Assignment Complex Problem

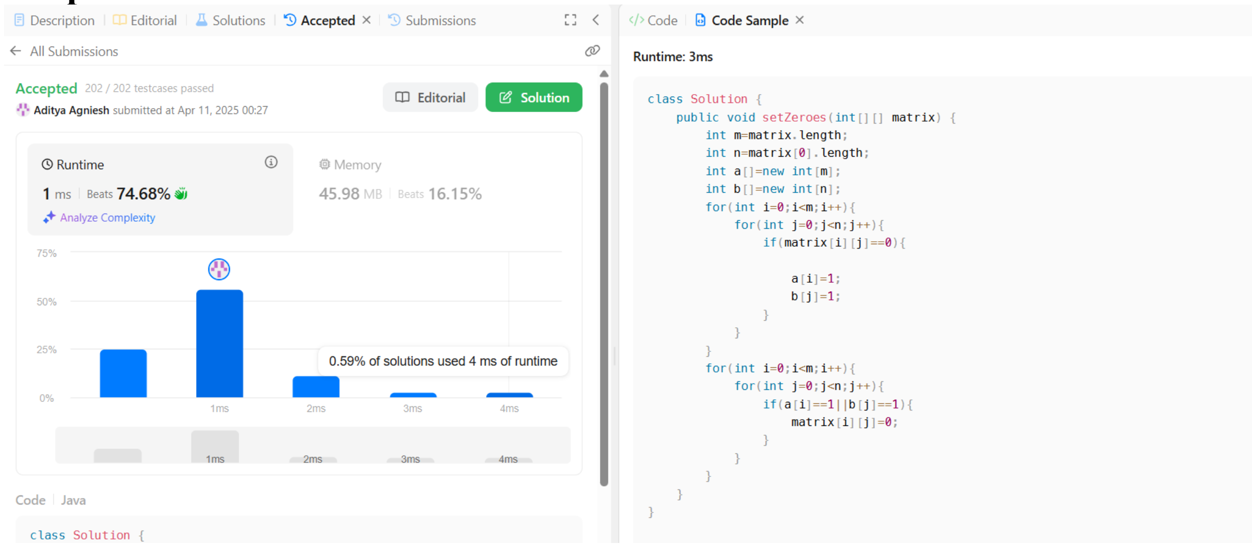
Student Name: Aditya Agniesh **UID:22BCS14644**
Branch: CSE **Section/Group:614_B**
Semester: 6th **Date of Performance:10/04/25**
Subject Name: AP lAB **Subject Code: 22CSP-351**

Ques 1: Set Matrix Zeroes

Code:

```
class Solution {
    public void setZeroes(int[][] matrix) {
      int rows = matrix.length, cols = matrix[0].length;
        List<Pair<Integer, Integer>> zeroPositions = new ArrayList<>();

        for (int row = 0; row < rows; row++) {
            for (int col = 0; col < cols; col++) {
                if (matrix[row][col] == 0) {
                    zeroPositions.add(new Pair<Integer, Integer>(row, col));
                }
            }
        }

        for (Pair<Integer, Integer> positions : zeroPositions) {
            int row = positions.getKey();
            int col = positions.getValue();

            for (int x = 0; x < cols; x++) {
                matrix[row][x] = 0;
            }

            for (int y = 0; y < rows; y++) {
                matrix[y][col] = 0;
            }
        }
    }
}
```

Output:



Accepted 202 / 202 testcases passed
Aditya Agniesh submitted at Apr 11, 2025 00:27

Runtime
1 ms | Beats 74.68%

Memory
45.98 MB | Beats 16.15%

0.59% of solutions used 4 ms of runtime

Code | Java

```
class Solution {
```

Runtime: 3ms

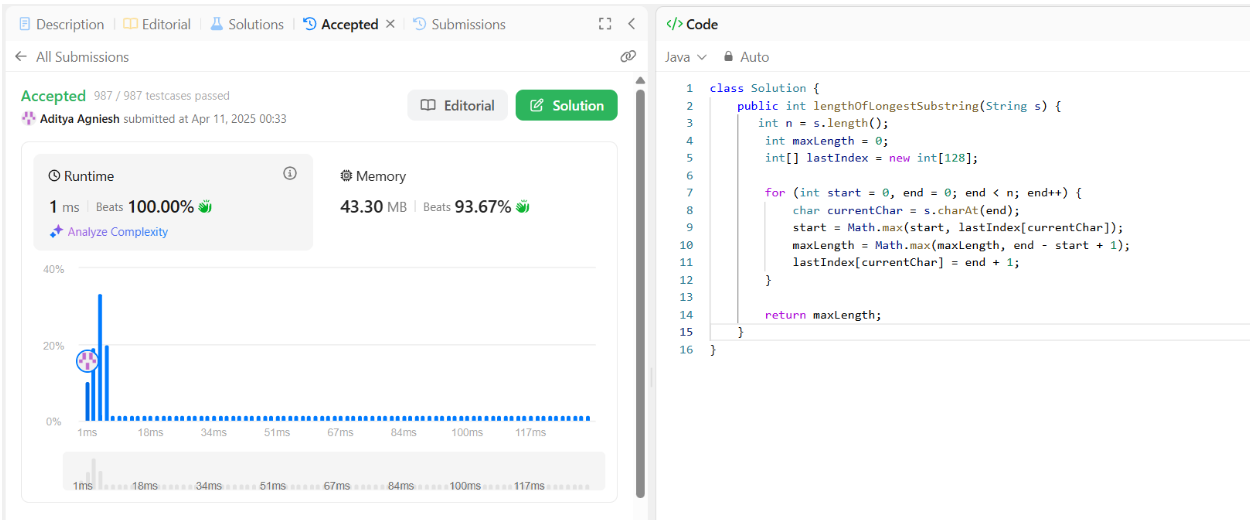
```
class Solution {
    public void setZeroes(int[][] matrix) {
        int m=matrix.length;
        int n=matrix[0].length;
        int a[]=new int[m];
        int b[]=new int[n];
        for(int i=0;i<m;i++){
            for(int j=0;j<n;j++){
                if(matrix[i][j]==0){

                    a[i]=1;
                    b[j]=1;
                }
            }
        }
        for(int i=0;i<m;i++){
            for(int j=0;j<n;j++){
                if(a[i]==1||b[j]==1){
                    matrix[i][j]=0;
                }
            }
        }
    }
}
```

