**AP Assignment 4**

**Q Longest Nice Substring**

Code:

class Solution {

public String longestNiceSubstring(String s) {

int[] res = solve(s, 0, s.length());

return s.substring(res[0], res[1]);

}

public int[] solve(String s, int start, int end) {

for (int i = start; i < end; i++) {

char ch = s.charAt(i);

if (Character.isLowerCase(ch) && s.substring(start, end).contains(Character.toUpperCase(ch) + ""))

continue;

else if (Character.isUpperCase(ch) && s.substring(start, end).contains(Character.toLowerCase(ch) + ""))

continue;

int[] first = solve(s, start, i);

int[] second = solve(s, i + 1, end);

if (first[1] - first[0] >= second[1] - second[0])

return first;

else

return second;

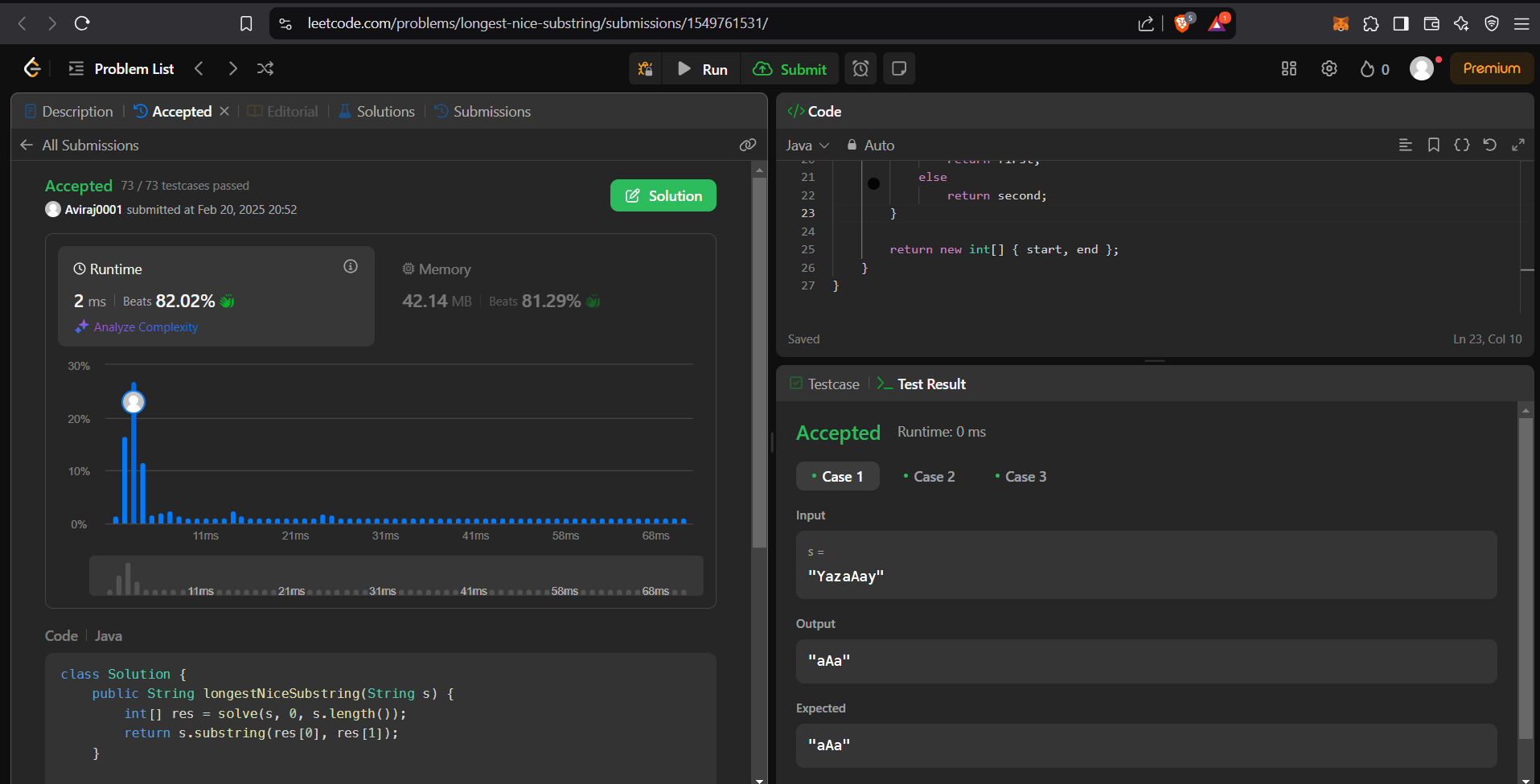
}

return new int[] { start, end };

