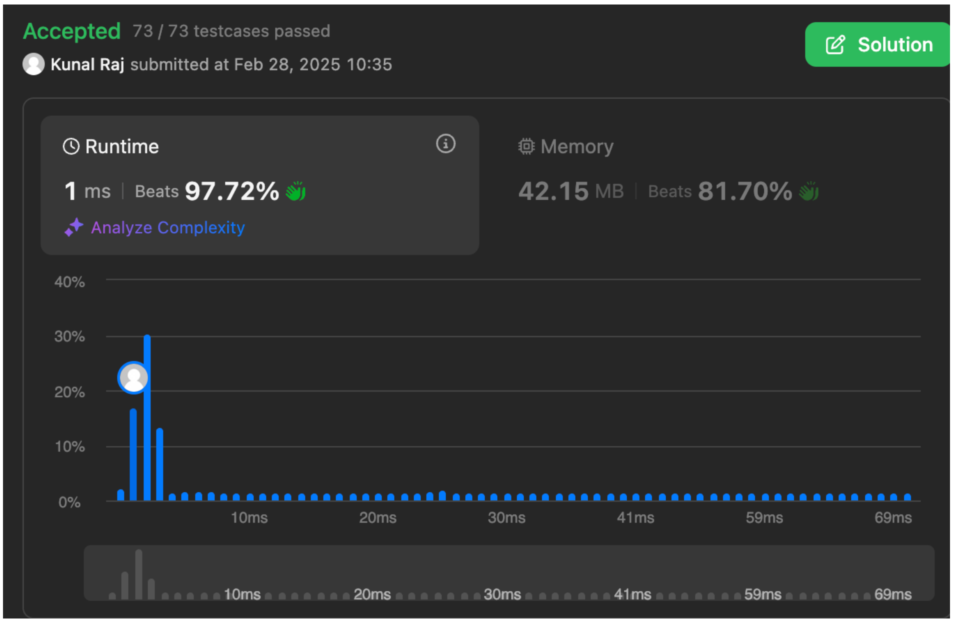
1. Longest Nice Substring:

```
 class Solution {
    public String longestNiceSubstring(String s) {
        if (s.length() < 2) return "";

        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
            if (s.indexOf(Character.toLowerCase(c)) == -1 ||
s.indexOf(Character.toUpperCase(c)) == -1) {
                String left = longestNiceSubstring(s.substring(0, i));
                String right = longestNiceSubstring(s.substring(i + 1));
                return left.length() >= right.length() ? left : right;
            }
        }

        return s;
    }
}
```



Accepted 73 / 73 testcases passed

Kunal Raj submitted at Feb 28, 2025 10:35

Runtime

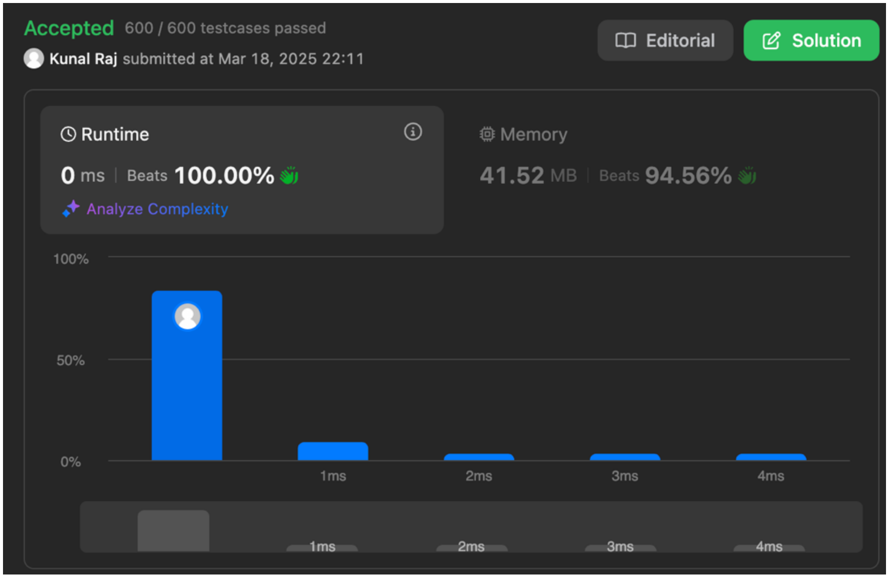
1 ms | Beats 97.72%

Memory

42.15 MB | Beats 81.70%

2. Reverse Bits:

```
public class Solution {
    // Treat n as an unsigned integer
    public int reverseBits(int n) {
        int result = 0;

        for (int i = 0; i < 32; i++) {
            result = (result << 1) | (n & 1);
            n >>= 1;
        }

        return result;
    }
}
```