Ques 2: Longest Substring Without Repeating Characters

Code:

```
class Solution {
    public int lengthOfLongestSubstring(String s) {
       int n = s.length();
        int maxLength = 0;
        int[] lastIndex = new int[128];

        for (int start = 0, end = 0; end < n; end++) {
            char currentChar = s.charAt(end);
            start = Math.max(start, lastIndex[currentChar]);
            maxLength = Math.max(maxLength, end - start + 1);
            lastIndex[currentChar] = end + 1;
        }

        return maxLength;
    }
}
```

Output:



Accepted 987 / 987 testcases passed
Aditya Agniesh submitted at Apr 11, 2025 00:33

Runtime 1 ms | Beats 100.00%
Memory 43.30 MB | Beats 93.67%

```java
class Solution {
    public int lengthOfLongestSubstring(String s) {
      int n = s.length();
       int maxLength = 0;
       int[] lastIndex = new int[128];

       for (int start = 0, end = 0; end < n; end++) {
           char currentChar = s.charAt(end);
           start = Math.max(start, lastIndex[currentChar]);
           maxLength = Math.max(maxLength, end - start + 1);
           lastIndex[currentChar] = end + 1;
       }

       return maxLength;
    }
}
```

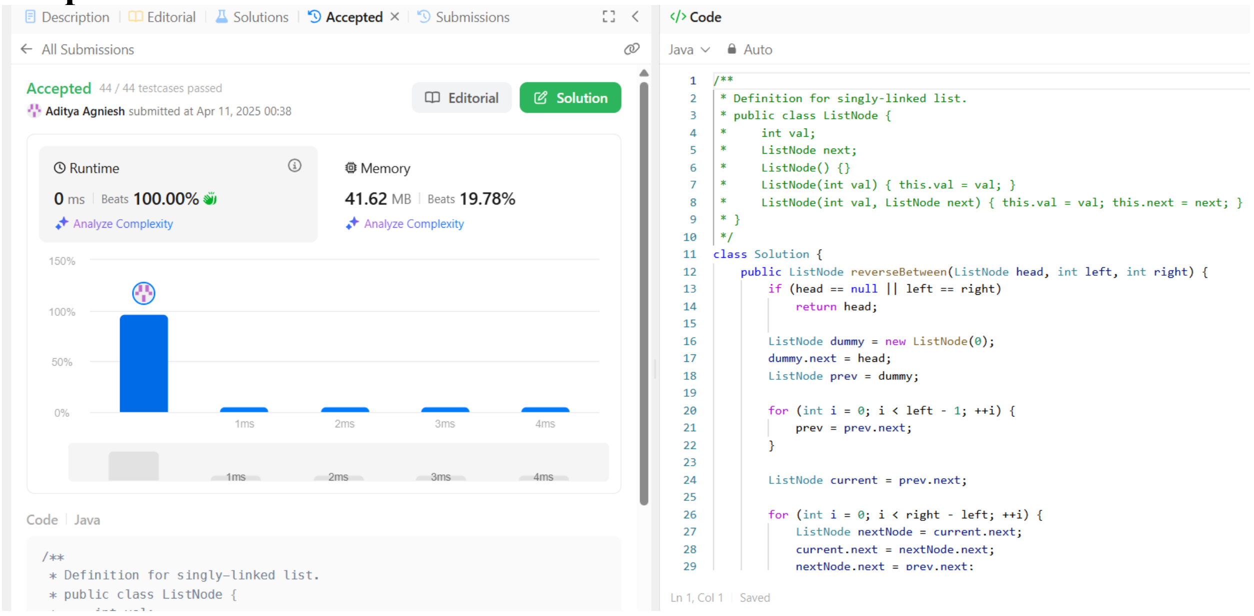
Ques 3: Reverse Linked List II

Code:

```java
class Solution {
    public ListNode reverseBetween(ListNode head, int left, int right) {
        if (head == null || left == right)
            return head;
        ListNode dummy = new ListNode(0);
        dummy.next = head;
        ListNode prev = dummy;
        for (int i = 0; i < left - 1; ++i) {
            prev = prev.next;
        }
        ListNode current = prev.next;
        for (int i = 0; i < right - left; ++i) {
            ListNode nextNode = current.next;
            current.next = nextNode.next;
            nextNode.next = prev.next;
            prev.next = nextNode;
      }
```

```
        return dummy.next;
    }
}
```

