Question: 1 Longest Nice Substring (1763)

Code:

class Solution {

public String longestNiceSubstring(String s) {

var result = "";

for (int i=0; i<s.length(); i++) {

for (int j=i+1; j<s.length(); j++) {

var current = s.substring(i, j+1);

if (isNice(current)) {

result = current.length() > result.length() ? current : result;

}

}

}

return result;

}

public boolean isNice(String s) {

// A=0, Z=25, a=32, z=57, index = char minus 'A'

int[] letters = new int[58];

for (int i=0; i<s.length(); i++) {

letters[s.charAt(i) - 'A']++;

}

for (int i=0; i<26; i++) {

if ((letters[i] == 0 && letters[i+32] != 0) || (letters[i] != 0 && letters[i+32] == 0)) {

return false;

}

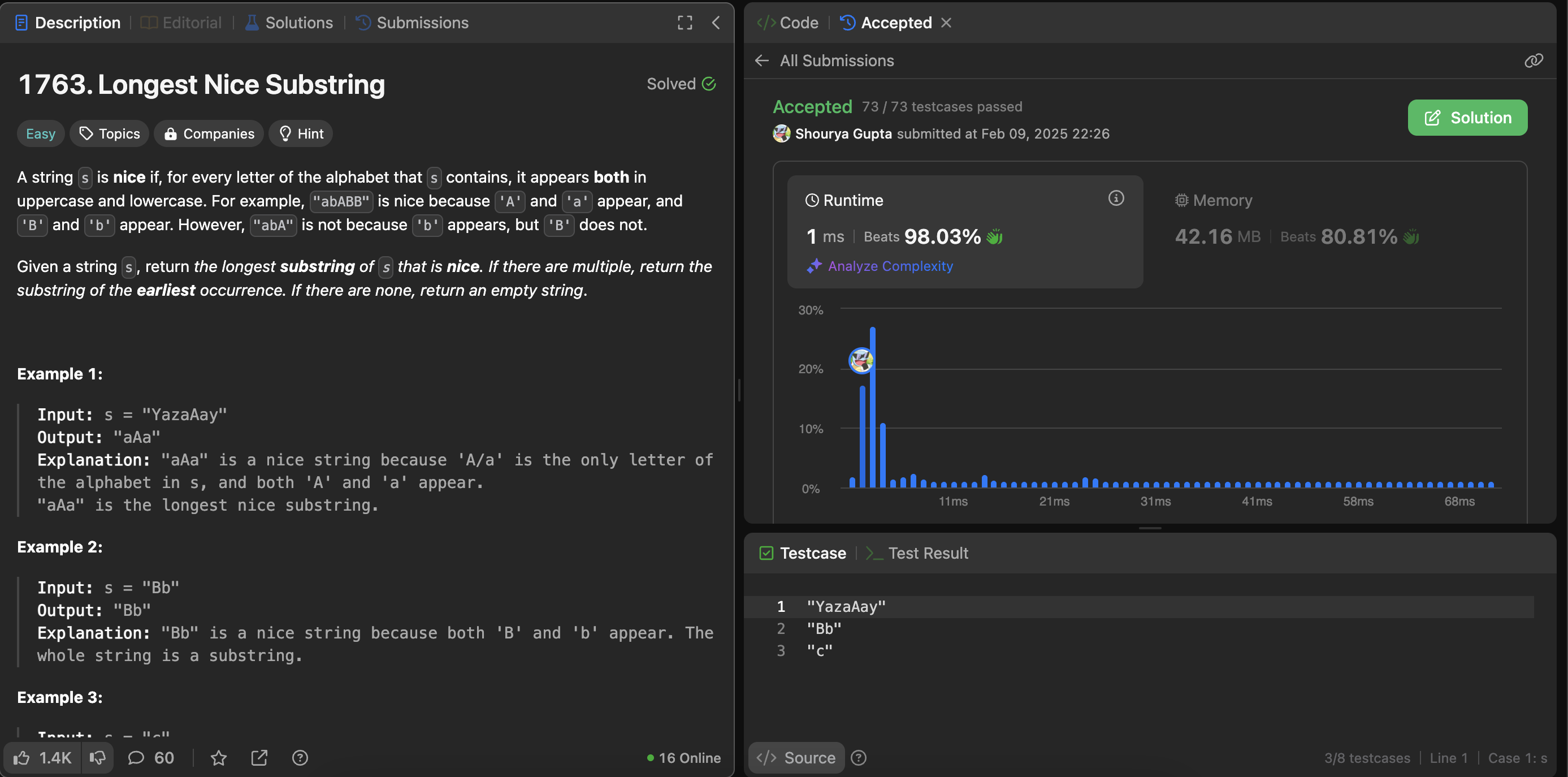
}

return true;

