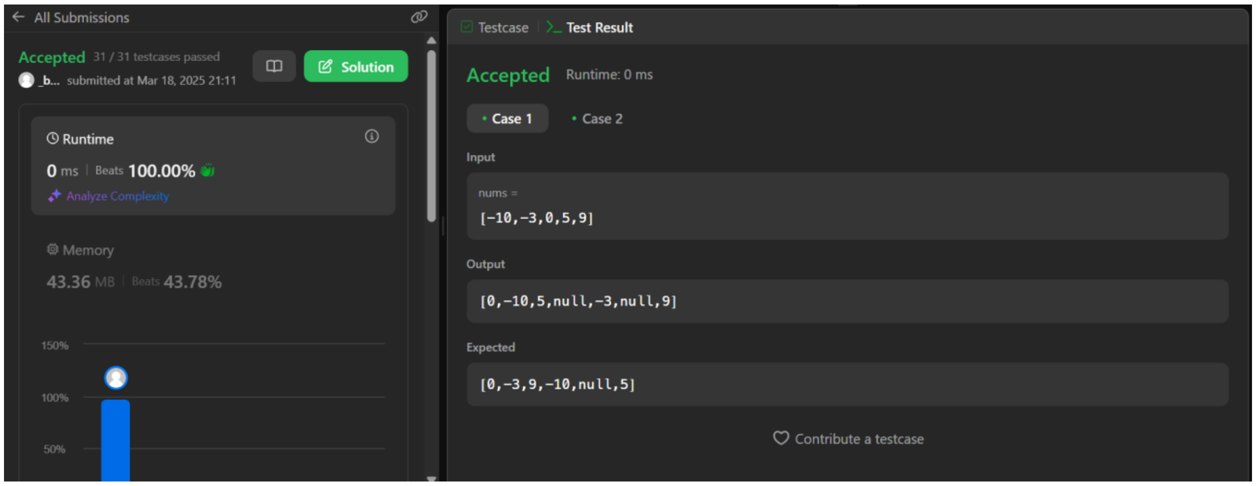
Experiment 6

Student Name: Barenya Behera **UID: 22BCS15121**
Branch: BE-CSE **Section/Group:FL_IOT-602/A**
Semester: 6th **Date of Performance: 18/03/25**
Subject Name: Advanced Programming Lab-2 **Subject Code: 22CSP-351**

1. Implementation/Code:

i. Convert Sorted Array to Binary Search Tree

```
class Solution {
  public TreeNode sortedArrayToBST(int[] nums) {
     return helper(nums, 0, nums.length - 1);
  }

  private TreeNode helper(int[] nums, int left, int right) {
     if (left > right) {
        return null; // base case
     }

     int mid = left + (right - left) / 2;
     TreeNode node = new TreeNode(nums[mid]);

     node.left = helper(nums, left, mid - 1);
     node.right = helper(nums, mid + 1, right);

     return node;
  }
}
```



