CU CHANDIGARH UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

WORKSHEET 8

Student Name: ANSH RAJ **UID:20BCS3434**

Branch: CSE **Section/Group: NTPP 603/B**

Semester: 06 **Date of Performance: 20/03/2025**

Subject Name: AP Lab II **Subject Code: 22CSP-351**

1. Aim:

a) Max Units on a Truck
Min Operations to Make Array Increasing **b)**
c) Remove Stones to Maximize Total

2. Source Code:

a.

```
class Solution {
public:
  int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
int ans = 0;
     ranges::sort(boxTypes,
ranges::greater{},
                 [](const vector<int>& boxType) { return boxType[1]; });
     for (const vector<int>& boxType : boxTypes)
{       const int boxes = boxType[0];       const
int units = boxType[1];       if (boxes >=
truckSize)         return ans + truckSize *
units;       ans += boxes * units;
truckSize -= boxes;
    }
    return ans;
  }
};
```

b.

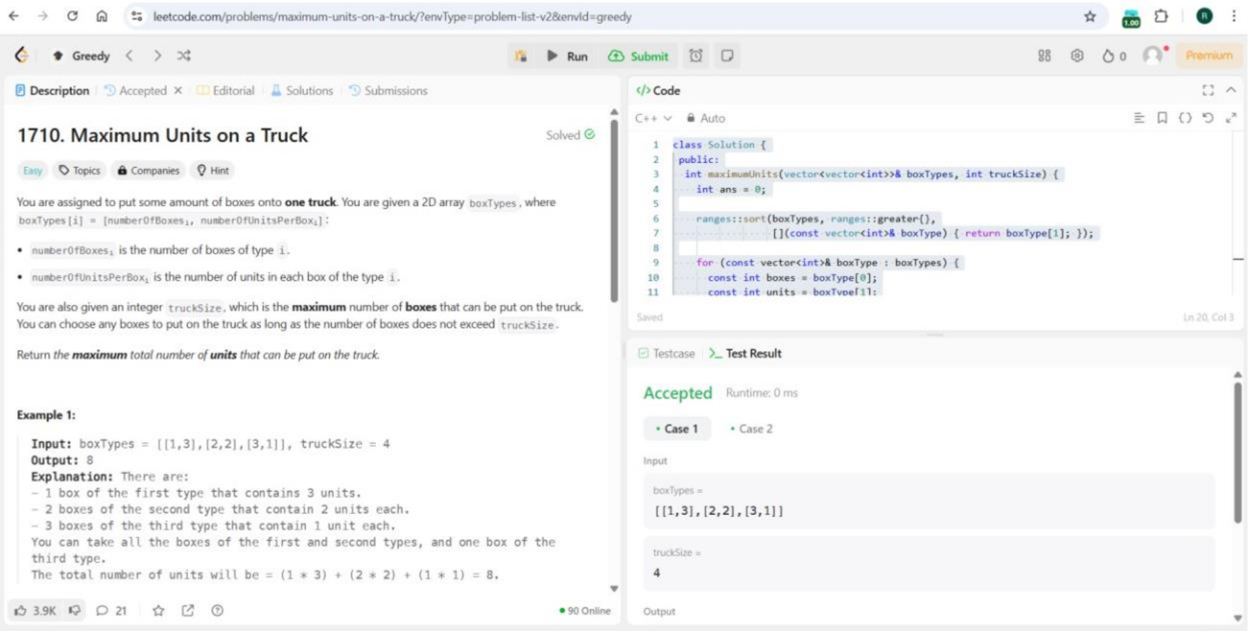
```
class Solution {
public:
  int minOperations(vector<int>& nums) {
int ans = 0;     int last = 0;
     for (const int num : nums) {
ans += max(0, last - num + 1);
last = max(num, last + 1);
    }
return ans;
  }
};
```

C.

```
class Solution {
public:
  int minStoneSum(vector<int>& piles, int k) {     int
ans = accumulate(piles.begin(), piles.end(), 0);
priority_queue<int> maxHeap;
     for (const int pile :
piles)       maxHeap.push(pile);
     for (int i = 0; i < k; ++i) {
const int maxPile = maxHeap.top();
maxHeap.pop();
      maxHeap.push(maxPile - maxPile / 2);
ans -= maxPile / 2;
    }
return ans;
  }
};
```

Screenshot of Outputs:

a.