}

}

Output:



Question 2: Reverse Bits (190)

Code:

public class Solution {

public int reverseBits(int n) {

int reversed = 0;

for (int i = 0; i < 32; i++) {

reversed = (reversed << 1) | (n & 1);

n >>>= 1;

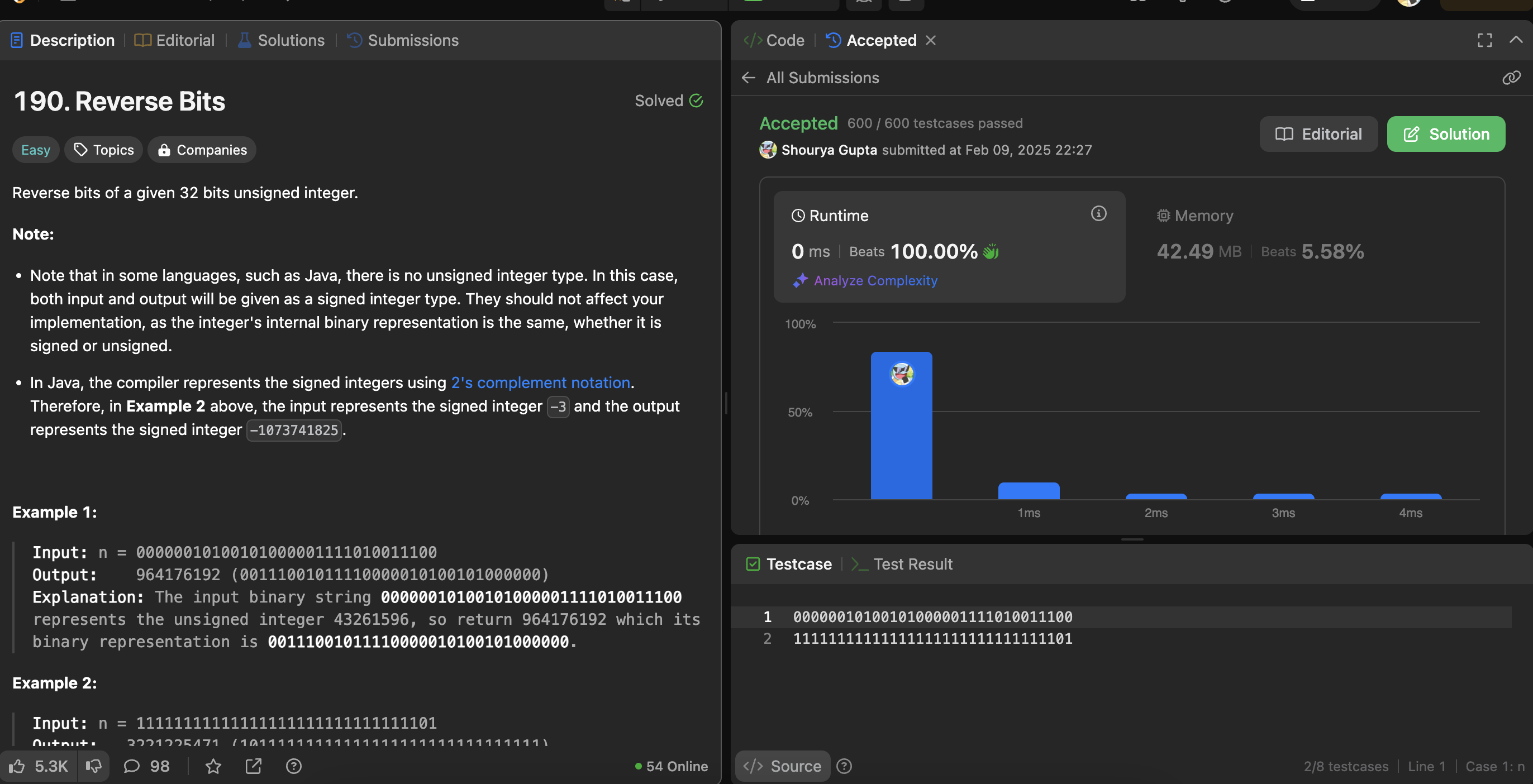
}

return reversed;