Accepted 600 / 600 testcases passed

Kunal Raj submitted at Mar 18, 2025 22:11

Runtime

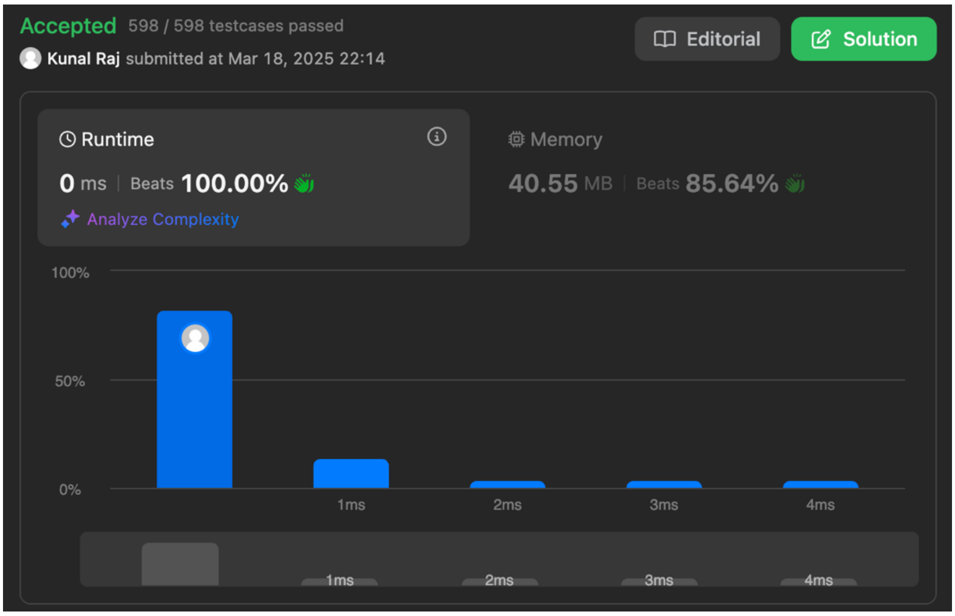
0 ms | Beats 100.00%

Memory

41.52 MB | Beats 94.56%

3. Number of 1 Bits:

```
public class Solution {
    public int hammingWeight(int n) {
        int count = 0;

        while (n != 0) {
            count += (n & 1);
            n >>>= 1;
        }

        return count;
    }
}
```