}

}

Output:



**Q Reverse Bits**

Code:

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;

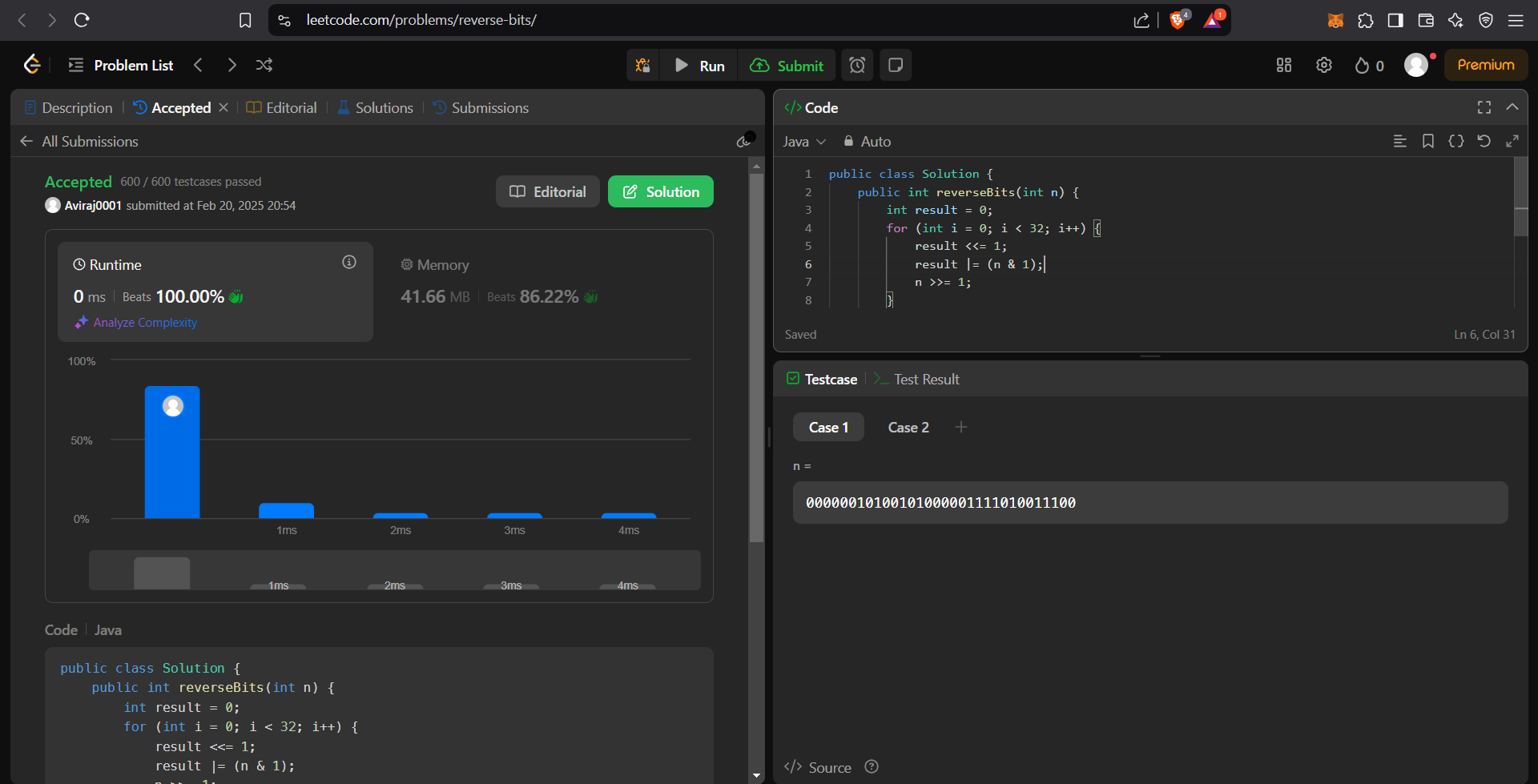
}

return result;