}

}

Output:



Question 3: Number of 1 bits (104)

Code:

public class Solution {

public int hammingWeight(int n) {

int count = 0;

while (n != 0) {

count += (n & 1);

n >>>= 1;

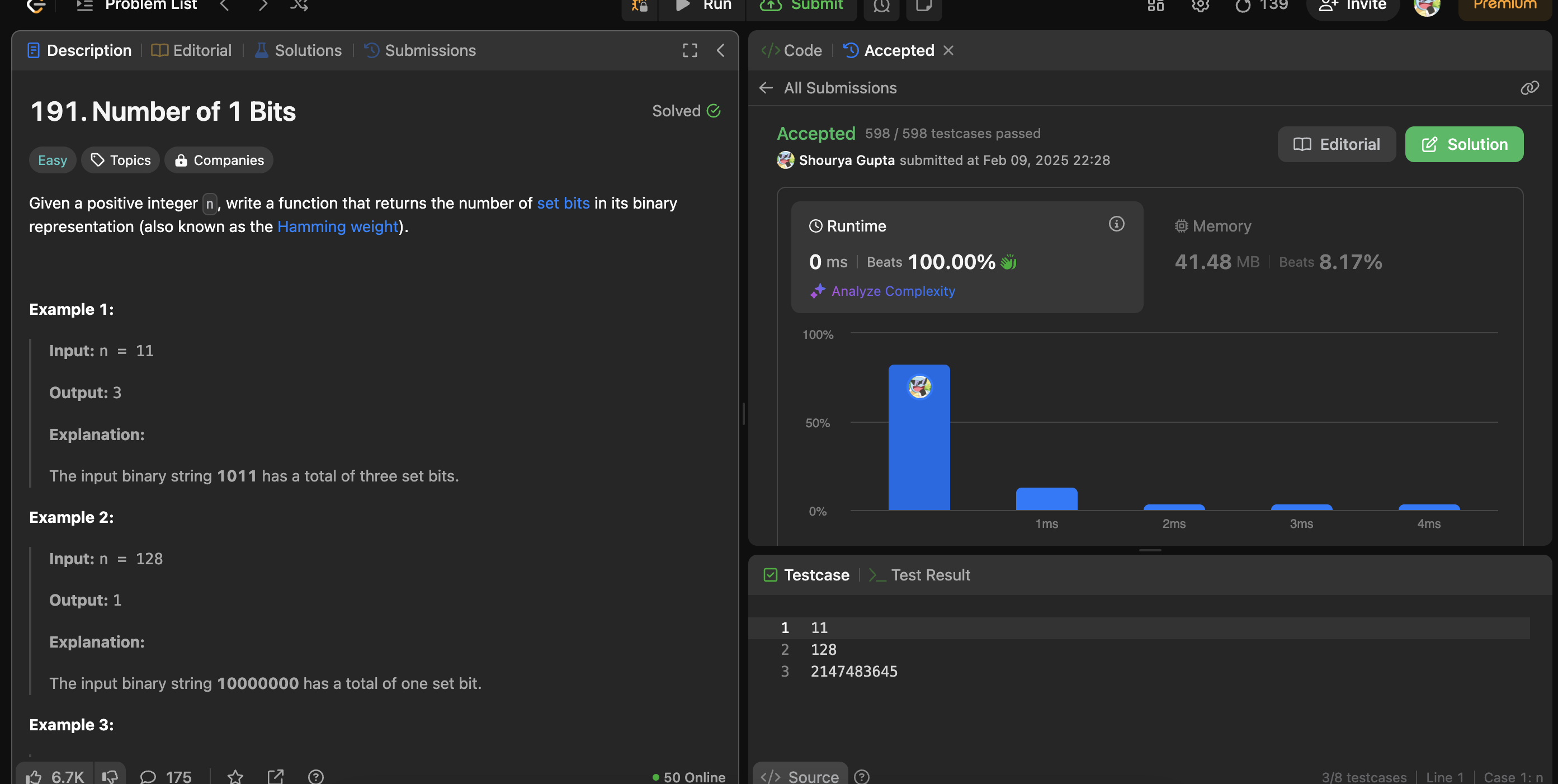
}

return count;

