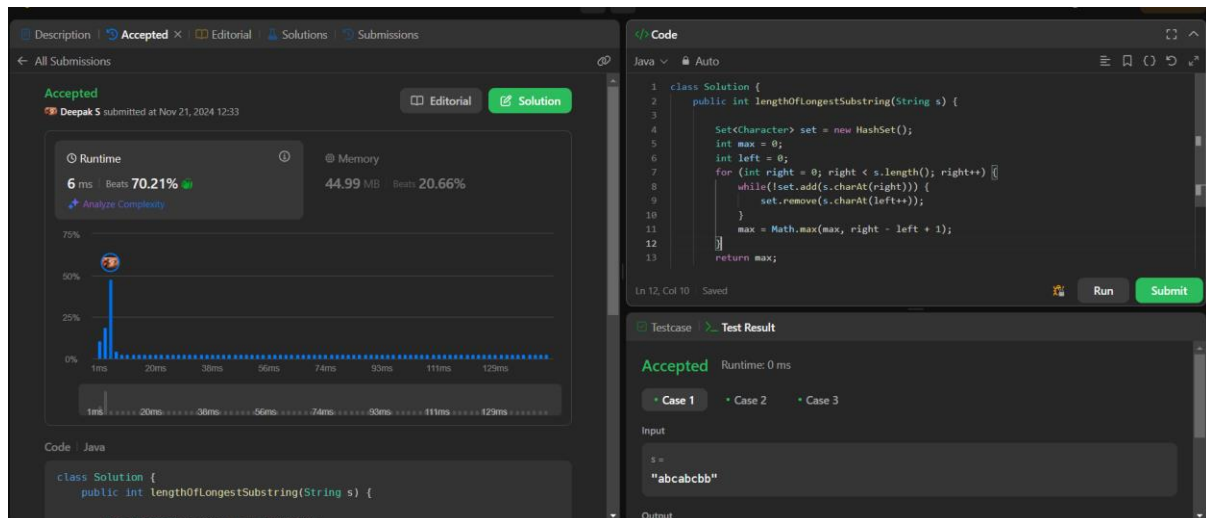
7.Longest Substring Without Repeating Characters

Code:

```
class Solution {
    public int lengthOfLongestSubstring(String s) {

        Set<Character> set = new HashSet();
        int max = 0;
        int left = 0;
        for (int right = 0; right < s.length(); right++) {
            while(!set.add(s.charAt(right))) {
                set.remove(s.charAt(left++));
            }
            max = Math.max(max, right - left + 1);
        }
        return max;
    }
}
```

Output:



8.Min Stack

Code:

```
class MinStack {
    Stack<Integer> s1;
    Stack<Integer> s2;

    public MinStack() {
        s1 = new Stack<>();
        s2 = new Stack<>();
    }

    public void push(int val) {
        s1.push(val);
        if (s2.isEmpty() || val <= s2.peek()) {
            s2.push(val);
        }
    }

    public void pop() {
        int popped = s1.pop();
        if (popped == s2.peek()) {
            s2.pop();
        }
    }

    public int top() {
        return s1.peek();
    }

    public int getMin() {
        return s2.peek();
    }
}

/**
 * Your MinStack object will be instantiated and called as such:
 * MinStack obj = new MinStack();
 * obj.push(val);
 * obj.pop();
 * int param_3 = obj.top();
 * int param_4 = obj.getMin();
 */
```


Output:

Description

Accepted

Editorial

Solutions

Submissions

All Submissions

Accepted

Deepak S submitted at Nov 21, 2024 13:17

Editorial

Solution

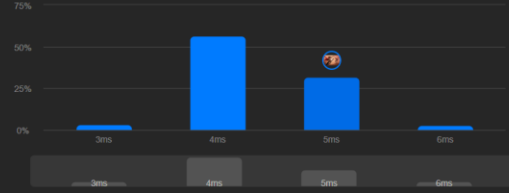
Runtime

5 ms | Beats 39.88%

Analyze Complexity

Memory

44.54 MB | Beats 84.75%



Runtime	Percentage
3ms	~5%
4ms	~55%
5ms	~30%
6ms	~5%

Code / Java

```
class MinStack {
    Stack<Integer> s1;
    Stack<Integer> s2;
}
```