}

}

Output:



**Q. Number of 1 bits.**

**Code:**

class Solution {

public int hammingWeight(int n) {

int count = 0;

while (n != 0) {

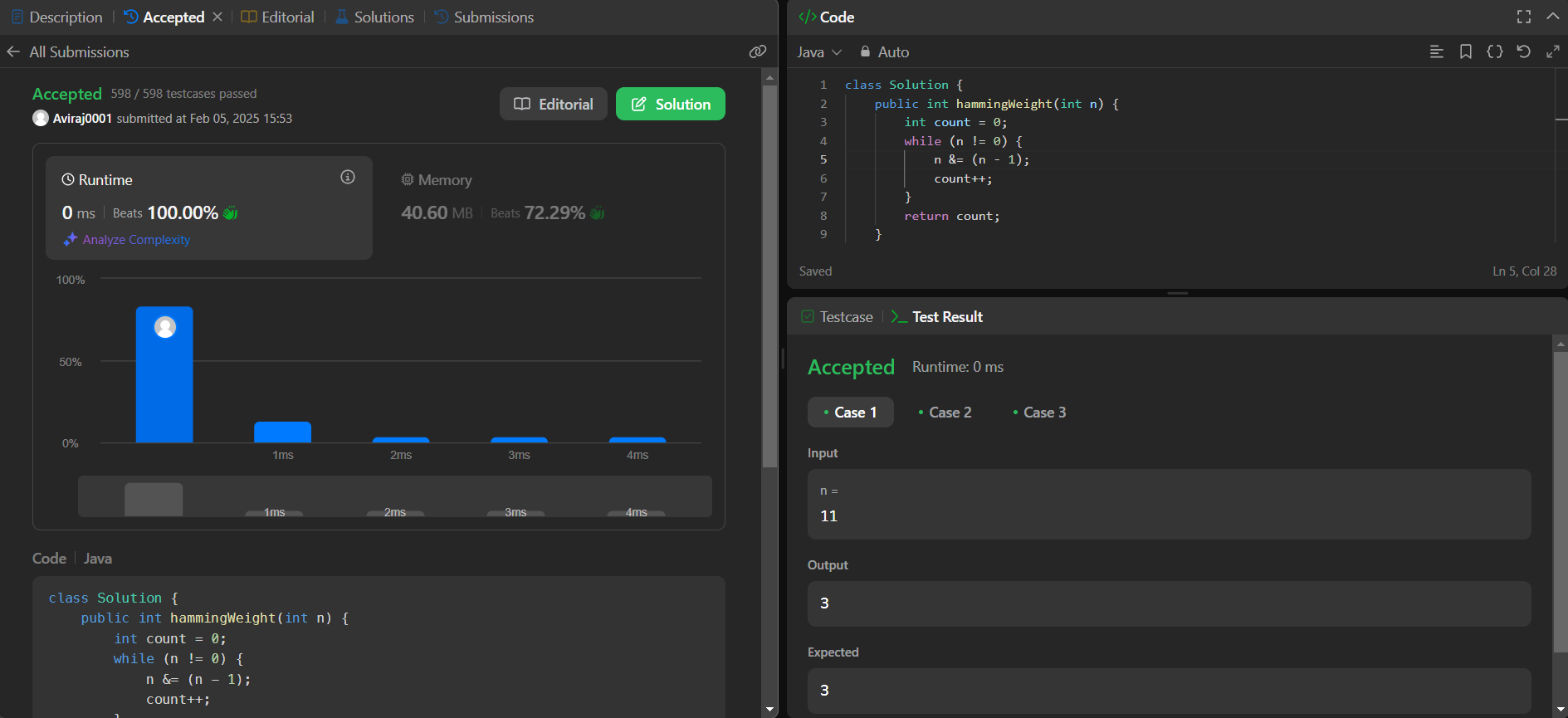
n &= (n - 1);

count++; }

return count;

}}

**Output:**

****

**Q. 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 row = 0, col = n - 1;

while (row < m && col >= 0) {

if (matrix[row][col] == target) {

return true;

} else if (matrix[row][col] > target) {

col--;

} else {

row++;