1710. Maximum Units on a Truck

Solved

Easy · Topics · Companies · Hint

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`$_i$ is the number of boxes of type `i`.
- `numberOfUnitsPerBox`$_i$ 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.
```

```cpp
class Solution {
public:
  int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
    int ans = 0;

    ranges::sort(boxTypes, ranges::greater{},
                 [](const vector<int>& boxType) { return boxType[1]; });

    for (const vector<int>& boxType : boxTypes) {
      const int boxes = boxType[0];
      const int units = boxType[1];
```

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

boxTypes =

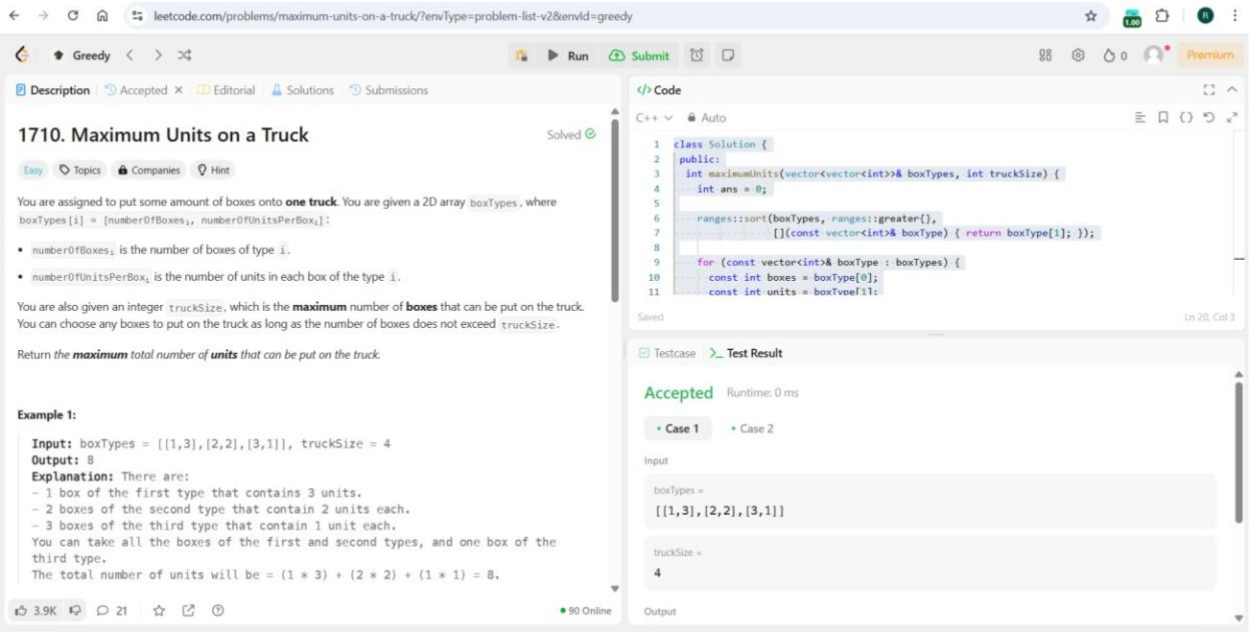
```
[[1,3],[2,2],[3,1]]
```

truckSize =

```
4
```

Output

b.


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

1710. Maximum Units on a Truck

Solved

Easy · Topics · Companies · Hint

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`$_i$ is the number of boxes of type `i`.
- `numberOfUnitsPerBox`$_i$ 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.
```

```cpp
class Solution {
public:
  int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
    int ans = 0;

    ranges::sort(boxTypes, ranges::greater{},
                 [](const vector<int>& boxType) { return boxType[1]; });

    for (const vector<int>& boxType : boxTypes) {
      const int boxes = boxType[0];
      const int units = boxType[1];
```