Output:



Description | Editorial | Solutions | Accepted × | Submissions

← All Submissions

Accepted 44 / 44 testcases passed
Aditya Agniesh submitted at Apr 11, 2025 00:38

Runtime
0 ms | Beats 100.00%

Memory
41.62 MB | Beats 19.78%

Code | Java

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
```

Code
Java Auto

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode reverseBetween(ListNode head, int left, int right) {
        if (head == null || left == right)
            return head;

        ListNode dummy = new ListNode(0);
        dummy.next = head;
        ListNode prev = dummy;

        for (int i = 0; i < left - 1; ++i) {
            prev = prev.next;
        }

        ListNode current = prev.next;

        for (int i = 0; i < right - left; ++i) {
            ListNode nextNode = current.next;
            current.next = nextNode.next;
            nextNode.next = prev.next;
```

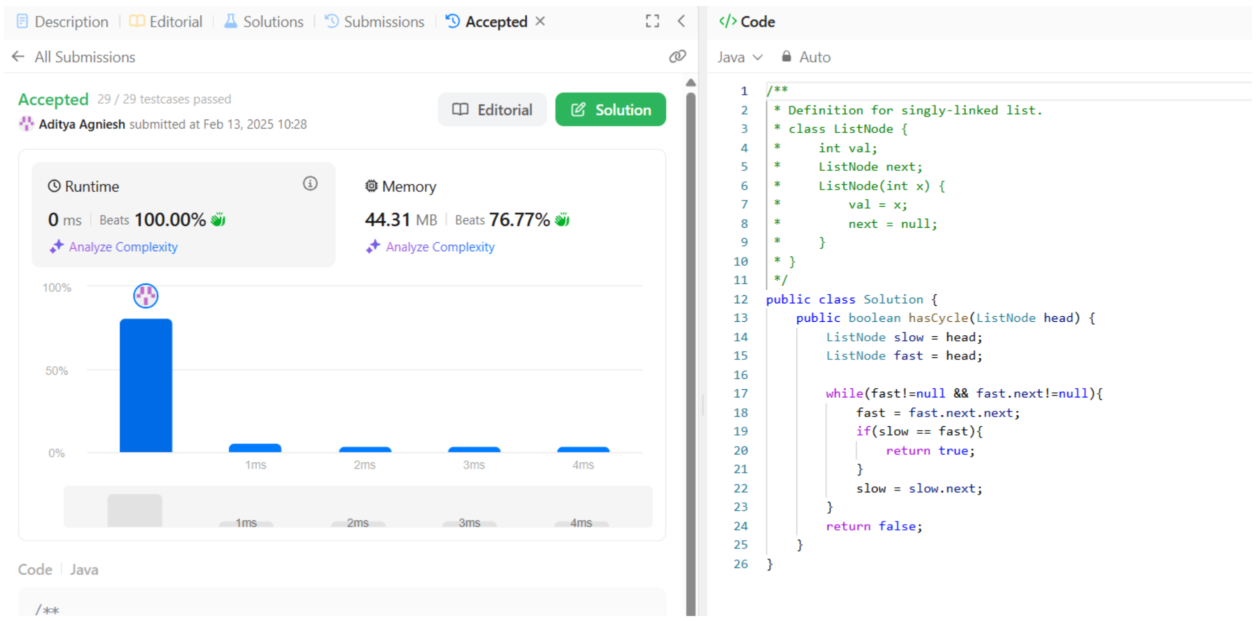
Ln 1, Col 1 | Saved

Ques 4: Detect a Cycle in a Linked List

Code:

```
public class Solution {
    public boolean hasCycle(ListNode head) {
        ListNode slow = head;
        ListNode fast = head;

        while(fast!=null && fast.next!=null){
            fast = fast.next.next;
            if(slow == fast){
                return true;
            }
            slow = slow.next;
        }
      return false;
    }
}
```

Output:



Accepted 29 / 29 testcases passed
Aditya Agniesh submitted at Feb 13, 2025 10:28

Runtime: 0 ms | Beats 100.00%
Memory: 44.31 MB | Beats 76.77%

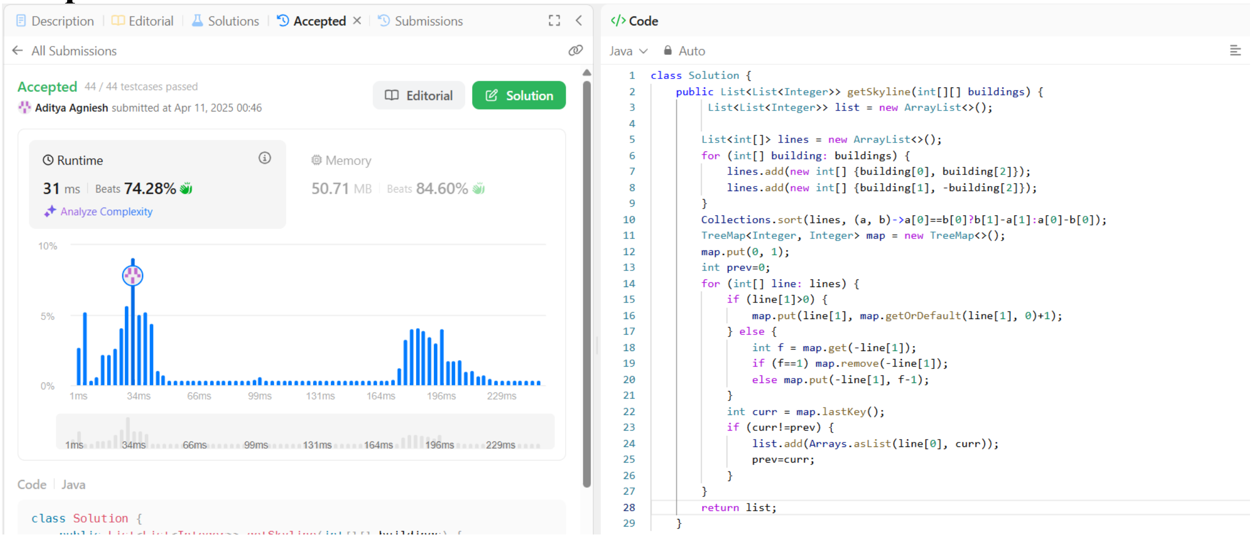
```
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) {
 *         val = x;
 *         next = null;
 *     }
 * }
 */
public class Solution {
    public boolean hasCycle(ListNode head) {
        ListNode slow = head;
        ListNode fast = head;

        while(fast!=null && fast.next!=null){
            fast = fast.next.next;
            if(slow == fast){
                return true;
            }
            slow = slow.next;
        }
        return false;
    }
}
```

Ques 5: The Skyline Problem

Code:

```
class Solution {
    public List<List<Integer>> getSkyline(int[][] buildings) {
        List<List<Integer>> list = new ArrayList<>();

        List<int[]> lines = new ArrayList<>();
        for (int[] building: buildings) {
            lines.add(new int[] {building[0], building[2]});
            lines.add(new int[] {building[1], -building[2]});
        }
        Collections.sort(lines, (a, b)->a[0]==b[0]?b[1]-a[1]:a[0]-b[0]);
        TreeMap<Integer, Integer> map = new TreeMap<>();
        map.put(0, 1);
        int prev=0;
        for (int[] line: lines) {
            if (line[1]>0) {
                map.put(line[1], map.getOrDefault(line[1], 0)+1);
            } else {
                int f = map.get(-line[1]);
                if (f==1) map.remove(-line[1]);
                else map.put(-line[1], f-1);
            }
            int curr = map.lastKey();
            if (curr!=prev) {
                list.add(Arrays.asList(line[0], curr));
                prev=curr;
            }
        }
        return list;
    }
}
```

Output:



Accepted 44 / 44 testcases passed
Aditya Agniesh submitted at Apr 11, 2025 00:46

Runtime
31 ms | Beats 74.28%

Memory
50.71 MB | Beats 84.60%

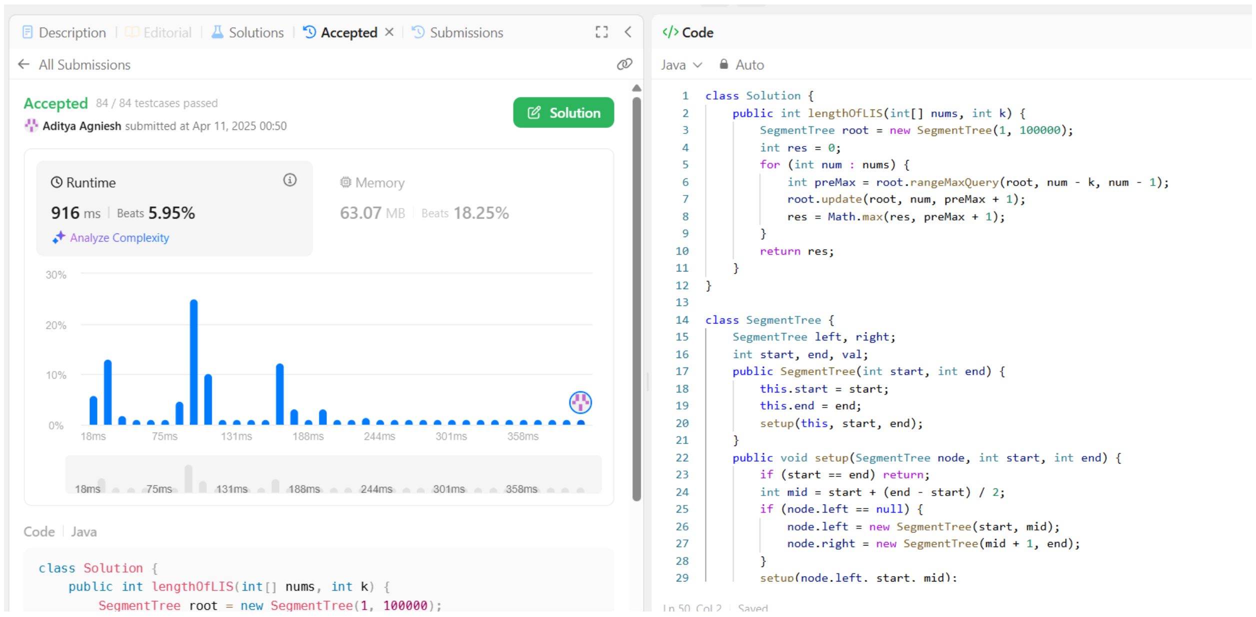
```
class Solution {
    public List<List<Integer>> getSkyline(int[][] buildings) {
        List<List<Integer>> list = new ArrayList<>();

        List<int[]> lines = new ArrayList<>();
        for (int[] building: buildings) {
            lines.add(new int[] {building[0], building[2]});
            lines.add(new int[] {building[1], -building[2]});
        }
        Collections.sort(lines, (a, b)->a[0]==b[0]?b[1]-a[1]:a[0]-b[0]);
        TreeMap<Integer, Integer> map = new TreeMap<>();
        map.put(0, 1);
        int prev=0;
        for (int[] line: lines) {
            if (line[1]>0) {
                map.put(line[1], map.getOrDefault(line[1], 0)+1);
            } else {
                int f = map.get(-line[1]);
                if (f==1) map.remove(-line[1]);
                else map.put(-line[1], f-1);
            }
            int curr = map.lastKey();
            if (curr!=prev) {
                list.add(Arrays.asList(line[0], curr));
                prev=curr;
            }
        }
        return list;
    }
```

Ques 6: Longest Increasing Subsequence II

Code:

```
class Solution {
    public int lengthOfLIS(int[] nums, int k) {
        SegmentTree root = new SegmentTree(1, 100000);
        int res = 0;
        for (int num : nums) {
            int preMax = root.rangeMaxQuery(root, num - k, num - 1);
            root.update(root, num, preMax + 1);
            res = Math.max(res, preMax + 1);
        }
        return res;
    }
}

class SegmentTree {
    SegmentTree left, right;
    int start, end, val;
    public SegmentTree(int start, int end) {
        this.start = start;
        this.end = end;
        setup(this, start, end);
    }
    public void setup(SegmentTree node, int start, int end) {
        if (start == end) return;
        int mid = start + (end - start) / 2;
        if (node.left == null) {
            node.left = new SegmentTree(start, mid);
            node.right = new SegmentTree(mid + 1, end);
        }
        setup(node.left, start, mid);
        setup(node.right, mid + 1, end);
        node.val = Math.max(node.left.val, node.right.val);
    }

    public void update(SegmentTree node, int index, int val) {
        if (index < node.start || index > node.end) return;
        if (node.start == node.end && node.start == index) {
            node.val = val;
            return;
        }
        update(node.left, index, val);
        update(node.right, index, val);
        node.val = Math.max(node.left.val, node.right.val);
    }

    public int rangeMaxQuery(SegmentTree node, int start, int end) {
        if (node.start > end || node.end < start) return 0;
        if (node.start >= start && node.end <= end) return node.val;
        return Math.max(rangeMaxQuery(node.left, start, end), rangeMaxQuery(node.right, start, end));
    }
}
```

Output:



Description | Editorial | Solutions | Accepted | Submissions

All Submissions

Accepted 84 / 84 testcases passed

Aditya Agniesh submitted at Apr 11, 2025 00:50

Runtime 916 ms | Beats 5.95%

Memory 63.07 MB | Beats 18.25%

Code | Java

```
class Solution {
    public int lengthOfLIS(int[] nums, int k) {
        SegmentTree root = new SegmentTree(1, 100000);
```

Code

Java Auto

```
class Solution {
    public int lengthOfLIS(int[] nums, int k) {
        SegmentTree root = new SegmentTree(1, 100000);
        int res = 0;
        for (int num : nums) {
            int preMax = root.rangeMaxQuery(root, num - k, num - 1);
            root.update(root, num, preMax + 1);
            res = Math.max(res, preMax + 1);
        }
        return res;
    }
}

class SegmentTree {
    SegmentTree left, right;
    int start, end, val;
    public SegmentTree(int start, int end) {
        this.start = start;
        this.end = end;
        setup(this, start, end);
    }
    public void setup(SegmentTree node, int start, int end) {
        if (start == end) return;
        int mid = start + (end - start) / 2;
        if (node.left == null) {
            node.left = new SegmentTree(start, mid);
            node.right = new SegmentTree(mid + 1, end);
        }
        setup(node.left, start, mid);
```

