Name : Ravi Prakash Singh

Uid : 22BCS14299

Section: Fl\_IOT\_602

Group: A

Que 1:[**Median of Two Sorted Arrays**](https://leetcode.com/problems/median-of-two-sorted-arrays/)

**Sol:** class Solution {

    public double findMedianSortedArrays(int[] nums1, int[] nums2) {

        int n = nums1.length;

        int m = nums2.length;

        int []ans = new int[n + m];

        int i = 0 ;

        int j = 0 ;

        int k = 0;

        while(j<n && k <m){

            if(nums1[j]>nums2[k]){

                ans[i++]=nums2[k++];

            }

            else {

                ans[i++]=nums1[j++];

            }

        }

        while(j<n){

            ans[i++]=nums1[j++];

        }

        while(k<m){

            ans[i++]=nums2[k++];

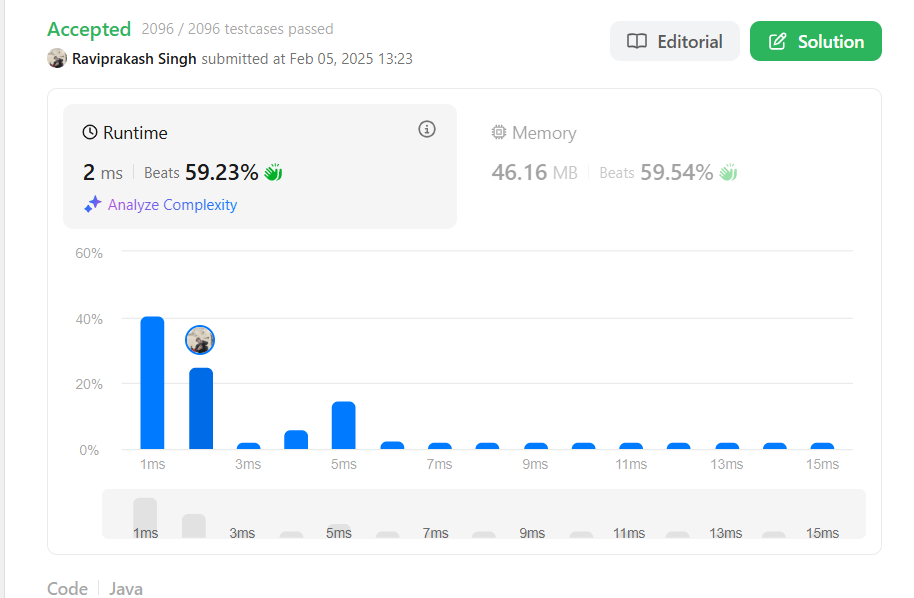
        }

        if((m+n)%2!=0 ) return (double)ans[(m+n)/2];

        else return (double)(ans[(m+n)/2 -1] + ans[(m+n)/2])/2;

    }

}

****

**Que 2:** [**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;

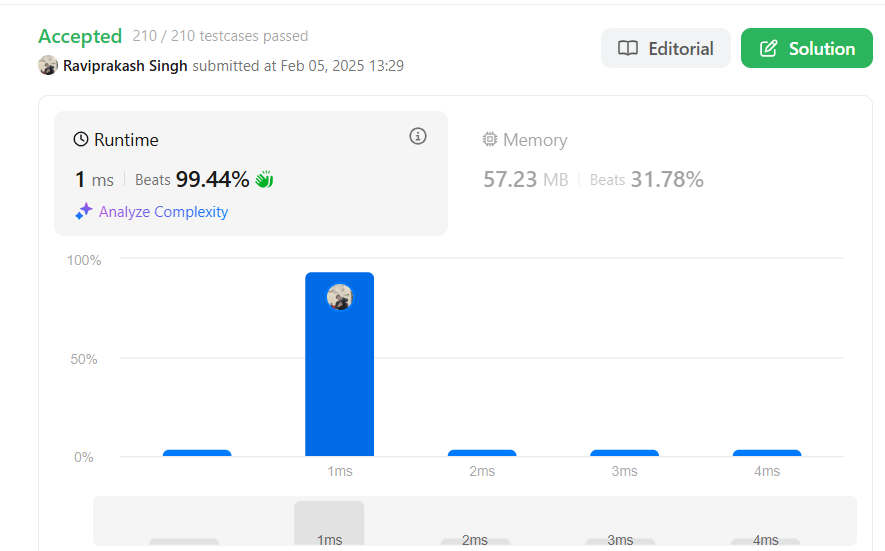
                if(currSum<0)currSum=0;

