WORKSHEET 8

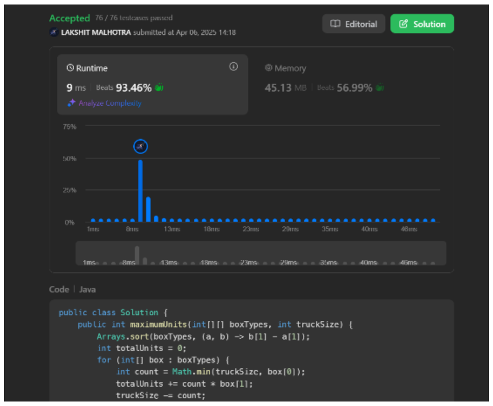
STUDENT NAME: LAKSHIT MALHOTRA **UID:** 22BCS13047
BRANCH: CSE **SECTION:** 22BCS_FL_IOT_601A
SEMESTER: 6 **DATE OF SUBMISSION:** 06/4/25
SUBJECT NAME: AP LAB -2 **SUBJECT CODE:** 22CSP-351

LEET CODE QUESTIONS :

1710 - MAXIMUM UNITS ON A TRUCK

PROBLEM: https://leetcode.com/problems/maximum-units-on-a-truck/

```
public class Solution {
    public int maximumUnits(int[][] boxTypes, int
truckSize) {
        Arrays.sort(boxTypes, (a, b) -> b[1] - a[1]);
        int totalUnits = 0;
        for (int[] box : boxTypes) {
            int count = Math.min(truckSize, box[0]);
            totalUnits += count * box[1];
            truckSize -= count;
            if (truckSize == 0) break;
        }
        return totalUnits;
    }
}
```

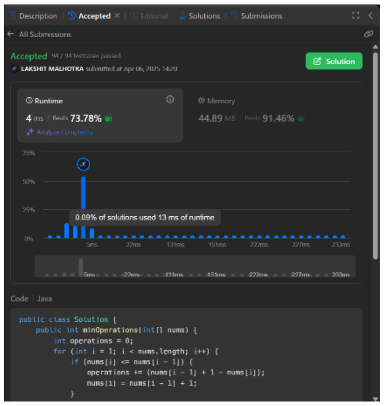

Accepted 76 / 76 testcases passed
LAKSHIT MALHOTRA submitted at Apr 06, 2025 14:18

Runtime 9 ms | Beats 93.46%
Memory 45.13 MB | Beats 56.99%

Code | Java
```
public class Solution {
    public int maximumUnits(int[][] boxTypes, int truckSize) {
        Arrays.sort(boxTypes, (a, b) -> b[1] - a[1]);
        int totalUnits = 0;
        for (int[] box : boxTypes) {
            int count = Math.min(truckSize, box[0]);
            totalUnits += count * box[1];
            truckSize -= count;
```

1827 - MINIMUM OPERATIONS TO MAKE THE ARRAY INCREASING

PROBLEM: https://leetcode.com/problems/minimum-operations-to-make-the-array-increasing/



Accepted 94 / 94 testcases passed
LAKSHIT MALHOTRA submitted at Apr 06, 2025 14:20
Runtime 4 ms | Beats 73.78%
Memory 44.89 MB | Beats 91.46%

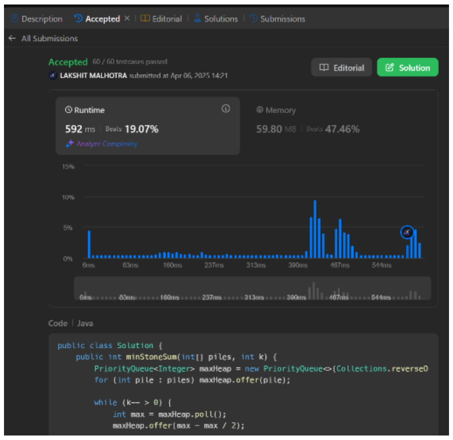
Code | Java
```
public class Solution {
    public int minOperations(int[] nums) {
        int operations = 0;
        for (int i = 1; i < nums.length; i++) {
            if (nums[i] <= nums[i - 1]) {
                operations += (nums[i - 1] + 1 - nums[i]);
                nums[i] = nums[i - 1] + 1;
            }
```

```
public class Solution {
    public int minOperations(int[] nums) {
        int operations = 0;
        for (int i = 1; i < nums.length; i++) {
            if (nums[i] <= nums[i - 1]) {
                operations += (nums[i - 1] + 1 - nums[i]);
                nums[i] = nums[i - 1] + 1;
            }
        }
        return operations;
```