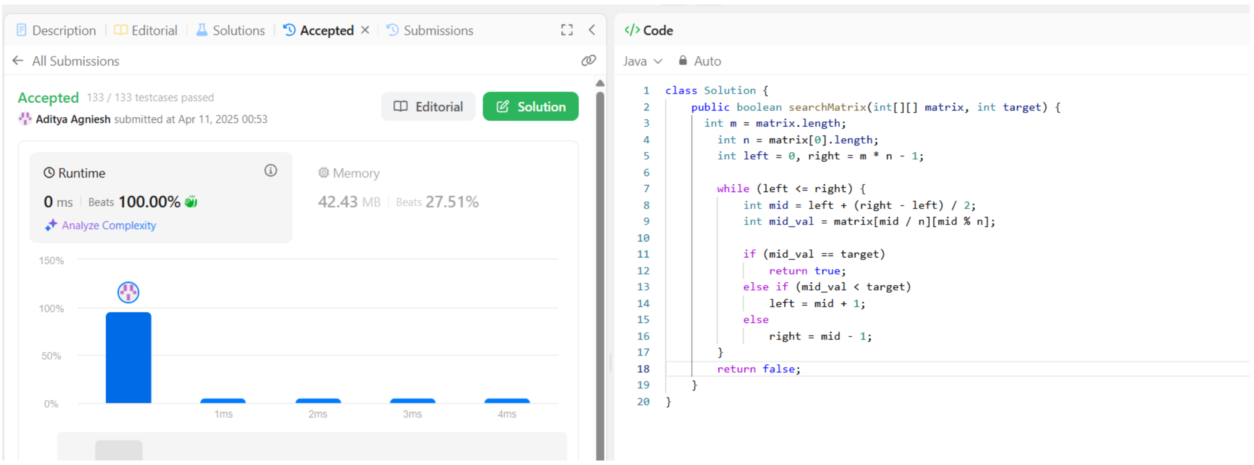
Ln 50, Col 2 Saved

Ques 7: Search a 2D Matrix II

Code:

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
      int m = matrix.length;
        int n = matrix[0].length;
        int left = 0, right = m * n - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            int mid_val = matrix[mid / n][mid % n];

            if (mid_val == target)
                return true;
            else if (mid_val < target)
                left = mid + 1;
            else
                right = mid - 1;
        }
        return false;
    }
}
```

Output:



Description | Editorial | Solutions | Accepted | Submissions

All Submissions

Accepted 133 / 133 testcases passed
Aditya Agniesh submitted at Apr 11, 2025 00:53

Runtime
0 ms | Beats 100.00%

Memory
42.43 MB | Beats 27.51%

Code
Java | Auto

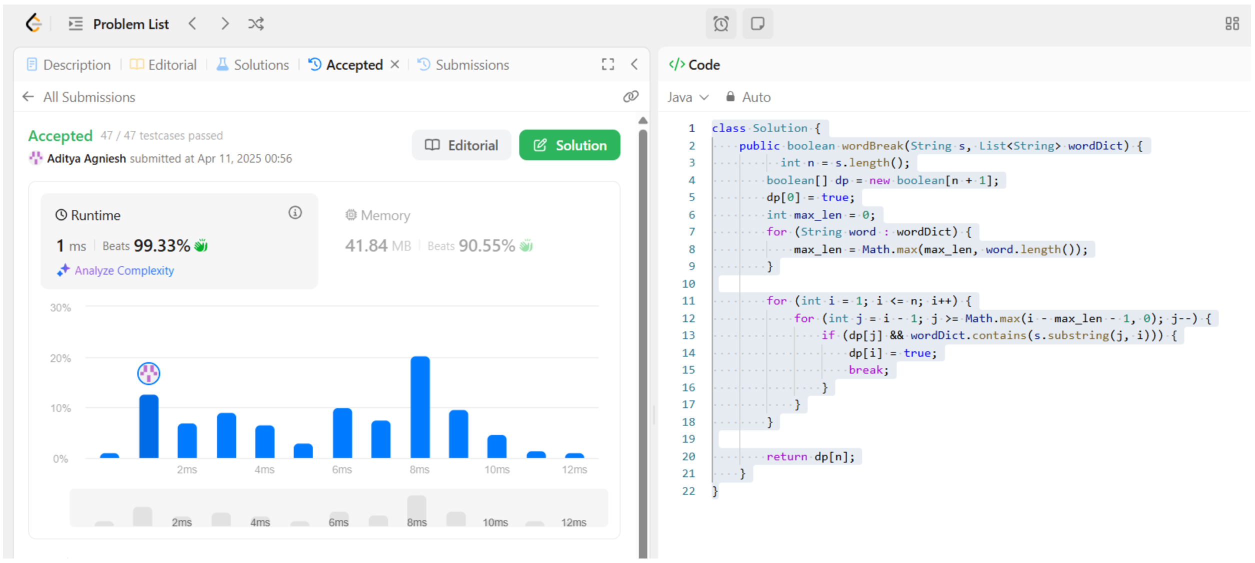
```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
      int m = matrix.length;
        int n = matrix[0].length;
        int left = 0, right = m * n - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            int mid_val = matrix[mid / n][mid % n];

            if (mid_val == target)
                return true;
            else if (mid_val < target)
                left = mid + 1;
            else
                right = mid - 1;
        }
        return false;
    }
}
```

Ques 8: Word Break

Code:

```
class Solution {
    public boolean wordBreak(String s, List<String> wordDict) {
          int n = s.length();
        boolean[] dp = new boolean[n + 1];
        dp[0] = true;
        int max_len = 0;
        for (String word : wordDict) {
            max_len = Math.max(max_len, word.length());
        }

        for (int i = 1; i <= n; i++) {
            for (int j = i - 1; j >= Math.max(i - max_len - 1, 0); j--) {
                if (dp[j] && wordDict.contains(s.substring(j, i))) {
                    dp[i] = true;
                    break;
                }
            }
        }

        return dp[n];
    }
}
```

Output:



Accepted 47 / 47 testcases passed
Aditya Agniesh submitted at Apr 11, 2025 00:56

Runtime
1 ms | Beats 99.33%
Analyze Complexity

Memory
41.84 MB | Beats 90.55%

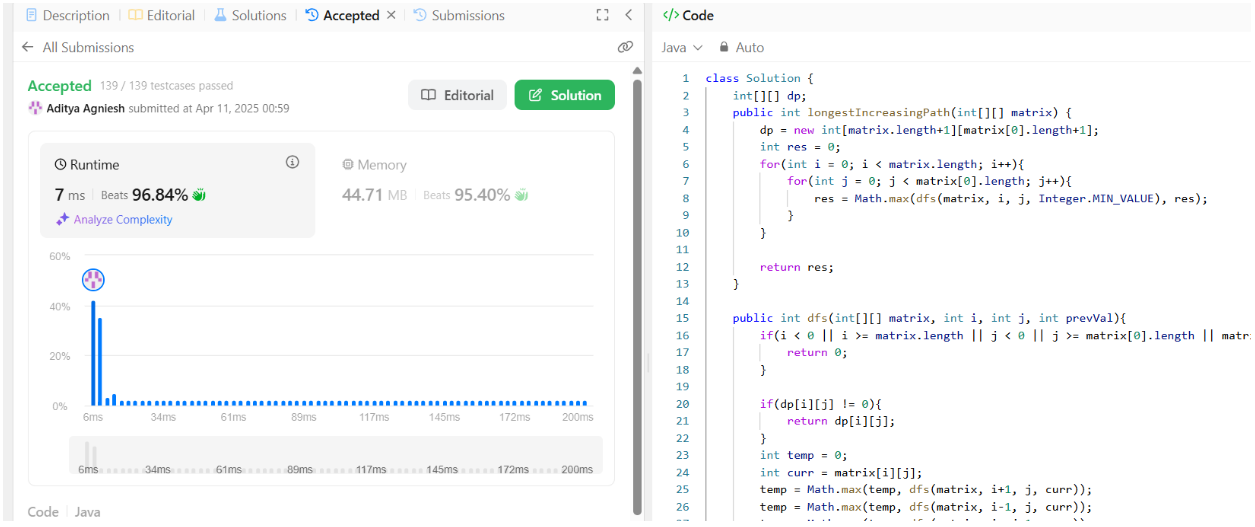
```
class Solution {
    public boolean wordBreak(String s, List<String> wordDict) {
        int n = s.length();
        boolean[] dp = new boolean[n + 1];
        dp[0] = true;
        int max_len = 0;
        for (String word : wordDict) {
            max_len = Math.max(max_len, word.length());
        }

        for (int i = 1; i <= n; i++) {
            for (int j = i - 1; j >= Math.max(i - max_len - 1, 0); j--) {
                if (dp[j] && wordDict.contains(s.substring(j, i))) {
                    dp[i] = true;
                    break;
                }
            }
        }

        return dp[n];
    }
}
```

Ques 9: Longest Increasing Path in a Matrix

Code:

```
class Solution {
    int[][] dp;
    public int longestIncreasingPath(int[][] matrix) {
        dp = new int[matrix.length+1][matrix[0].length+1];
        int res = 0;
        for(int i = 0; i < matrix.length; i++){
            for(int j = 0; j < matrix[0].length; j++){
                res = Math.max(dfs(matrix, i, j, Integer.MIN_VALUE), res);
            }
        }

        return res;
    }

    public int dfs(int[][] matrix, int i, int j, int prevVal){
        if(i < 0 || i >= matrix.length || j < 0 || j >= matrix[0].length || matrix[i][j] <=
prevVal){
            return 0;
        }

        if(dp[i][j] != 0){
            return dp[i][j];
        }
        int temp = 0;
        int curr = matrix[i][j];
        temp = Math.max(temp, dfs(matrix, i+1, j, curr));
        temp = Math.max(temp, dfs(matrix, i-1, j, curr));
        temp = Math.max(temp, dfs(matrix, i, j+1, curr));
        temp = Math.max(temp, dfs(matrix, i, j-1, curr));