Accepted 598 / 598 testcases passed

Kunal Raj submitted at Mar 18, 2025 22:14

Runtime

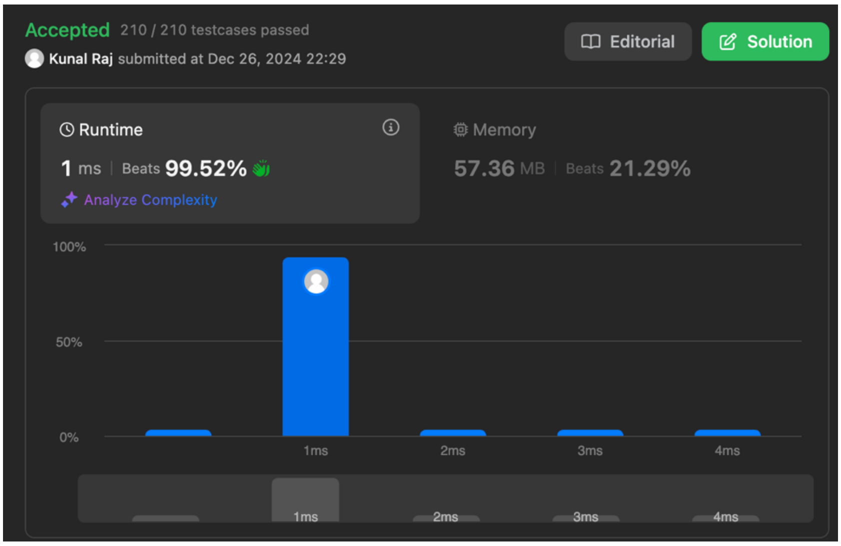
0 ms | Beats 100.00%

Memory

40.55 MB | Beats 85.64%

4. Maximum Subarray:

```
class Solution {
    public int maxSubArray(int[] nums) {
        int n = nums.length;
        int max = nums[0];
        int maxend = nums[0];

        for (int i = 1; i < n; i++) {
            maxend = Math.max(nums[i], maxend + nums[i]);
            max = Math.max(max, maxend);
        }
        return max;
    }
}
```



