Experiment 8

Student Name: Harsh. **UID: 22BCS12488**
Branch: CSE **Section/Group: NTPP 603B**
Semester: 6 **Date of Performance:21 /03/25**
Subject Name: AP Lab 2 **Subject Code:22CSP-351**

1. Aim:

a. Max Units on a Truck
b. Minimum Operations to Make Array Increasing
c. Maximum Score from Removing Substrings
d. Minimum Operations to Make a Subsequence

2. Code:

a.
```java
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) {
            if (truckSize <= 0) break;
            int count = Math.min(box[0], truckSize);
            maxUnits += count * box[1];
            truckSize -= count;
        }

        return maxUnits;
    }
}
```

b.
```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]) {
                operations += nums[i - 1] - nums[i] + 1;
                nums[i] = nums[i - 1] + 1;
            }
        }

        return operations;
    }
}
```

c.

```
class Solution {
    public int maximumGain(String s, int x, int y) {
         int points = 0;

        // First, remove the higher-point substring
        if (x >= y) {
            points += removeSubstring(s, "ab", x);
            s = removeSubstringReturnString(s, "ab");
            points += removeSubstring(s, "ba", y);
        } else {
            points += removeSubstring(s, "ba", y);
            s = removeSubstringReturnString(s, "ba");
            points += removeSubstring(s, "ab", x);
        }

        return points;
    }

    // Removes the target substring and returns the updated
string
    private String removeSubstringReturnString(String s, String
target) {
        StringBuilder sb = new StringBuilder(s);
        int index = sb.indexOf(target);
        while (index != -1) {
            sb.delete(index, index + target.length());
            index = sb.indexOf(target);
        }
        return sb.toString();
    }
    // Count the number of times a substring can be removed and
adds points
    private int removeSubstring(String s, String target, int
points) {
        int totalPoints = 0;
        StringBuilder sb = new StringBuilder(s);

        int index = sb.indexOf(target);
        while (index != -1) {
            sb.delete(index, index + target.length());
            totalPoints += points;
            index = sb.indexOf(target);
        }

        return totalPoints;
    }
}
```

d.
```
class Solution {

    public int minOperations(int[] target, int[] arr) {

        Map<Integer, Integer> targetIndexMap = new HashMap<>();
        for (int i = 0; i < target.length; i++) {
            targetIndexMap.put(target[i], i);
        }


        List<Integer> transformedArr = new ArrayList<>();
        for (int num : arr) {
            if (targetIndexMap.containsKey(num)) {
                transformedArr.add(targetIndexMap.get(num));
            }
        }


        return target.length - lengthOfLIS(transformedArr);
    }

    private int lengthOfLIS(List<Integer> nums) {
        if (nums.isEmpty()) return 0;

        List<Integer> lis = new ArrayList<>();
        for (int num : nums) {
            int pos = binarySearch(lis, num);
            if (pos < lis.size()) {
                lis.set(pos, num);
            } else {
                lis.add(num);
            }
        }

        return lis.size();
    }

    private int binarySearch(List<Integer> lis, int target) {
        int left = 0, right = lis.size();
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (lis.get(mid) < target) {
                left = mid + 1;
            } else {
                right = mid;
            }
        }
        return left;
    }
}
```

