Assignment

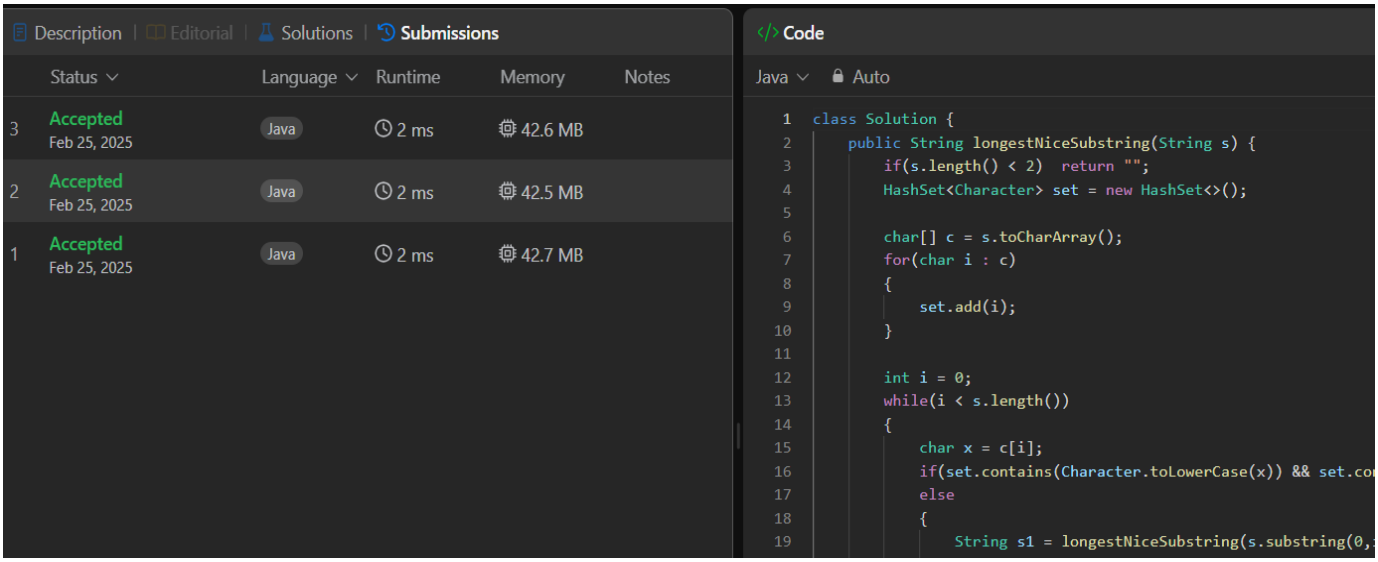
Name – JIGYA JAIN
UID – 22BCS50101
Section – 609-A

1. <u>Longest Nice Substring</u> –

Code -

```
class Solution {
   public String longestNiceSubstring(String s) {
      if(s.length() < 2)  return "";
      HashSet<Character> set = new HashSet<>();
      char[] c = s.toCharArray();
      for(char i : c)   set.add(i);
      int i = 0;
      while(i < s.length())
      {
         char x = c[i];
         if(set.contains(Character.toLowerCase(x)) &&
set.contains(Character.toUpperCase(x)))  i++;
         else {
            String s1 = longestNiceSubstring(s.substring(0,i));
            String s2 = longestNiceSubstring(s.substring(i+1));
            return s1.length() >= s2.length() ? s1 : s2;
         }
     }  return s;
 }}
```