Code

Auto

```
14
15
16
17 public void pop() {
18     int popped = s1.pop();
19     if (popped == s2.peek()) {
20         s2.pop();
21     }
22 }
23
```

Ln 21, Col 10 | Saved

Run

Submit

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1

Input

["MinStack","push","push","push","getMin","pop","top","getMin"]

[[[]],[-2],[0],[-3],[[],[],[],[]]]

Output

[null,null,null,null,-3,null,0,-2]

9.Search Insert Position0

Code:

```
class Solution {
    public int searchInsert(int[] nums, int target) {
        return binarySearch(nums, target);
    }

    private int binarySearch(int[] nums, int target) {
        int low = 0;
        int high = nums.length - 1;

        while (low <= high) {
            int mid = (low + high) / 2;

            if (nums[mid] > target) {
                high = mid - 1;
            } else if (nums[mid] < target) {
                low = mid + 1;
            } else {
                return mid;
            }
        }

        return low;
    }
}
```

Output:

The screenshot displays a code editor interface for a Java solution. On the left, a submission status panel shows 'Accepted' for a submission by 'Deepak S' on Nov 21, 2024. It includes performance metrics: Runtime 0 ms (Beats 100.00%) and Memory 42.94 MB (Beats 41.40%). Below this, a message states 'Sorry, there are not enough accepted submissions to show data'. The main editor shows the Java code for the 'Search Insert Position' problem. The code is as follows:

```
1 class Solution {
2     public int searchInsert(int[] nums, int target) {
3         return binarySearch(nums, target);
4     }
5
6     private int binarySearch(int[] nums, int target) {
7         int low = 0;
8         int high = nums.length - 1;
9
10        while (low <= high) {
11            int mid = (low + high) / 2;
12
13            if (nums[mid] > target) {
14                high = mid - 1;
15            } else if (nums[mid] < target) {
16                low = mid + 1;
17            } else {
18                return mid;
19            }
20        }
21
22        return low;
23    }
24 }
```

At the bottom, the 'Testcase' tab shows 'Accepted' for all three cases (Case 1, Case 2, Case 3) with a runtime of 0 ms.

10.Evaluate Reverse Polish Notation

```
import java.util.Stack;

class Solution {
    public int evalRPN(String[] tokens) {
        Stack<Integer> st = new Stack<>();

        for (String token : tokens) {
            if (token.matches("-?\\d+")) {
                st.push(Integer.parseInt(token));
            } else {
                int b = st.pop();
                int a = st.pop();
                switch (token) {
                    case "+":
                        st.push(a + b);
                        break;
                    case "-":
                        st.push(a - b);
                        break;
                    case "*":
                        st.push(a * b);
                        break;
                    case "/":
                        st.push(a / b);
                        break;
                }
            }
        }

        return st.peek();
    }
}
```

Output:

The screenshot displays a submission interface for the 'Evaluate Reverse Polish Notation' problem. On the left, the submission status is 'Accepted' for user 'Deepak S' on Nov 21, 2024. Performance metrics show a runtime of 24 ms (beats 5.39%) and memory usage of 46.89 MB (beats 6.28%). A bar chart illustrates the runtime distribution across different time intervals. The code editor on the right shows the Java solution, which uses a stack to evaluate the RPN expression. The 'Testcase' section shows 'Case 1' with an input of `tokens = ["2","1","+","3","+"]` and an output of `9`. The 'Test Result' section indicates the solution was 'Accepted' with a runtime of 0 ms.