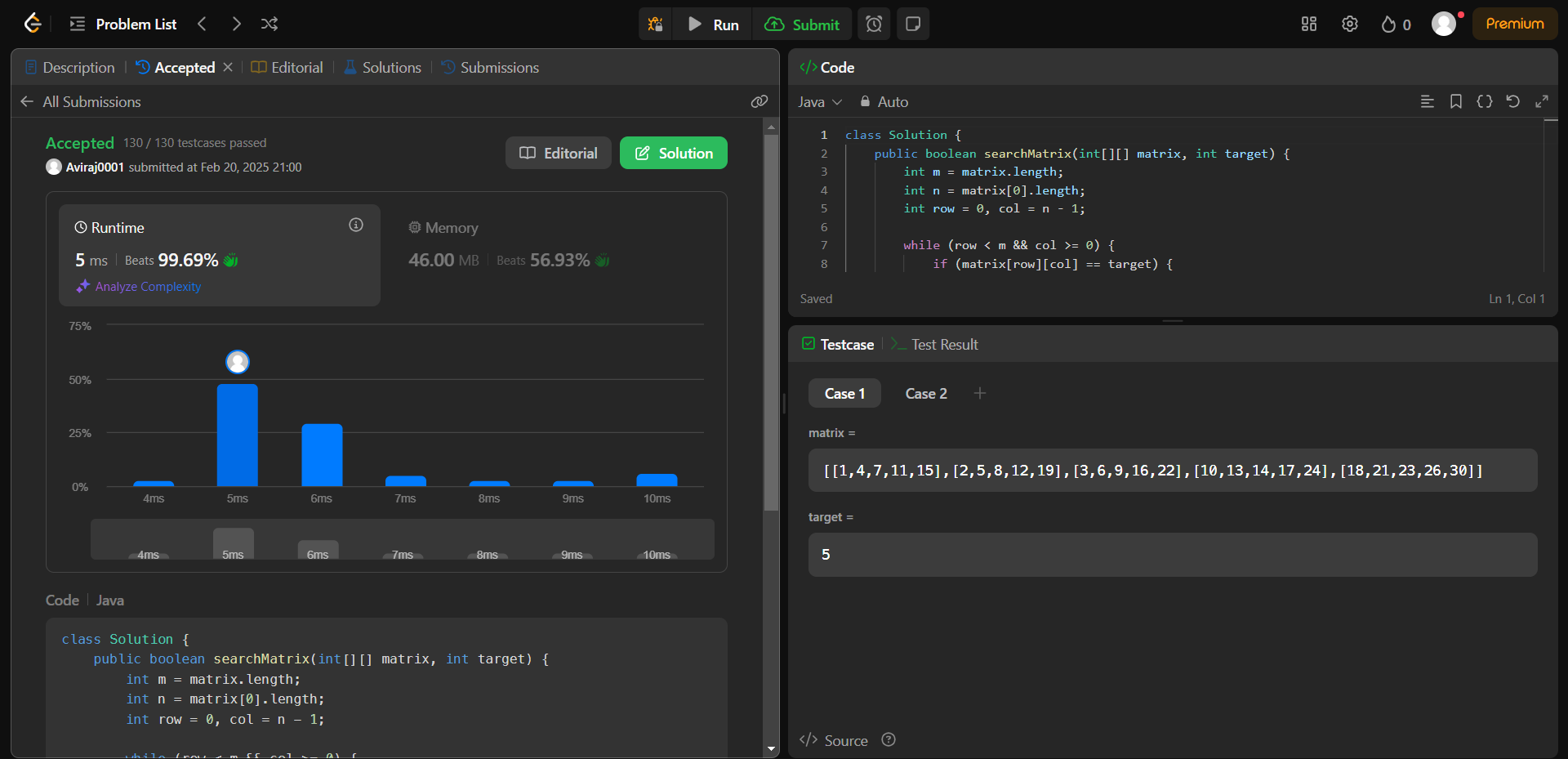
}

}

return false;

}

}**Output:**

****

**Q. Super Pow.**

**Code:**

class Solution {

private static final int MOD = 1337;

private int pow(int a, int b) {

int result = 1;

a %= MOD; // Taking mod to prevent overflow

for (int i = 0; i < b; i++) {

result = (result \* a) % MOD;

}

return result;