Submission –

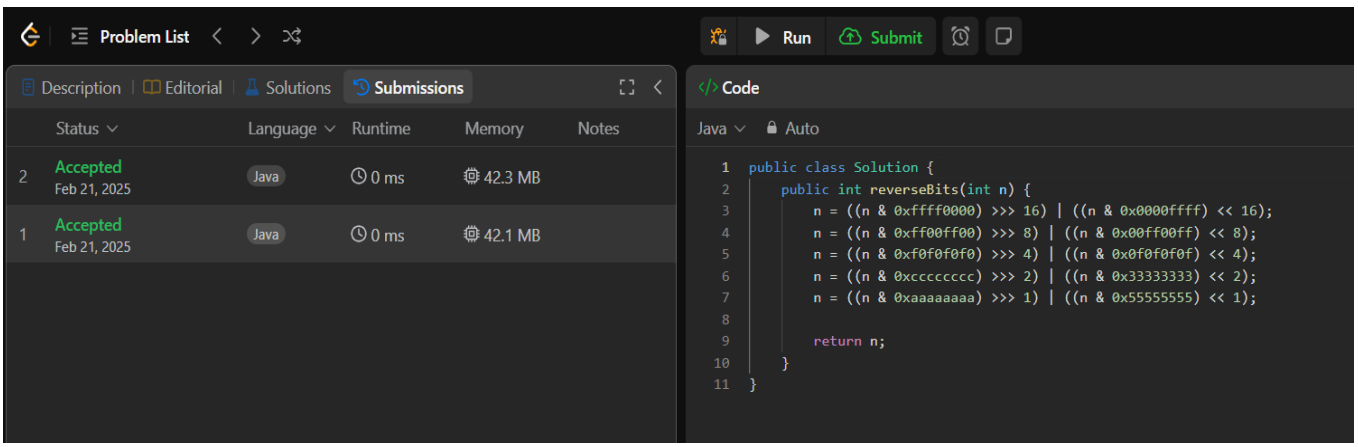


2. <u>Reverse Bits</u> –

Code –

```
public class Solution {
    public int reverseBits(int n) {
        n = ((n & 0xffff0000) >>> 16) | ((n & 0x0000ffff) << 16);
        n = ((n & 0xff00ff00) >>> 8) | ((n & 0x00ff00ff) << 8);
        n = ((n & 0xf0f0f0f0) >>> 4) | ((n & 0x0f0f0f0f) << 4);
        n = ((n & 0xcccccccc) >>> 2) | ((n & 0x33333333) << 2);
        n = ((n & 0xaaaaaaaa) >>> 1) | ((n & 0x55555555) << 1);
        return n;
    }
}
```

