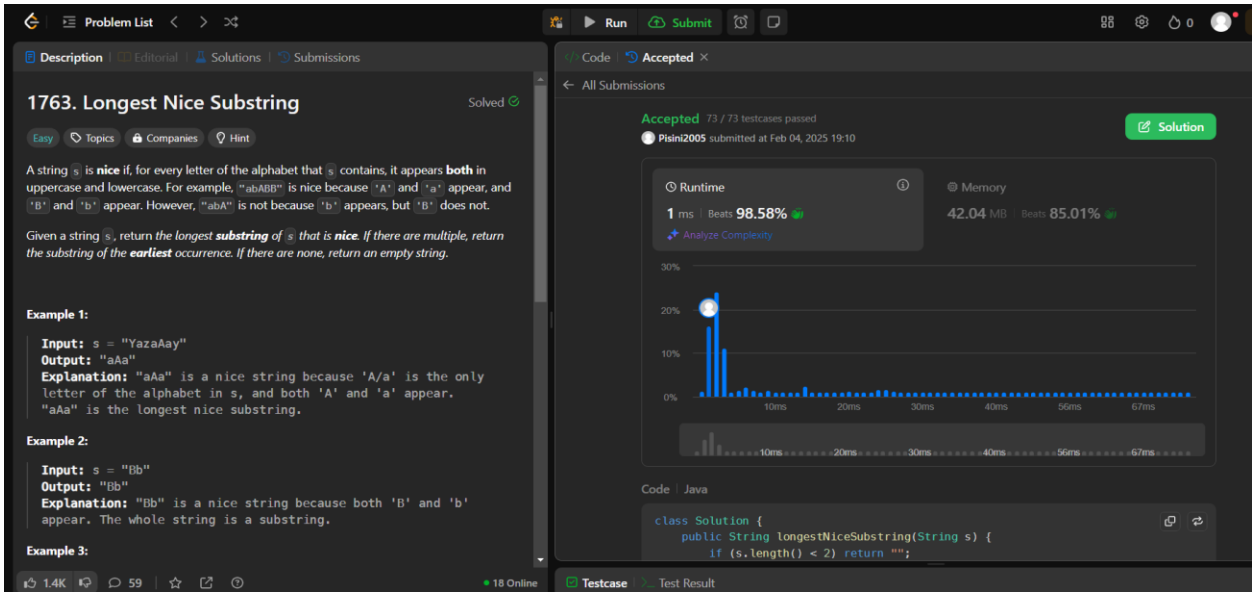
ASSIGNMENT -4 (ADVANCED PROGRAMMING)

Tapan Kumar– 22BCS10806

1. **Problem 1:** Longest Nice Substring
2. **Implementation/Code:**

```
class Solution {
  public String longestNiceSubstring(String s) {
    if (s.length() < 2) return "";
    for (int i = 0; i < s.length(); i++) {
      char ch = s.charAt(i);
      if (s.contains(Character.toString(Character.toLowerCase(ch))) &&
        s.contains(Character.toString(Character.toUpperCase(ch)))) {
        continue;
      }
      String left = longestNiceSubstring(s.substring(0, i));
      String right = longestNiceSubstring(s.substring(i + 1));
      return left.length() >= right.length() ? left : right;
    }
    return s; }}
```

