Experiment 7

Student Name: Suraj Raj
UID: 22BCS15623
Branch: BE-CSE
Section/Group: 22BCS_IOT-638/A
Semester: 6th
Subject Code: 22CSP-351
Subject Name: AP Lab-II

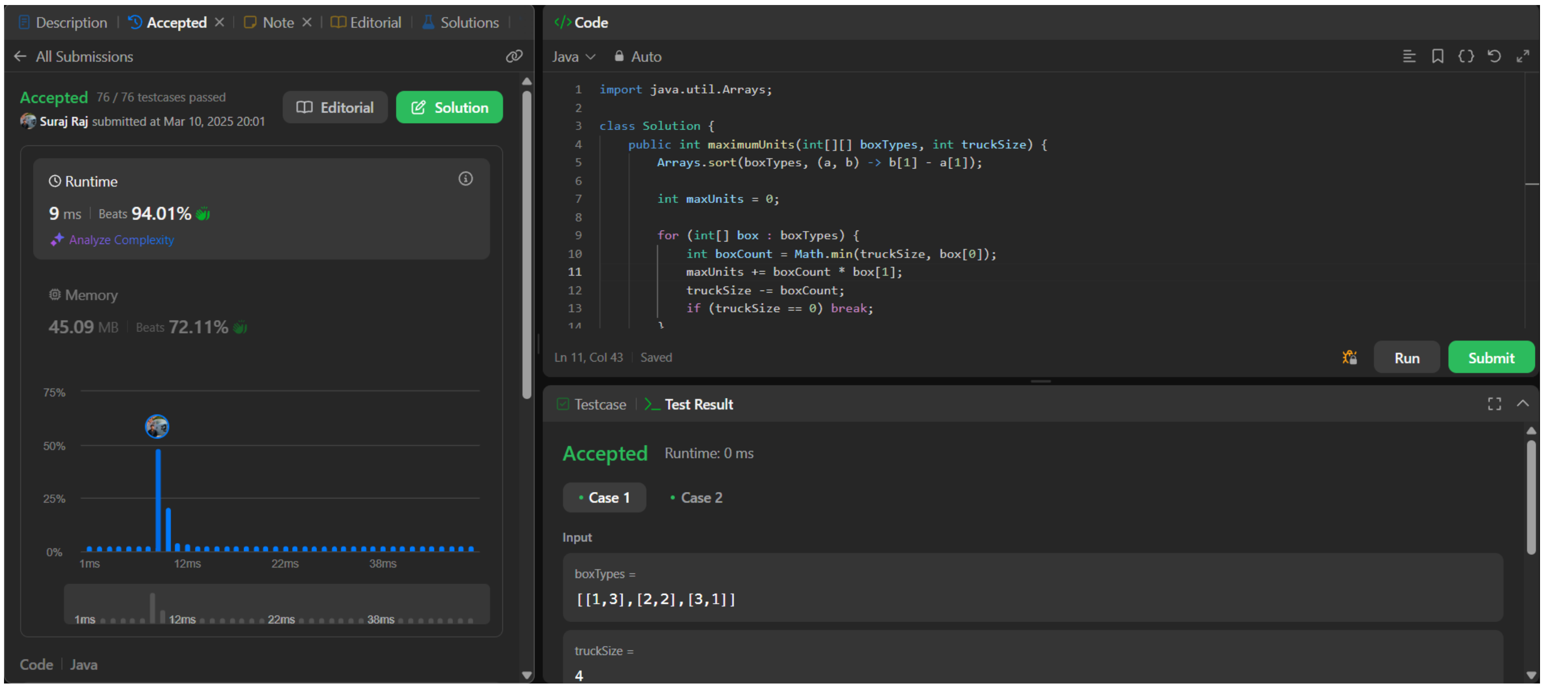
A. Maximum Units on a Truck

1. **Aim:** You are assigned to put some amount of boxes onto one truck. You are given a 2D array boxTypes, where boxTypes[i] = [numberOfBoxes$_i$, numberOfUnitsPerBox$_i$]:
   - numberOfBoxes is the number of boxes of type i.
   - numberOfUnitsPerBox$_i$ is the number of units in each box of the type i.

2. Code

```
import java.util.Arrays;

class Solution {
    public int maximumUnits(int[][] boxTypes, int truckSize) {
        Arrays.sort(boxTypes, (a, b) -> b[1] - a[1]);

        int maxUnits = 0;

        for (int[] box : boxTypes) {
            int boxCount = Math.min(truckSize, box[0]);
            maxUnits += boxCount * box[1];
            truckSize -= boxCount;
            if (truckSize == 0) break;
        }

        return maxUnits;
    }
}
```

3. Output:



```java
import java.util.Arrays;

class Solution {
    public int maximumUnits(int[][] boxTypes, int truckSize) {
        Arrays.sort(boxTypes, (a, b) -> b[1] - a[1]);

        int maxUnits = 0;

        for (int[] box : boxTypes) {
            int boxCount = Math.min(truckSize, box[0]);
            maxUnits += boxCount * box[1];
            truckSize -= boxCount;
            if (truckSize == 0) break;
        }
```

boxTypes =
[[1,3],[2,2],[3,1]]

truckSize =
4

4. **Link:** https://leetcode.com/problems/maximum-units-on-a-truck/submissions/1569186594/