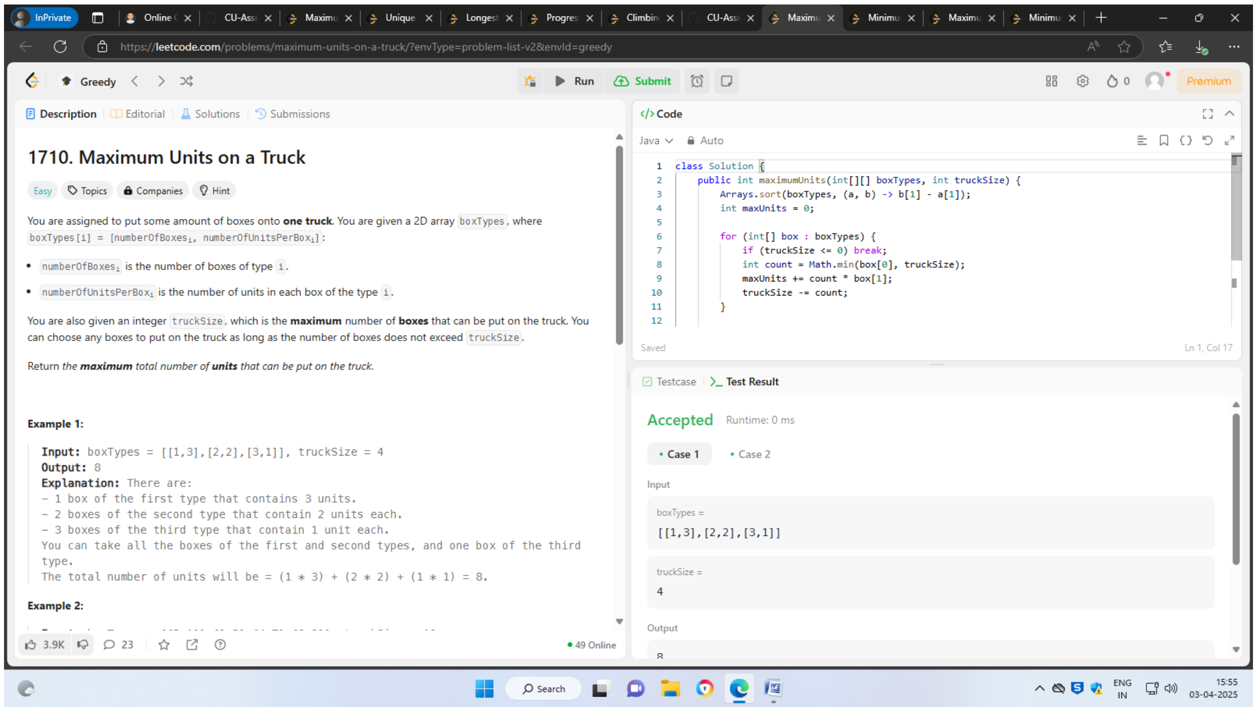
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

3. Output:

a.



https://leetcode.com/problems/maximum-units-on-a-truck/?envType=problem-list-v2&envId=greedy

1710. Maximum Units on a Truck

Easy

You are assigned to put some amount of boxes onto **one truck**. You are given a 2D array `boxTypes`, where `boxTypes[i] = [numberOfBoxesi, numberOfUnitsPerBoxi]`:

- `numberOfBoxesi` is the number of boxes of type `i`.
- `numberOfUnitsPerBoxi` is the number of units in each box of the type `i`.

You are also given an integer `truckSize`, which is the **maximum** number of **boxes** that can be put on the truck. You can choose any boxes to put on the truck as long as the number of boxes does not exceed `truckSize`.

Return *the **maximum** total number of **units** that can be put on the truck.*

Example 1:

```
Input: boxTypes = [[1,3],[2,2],[3,1]], truckSize = 4
Output: 8
Explanation: There are:
- 1 box of the first type that contains 3 units.
- 2 boxes of the second type that contain 2 units each.
- 3 boxes of the third type that contain 1 unit each.
You can take all the boxes of the first and second types, and one box of the third
type.
The total number of units will be = (1 * 3) + (2 * 2) + (1 * 1) = 8.
```

Example 2:

```java
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) {
            if (truckSize <= 0) break;
            int count = Math.min(box[0], truckSize);
            maxUnits += count * box[1];
            truckSize -= count;
        }

