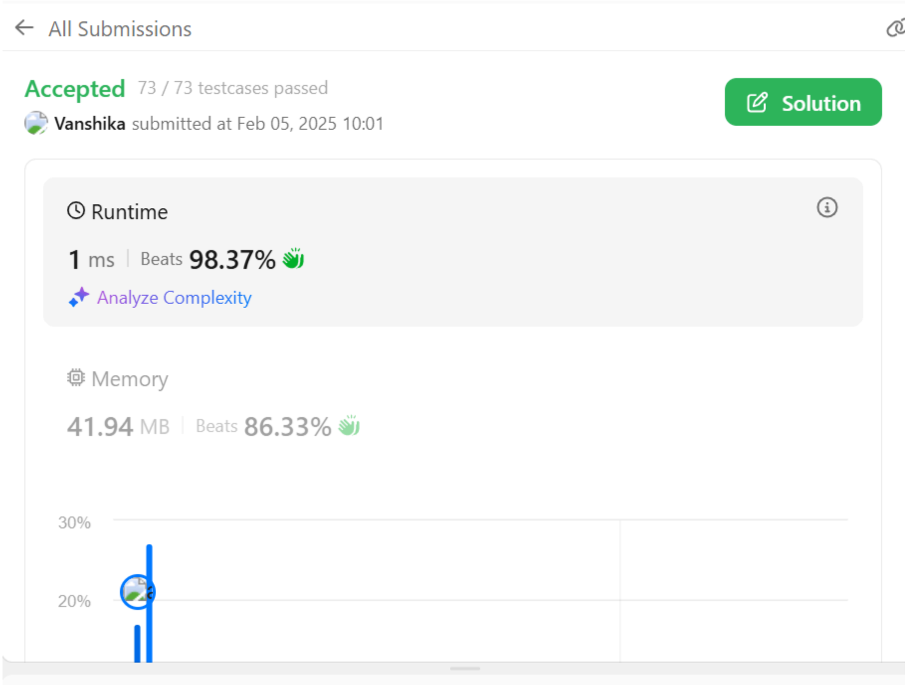
Name: Vanshika **UID: 22BCS15478**

Assignment-4

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.contains(Character.toString(Character.toUpperCase(c))) &&
                s.contains(Character.toString(Character.toLowerCase(c)))) {
                continue;
            }
            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