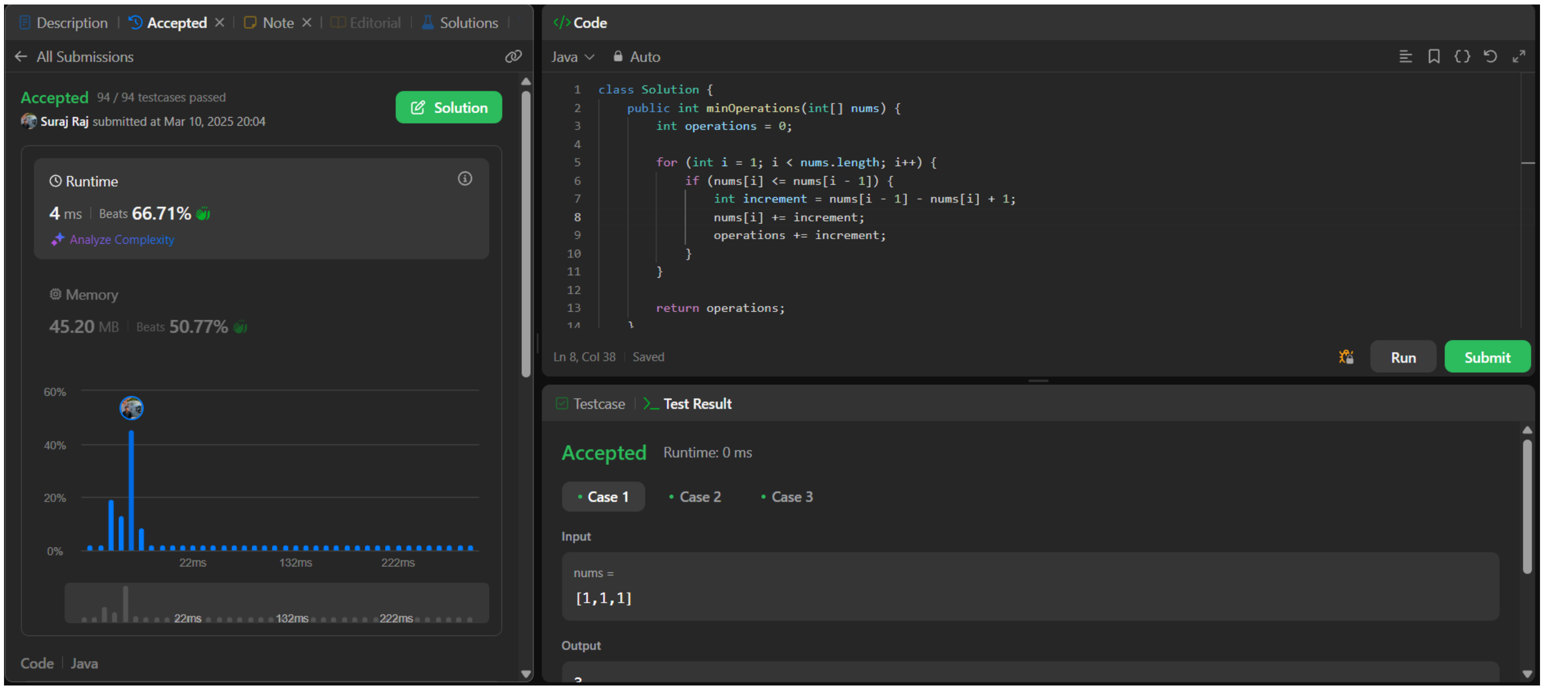
B. Minimum Operations to Make the Array Increasing

1. **Aim:** You are given an integer array nums (**0-indexed**). In one operation, you can choose an element of the array and increment it by 1.
    - For example, if nums = [1,2,3], you can choose to increment nums[1] to make nums = [1,**3**,3].
    - Return *the **minimum** number of operations needed to make* nums ***strictly increasing***.

2. **Code:**

```java
class Solution {
    public int minOperations(int[] nums) {
        int operations = 0;

        for (int i = 1; i < nums.length; i++) {
            if (nums[i] <= nums[i - 1]) {
                int increment = nums[i - 1] - nums[i] + 1;
                nums[i] += increment;
                operations += increment;
            }
        }

        return operations;
    }
}
```

3. Output:



Accepted 94 / 94 testcases passed
Suraj Raj submitted at Mar 10, 2025 20:04

Runtime 4 ms | Beats 66.71%

Memory 45.20 MB | Beats 50.77%

```
class Solution {
    public int minOperations(int[] nums) {
        int operations = 0;

        for (int i = 1; i < nums.length; i++) {
            if (nums[i] <= nums[i - 1]) {
                int increment = nums[i - 1] - nums[i] + 1;
                nums[i] += increment;
                operations += increment;
            }
        }

        return operations;
```