3. **Output:**

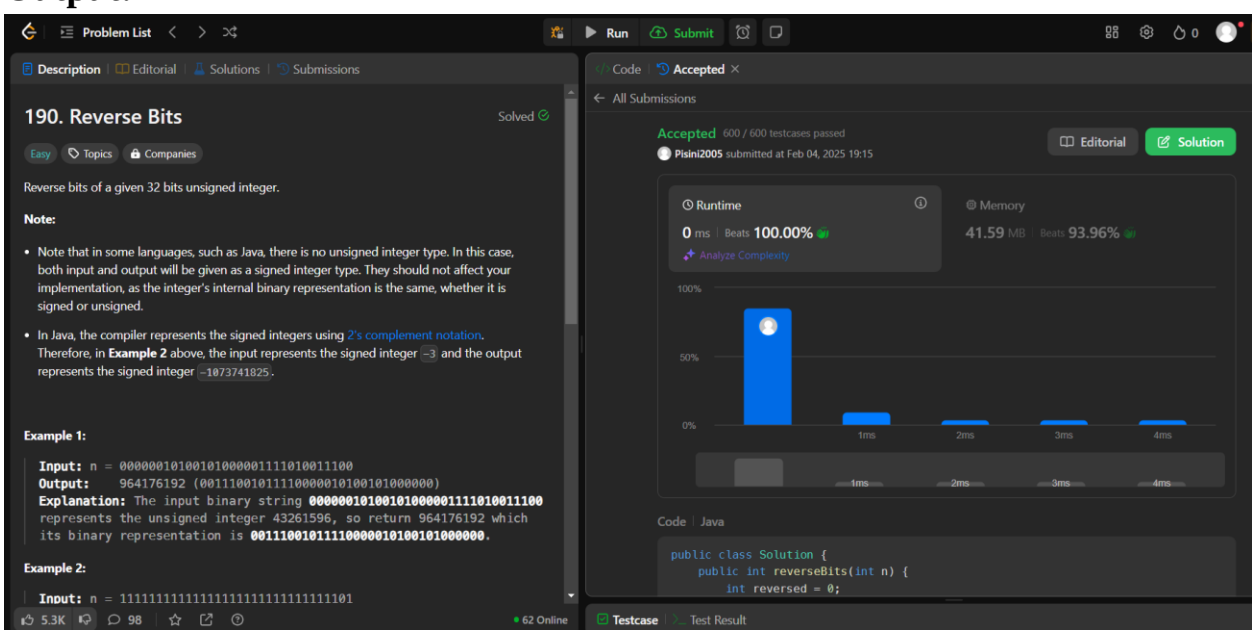


1. **Problem 2:** Reverse Bits

2. **Implementation/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;
    }
}
```

3. **Output:**

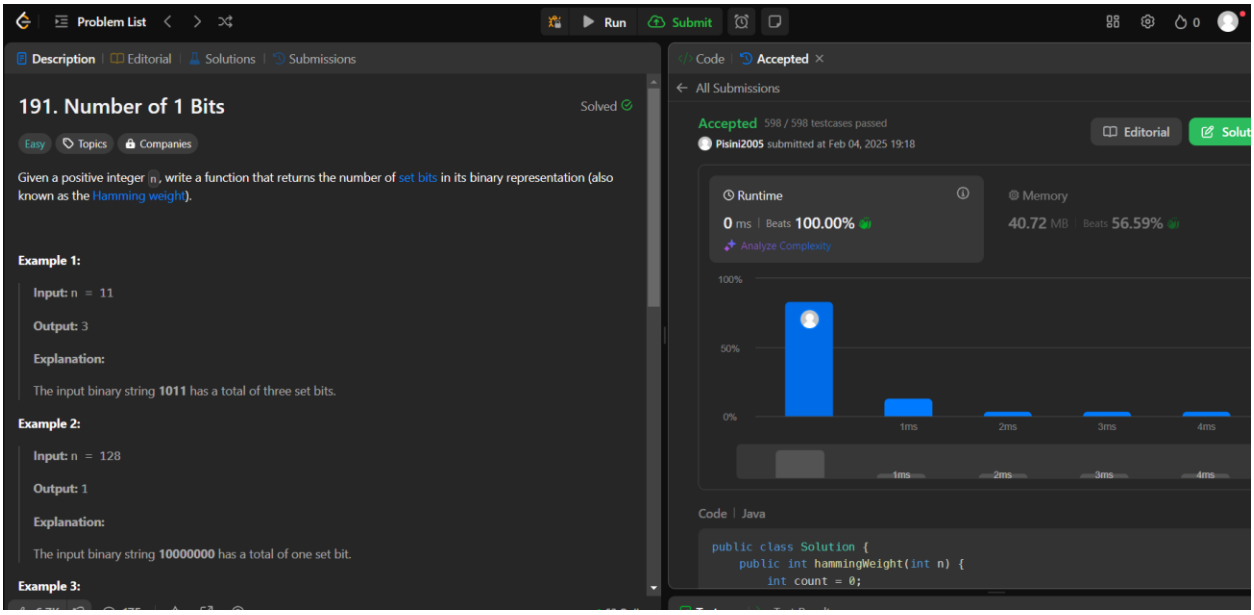


1. **Problem 3:** Number of 1 bits

2. **Implementation/code:**

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

3. **Output:**

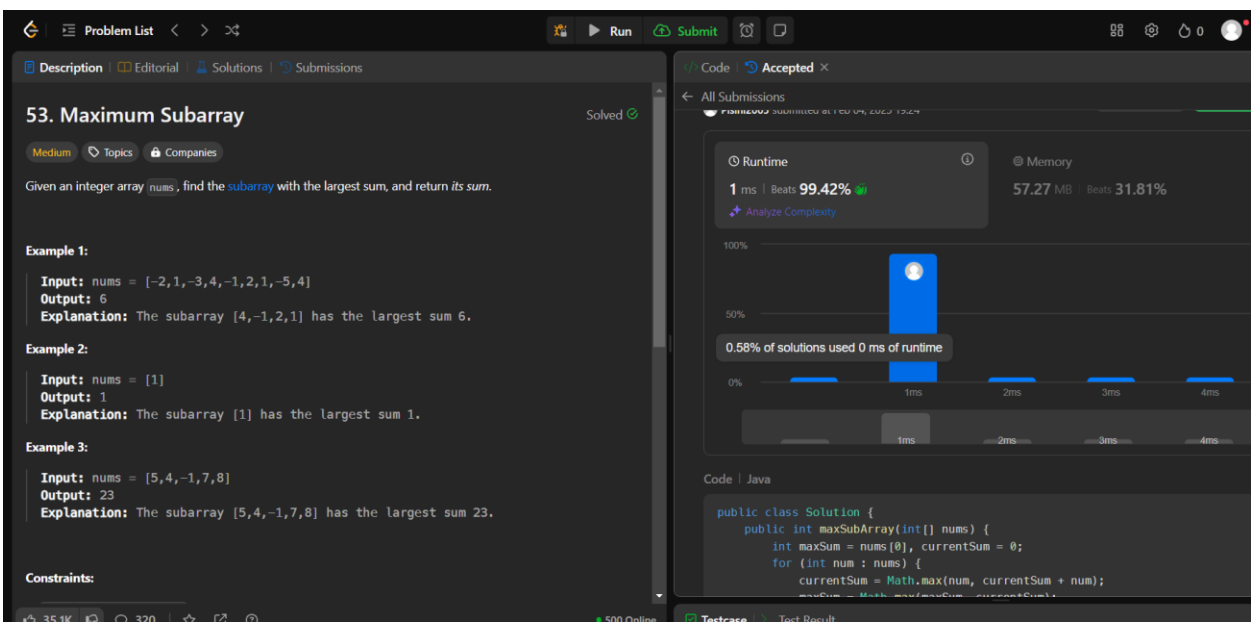


1. **Problem 4: Maximum Sub array**

2. **Implementation/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;
    }
}
```