All Submissions

Accepted 31 / 31 testcases passed
_b... submitted at Mar 18, 2025 21:11

Runtime 0 ms | Beats 100.00%

Memory 43.36 MB | Beats 43.78%

Testcase | Test Result

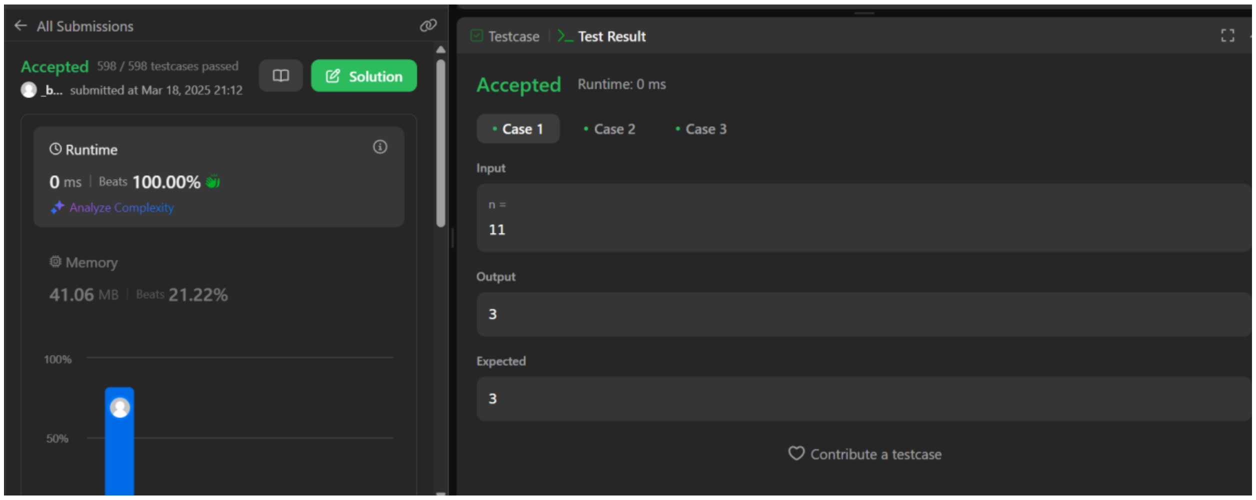
Accepted Runtime: 0 ms

Case 1 · Case 2

Input
```
nums =
[-10,-3,0,5,9]
```
Output
```
[0,-10,5,null,-3,null,9]
```
Expected
```
[0,-3,9,-10,null,5]
```

ii. Number of 1 Bits

```
class Solution {
   public int hammingWeight(int n) {
       int count = 0;
       while (n > 0) {
           n = n & (n - 1);
           count++;
       }
       return count;
   }
}
```



All Submissions

Accepted 598 / 598 testcases passed
_b... submitted at Mar 18, 2025 21:12

Runtime 0 ms | Beats 100.00%

Memory 41.06 MB | Beats 21.22%

Testcase | Test Result

Accepted Runtime: 0 ms

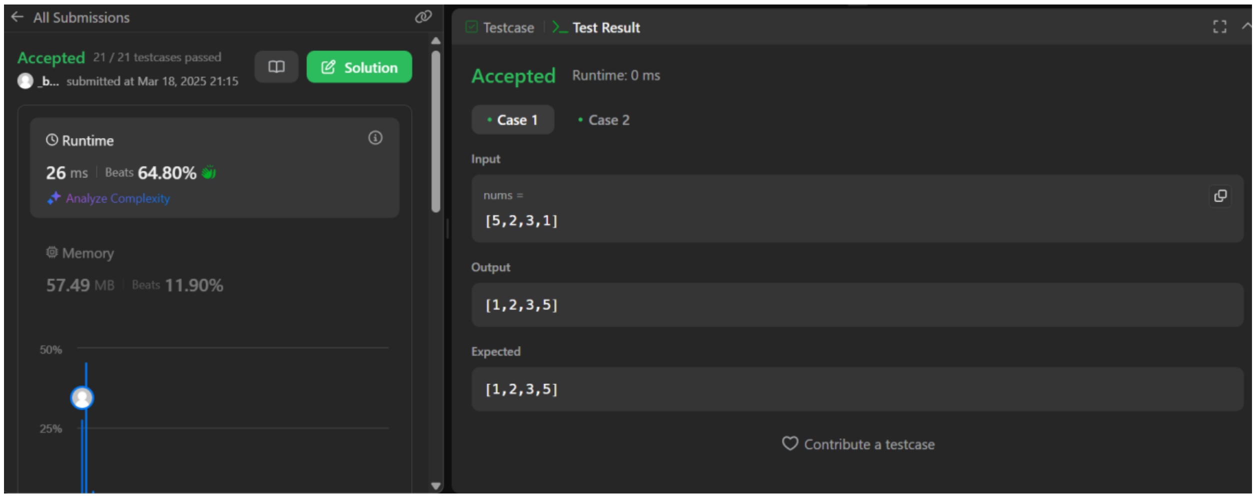
Case 1 · Case 2 · Case 3

Input
```
n =
11
```
Output
```
3
```
Expected
```
3
```

iii. Sort An Array

```
class Solution {
  public int[] sortArray(int[] nums) {
     if (nums == null || nums.length <= 1) {
        return nums;
     }
     mergeSort(nums, 0, nums.length - 1);
     return nums;
  }

  private void mergeSort(int[] nums, int left, int right) {
     if (left >= right) {
        return;
     }

     int mid = left + (right - left) / 2;
     mergeSort(nums, left, mid);
     mergeSort(nums, mid + 1, right);
     merge(nums, left, mid, right);
  }

  private void merge(int[] nums, int left, int mid, int right) {
     int[] temp = new int[right - left + 1];
     int i = left, j = mid + 1, k = 0;

     while (i <= mid && j <= right) {
        if (nums[i] <= nums[j]) {
           temp[k++] = nums[i++];
        } else {
           temp[k++] = nums[j++];
        }
     }
```

```
        while (i <= mid) {
            temp[k++] = nums[i++];
        }

        while (j <= right) {
            temp[k++] = nums[j++];
        }

        for (int p = 0; p < temp.length; p++) {
            nums[left + p] = temp[p];
        }
    }
}
```