Ln 8, Col 38 | Saved

Accepted Runtime: 0 ms

Input

nums =
[1,1,1]

Output

4. Link : https://leetcode.com/problems/minimum-operations-to-make-the-array-increasing/submissions/1569189850/

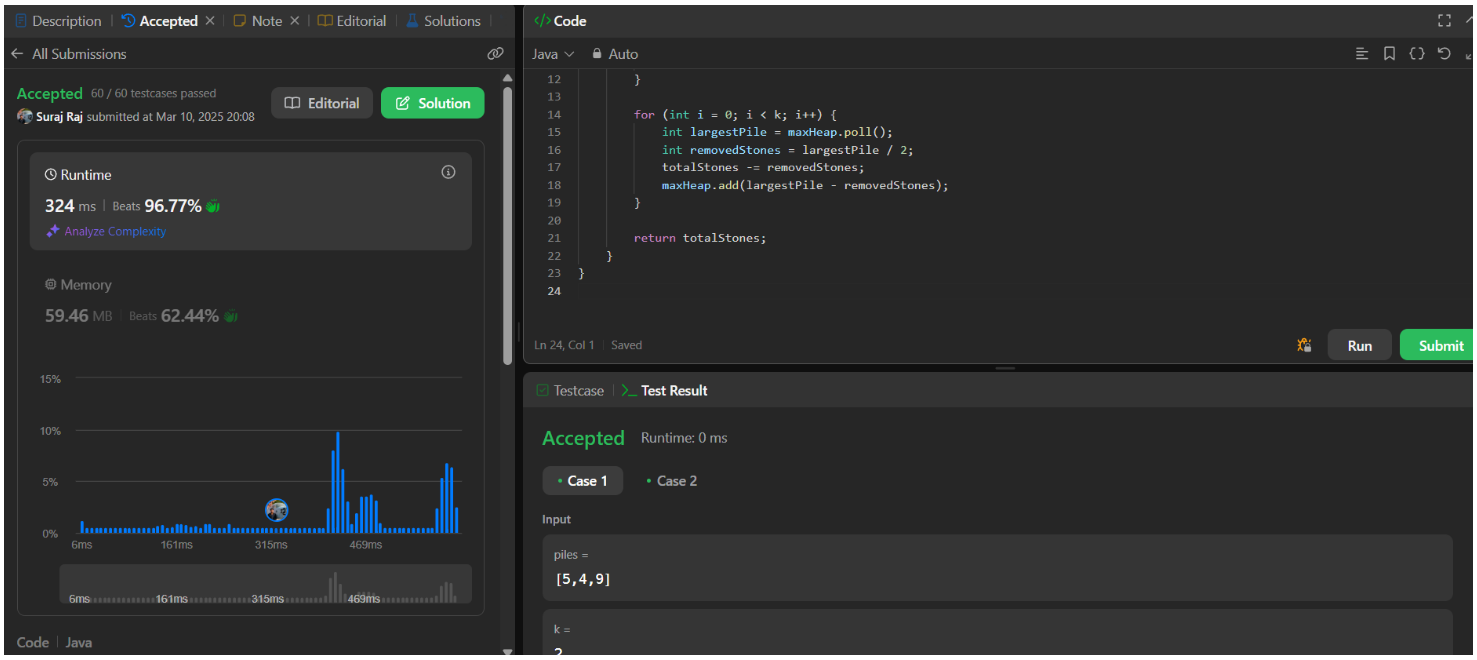
C. Remove Stones to Minimize the Total

1. Aim: You are given a 0-indexed integer array piles, where piles[i] represents the number of stones in the ith pile, and an integer k. You should apply the following operation exactly k times:

2. Code:

```
import java.util.PriorityQueue;

class Solution {
    public int minStoneSum(int[] piles, int k) {
        PriorityQueue<Integer> maxHeap = new PriorityQueue<>((a, b) -> b - a);

        int totalStones = 0;

        for (int pile : piles) {
            maxHeap.add(pile);
            totalStones += pile;
        }

        for (int i = 0; i < k; i++) {
            int largestPile = maxHeap.poll();
            int removedStones = largestPile / 2;
            totalStones -= removedStones;
            maxHeap.add(largestPile - removedStones);
        }

        return totalStones;
    }
}
```

3. Output:



Accepted 60 / 60 testcases passed
Suraj Raj submitted at Mar 10, 2025 20:08

Runtime 324 ms | Beats 96.77%

Memory 59.46 MB | Beats 62.44%

```
        }

        for (int i = 0; i < k; i++) {
            int largestPile = maxHeap.poll();
            int removedStones = largestPile / 2;
            totalStones -= removedStones;
            maxHeap.add(largestPile - removedStones);
        }

        return totalStones;
    }
}
```

Accepted Runtime: 0 ms

Input
piles =
[5,4,9]
k =
2

4. Link: https://leetcode.com/problems/remove-stones-to-minimize-the-total/submissions/1569193429/