Vanshika submitted at Feb 05, 2025 10:01

Runtime

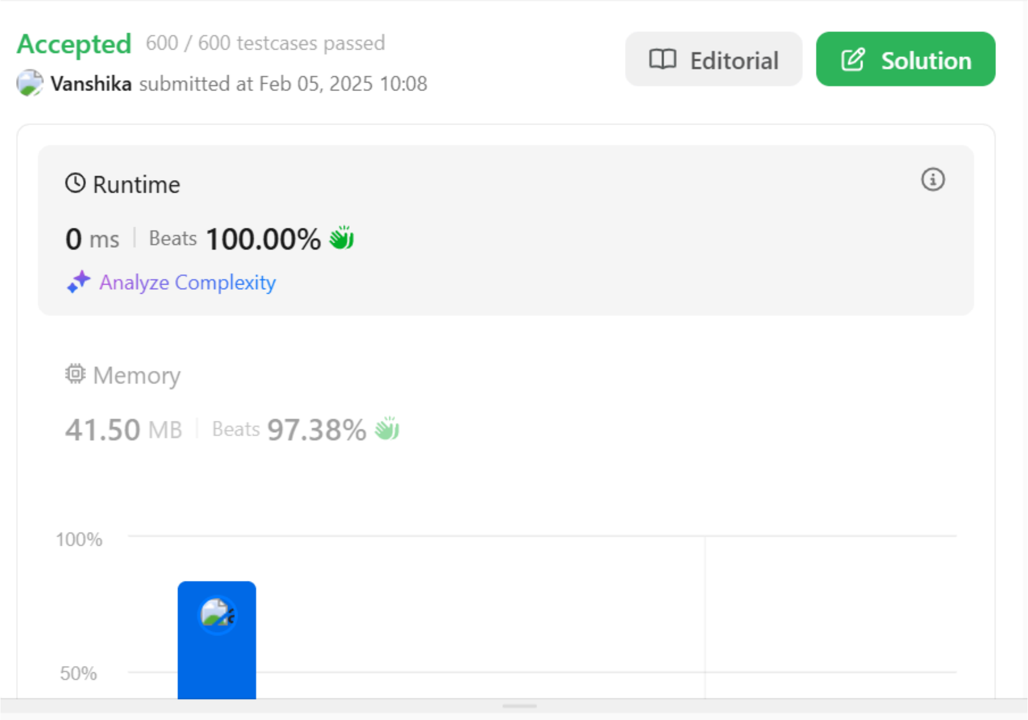
1 ms | Beats 98.37%

Memory

41.94 MB | Beats 86.33%

2. Reverse Bits

```
public class Solution {
    public int reverseBits(int n) {
        int result = 0;
        for (int i = 0; i < 32; i++) {
            result <<= 1;
            result |= (n & 1);
            n >>= 1;
        }
        return result;
    }
}
```