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

boxTypes =

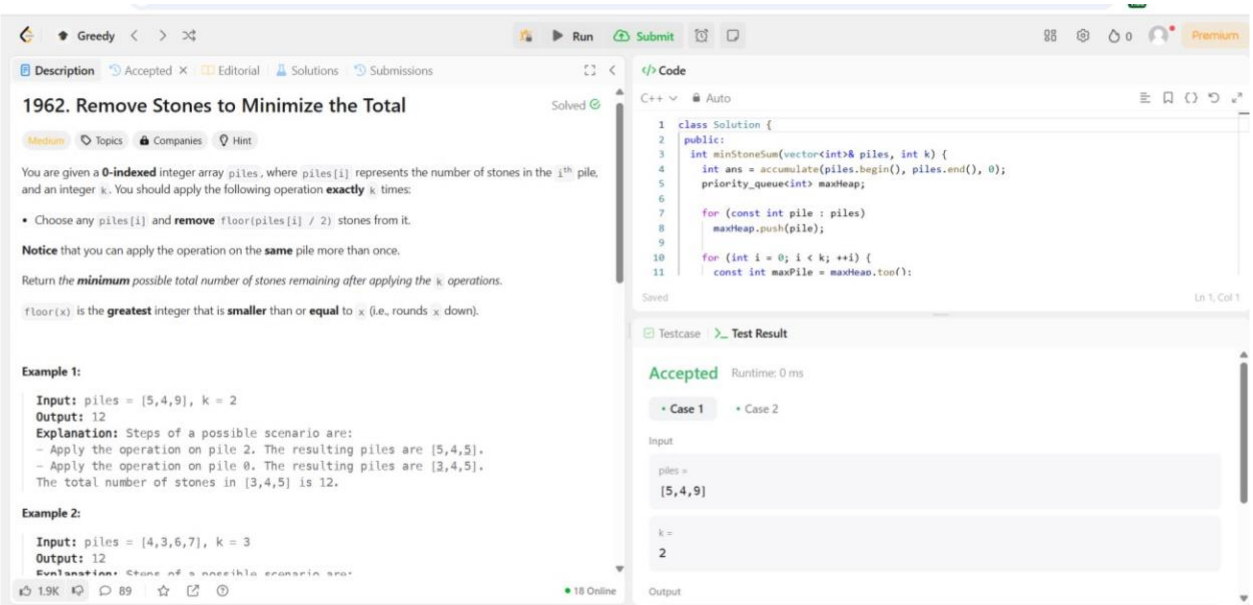
```
[[1,3],[2,2],[3,1]]
```

truckSize =

```
4
```

Output

c.



1962. Remove Stones to Minimize the Total

Solved

Medium · Topics · Companies · Hint

You are given a **0-indexed** integer array `piles`, where `piles[i]` represents the number of stones in the i^{th} pile, and an integer `k`. You should apply the following operation **exactly** `k` times:

- Choose any `piles[i]` and **remove** `floor(piles[i] / 2)` stones from it.

Notice that you can apply the operation on the **same** pile more than once.

Return *the **minimum** possible total number of stones remaining after applying the* `k` *operations.*

`floor(x)` is the **greatest** integer that is **smaller** than or **equal** to `x` (i.e., rounds `x` down).

Example 1:

```
Input: piles = [5,4,9], k = 2
Output: 12
Explanation: Steps of a possible scenario are:
- Apply the operation on pile 2. The resulting piles are [5,4,5].
- Apply the operation on pile 0. The resulting piles are [3,4,5].
The total number of stones in [3,4,5] is 12.
```

Example 2:

```
Input: piles = [4,3,6,7], k = 3
Output: 12
Explanation: Steps of a possible scenario are:
```

```
class Solution {
public:
  int minStoneSum(vector<int>& piles, int k) {
    int ans = accumulate(piles.begin(), piles.end(), 0);
    priority_queue<int> maxHeap;

    for (const int pile : piles)
      maxHeap.push(pile);

    for (int i = 0; i < k; ++i) {
      const int maxPile = maxHeap.top();
```

Saved

Ln 1, Col 1

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

```
piles =
[5,4,9]
```

```
k =
2
```

Output

3. Learning Outcomes

(i) Learned about Greedy Programming.