Submission –

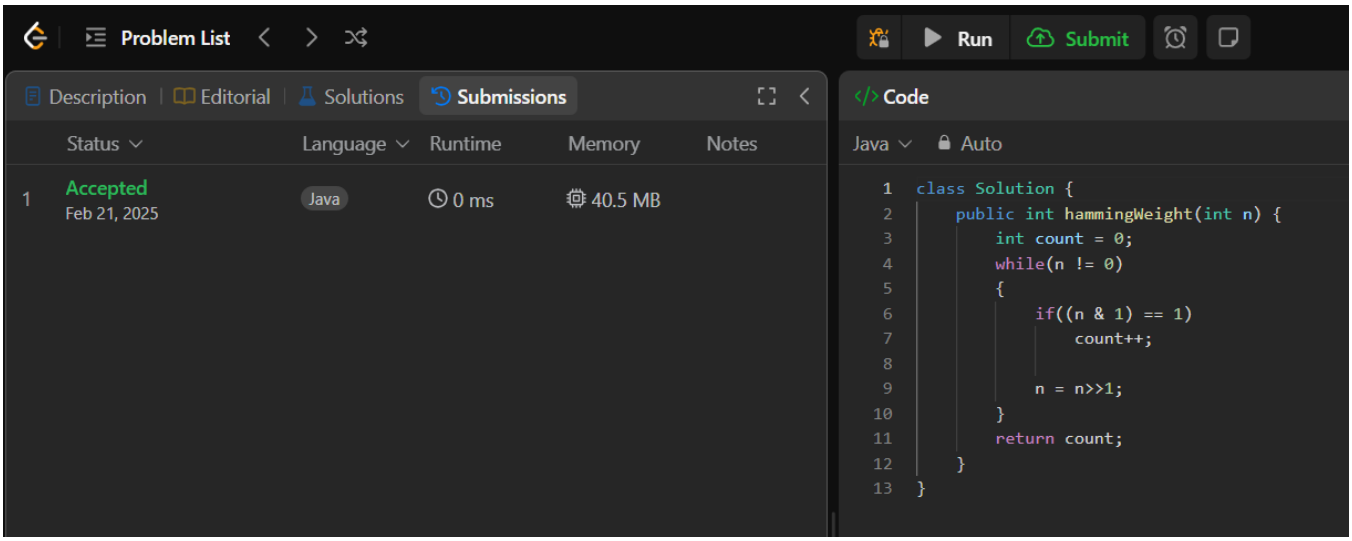


3. <u>Number of 1 Bits</u> –

Code –

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

Submission –

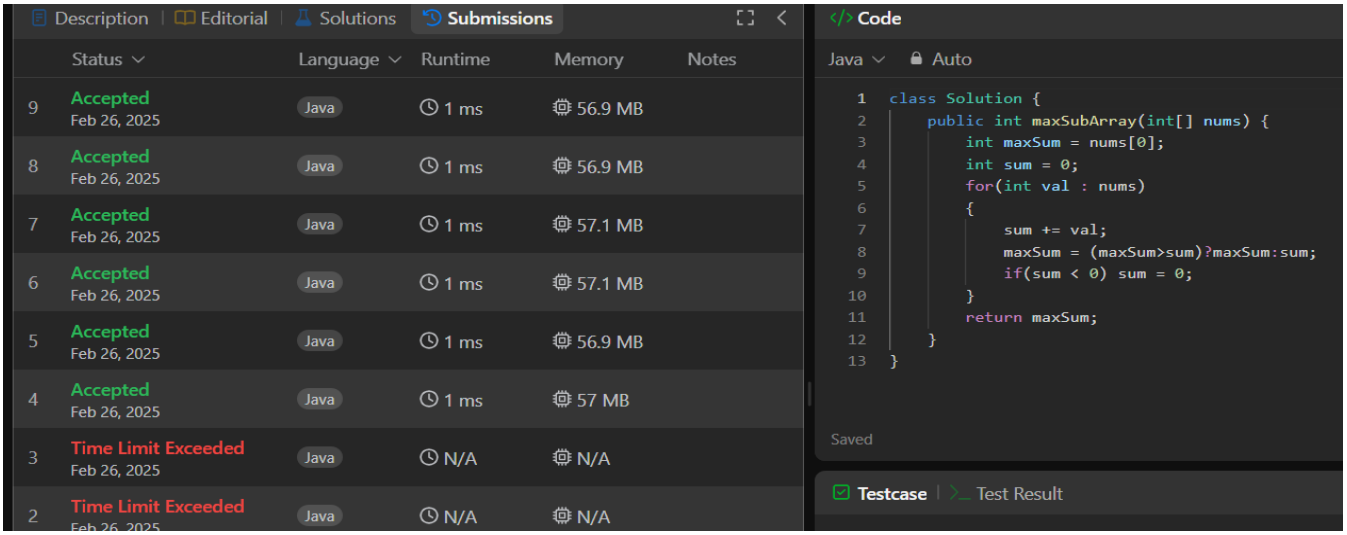


4. <u>Maximum Subarray</u> –

Code –

```
class Solution {
    public int maxSubArray(int[] nums) {
        int maxSum = nums[0];
        int sum = 0;
        for(int val : nums)
        {
            sum += val;
            maxSum = (maxSum>sum)?maxSum:sum;
            if(sum < 0) sum = 0;
        }
        return maxSum;
    }
}
```