```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

boxTypes =

```
[[1,3],[2,2],[3,1]]
```

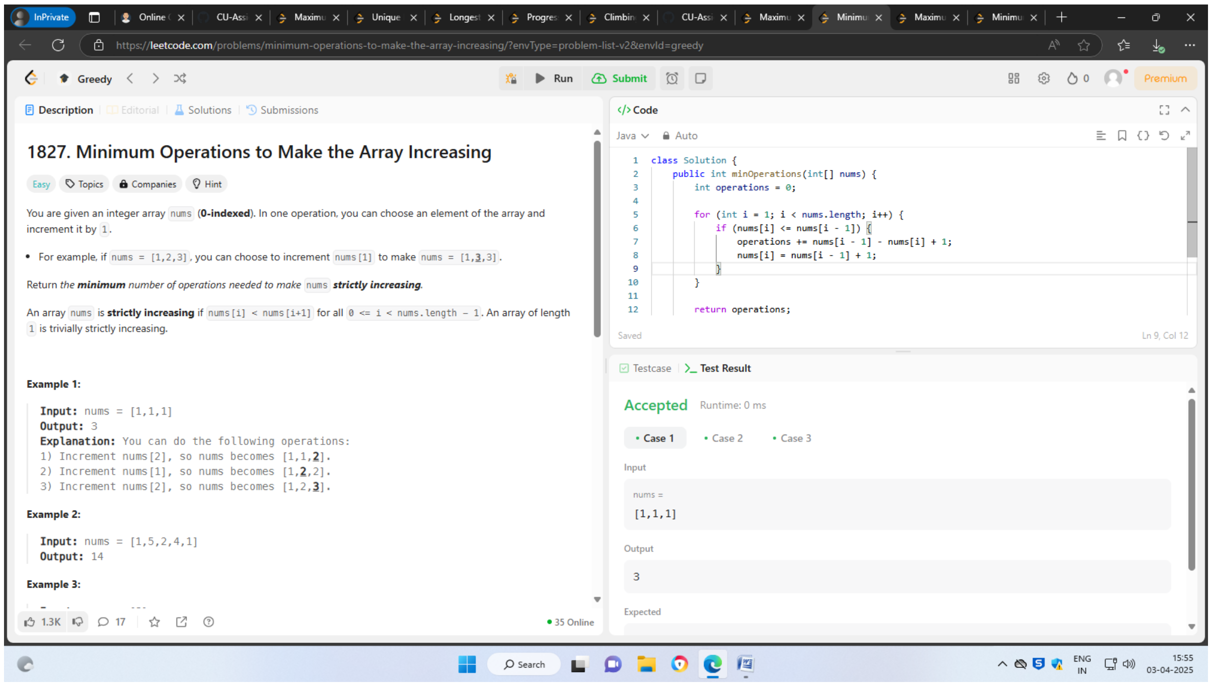
truckSize =

```
4
```

Output

```
8
```

b.



https://leetcode.com/problems/minimum-operations-to-make-the-array-increasing/?envType=problem-list-v2&envId=greedy

1827. Minimum Operations to Make the Array Increasing

Easy

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**.*

An array `nums` is **strictly increasing** if `nums[i] < nums[i+1]` for all `0 <= i < nums.length - 1`. An array of length `1` is trivially strictly increasing.

Example 1:

```
Input: nums = [1,1,1]
Output: 3
Explanation: You can do the following operations:
1) Increment nums[2], so nums becomes [1,1,2].
2) Increment nums[1], so nums becomes [1,2,2].
3) Increment nums[2], so nums becomes [1,2,3].
```