Accepted 210 / 210 testcases passed

Kunal Raj submitted at Dec 26, 2024 22:29

Runtime

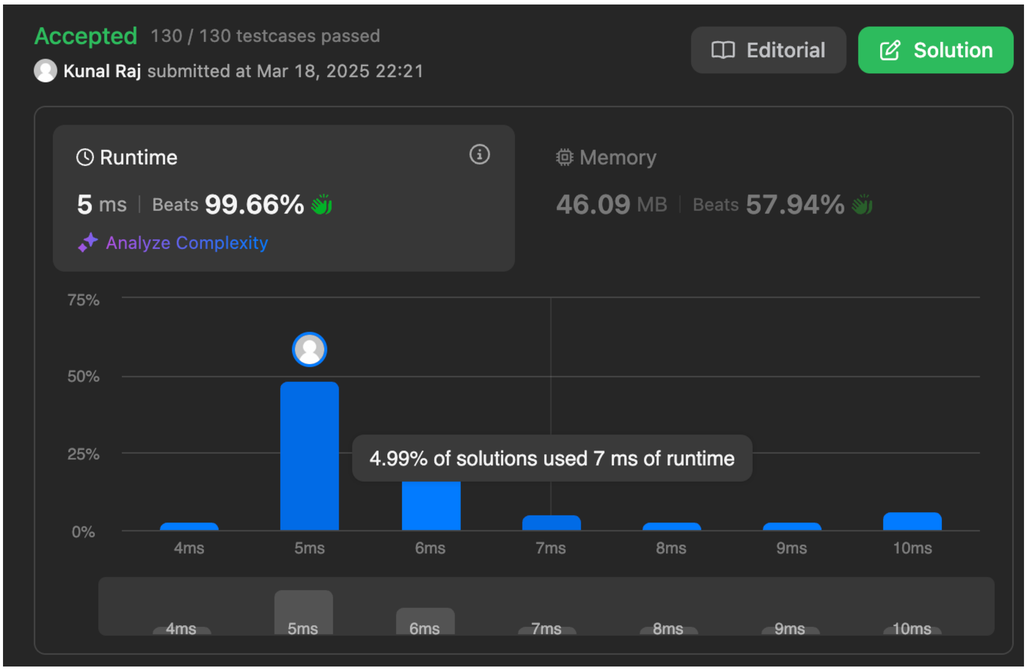
1 ms | Beats 99.52%

Memory

57.36 MB | Beats 21.29%

5. Search a 2D Matrix II:

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

        while (row < rows && col >= 0) {
            if (matrix[row][col] == target) return true;
            else if (matrix[row][col] > target) col--;
            else row++;
        }
        return false;
    }
}
```



Accepted 130 / 130 testcases passed

Kunal Raj submitted at Mar 18, 2025 22:21

Runtime

5 ms | Beats 99.66%

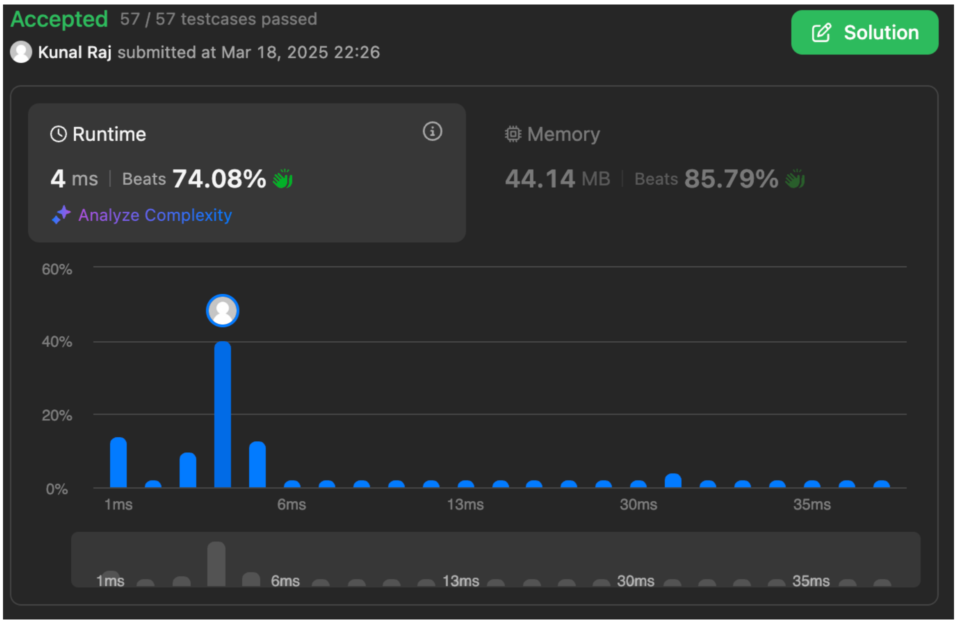
Memory

46.09 MB | Beats 57.94%

4.99% of solutions used 7 ms of runtime

6. Super Pow:

```
class Solution {
    private static final int MOD = 1337;

    private int modPow(int a, int b) {
        int result = 1;
        a %= MOD;

        while (b > 0) {
            if (b % 2 == 1) {
                result = (result * a) % MOD;
            }
            a = (a * a) % MOD;
            b /= 2;
        }
        return result;
    }

    public int superPow(int a, int[] b) {
        int result = 1;
        for (int digit : b) {
            result = (modPow(result, 10) * modPow(a, digit)) % MOD;
        }
        return result;
    }
}
```



Accepted 57 / 57 testcases passed

Kunal Raj submitted at Mar 18, 2025 22:26

Runtime

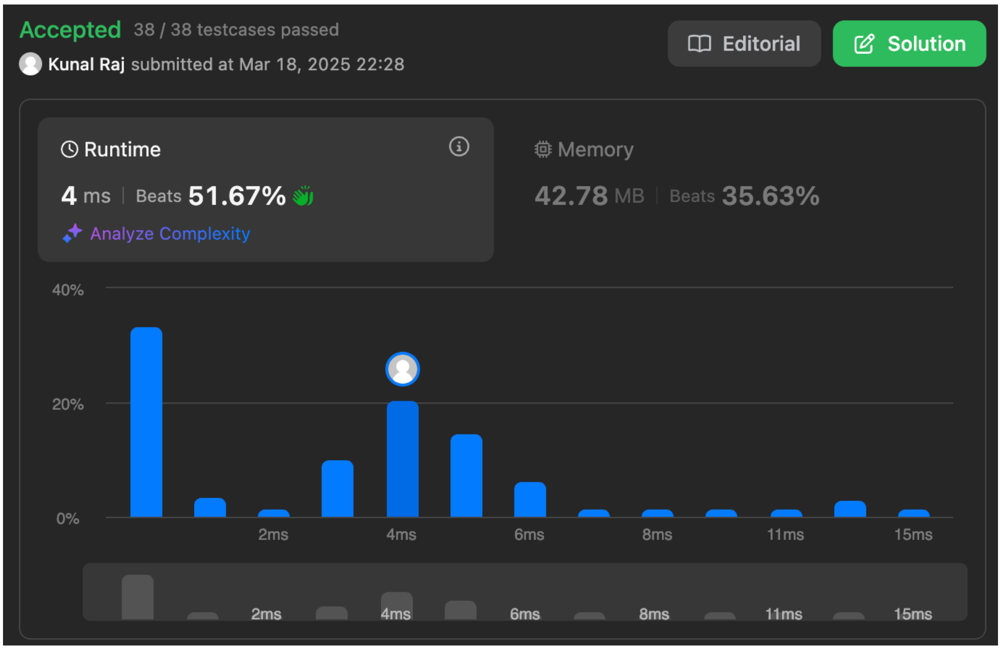
4 ms | Beats 74.08%

Memory

44.14 MB | Beats 85.79%

7. Beautiful Array:

```
import java.util.ArrayList;
import java.util.List;

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

        while (res.size() < n) {
            List<Integer> next = new ArrayList<>();

            for (int x : res) {
                if (2 * x - 1 <= n) {
                    next.add(2 * x - 1);
                }
            }

            for (int x : res) {
                if (2 * x <= n) {
                    next.add(2 * x);
                }
            }

            res = next;
        }

        // Convert list to array
        int[] result = new int[n];
        for (int i = 0; i < n; i++) {
            result[i] = res.get(i);
        }
        return result;
    }
}
```



Accepted 38 / 38 testcases passed

Kunal Raj submitted at Mar 18, 2025 22:28

Runtime

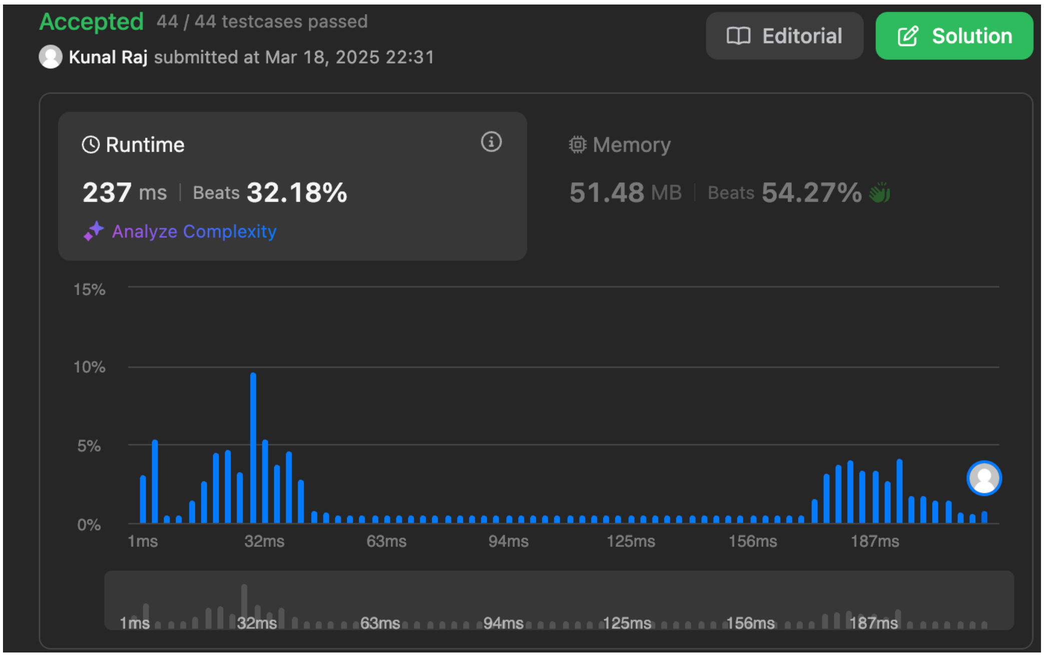
4 ms | Beats 51.67%

Memory

42.78 MB | Beats 35.63%

8. The Skyline Problem:

```
import java.util.*;

class Solution {
    public List<List<Integer>> getSkyline(int[][] buildings) {
        List<int[]> points = new ArrayList<>();
        for (int[] b : buildings) {
            points.add(new int[]{b[0], -b[2]});
            points.add(new int[]{b[1], b[2]});
        }

        Collections.sort(points, (a, b) -> {
            if (a[0] != b[0]) return a[0] - b[0];
            return a[1] - b[1];
        });