Submission –

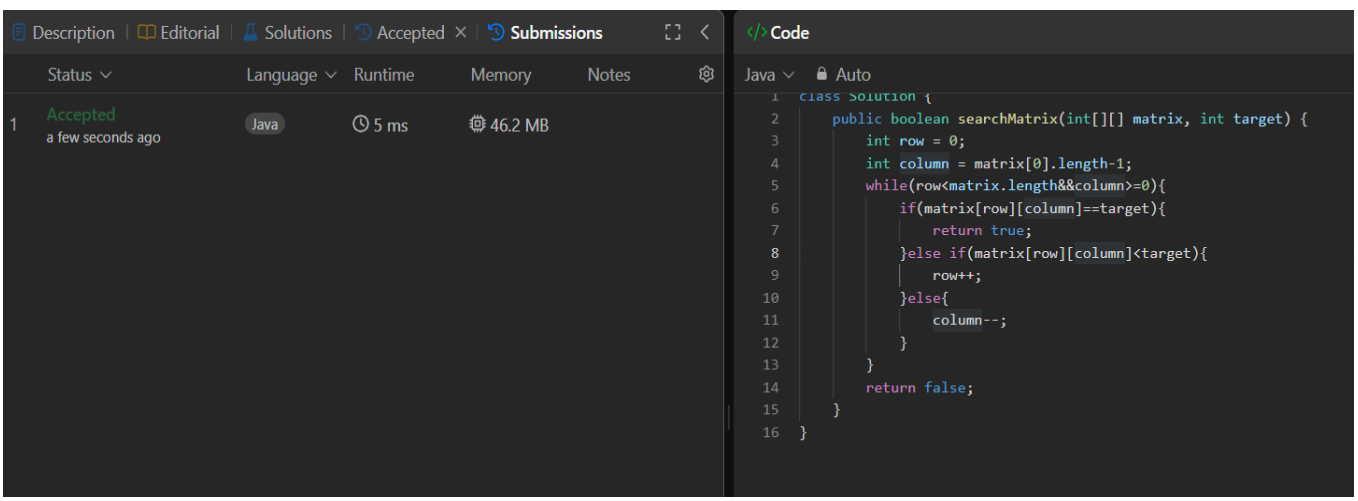


5. Search a 2D Matrix II –

Code –

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int row = 0;
        int column = matrix[0].length-1;
        while(row<matrix.length&&column>=0){
            if(matrix[row][column]==target){
                return true;
            }else if(matrix[row][column]<target){
                row++;
            }else{
                column--;
            }
        }
        return false;
    }
}
```

Submission –



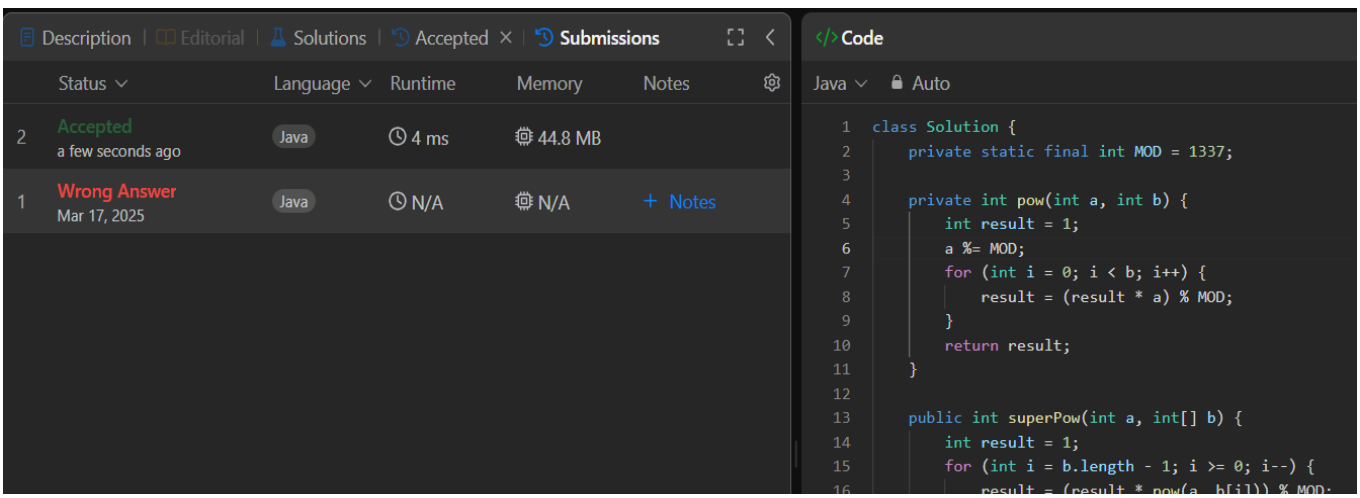
6. Super Pow –

Code –

