Name : Ravi Prakash Singh Uid: 22BCS14299

Section: FL\_IOT\_602/A

[**190. Reverse Bits**](https://leetcode.com/problems/reverse-bits/)

Sol : public class Solution {

    // you need treat n as an unsigned value

    public int reverseBits(int n) {

        int result = 0;

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

            result <<= 1;

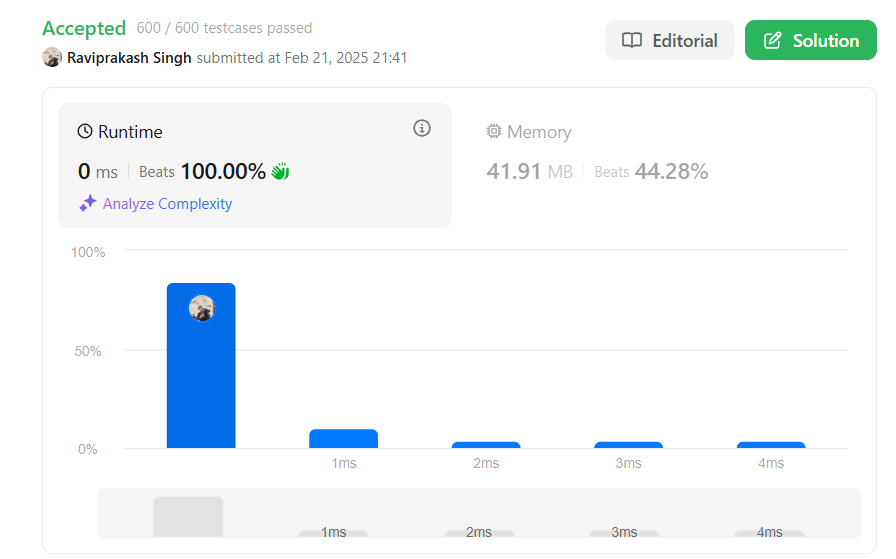
            result |= (n & 1);

            n >>= 1;

        }

        return result;

    }

}

[**191. Number of 1 Bits**](https://leetcode.com/problems/number-of-1-bits/)

Sol:

class Solution {

    public int hammingWeight(int n) {

        int count = 0;

        while (n != 0) {

            count += (n & 1);

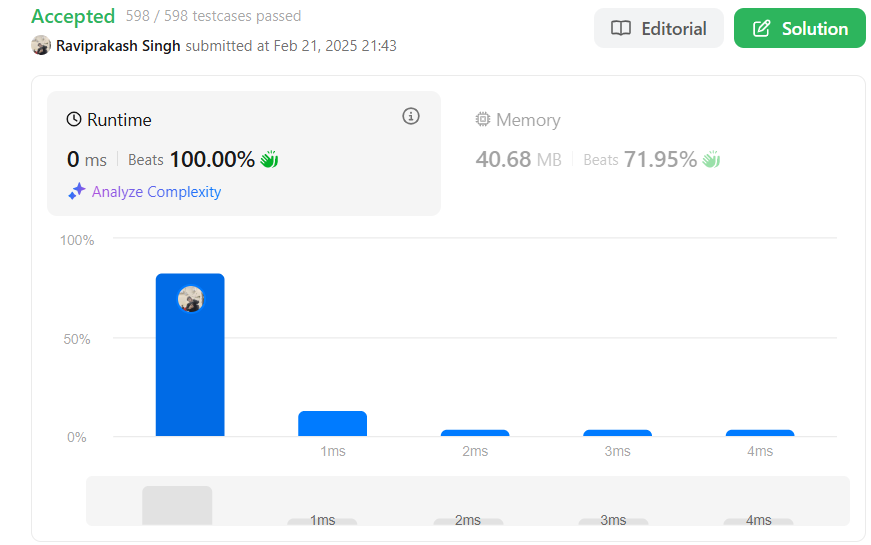
            n >>>= 1;

        }

        return count;

    }

}



[**53. Maximum Subarray**](https://leetcode.com/problems/maximum-subarray/)

Sol:

class Solution {

    public int maxSubArray(int[] nums) {

        int max = nums[0];

        int currSum = 0;

        for(int i : nums){

                currSum+=i;

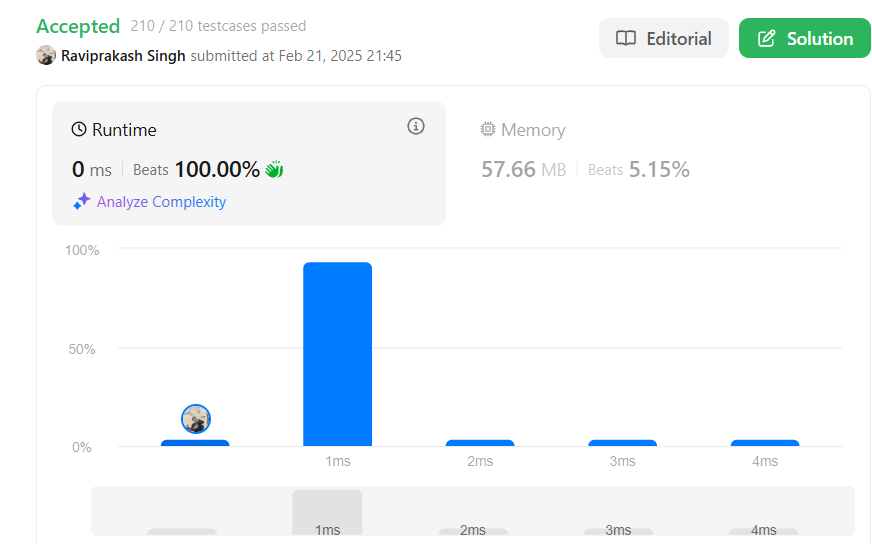
                max = max<currSum?currSum:max;

                if(currSum<0)currSum=0;