        PriorityQueue<Integer> maxHeap = new
PriorityQueue<>(Collections.reverseOrder());
        maxHeap.add(0);

        int prevHeight = 0;
        List<List<Integer>> skyline = new ArrayList<>();

        for (int[] p : points) {
            int x = p[0], height = p[1];

            if (height < 0) {
                maxHeap.add(-height);
            } else {
                maxHeap.remove(height);
            }

            int currHeight = maxHeap.peek();

            if (currHeight != prevHeight) {
                skyline.add(Arrays.asList(x, currHeight));
                prevHeight = currHeight;
            }
        }

        return skyline;
    }
}
```



Accepted 44 / 44 testcases passed

Kunal Raj submitted at Mar 18, 2025 22:31

Runtime

237 ms | Beats 32.18%

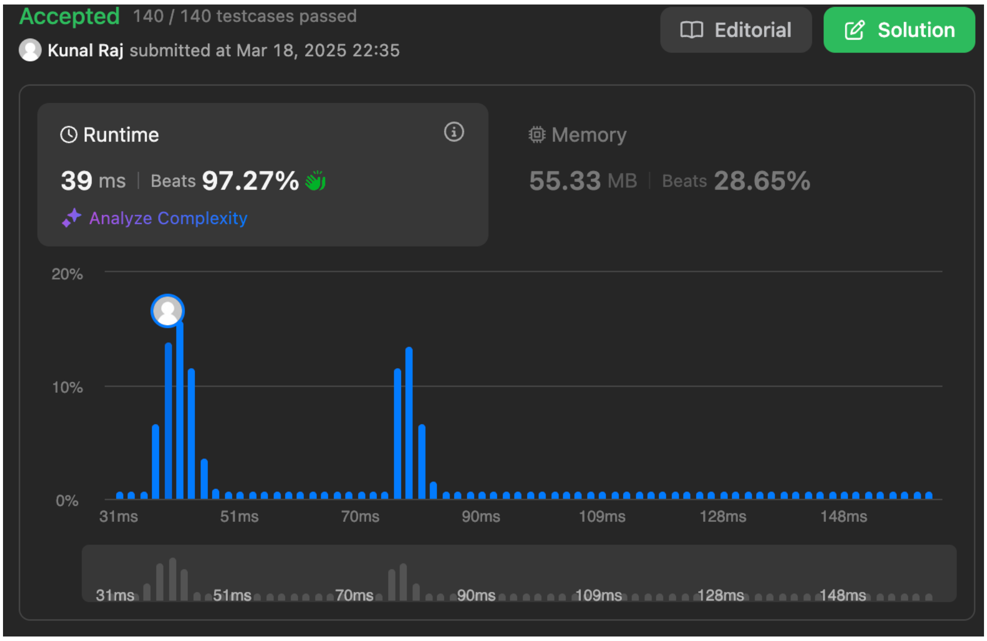
Analyze Complexity

Memory

51.48 MB | Beats 54.27%

9. Reverse Pairs:

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

    private int mergeSort(int[] nums, int left, int right) {
        if (left >= right) return 0;

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

        count += countPairs(nums, left, mid, right);
        merge(nums, left, mid, right);
        return count;
    }

    private int countPairs(int[] nums, int left, int mid, int right) {
        int count = 0, j = mid + 1;
        for (int i = left; i <= mid; i++) {
            while (j <= right && nums[i] > 2L * nums[j]) {
                j++;
            }
            count += (j - (mid + 1));
        }
        return count;
    }
    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++];
        System.arraycopy(temp, 0, nums, left, temp.length);
    }
}
```



