**Assignment 4**

**Name: Aditi Singh UID: 22BCS10670**

**Section: FL\_IOT\_602 Group: A**

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

**Sol:** class Solution {

    public String longestNiceSubstring(String s) {

        if(s.isEmpty()){

            return "";

        }

        int index = find(s);

        if(index == -1){

            return s;

        }

        else{

            String left = longestNiceSubstring(s.substring(0,index));

            String right = longestNiceSubstring(s.substring(index+1));

            return left.length()>=right.length()?left:right;

        }

    }

    private int find(String s){

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

            char c= s.charAt(i);

            int index;

            if(c>='A' && c<='Z'){

                index = s.indexOf((char)(c - 'A'+'a'));

            }else{

                index = s.indexOf((char)(c - 'a'+'A'));

            }

            if(index==-1){

                return i;

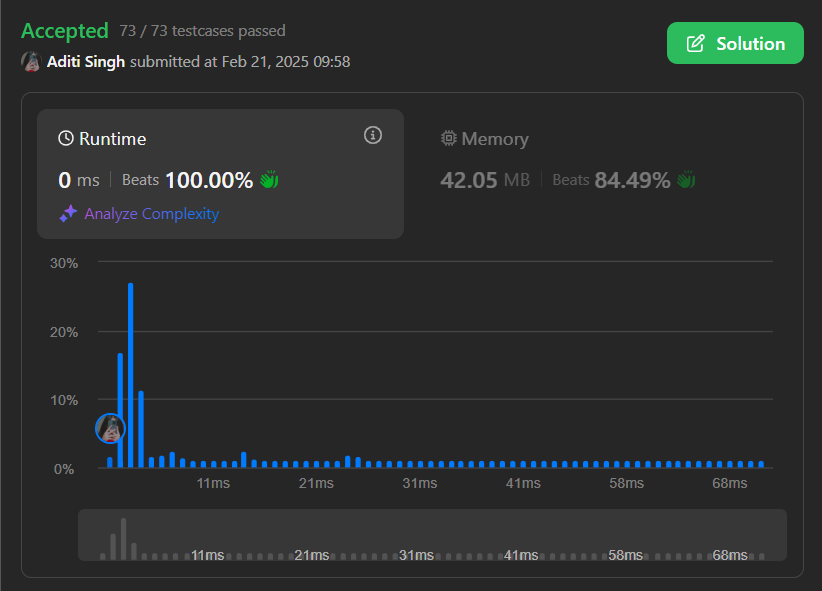
            }

        }return -1;

    }

}

**Output:**

****

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

**Sol:** 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;

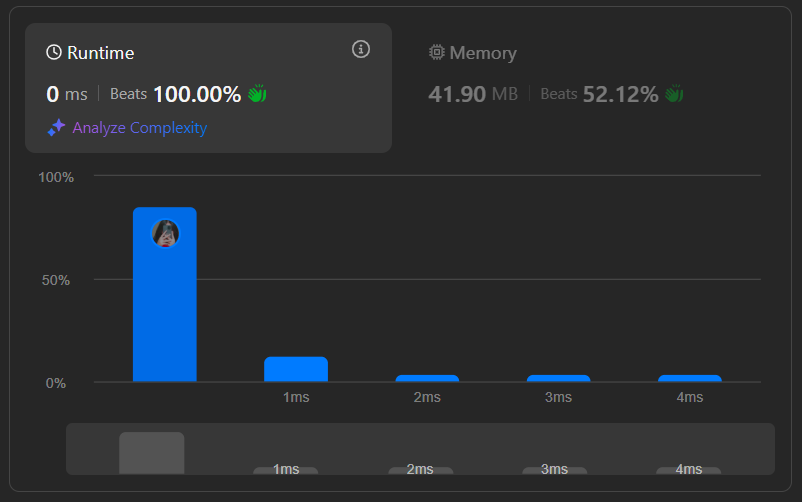
        }

        return result;

    }

}

**Output:**

****

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

**Sol:** public class Solution {

    public int hammingWeight(int n) {

        int res = 0;

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

            if (((n >> i) & 1) == 1) {

                res += 1;