}

}

OUTPUT:



Question 4: Maximum Subarray (53)

Code:

public class Solution {

public int maxSubArray(int[] nums) {

int maxSum = nums[0], currentSum = 0;

for (int num : nums) {

currentSum = Math.max(num, currentSum + num);

maxSum = Math.max(maxSum, currentSum);

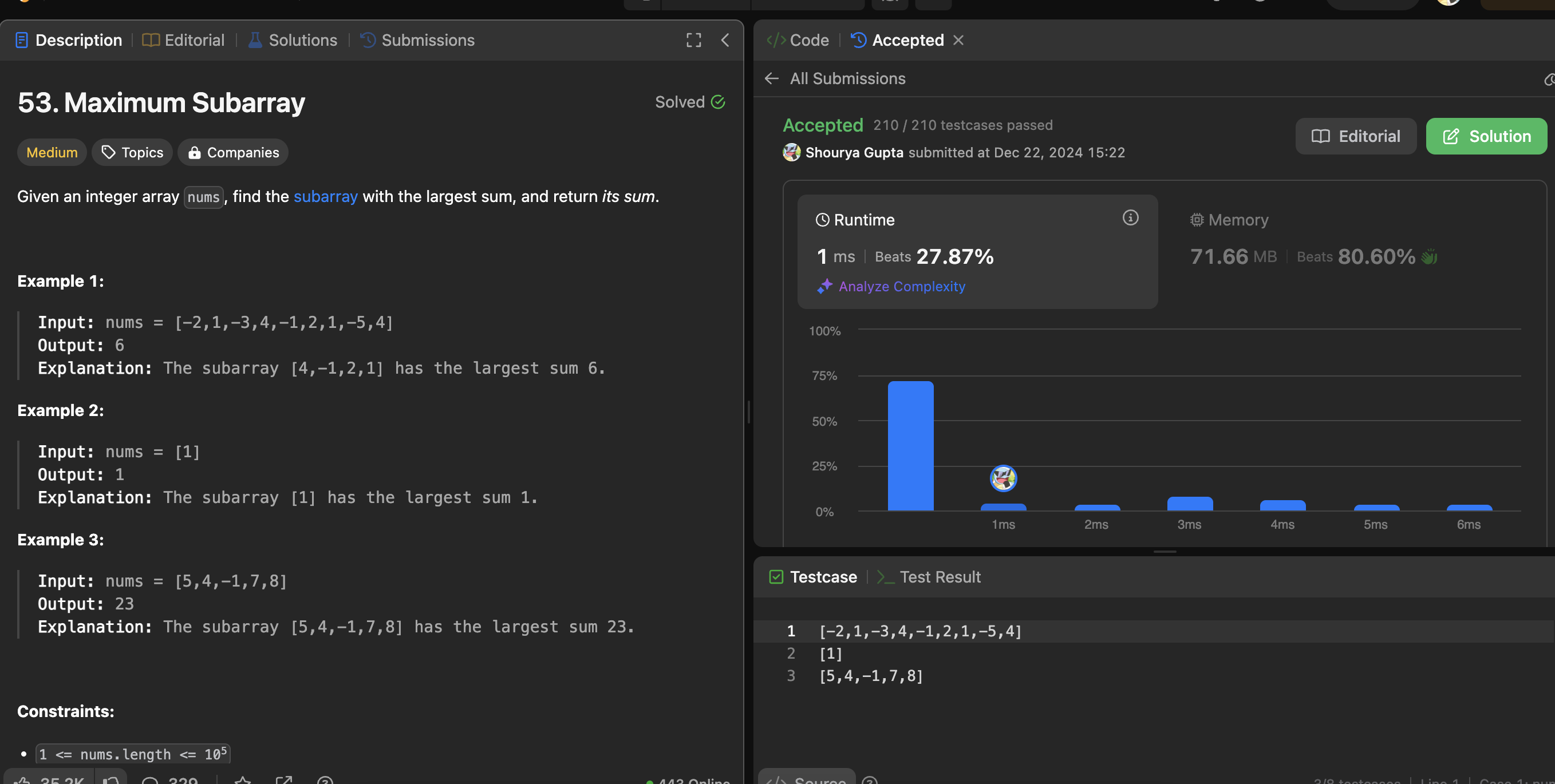
}

return maxSum;

}

}

Output:



Question 5: Search a 2D matrix II (240)