        }

        return max;

    }

}



Que 3: [**Merge Sorted Array**](https://leetcode.com/problems/merge-sorted-array/)

Sol: class Solution {

public void merge(int[] nums1, int m, int[] nums2, int n) {

int k = m+n-1;

int i = m-1 , j= n-1 ;

while(j>=0&&i>=0){

if(nums1[i]<nums2[j]){

nums1[k--]=nums2[j--];

}

else{

nums1[k--]=nums1[i--];

}

}

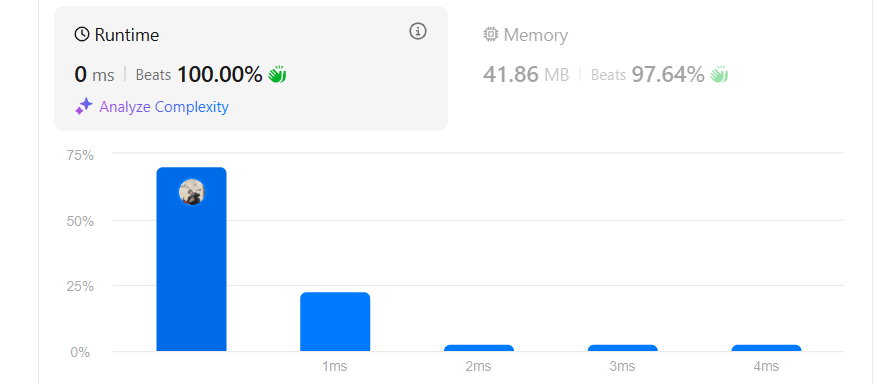
while(j>=0){

nums1[k--]=nums2[j--];

}

}

}



Que 4: [**Sort Colors**](https://leetcode.com/problems/sort-colors/)

class Solution {

public void sortColors(int[] nums) {

// using three pointer concept

int n = nums.length;

int curr = 0 , i= 0 , j= n-1; //j 2 ko aur i 0 ko hold karega

while(curr<=j){

if(nums[curr]== 0 ){

int temp = nums[curr];

nums[curr] = nums[i];

nums[i] = temp;

i++;

curr++;

}

else if( nums[curr]==2){

int temp = nums[curr];

nums[curr] = nums[j];

nums[j] = temp;

j--;

}

else{

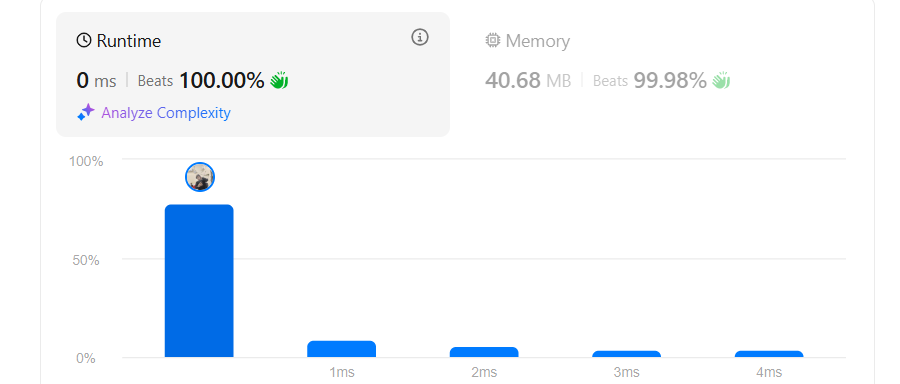
curr++;

}

}

}

}



Que 6: [**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);

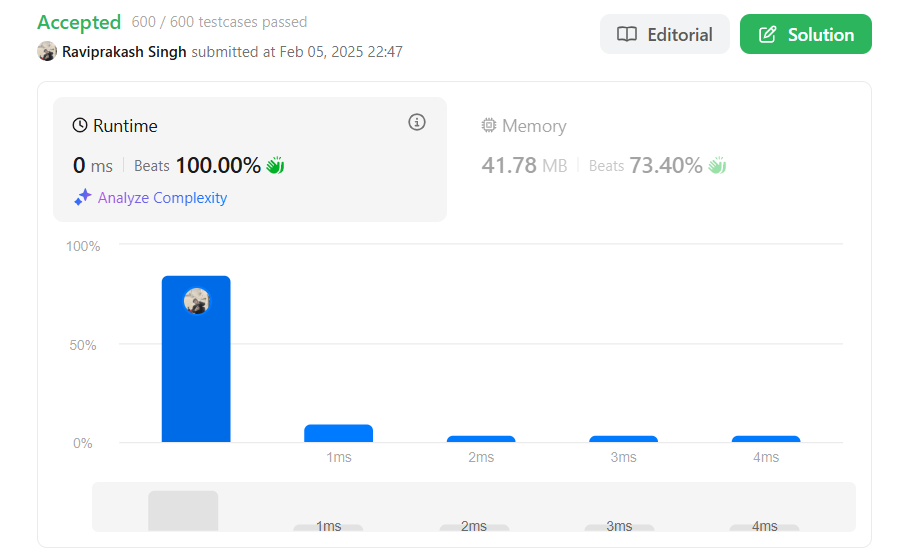
            n >>= 1;

        }

        return result;

    }

}



Que 7: [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);

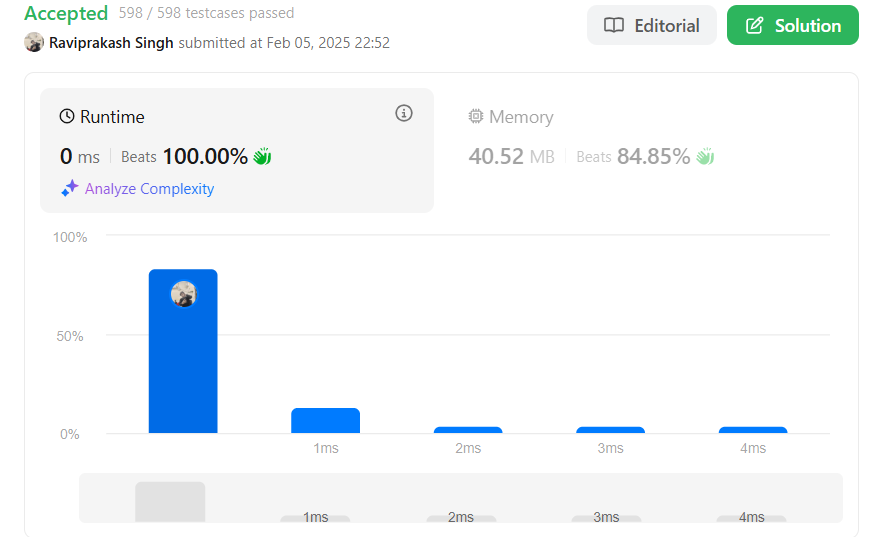
            n >>>= 1;

        }

        return count;

    }

}



Que 8: [**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--;

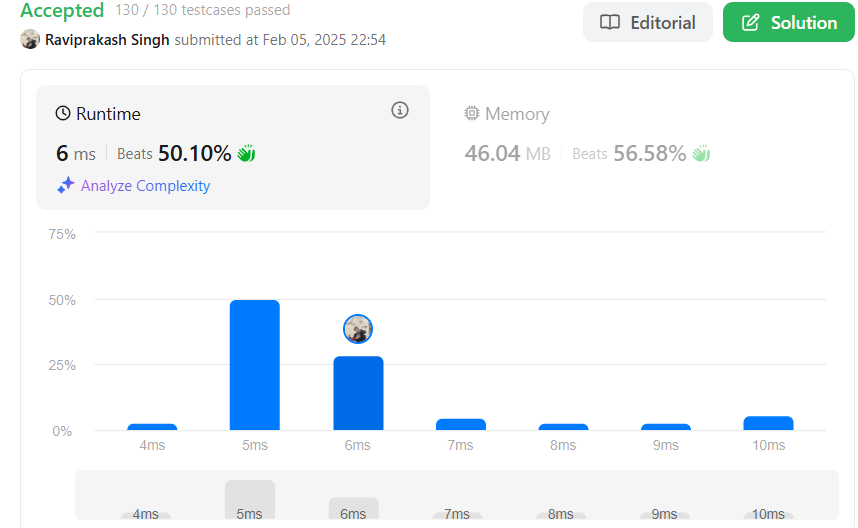
            }

        }

        return false;

    }

}



Que 10: [**Super Pow**](https://leetcode.com/problems/super-pow/)

Sol:

class Solution {

    private static final int MOD = 137;

    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++) {

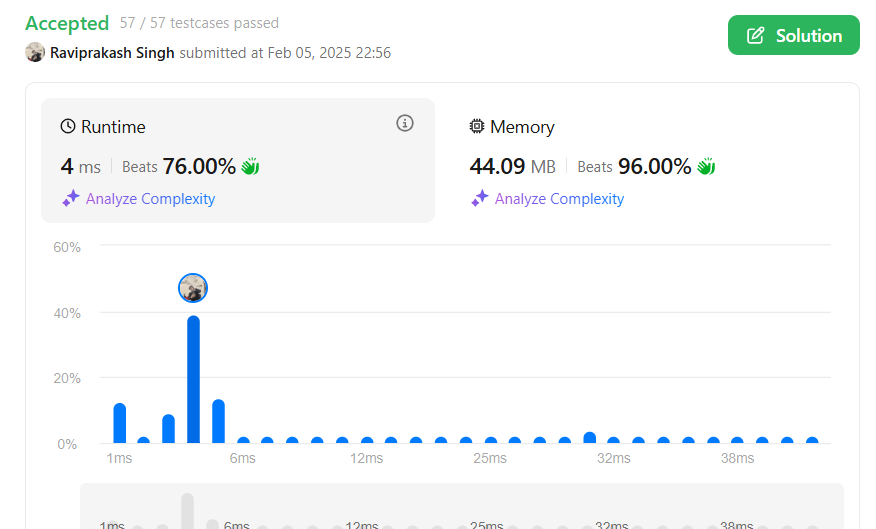
            result = modPow(result, 10) \* modPow(a, b[i]) % MOD;

        }

        return result;

    }

}



Que 11: [Beautiful Array](https://leetcode.com/problems/beautiful-array/)

Sol:

class Solution {

    public int[] beautifulArray(int n) {

        return buildBeautifulArray(n);

    }

    private int[] buildBeautifulArray(int n) {

        if (n == 1) {

            return new int[] { 1 };  // Base case

        }

        int[] odd = buildBeautifulArray((n + 1) / 2);  // For odd numbers

        int[] even = buildBeautifulArray(n / 2);  // For even numbers

        int[] result = new int[n];

        int idx = 0;

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

            result[idx++] = 2 \* odd[i] - 1;

        }

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

            result[idx++] = 2 \* even[i];

        }

        return result;

    }

}



Que 12: [**First Bad Version**](https://leetcode.com/problems/first-bad-version/)

Sol:

/\* The isBadVersion API is defined in the parent class VersionControl.

      boolean isBadVersion(int version); \*/

public class Solution extends VersionControl {

    public int firstBadVersion(int n) {

        int left = 1; // Start from the first version

        int right = n; // End at the nth version

        while (left < right) {

            int mid = left + (right - left) / 2; // Avoid potential overflow

            if (isBadVersion(mid)) {

                right = mid; // The first bad version is at mid or before it

            } else {

                left = mid + 1; // The first bad version is after mid

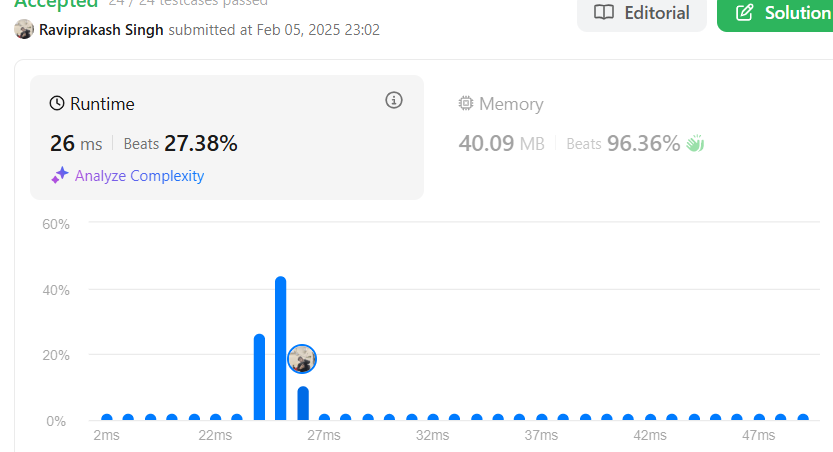
            }

        }

        return left; // At the end of the loop, left == right, which is the first bad version

    }

}



Que 13: [**Find Peak Element**](https://leetcode.com/problems/find-peak-element/)

Sol:

class Solution {

    public int findPeakElement(int[] nums) {

        int left = 0;

        int right = nums.length - 1;

        while (left < right) {

            if (nums[mid] > nums[mid + 1]) {

                right = mid;

            } else {

                // Peak lies on the right side

                left = mid + 1;