Example 2:

```
Input: nums = [1,5,2,4,1]
Output: 14
```

Example 3:

```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]) {
                operations += nums[i - 1] - nums[i] + 1;
                nums[i] = nums[i - 1] + 1;
            }
        }

        return operations;
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =

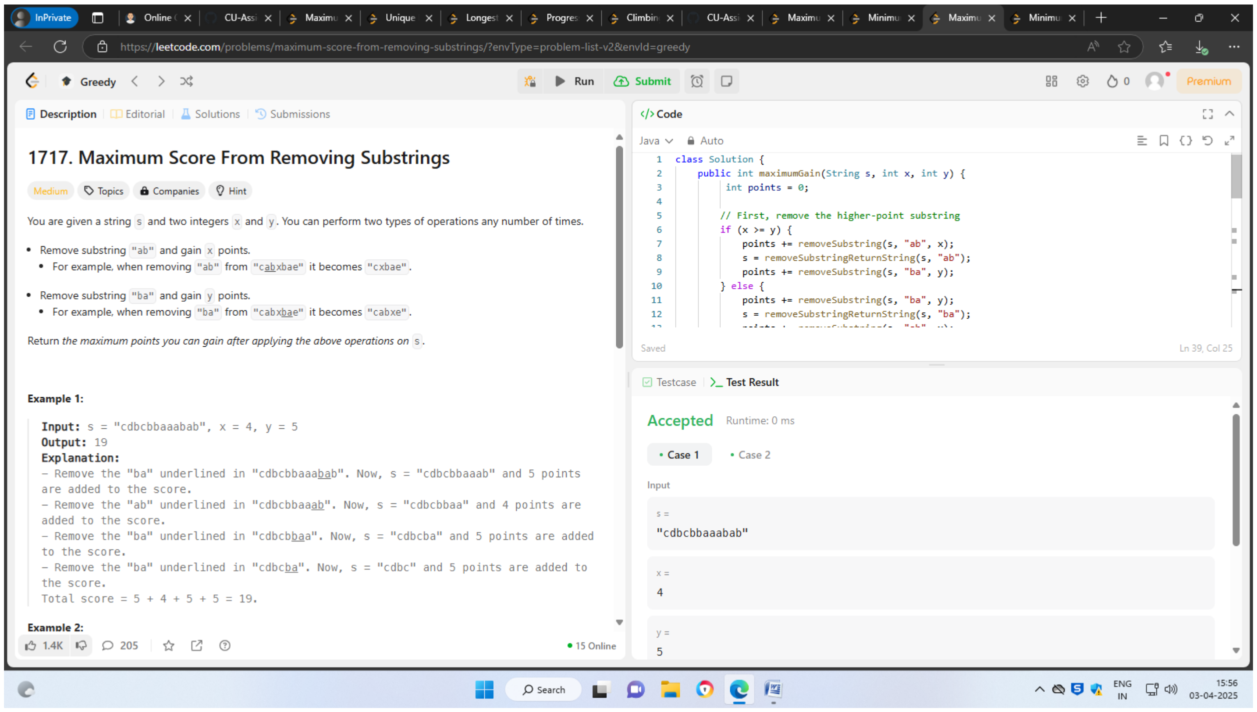
```
[1,1,1]
```

Output

```
3
```

Expected

c.



https://leetcode.com/problems/maximum-score-from-removing-substrings/?envType=problem-list-v2&envId=greedy

1717. Maximum Score From Removing Substrings

Medium · Topics · Companies · Hint

You are given a string `s` and two integers `x` and `y`. You can perform two types of operations any number of times.

- Remove substring `"ab"` and gain `x` points.
  - For example, when removing `"ab"` from `"cabxbae"` it becomes `"cxbae"`.
- Remove substring `"ba"` and gain `y` points.
  - For example, when removing `"ba"` from `"cabxbae"` it becomes `"cabxe"`.

Return *the maximum points you can gain after applying the above operations on* `s`.

Example 1:

```
Input: s = "cdbcbbaaabab", x = 4, y = 5
Output: 19
Explanation:
– Remove the "ba" underlined in "cdbcbbaaabab". Now, s = "cdbcbbaaab" and 5 points
are added to the score.
– Remove the "ab" underlined in "cdbcbbaaab". Now, s = "cdbcbbaa" and 4 points are
added to the score.
– Remove the "ba" underlined in "cdbcbbaa". Now, s = "cdbcba" and 5 points are added
to the score.
– Remove the "ba" underlined in "cdbcba". Now, s = "cdbc" and 5 points are added to
the score.
Total score = 5 + 4 + 5 + 5 = 19.
```

Example 2:

1.4K · 205 · 15 Online

Code — Java

```java
class Solution {
    public int maximumGain(String s, int x, int y) {
        int points = 0;