3. **Output:**

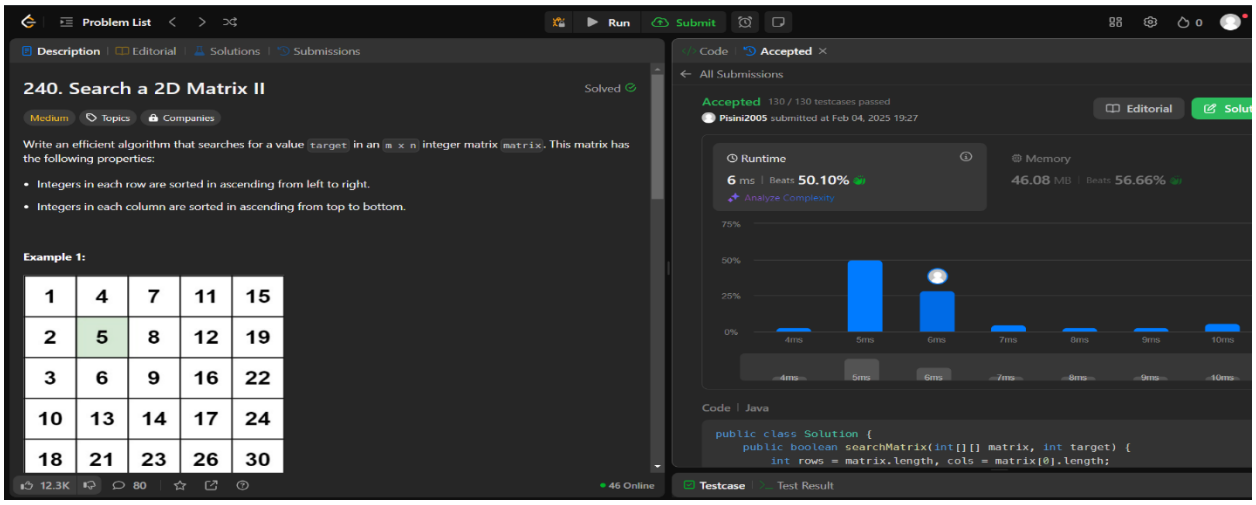


1. **Problem 5: Search a 2D Matrix II**

2. **Implementation/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;
    }}
```

3. **Output:**



1. **Problem 6: Super Pow**

2. **Implementation/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;    }}
```

3. **Output:**