        return dp[i][j] = ++temp;
    }
}
```

Output:


Description | Editorial | Solutions | Accepted × | Submissions
← All Submissions
Editorial
Solution
Runtime
Memory
Analyze Complexity
60%
40%
20%
0%
6ms 34ms 61ms 89ms 117ms 145ms 172ms 200ms
Code | Java
Code
Java
Auto

Accepted 139 / 139 testcases passed
Aditya Agniesh submitted at Apr 11, 2025 00:59

7 ms | Beats 96.84%

44.71 MB | Beats 95.40%

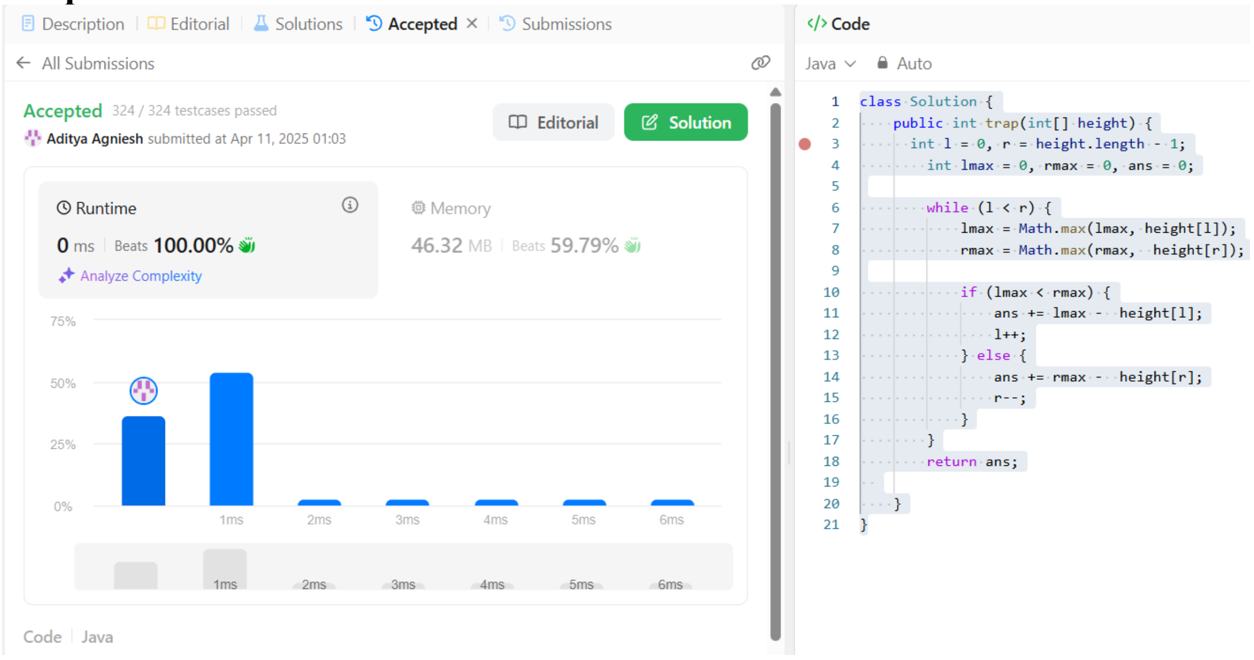
```
class Solution {
    int[][] dp;
    public int longestIncreasingPath(int[][] matrix) {
        dp = new int[matrix.length+1][matrix[0].length+1];
        int res = 0;
        for(int i = 0; i < matrix.length; i++){
            for(int j = 0; j < matrix[0].length; j++){
                res = Math.max(dfs(matrix, i, j, Integer.MIN_VALUE), res);
            }
        }

        return res;
    }

    public int dfs(int[][] matrix, int i, int j, int prevVal){
        if(i < 0 || i >= matrix.length || j < 0 || j >= matrix[0].length || matr
            return 0;
        }

        if(dp[i][j] != 0){
            return dp[i][j];
        }
        int temp = 0;
        int curr = matrix[i][j];
        temp = Math.max(temp, dfs(matrix, i+1, j, curr));
        temp = Math.max(temp, dfs(matrix, i-1, j, curr));
```

Ques 10: Trapping Rain Water

Code:

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

        while (l < r) {
            lmax = Math.max(lmax, height[l]);
            rmax = Math.max(rmax,  height[r]);

            if (lmax < rmax) {
                ans += lmax -  height[l];
                l++;
            } else {
                ans += rmax -  height[r];
                r--;
            }
        }
        return ans;

    }
}
```

Output:



Accepted 324 / 324 testcases passed
Aditya Agniesh submitted at Apr 11, 2025 01:03

Runtime 0 ms | Beats 100.00%
Memory 46.32 MB | Beats 59.79%

Code | Java

```java
class Solution {
    public int trap(int[] height) {
      int l = 0, r = height.length - 1;
        int lmax = 0, rmax = 0, ans = 0;

        while (l < r) {
            lmax = Math.max(lmax, height[l]);
            rmax = Math.max(rmax,  height[r]);

            if (lmax < rmax) {
                ans += lmax -  height[l];
                l++;
            } else {
                ans += rmax -  height[r];
                r--;
            }
        }
        return ans;

    }
}
```