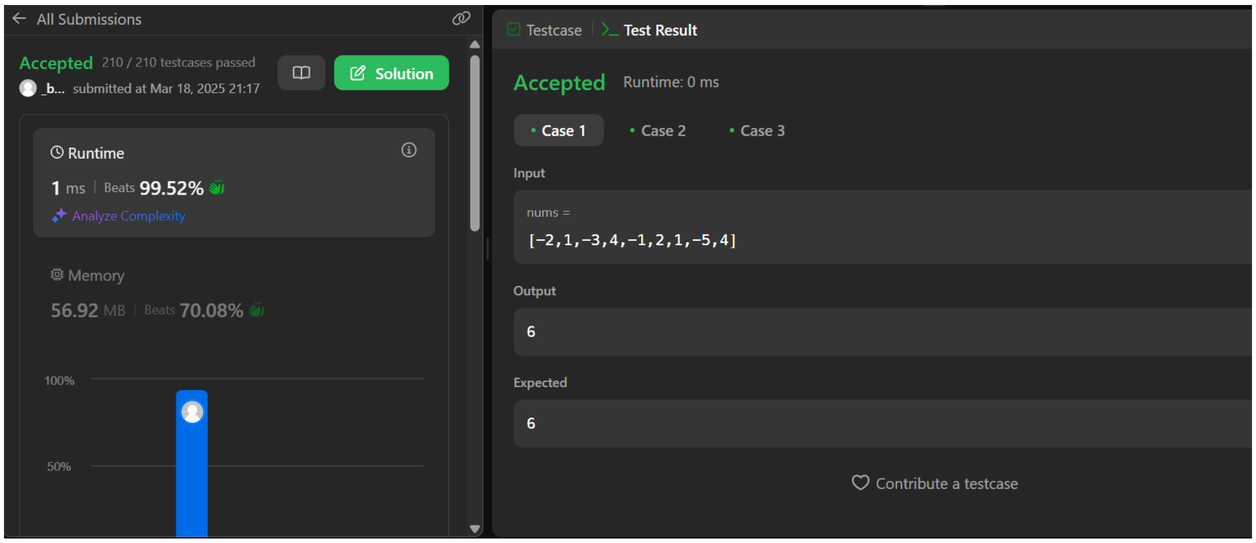


```
All Submissions
Accepted 21 / 21 testcases passed
_b... submitted at Mar 18, 2025 21:15
Solution

Runtime
26 ms | Beats 64.80%
Analyze Complexity

Memory
57.49 MB | Beats 11.90%

50%
25%
```

```
Testcase | Test Result

Accepted Runtime: 0 ms
Case 1   Case 2

Input
nums =
[5,2,3,1]

Output
[1,2,3,5]

Expected
[1,2,3,5]

Contribute a testcase
```

iv. Maximum Subarray

```
class Solution {
    public int maxSubArray(int[] nums) {
        int max=nums[0];
        int current=nums[0];
        for(int i=1;i<nums.length;i++){
            current=Math.max(nums[i],current+nums[i]);
            max=Math.max(current,max);
        }
        return max;
    }
}
```



All Submissions

Accepted 210 / 210 testcases passed

_b... submitted at Mar 18, 2025 21:17

Runtime

1 ms | Beats 99.52%

Memory

56.92 MB | Beats 70.08%

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
nums =
[-2,1,-3,4,-1,2,1,-5,4]
```

Output