1962 - REMOVE STONES TO MINIMIZE THE TOTAL

PROBLEM: https://leetcode.com/problems/remove-stones-to-minimize-the-total/

```
public class Solution {
    public int minStoneSum(int[] piles, int k) {
        PriorityQueue<Integer> maxHeap = new
PriorityQueue<>(Collections.reverseOrder());
        for (int pile : piles) maxHeap.offer(pile);
        while (k-- > 0) {
            int max = maxHeap.poll();
            maxHeap.offer(max - max / 2);
        }
        int sum = 0;
        while (!maxHeap.isEmpty()) sum +=
maxHeap.poll();
        return sum;
    }
}
```



Accepted 60 / 60 testcases passed
LAKSHIT MALHOTRA submitted at Apr 06, 2025 14:21

Runtime
592 ms | Beats 19.07%

Memory
59.80 MB | Beats 47.46%

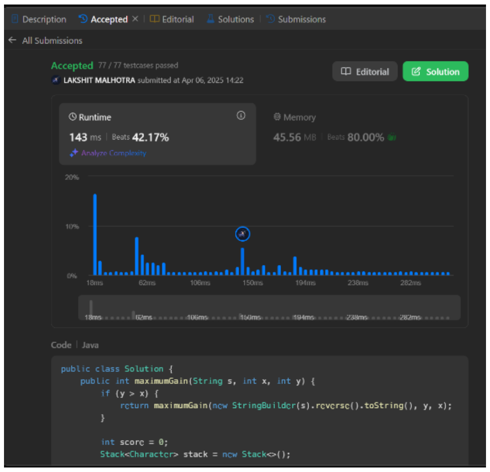
Code | Java

```
public class Solution {
    public int minStoneSum(int[] piles, int k) {
        PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Collections.reverseO
        for (int pile : piles) maxHeap.offer(pile);

        while (k-- > 0) {
            int max = maxHeap.poll();
            maxHeap.offer(max - max / 2);
```

1717 - MAXIMUM SCORE FROM REMOVING SUBSTRINGS

PROBLEM: https://leetcode.com/problems/maximum-score-from-removing-substrings/

```
public class Solution {
    public int maximumGain(String s, int x, int y) {
        if (y > x) {
            return maximumGain(new StringBuilder(s).reverse().toString(), y, x);
        }
        int score = 0;
        Stack<Character> stack = new Stack<>();
        for (char c : s.toCharArray()) {
            if (!stack.isEmpty() && stack.peek() == 'a' && c
== 'b') {
                stack.pop();
                score += x;
            } else {
                stack.push(c);
            }
        }
        StringBuilder remaining = new StringBuilder();
        for (char c : stack) remaining.append(c);
        stack.clear();
```



Accepted 77 / 77 testcases passed
LAKSHIT MALHOTRA submitted at Apr 06, 2025 14:22

Runtime
143 ms | Beats 42.17%

Memory
45.56 MB | Beats 80.00%

Code | Java

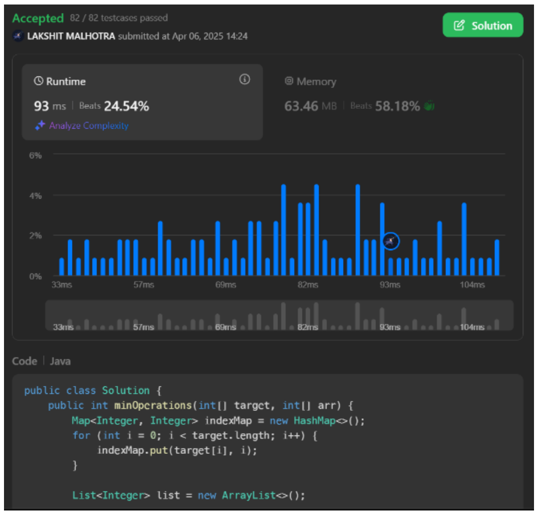
```
public class Solution {
    public int maximumGain(String s, int x, int y) {
        if (y > x) {
            return maximumGain(new StringBuilder(s).reverse().toString(), y, x);
        }

        int score = 0;
        Stack<Character> stack = new Stack<>();
```

```
 for (char c : remaining.toString().toCharArray()) {
            if (!stack.isEmpty() && stack.peek() == 'b' && c == 'a') {
                stack.pop();
                score += y;
            } else {
                stack.push(c);
            }
        }
        return score;
    }
}
```