Accepted 140 / 140 testcases passed

Kunal Raj submitted at Mar 18, 2025 22:35

Runtime

39 ms | Beats 97.27%

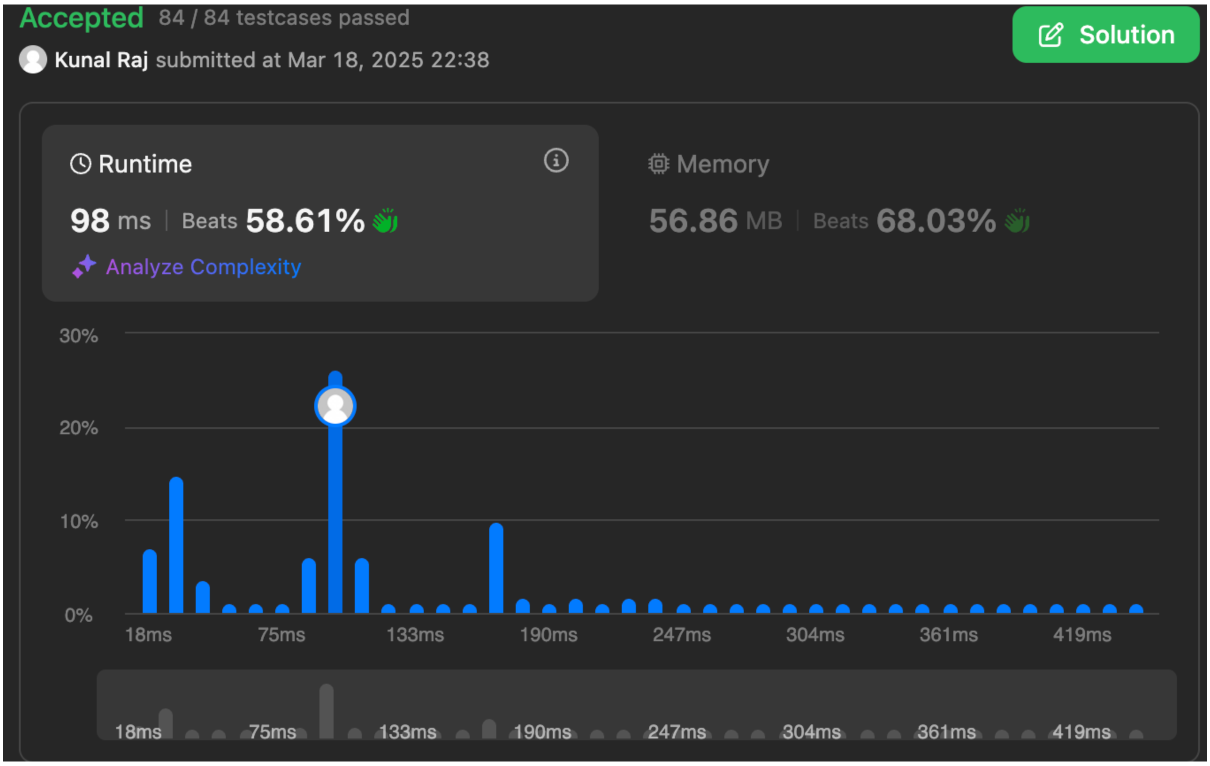
Analyze Complexity

Memory

55.33 MB | Beats 28.65%

10. Longest Increasing Subsequence II:

```
class Solution {
    public int lengthOfLIS(int[] nums, int k) {
        int maxVal = 0;
        for (int num : nums) {
            maxVal = Math.max(maxVal, num);
        }

        SegmentTree segTree = new SegmentTree(maxVal + 1);
        int maxLength = 0;
        for (int num : nums) {
            int bestPrevLIS = segTree.query(Math.max(0, num - k), num - 1);
            int newLIS = bestPrevLIS + 1;
            segTree.update(num, newLIS);
            maxLength = Math.max(maxLength, newLIS);
        }

        return maxLength;
    }
}

class SegmentTree {
    private int[] tree;
    private int size;

    public SegmentTree(int n) {
        size = n;
        tree = new int[4 * n];
    }

    public void update(int index, int value) {
        update(1, 0, size - 1, index, value);
    }

    private void update(int node, int start, int end, int index, int value) {
        if (start == end) {
            tree[node] = Math.max(tree[node], value);
            return;
        }
        int mid = (start + end) / 2;
        if (index <= mid) {
            update(2 * node, start, mid, index, value);
        } else {
            update(2 * node + 1, mid + 1, end, index, value);
        }
        tree[node] = Math.max(tree[2 * node], tree[2 * node + 1]);
    }

    public int query(int left, int right) {
        if (left > right) return 0;
        return query(1, 0, size - 1, left, right);
    }

    private int query(int node, int start, int end, int left, int right) {
        if (start > right || end < left) return 0;
        if (start >= left && end <= right) return tree[node];
        int mid = (start + end) / 2;
        int leftMax = query(2 * node, start, mid, left, right);
        int rightMax = query(2 * node + 1, mid + 1, end, left, right);
        return Math.max(leftMax, rightMax);
    }
}
```



Accepted 84 / 84 testcases passed

Kunal Raj submitted at Mar 18, 2025 22:38

Runtime

98 ms | Beats 58.61%

Memory

56.86 MB | Beats 68.03%