```
6
```

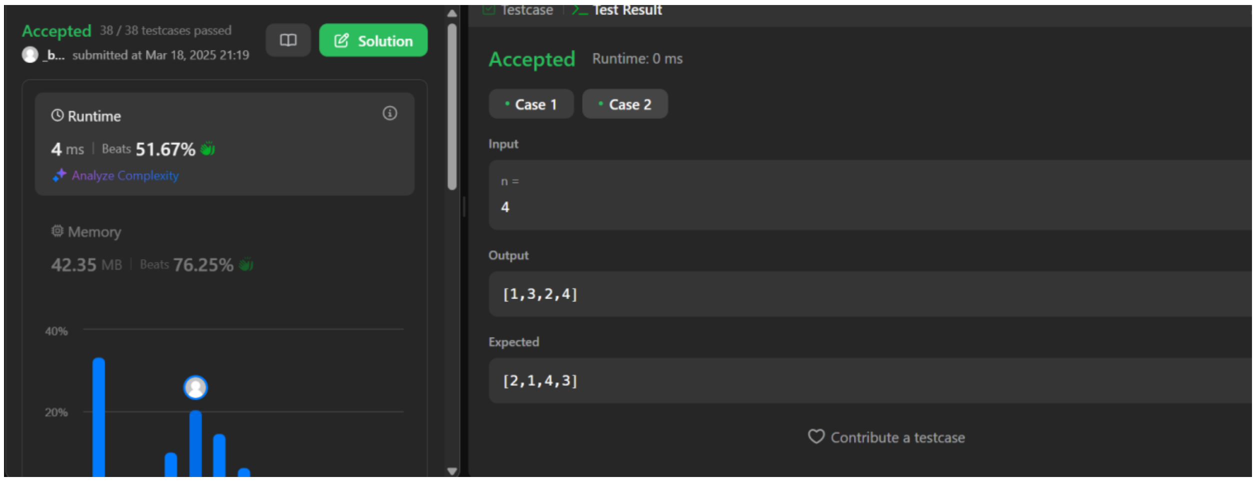
Expected

```
6
```

v. Beautiful Array

```
class Solution {
  public int[] beautifulArray(int n) {
      List<Integer> result = new ArrayList<>();
      result.add(1);

      while (result.size() < n) {
          List<Integer> temp = new ArrayList<>();
          for (int x : result) {
              if (2 * x - 1 <= n) {
                  temp.add(2 * x - 1);
              }
          }
          for (int x : result) {
              if (2 * x <= n) {
                  temp.add(2 * x);
              }
```

```
                }
                result = temp;
            }

            // Convert List<Integer> to int[]
            int[] arr = new int[n];
            for (int i = 0; i < n; i++) {
                arr[i] = result.get(i);
            }

            return arr;
        }
    }
```


Accepted 38 / 38 testcases passed
_b... submitted at Mar 18, 2025 21:19

Accepted Runtime: 0 ms

Input

```
n =
4
```

Output

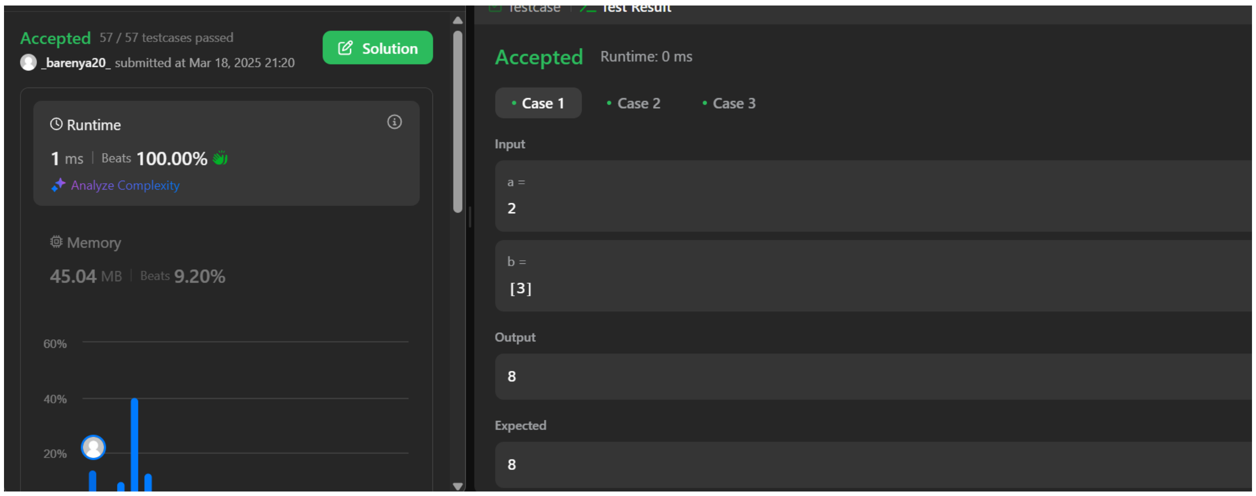
```
[1,3,2,4]
```

Expected

```
[2,1,4,3]
```

vi. Super Pow

```
class Solution {
    private static final int MOD = 1337;
    private int powerMod(int x, int y, int mod) {
        int result = 1;
        x = x % mod;
        while (y > 0) {
            if (y % 2 == 1) {
                result = (result * x) % mod;
            }
            y /= 2;
            x = (x * x) % mod;
        }
        return result;
    }
    public int superPow(int a, int[] b) {
        a %= MOD;
        int exponent = 0;
        for (int digit : b) {
            exponent = (exponent * 10 + digit) % 1140;
        }
        if (exponent == 0) {
            exponent = 1140;
        }
        return powerMod(a, exponent, MOD);
    }
}
```

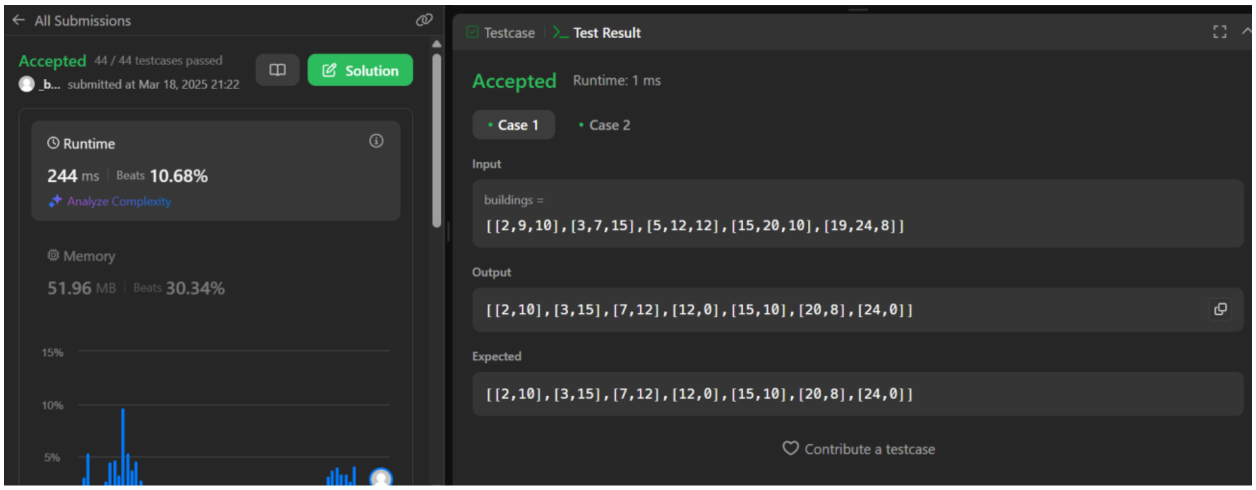


```
Accepted 57 / 57 testcases passed
_barenya20_ submitted at Mar 18, 2025 21:20
Runtime
1 ms | Beats 100.00%
Memory
45.04 MB | Beats 9.20%

Accepted Runtime: 0 ms
Case 1  Case 2  Case 3
Input
a =
2
b =
[3]
Output
8
Expected
8
```

vii. The SkyLine Problem

```
class Solution {
    public List<List<Integer>> getSkyline(int[][] buildings) {
        List<int[]> events = new ArrayList<>();
        // Create start and end events
        for (int[] building : buildings) {
            int start = building[0], end = building[1], height = building[2];
            events.add(new int[]{start, -height}); // Start of building
            events.add(new int[]{end, height});    // End of building
        }

        // Sort events
        events.sort((a, b) -> {
            if (a[0] != b[0]) {
                return Integer.compare(a[0], b[0]);
            } else {
                return Integer.compare(a[1], b[1]);
            }
        });

        List<List<Integer>> result = new ArrayList<>();
                    PriorityQueue<Integer>     maxHeap     =     new
PriorityQueue<>(Collections.reverseOrder());
        maxHeap.add(0); // Initial ground height
        int prevMaxHeight = 0;

        for (int[] event : events) {
            int x = event[0], h = event[1];

            if (h < 0) {
                maxHeap.add(-h); // Start of building, add height
            } else {
                maxHeap.remove(h); // End of building, remove height
            }
```

```
                int currMaxHeight = maxHeap.peek();
                if (currMaxHeight != prevMaxHeight) {
                    result.add(Arrays.asList(x, currMaxHeight));
                    prevMaxHeight = currMaxHeight;
                }
            }

            return result;
        }
    }
```



Accepted 44 / 44 testcases passed
_b... submitted at Mar 18, 2025 21:22

Runtime
244 ms | Beats 10.68%

Memory
51.96 MB | Beats 30.34%

Accepted Runtime: 1 ms

Input
buildings =
```
[[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]
```
Output
```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```
Expected
```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```