}

public int superPow(int a, int[] b) {

int result = 1;

for (int i = b.length - 1; i >= 0; i--) {

result = (result \* pow(a, b[i])) % MOD;

a = pow(a, 10); // Power up for the next iteration

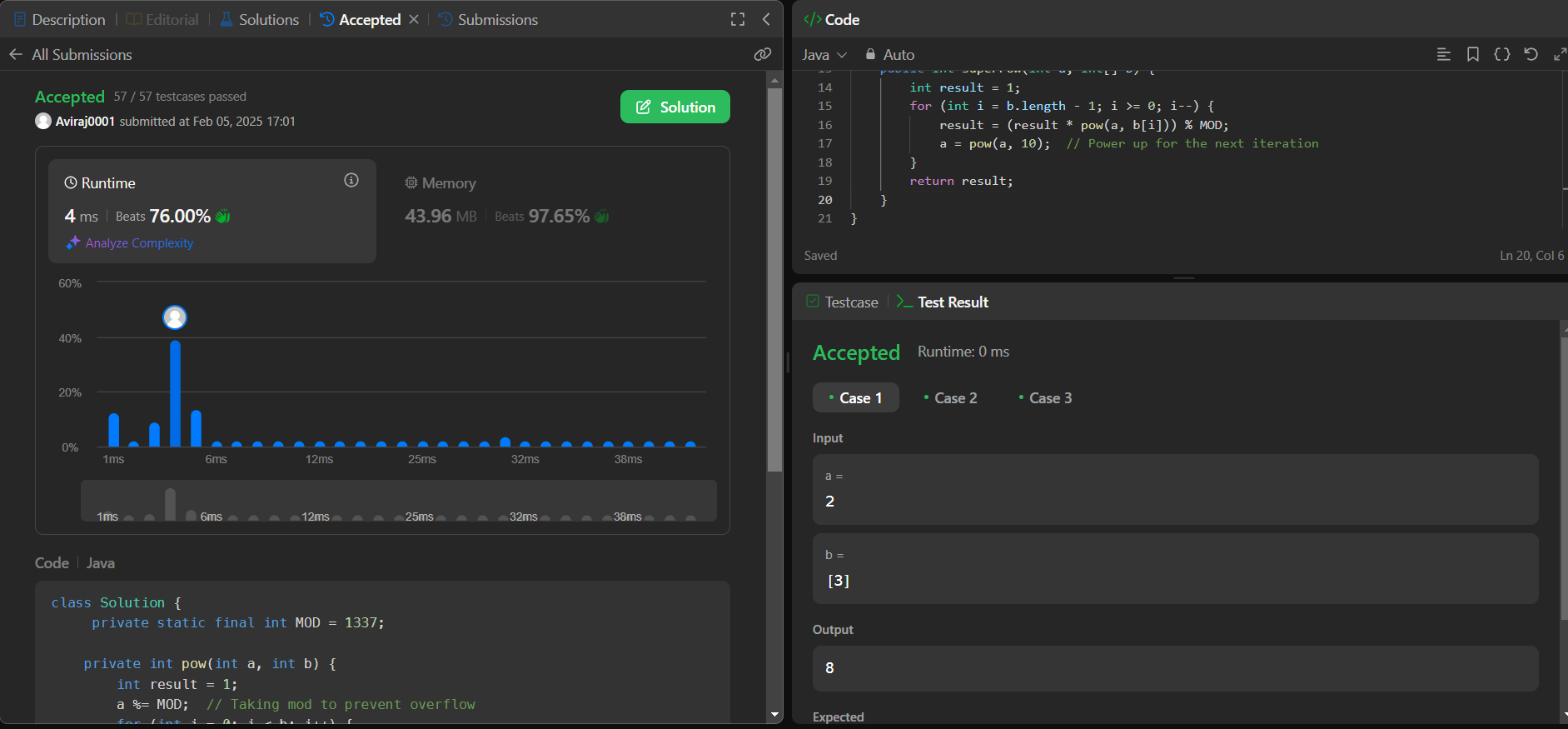
}

return result;

}

}

**Output:**

****

**Q Maximum Subarray**

**Code:**

class Solution {

public int maxSubArray(int[] nums) {

int maxSum = nums[0];

int 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;

}

}

**Output:**