```
class Solution {
    private static final int MOD = 1337;

    private int pow(int a, int b) {
        int result = 1;
        a %= MOD;
        for (int i = 0; i < b; i++) {
```

```
            result = (result * a) % MOD;
        }
        return result;
    }

    public int superPow(int a, int[] b) {
        int result = 1;
        for (int i = b.length - 1; i >= 0; i--) {
            result = (result * pow(a, b[i])) % MOD;
            a = pow(a, 10);
        }
        return result;
    }
}
```

Submission –



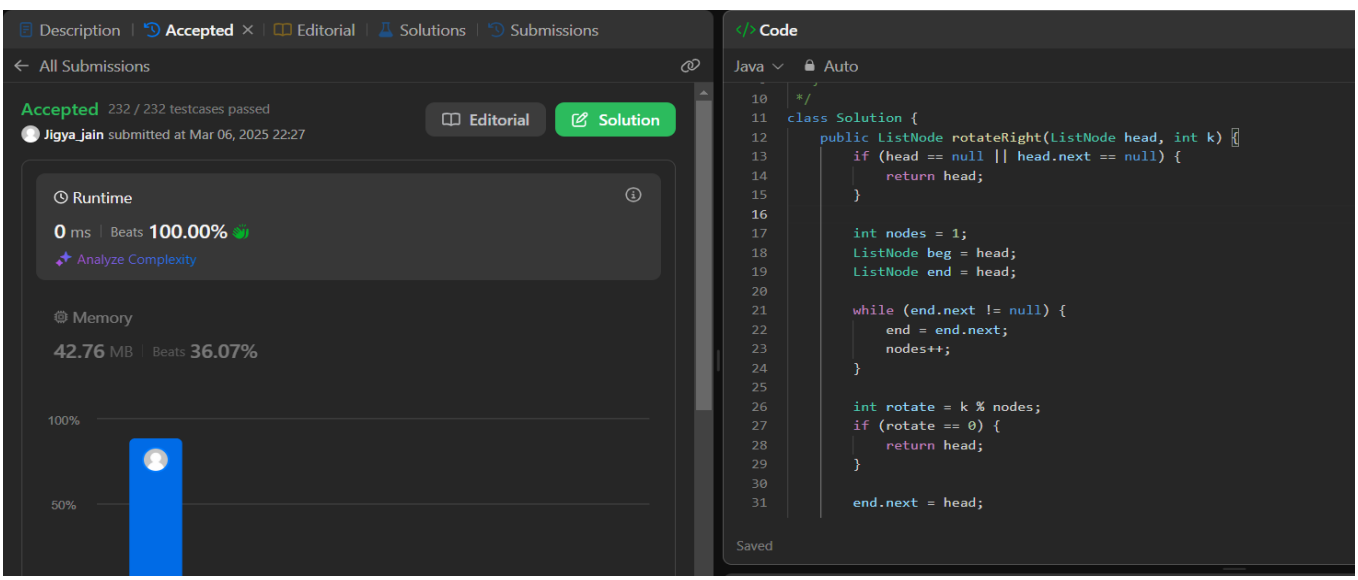
7. <u>Beautiful Array</u> –

Code –

```
class Solution {
    public ListNode rotateRight(ListNode head, int k) {
        if (head == null || head.next == null) {
            return head;
        }

        int nodes = 1;
        ListNode beg = head;
        ListNode end = head;

        while (end.next != null) {
```

```
            end = end.next;
            nodes++;
        }

        int rotate = k % nodes;
        if (rotate == 0) {
            return head;
        }
        end.next = head;