        // First, remove the higher-point substring
        if (x >= y) {
            points += removeSubstring(s, "ab", x);
            s = removeSubstringReturnString(s, "ab");
            points += removeSubstring(s, "ba", y);
        } else {
            points += removeSubstring(s, "ba", y);
            s = removeSubstringReturnString(s, "ba");
```

Saved · Ln 39, Col 25

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

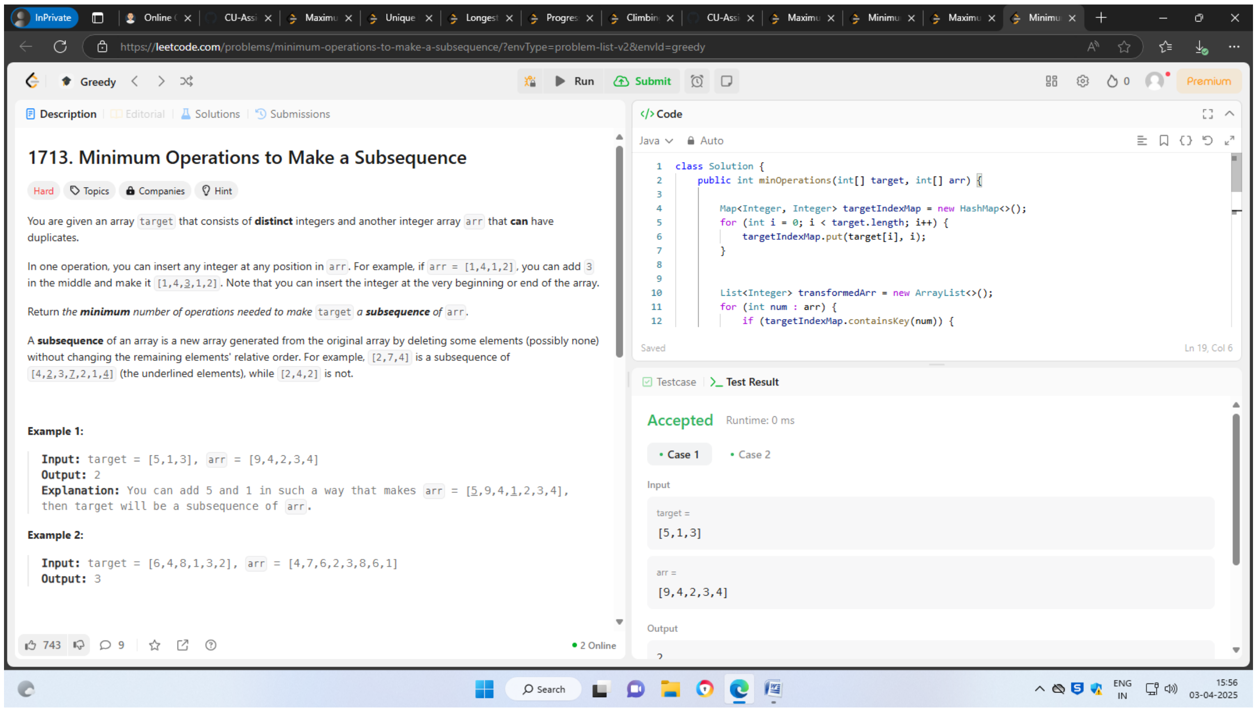
s =
"cdbcbbaaabab"

x =
4

y =
5

15:56 03-04-2025

d.



https://leetcode.com/problems/minimum-operations-to-make-a-subsequence/?envType=problem-list-v2&envId=greedy

1713. Minimum Operations to Make a Subsequence

Hard · Topics · Companies · Hint

You are given an array `target` that consists of **distinct** integers and another integer array `arr` that **can** have duplicates.

In one operation, you can insert any integer at any position in `arr`. For example, if `arr = [1,4,1,2]`, you can add `3` in the middle and make it `[1,4,3,1,2]`. Note that you can insert the integer at the very beginning or end of the array.

Return *the **minimum** number of operations needed to make* `target` *a **subsequence** of* `arr`.

A **subsequence** of an array is a new array generated from the original array by deleting some elements (possibly none) without changing the remaining elements' relative order. For example, `[2,7,4]` is a subsequence of `[4,2,3,7,2,1,4]` (the underlined elements), while `[2,4,2]` is not.

Example 1:

```
Input: target = [5,1,3], arr = [9,4,2,3,4]
Output: 2
Explanation: You can add 5 and 1 in such a way that makes arr = [5,9,4,1,2,3,4],
then target will be a subsequence of arr.
```

Example 2:

```
Input: target = [6,4,8,1,3,2], arr = [4,7,6,2,3,8,6,1]
Output: 3
```

743 · 9 · 2 Online

Code — Java

```java
class Solution {
    public int minOperations(int[] target, int[] arr) {

        Map<Integer, Integer> targetIndexMap = new HashMap<>();
        for (int i = 0; i < target.length; i++) {
            targetIndexMap.put(target[i], i);
        }


        List<Integer> transformedArr = new ArrayList<>();
        for (int num : arr) {
            if (targetIndexMap.containsKey(num)) {
```

Saved · Ln 19, Col 6

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

target =
[5,1,3]

arr =
[9,4,2,3,4]

Output

2

15:56 03-04-2025