Accepted
Deepak S submitted at Nov 21, 2024 13:59

Runtime: 24 ms | Beats: 5.39%
Memory: 46.89 MB | Beats: 6.28%

Code: Java

```
import java.util.Stack;

class Solution {
    public int evalRPN(String[] tokens) {
        Stack<Integer> st = new Stack<>();

        for (String token : tokens) {
            if (token.matches("-?\\d+")) {
                st.push(Integer.parseInt(token));
            } else {
                int b = st.pop();
                int a = st.pop();
                switch (token) {
                    case "+":
                        st.push(a + b);
                        break;
                    case "-":
                        st.push(a - b);
                        break;
                    case "*":
                        st.push(a * b);
                        break;
                    case "/":
                        st.push(a / b);
                        break;
                }
            }
        }

        return st.peek();
    }
}
```

Testcase: Case 1
Input: tokens = ["2","1","+","3","+"]
Output: 9
Test Result: Accepted Runtime: 0 ms

11.Simplify Path

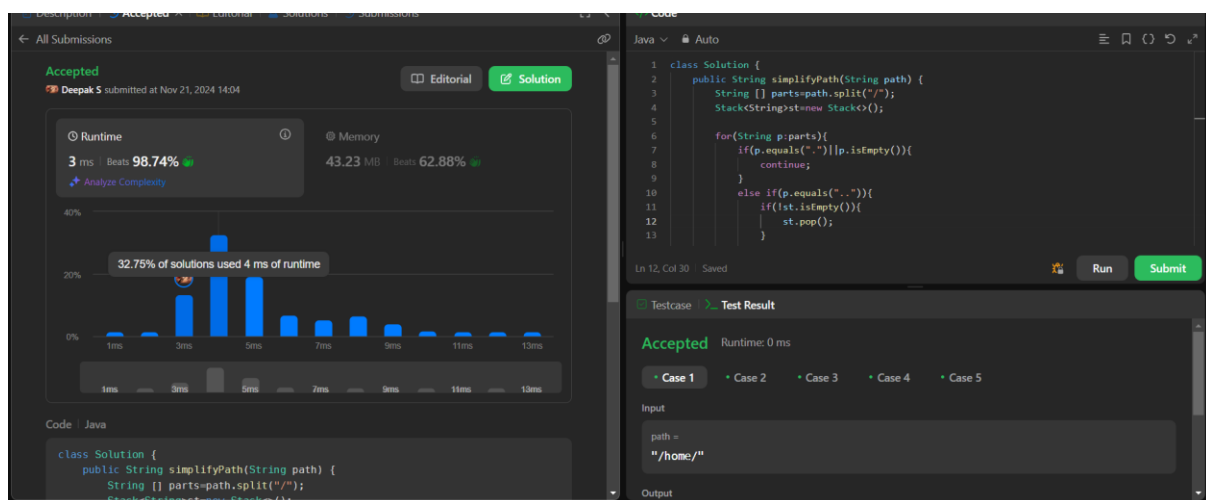
Code

```
class Solution {
    public String simplifyPath(String path) {
        String [] parts=path.split("/");
        Stack<String>st=new Stack<>();

        for(String p:parts){
            if(p.equals(".")||p.isEmpty()){
                continue;
            }
            else if(p.equals("..")){
                if(!st.isEmpty()){
                    st.pop();
                }
            }
            else{
                st.push(p);
            }
        }
        StringBuilder res=new StringBuilder();
        for(String p:st){
            res.append('/').append(p);
        }

        return res.length()>0?res.toString():"/";
    }
}
```

Output:



12.Find Peak Element

Code:

```
class Solution {
    public int findPeakElement(int[] nums) {
        int left = 0;
        int right = nums.length - 1;
        while (left < right) {
            int mid = (left + right) / 2;
            if (nums[mid] > nums[mid + 1]) {
                right = mid;
            } else {
                left = mid + 1;
            }
        }
        return left;
    }
}
```

Output:

The screenshot displays a code editor interface for a problem titled "Find Peak Element". The left sidebar shows the submission status as "Accepted" for user "Deepak S" on Nov 21, 2024. Performance metrics are listed: Runtime 0 ms (Beats 100.00%) and Memory 42.27 MB (Beats 59.70%). The main editor shows the Java code for the solution, which is a binary search implementation. The code is as follows:

```
1 class Solution {
2     public int findPeakElement(int[] nums) {
3         int left = 0;
4         int right = nums.length - 1;
5         while (left < right) {
6             int mid = (left + right) / 2;
7             if (nums[mid] > nums[mid + 1]) {
8                 right = mid;
9             } else {
10                 left = mid + 1;
11             }
12         }
13         return left;
14     }
15 }
```

At the bottom, the "Test Result" section shows "Accepted" with a runtime of 0 ms for "Case 1".