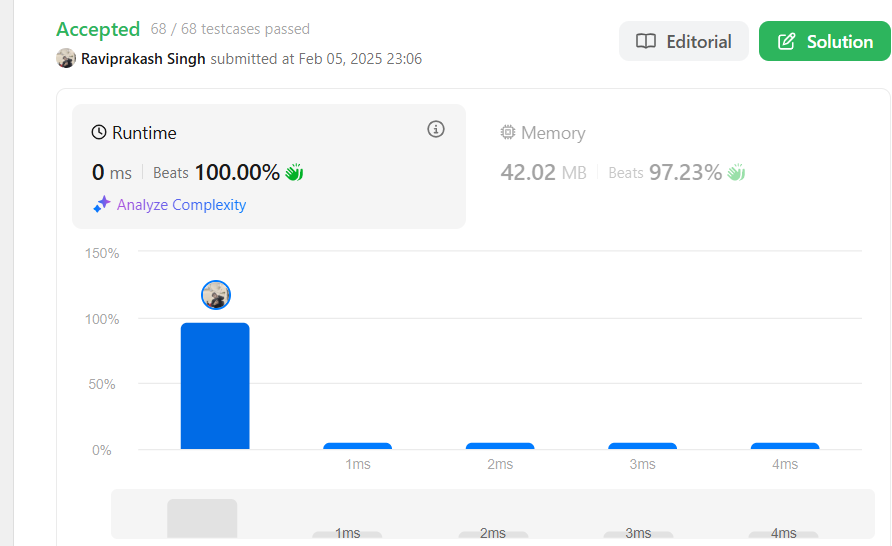
            }

        }

        return left;

    }

}



Que 14: [**Search in Rotated Sorted Array**](https://leetcode.com/problems/search-in-rotated-sorted-array/)

Sol:

class Solution {

    public int search(int[] nums, int target) {

        int left = 0;

        int right = nums.length - 1;

        while (left <= right) {

            int mid = left + (right - left) / 2;

            // If the target is found at mid, return its index

            if (nums[mid] == target) {

                return mid;

            }

            // Determine which part is sorted

            if (nums[left] <= nums[mid]) {

                // Left part is sorted

                if (target >= nums[left] && target < nums[mid]) {

                    right = mid - 1;  // target is in the left half

                } else {

                    left = mid + 1;  // target is in the right half

                }

            } else {

                if (target > nums[mid] && target <= nums[right]) {

                    left = mid + 1;  // target is in the right half

                } else {

                    right = mid - 1;  // target is in the left half

                }

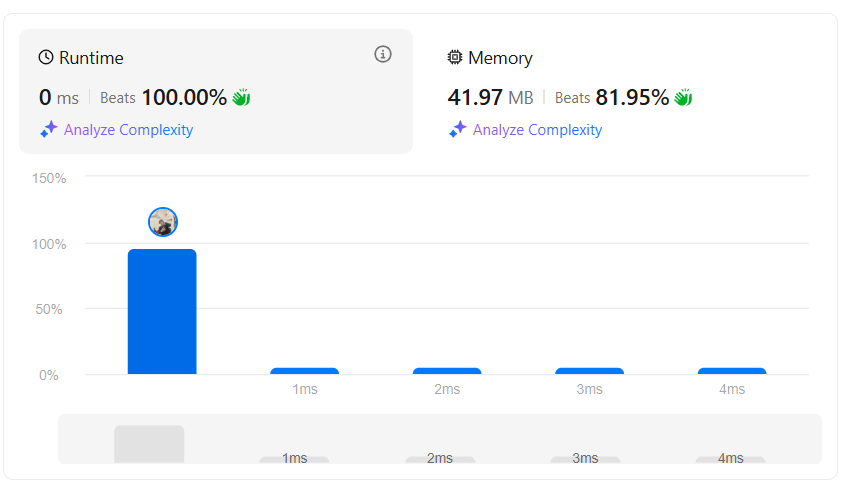
            }

        }

        return -1;

    }

}



Que 15: [**Wiggle Sort II**](https://leetcode.com/problems/wiggle-sort-ii/)

Sol:

import java.util.Arrays;

class Solution {

    public void wiggleSort(int[] nums) {

        int n = nums.length;

        int[] sorted = nums.clone();

        Arrays.sort(sorted);

        int mid = (n + 1) / 2;

        int left = mid - 1;

        int right = n - 1;

        for (int i = 0; i < n; i++) {

            if (i % 2 == 0) {

                nums[i] = sorted[left--];

            } else {

                nums[i] = sorted[right--];

            }

        }

    }

}