        ListNode newTail = head;
        for (int i = 0; i < nodes - rotate - 1; i++) {
            newTail = newTail.next;
        }
        ListNode newHead = newTail.next;
        newTail.next = null;
        return newHead;
    }
}
```

Submission –



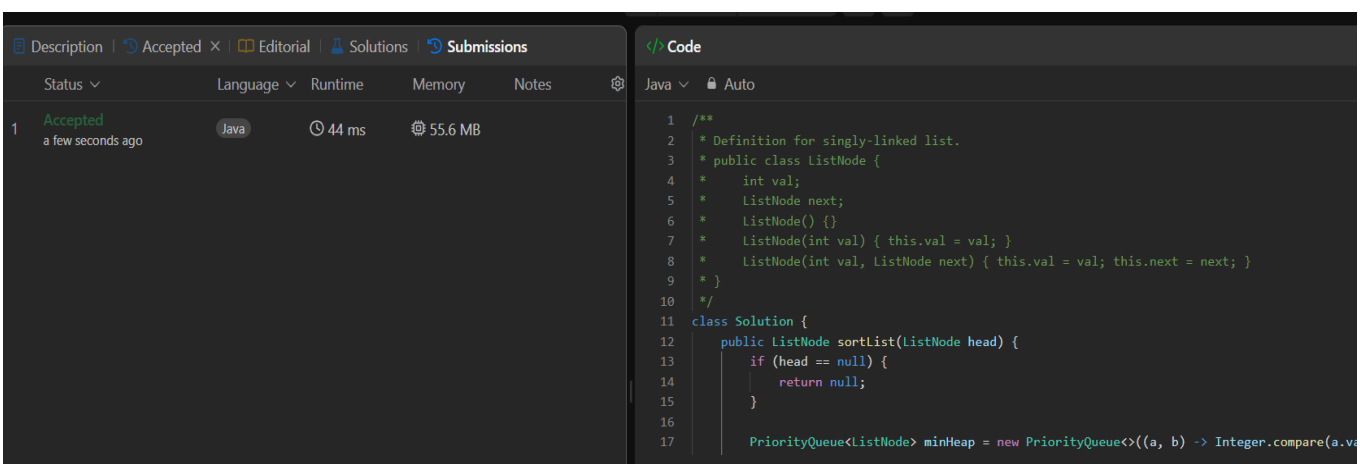
8. <u>The Skyline Problem</u> –

Code –

```
class Solution {
    public ListNode sortList(ListNode head) {
        if (head == null) {
            return null;
```

```
        }

        PriorityQueue<ListNode> minHeap = new PriorityQueue<>((a, b) ->
Integer.compare(a.val, b.val));

        while (head != null) {
            minHeap.add(head);
            head = head.next;
        }

        ListNode dummy = new ListNode(0);
        ListNode temp = dummy;

        while (!minHeap.isEmpty()) {
            temp.next = minHeap.poll();
            temp = temp.next;
        }

        temp.next = null; // Ensure the last node points to null
        return dummy.next;
    }
}
```

Submission –



9. <u>Reverse Pairs</u> –

Code –

```
class Solution {

    public void merge(int[] arr, int low, int mid, int high) {
        ArrayList<Integer> temp = new ArrayList<>();
```

```
    int left = low;
    int right = mid+1;

    while(left <= mid && right <= high) {
        if(arr[left] <= arr[right]) {
            temp.add(arr[left++]);
        } else {
            temp.add(arr[right++]);
        }
    }

    while(left <= mid) temp.add(arr[left++]);
    while(right <= high) temp.add(arr[right++]);
    for(int i=low; i<=high; i++) {
        arr[i] = temp.get(i-low);
    }
}

public int countPairs(int[] arr, int low, int mid, int high) {
    int right = mid + 1;
    int cnt = 0;
    for(int i=low; i<=mid; i++) {
        while(right <= high && (long) arr[i] > 2L * arr[right])
        right++;
        cnt += (right - (mid + 1));
    }
    return cnt;
}

public int mergeSort(int[] arr, int low, int high) {
    int cnt = 0;
    if(low >= high) return cnt;
    int mid = (low + high) / 2;
    cnt += mergeSort(arr,low,mid);
    cnt += mergeSort(arr,mid+1,high);
    cnt += countPairs(arr,low,mid,high);
    merge(arr,low,mid,high);
    return cnt;
}
```

```
    public int reversePairs(int[] nums) {
        int n = nums.length;
        return mergeSort(nums, 0, n-1);
    }
}
```

Submission –