Accepted 600 / 600 testcases passed

Vanshika submitted at Feb 05, 2025 10:08

Runtime

0 ms | Beats 100.00%

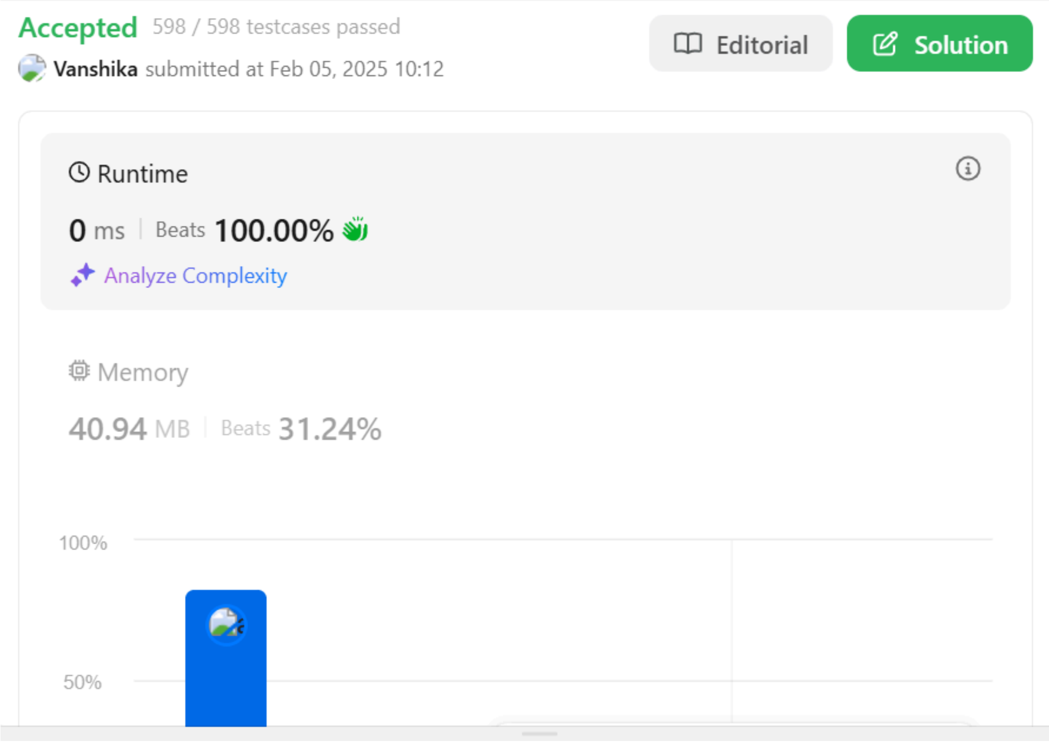
Analyze Complexity

Memory

41.50 MB | Beats 97.38%

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

Vanshika submitted at Feb 05, 2025 10:12

Runtime

0 ms | Beats 100.00%

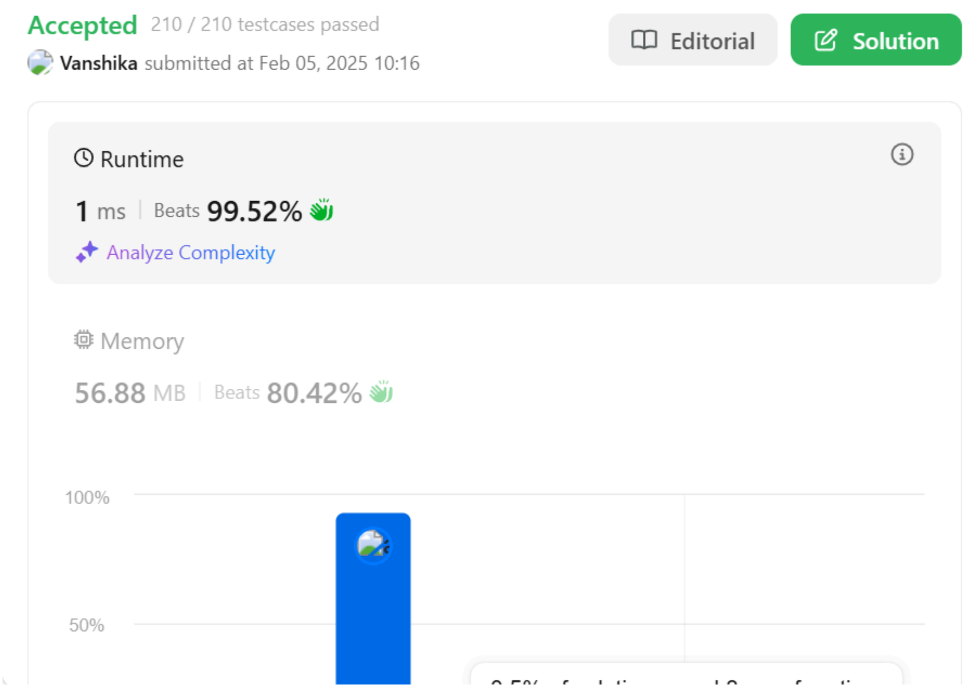
Analyze Complexity

Memory

40.94 MB | Beats 31.24%

4. **Maximum Subarray**

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



Accepted 210 / 210 testcases passed

Vanshika submitted at Feb 05, 2025 10:16

Runtime

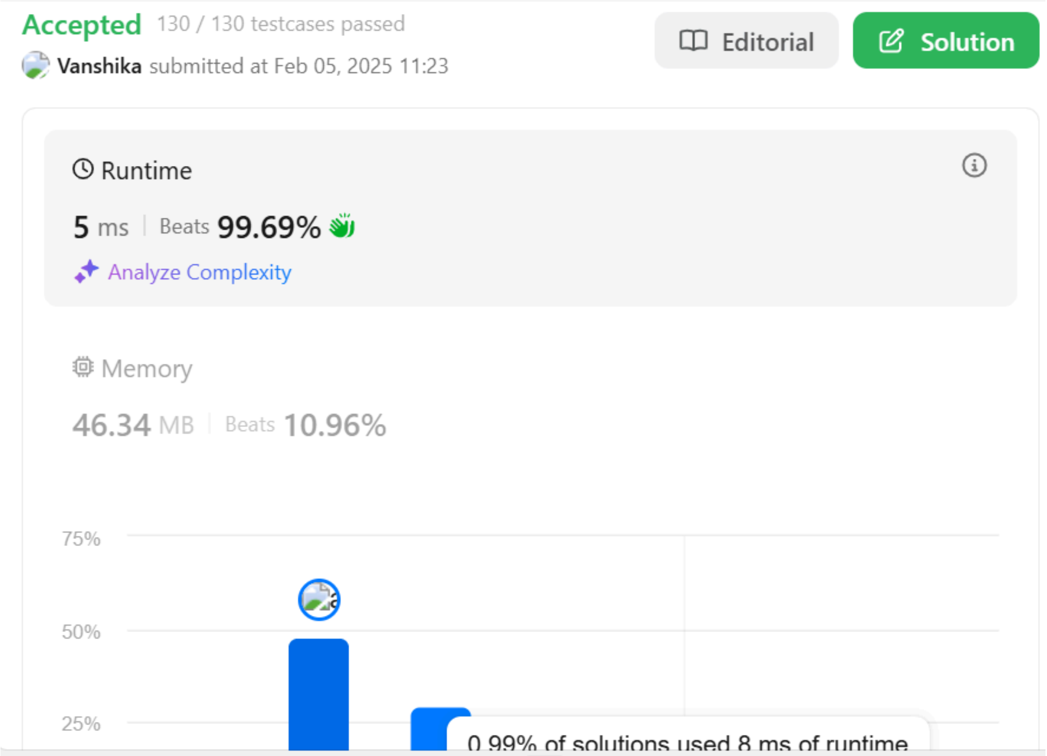
1 ms | Beats 99.52%

Memory

56.88 MB | Beats 80.42%

5. Search a 2D Matrix II

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int row = 0, col = matrix[0].length - 1;
        while (row < matrix.length && 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
Vanshika submitted at Feb 05, 2025 11:23

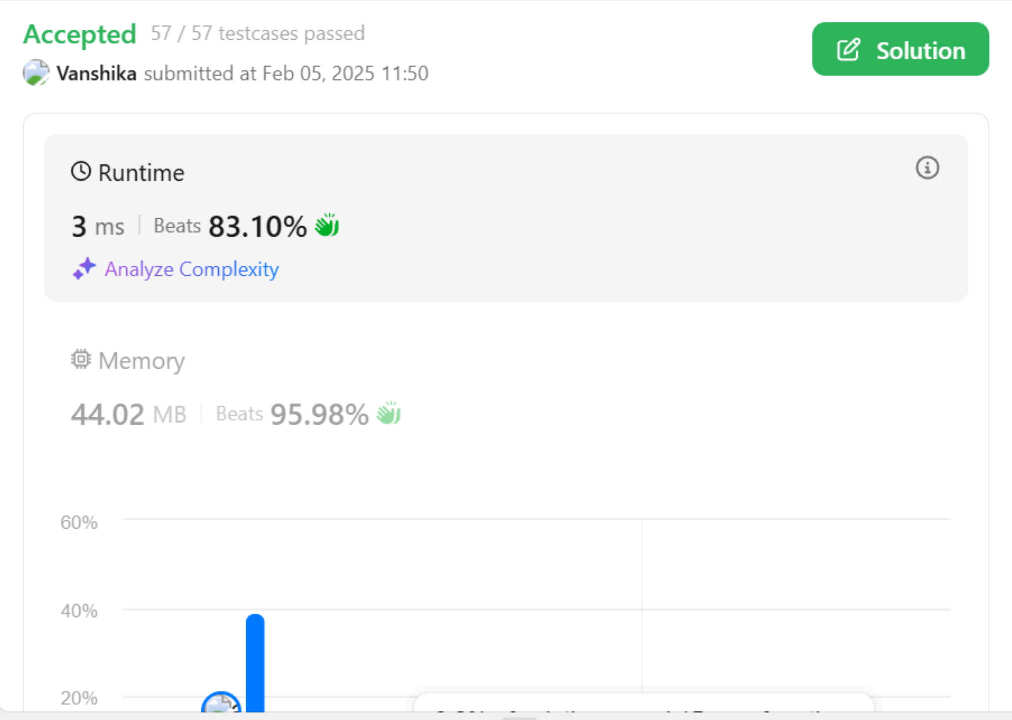
Runtime
5 ms | Beats 99.69%
Analyze Complexity

Memory
46.34 MB | Beats 10.96%

0.99% of solutions used 8 ms of runtime

6. **Super Pow**

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

    public int superPow(int a, int[] b) {
        a %= MOD;
        return helper(a, b, b.length);
    }

    private int helper(int a, int[] b, int length) {
        if (length == 0) return 1;

        int lastDigit = b[length - 1];
        int remainingPow = helper(a, b, length - 1);

        return powerMod(remainingPow, 10) * powerMod(a, lastDigit) % MOD;
    }

    private int powerMod(int base, int exp) {
        int result = 1;
        while (exp > 0) {
            if (exp % 2 == 1) {
                result = result * base % MOD;
            }
            base = base * base % MOD;
            exp /= 2;
        }
        return result;
    }
}
```



Accepted 57 / 57 testcases passed

Vanshika submitted at Feb 05, 2025 11:50

Runtime

3 ms | Beats 83.10%

Memory

44.02 MB | Beats 95.98%