1713 - MINIMUM OPERATIONS TO MAKE A SUBSEQUENCE

PROBLEM: https://leetcode.com/problems/minimum-operations-to-make-a-subsequence/

```
public class Solution {
    public int minOperations(int[] target, int[] arr) {
        Map<Integer, Integer> indexMap = new
HashMap<>();
        for (int i = 0; i < target.length; i++) {
            indexMap.put(target[i], i);
        }
        List<Integer> list = new ArrayList<>();
        for (int num : arr) {             if
(indexMap.containsKey(num)) {
                int idx = indexMap.get(num);
                int pos = Collections.binarySearch(list, idx);
                if (pos < 0) pos = -(pos + 1);
                if (pos == list.size()) list.add(idx);
                else list.set(pos, idx);
            }
        }
        return target.length - list.size();
    }
}
```



Accepted 82 / 82 testcases passed

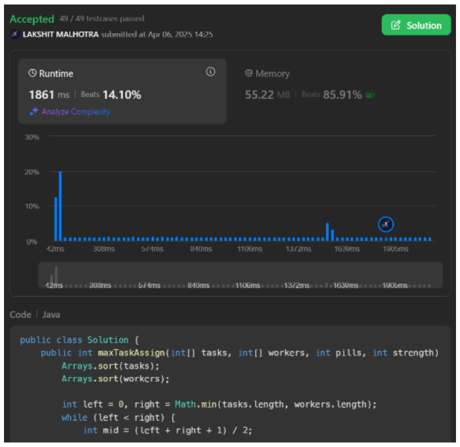
LAKSHIT MALHOTRA submitted at Apr 06, 2025 14:24

```
public class Solution {
    public int minOperations(int[] target, int[] arr) {
        Map<Integer, Integer> indexMap = new HashMap<>();
        for (int i = 0; i < target.length; i++) {
            indexMap.put(target[i], i);
        }

        List<Integer> list = new ArrayList<>();
```

2071 - MAXIMUM NUMBER OF TASKS YOU CAN ASSIGN

PROBLEM: https://leetcode.com/problems/maximum-number-of-tasks-you-can-assign/

```
public class Solution {
    public int maxTaskAssign(int[] tasks, int[] workers, int pills, int strength) {
        Arrays.sort(tasks);
        Arrays.sort(workers);
int left = 0, right = Math.min(tasks.length, workers.length);
        while (left < right) {
            int mid = (left + right + 1) / 2;
            if (canAssign(mid, tasks, workers, pills, strength)) {
                left = mid;
            } else {
                right = mid - 1;
            }
        }
        return left;
    }
    private boolean canAssign(int k, int[] tasks, int[] workers,
int pills, int strength) {
        TreeMap<Integer, Integer> taskMap = new TreeMap<>();
        for (int i = 0; i < k; i++) {
            taskMap.put(tasks[i], taskMap.getOrDefault(tasks[i], 0) + 1);
        }
        for (int i = workers.length - k; i < workers.length; i++) {
            Integer normal = taskMap.floorKey(workers[i]);
            if (normal != null) {
                taskMap.put(normal, taskMap.get(normal) - 1);
                if (taskMap.get(normal) == 0) taskMap.remove(normal);
                continue;
            }
            if (pills == 0) return false;
            Integer boosted = taskMap.floorKey(workers[i] + strength);
            if (boosted == null) return false;
            taskMap.put(boosted, taskMap.get(boosted) - 1);
            if (taskMap.get(boosted) == 0) taskMap.remove(boosted);
            pills--;
        }
        return true;
    }
}
```



Accepted 49 / 49 testcases passed
LAKSHIT MALHOTRA submitted at Apr 06, 2025 14:25

Runtime
1861 ms | Beats 14.10%

Memory
55.22 MB | Beats 85.91%

Code | Java

```
public class Solution {
    public int maxTaskAssign(int[] tasks, int[] workers, int pills, int strength)
        Arrays.sort(tasks);
        Arrays.sort(workers);

        int left = 0, right = Math.min(tasks.length, workers.length);
        while (left < right) {
            int mid = (left + right + 1) / 2;
```