        }

        return max;

    }

}

[**1763. Longest Nice Substring**](https://leetcode.com/problems/longest-nice-substring/)

Sol: class Solution {

public String longestNiceSubstring(String s) {

Set<Character> charSet = new HashSet<>();

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

charSet.add(s.charAt(i));

}

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

if (charSet.contains(Character.toUpperCase(s.charAt(i))) &&

charSet.contains(Character.toLowerCase(s.charAt(i)))) {

continue;}

String s1 = longestNiceSubstring(s.substring(0, i));

String s2 = longestNiceSubstring(s.substring(i+1));

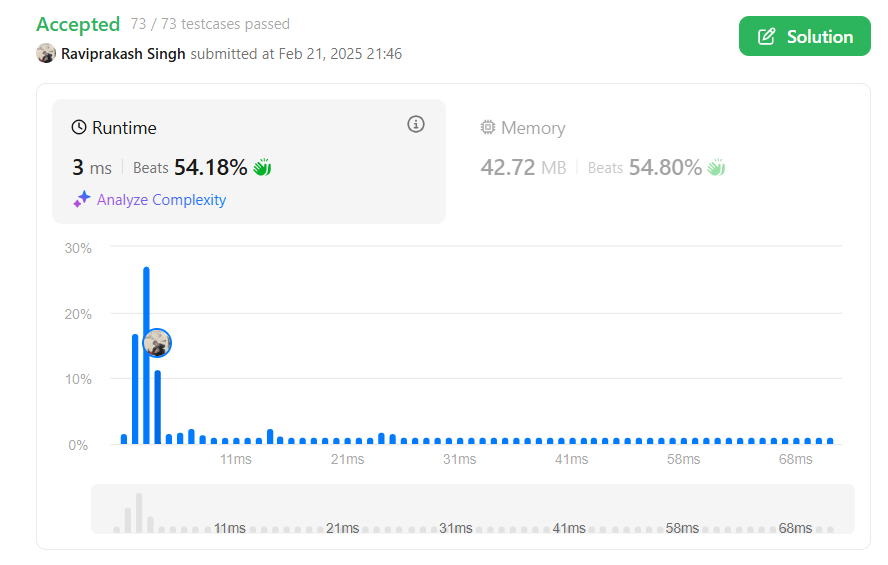
return s1.length()>= s2.length() ? s1 : s2;

}

return s;

}

}



[**240. Search a 2D Matrix II**](https://leetcode.com/problems/search-a-2d-matrix-ii/)

Sol:

class Solution {

    public boolean searchMatrix(int[][] matrix, int target) {

        if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {

            return false;    }

        int row = 0;

        int col = matrix[0].length - 1;

        while (row < matrix.length && col >= 0) {

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

                return true;

            } else if (matrix[row][col] < target)row ++;

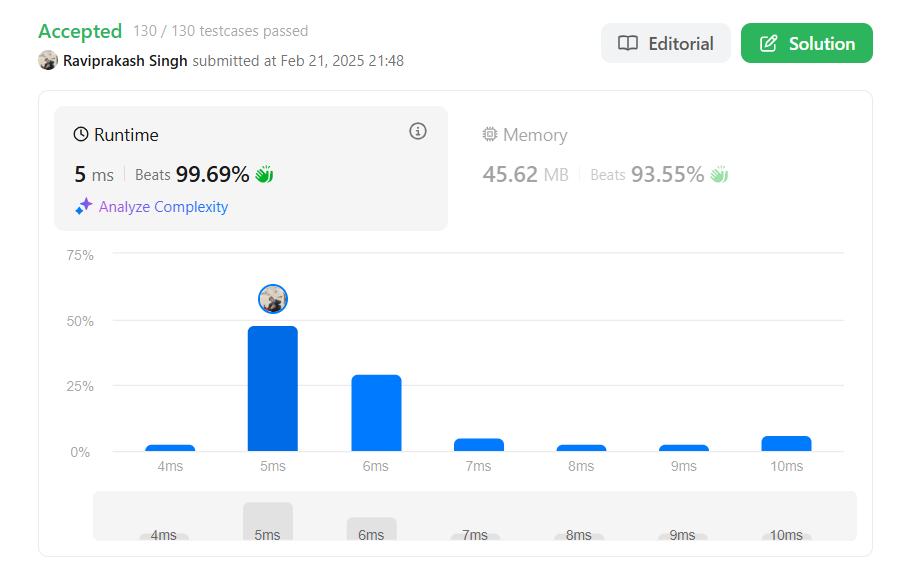
           else  col--;

        }

        return false;

    }

}



[**372. Super Pow**](https://leetcode.com/problems/super-pow/)

Sol:

class Solution {

    private static final int MOD = 1337;

    private int modPow(int a, int b) {

        int result = 1;

        a = 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;

        a = a % MOD;

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

            result = modPow(result, 10) \* modPow(a, b[i]) % MOD;

        }

        return result;

    }

}