            }

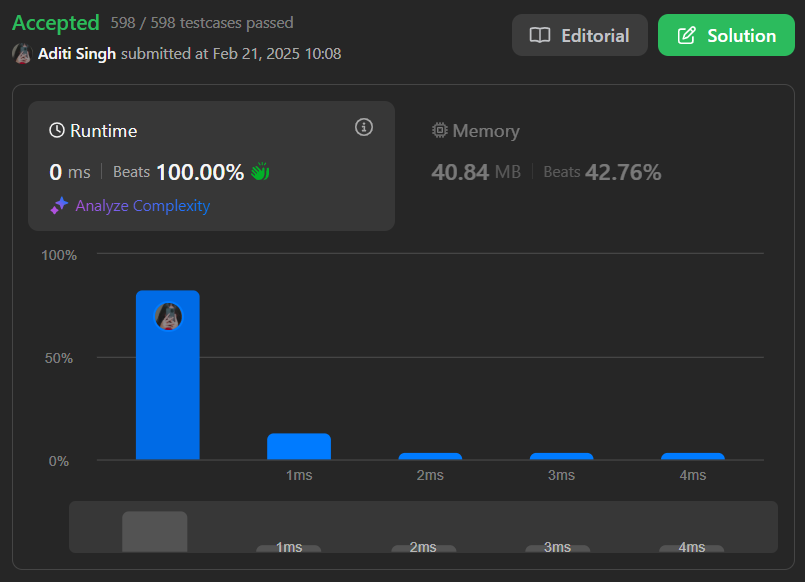
        }

        return res;

    }

}

**Output:**

****

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

**Sol:** class Solution {

    public int maxSubArray(int[] nums) {

        int max=Integer.MIN\_VALUE;

        int c=0;

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

         int num = nums[i];

        if (c < 0) {

        c = 0;

    }

      c=c+ num;

     max = Math.max(max, c);

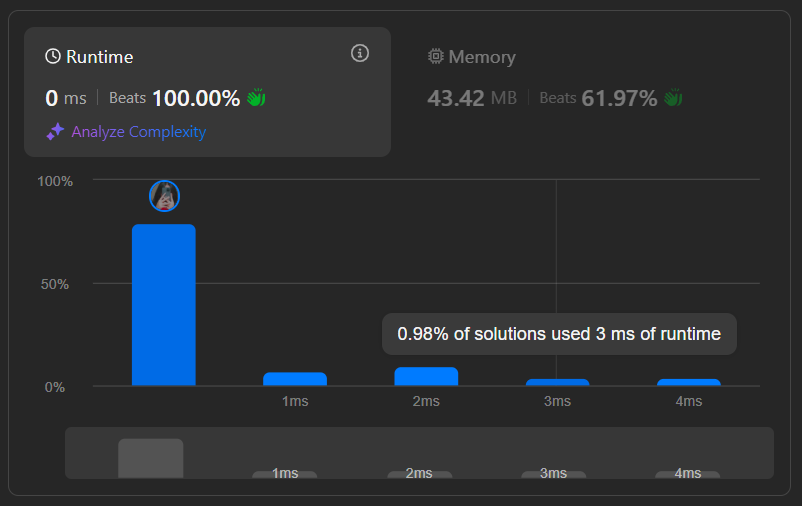
}

        return max;

    }

}

**Output:**

****

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

**Sol:** class Solution {

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

        int n = matrix.length, m = matrix[0].length;

        int row = 0, col = m-1;

        while(row < n && col >= 0){

            if(matrix[row][col] == target) return true;

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

            else col--;

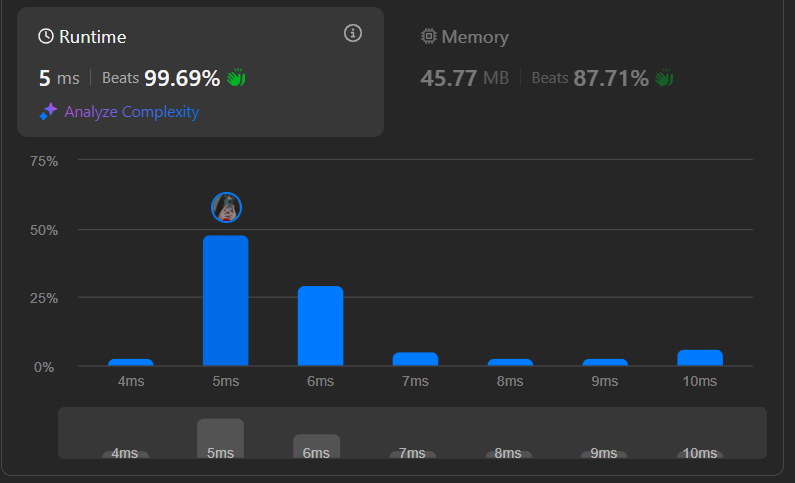
        }

        return false;

    }

}

**Output:**

****

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

**Sol:** class Solution {

    private int binExp(int a, int b, int M) {

        int res = 1;

        a %= M;

        while (b > 0) {

            if ((b & 1) != 0)

                res = (res \* a) % M;

            a = (a \* a) % M;

            b >>= 1;

        }

        return res;

    }

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

        int m = 1140;

        int exp = 0;

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

            exp = (exp \* 10 + b[i]) % m;

        if (exp == 0)

            exp = 1140;

        return binExp(a, exp, 1337);

    }

}

**Output:**  