13.Find First and Last Position of Element in Sorted Array

Code:

```
class Solution {
    public int[] searchRange(int[] nums, int target) {
        int[] result = {-1, -1};

        for (int i = 0; i < nums.length; i++) {
            if (nums[i] == target) {
                if (result[0] == -1) result[0] = i;
                result[1] = i;
            }
        }

        return result;
    }
}
```

Output:

The screenshot displays a code editor interface with the following components:

- Top Bar:** Includes navigation icons and a language selector set to "Java".
- Left Panel:** Shows submission status "Accepted" for user "Deepak S" on Nov 21, 2024. It also displays performance metrics: Runtime 1 ms (Beats 17.84%) and Memory 45.88 MB (Beats 52.02%).
- Code Editor:** Contains the Java code for the `searchRange` method, which iterates through the array to find the first and last occurrences of the target.
- Right Panel:** Shows the "Test Result" section with a green "Accepted" status and a runtime of 0 ms. It also includes buttons for "Run" and "Submit".

14.Search a 2D Matrix:

Code:

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int rows = matrix.length, cols = matrix[0].length;
        int left = 0, right = rows * cols - 1;

        while (left <= right) {
            int mid = (left + right) / 2;
            int midValue = matrix[mid / cols][mid % cols];

            if (midValue == target) {
                return true;
            } else if (midValue < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return false;
    }
}
```

Output:

The screenshot displays a code editor interface for a problem titled "14. Search a 2D Matrix". The left sidebar shows submission statistics: "Accepted", "Deepak S" submitted at Nov 22, 2024 10:11, "Runtime: 0 ms", "Beats: 100.00%", "Memory: 42.44 MB", and "Beats: 12.35%". The main editor area shows the Java code for the solution. The right sidebar shows the "Test Result" section, which indicates "Accepted" with a "Runtime: 0 ms".

Code: Java

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int rows = matrix.length, cols = matrix[0].length;
        int left = 0, right = rows * cols - 1;

        while (left <= right) {
            int mid = (left + right) / 2;
            int midValue = matrix[mid / cols][mid % cols];

            if (midValue == target) {
                return true;
            } else if (midValue < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return false;
    }
}
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

15. Find Minimum in Rotated Sorted Array

Code:

```
class Solution {
    public int findMin(int[] nums) {
        int left = 0;
        int right = nums.length - 1;
        while(left < right){
            int mid = left + (right - left) / 2;
            if (nums[mid] <= nums[right]){
                right = mid;
            }
            else{
                left = mid + 1;
            }
        }
        return nums[left];
    }
}
```

Output:

The screenshot displays a code editor interface with two main panels. The left panel shows the submission status as 'Accepted' for a user named 'Deepak S' on Nov 22, 2024. It includes performance metrics: Runtime is 0 ms (Beats 100.00%) and Memory is 41.60 MB (Beats 94.00%). A bar chart visualizes the runtime performance, showing a single blue bar at 100%. The right panel shows the Java code for the 'findMin' method, which implements a binary search algorithm to find the minimum element in a rotated sorted array. The code is as follows:

```
1 class Solution {
2     public int findMin(int[] nums) {
3         int left = 0;
4         int right = nums.length - 1;
5         while(left < right){
6             int mid = left + (right - left) / 2;
7             if (nums[mid] <= nums[right]){
8                 right = mid;
9             }
10            else{
11                left = mid + 1;
12            }
13        }
14        return nums[left];
15    }
16 }
```

At the bottom, the 'Test Result' section shows 'Accepted' with a runtime of 0 ms for Case 1.

16.Valid Parentheses

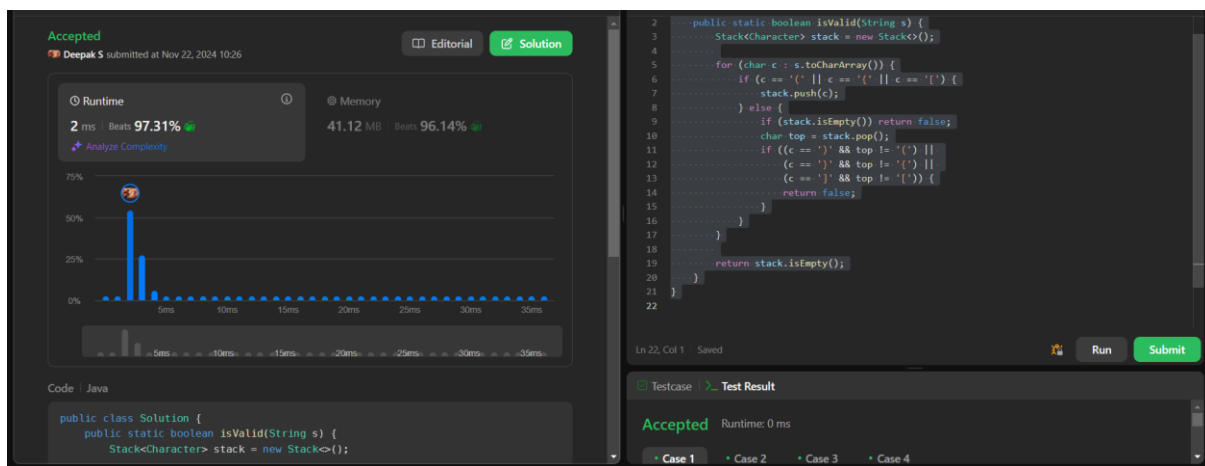
Code:

```
public class Solution {
    public static boolean isValid(String s) {
        Stack<Character> stack = new Stack<>();

        for (char c : s.toCharArray()) {
            if (c == '(' || c == '{' || c == '[') {
                stack.push(c);
            } else {
                if (stack.isEmpty()) return false;
                char top = stack.pop();
                if ((c == ')' && top != '(') ||
                    (c == '}' && top != '{') ||
                    (c == ']' && top != '[')) {
                    return false;
                }
            }
        }

        return stack.isEmpty();
    }
}
```

Output:



17.Find First and Last Position of Element in Sorted Array

Code:

```
class Solution {
    public int[] searchRange(int[] nums, int target) {
        int[] result = {-1, -1};

        for (int i = 0; i < nums.length; i++) {
            if (nums[i] == target) {
                if (result[0] == -1) result[0] = i;
                result[1] = i;
            }
        }

        return result;
    }
}
```

Output:

The screenshot displays a code editor interface with the following components:

- Top Bar:** Includes tabs for 'Description', 'Accepted', 'Note', 'Editorial', 'Solutions', and 'Submissions'. The 'Accepted' tab is active.
- Left Panel (Submission Details):**
 - Shows 'All Submissions' for the problem.
 - Indicates the submission is 'Accepted' by 'Deepak S' on 'Nov 22, 2024 10:31'.
 - Provides performance metrics: Runtime of 1 ms (Beats 17.84%) and Memory of 45.07 MB (Beats 99.90%).
 - Includes a bar chart showing the submission's performance relative to others.
 - Contains a 'Code' section with a snippet of the Java solution.
- Right Panel (Code Editor):**
 - Shows the full Java code for the 'Solution' class, which implements the 'searchRange' method.
 - Includes a 'Run' button and a 'Submit' button.
 - At the bottom, it shows 'Testcase' and 'Test Result' sections, indicating 'Accepted' with a runtime of 0 ms.