Code:

public 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) {

row++;

} else {

col--; }

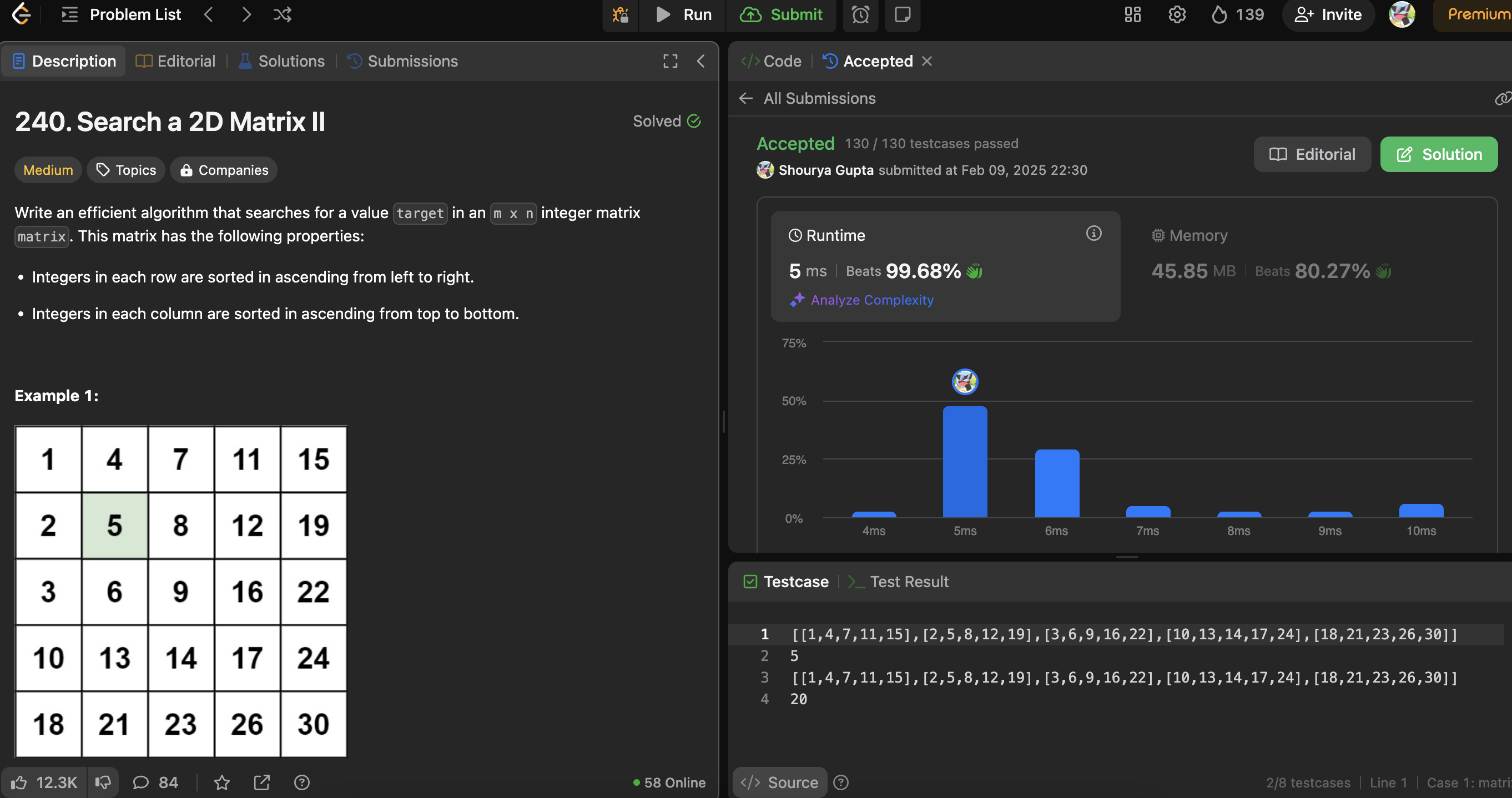
}

return false;

}

}

Output:



Question 6: Super Pow (372)  
Code:

public class Solution {

private static final int MOD = 1337;

private int pow(int a, int b) {

int res = 1;

a %= MOD;

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

res = (res \* a) % MOD;

}

return res;

}

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

int res = 1;

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

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

a = pow(a, 10);

}

return res;

}

}

Output:

