

Name =Raja Babu

UID: 22BCS11971

Sec: 614-B

AP-Experiment-8

1. Max Units on a Truck

The screenshot shows a LeetCode submission for the problem "Maximum Units on a Truck". The submission is in C++ and has been accepted. The runtime is 3 ms, which beats 67.22% of other submissions. The memory usage is 20.88 MB, which beats 25.73% of other submissions. The code is a greedy solution that sorts the boxes by the number of units per box in descending order and then picks boxes until the truck is full.

Accepted 76 / 76 testcases passed
Submitted at Apr 17, 2025 09:42

Runtime 3 ms | Beats 67.22%
Memory 20.88 MB | Beats 25.73%

```
class Solution {
public:
    static bool myfunction(vector<int>& a, vector<int>& b){
        return a[1] > b[1];
    }
};
```

```
int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
    //custom sort (in increasing order of numberOfUnitsPerBox as we have to return maximum
    total number of units )
    sort(boxTypes.begin(), boxTypes.end(), myfunction);
    //greedily pick boxes till capacity is full
    int ans=0;
    for(auto box: boxTypes){
        int x=min(box[0], truckSize); //choose minimum boxes from available boxes and capacity
        left
        ans+=x*box[1]; //adding units in ans
        truckSize-=x; //reduce the capacity
        if(truckSize==0) break; //capacity full
    }
    return ans;
}
```

2. Min Operations to Make Array Increasing

The screenshot shows a LeetCode submission for the problem "Minimum Operations to Make the Array Increasing". The submission is in C++ and has been accepted. The runtime is 3 ms. The code is a greedy solution that iterates through the array and increments elements that are not greater than the previous element.

1827. Minimum Operations to Make the Array Increasing Solved

Easy Topics Companies Hint

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 $0 \leq i < \text{nums.length} - 1$. 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]`

```
class Solution {
public:
    int minOperations(vector<int>& nums) {
        int counter = 0;
        for(int i = 0; i < nums.size() - 1; i++)
        {
            while(nums[i] >= nums[i+1])
            {
                nums[i+1]++;
                counter++;
            }
        }
        return counter;
    }
};
```

3. Remove Stones to Maximize Total

162. Remove Stones to Minimize the Total

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** $\text{floor}(\text{piles}[i] / 2)$ 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.

$\text{floor}(x)$ 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

```
C++  
int minStoneSum(vector<int>& piles, int k) {  
    priority_queue<int> pq(piles.begin(), piles.end());  
    int ans = accumulate(piles.begin(), piles.end(), 0);  
    while(k-- && !pq.empty())  
    {  
        int temp = pq.top();  
        pq.pop();  
        ans -= (temp / 2);  
        pq.push(temp / 2);  
        k--;  
    }  
    return ans;  
}
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

`piles = [5, 4, 9]`

4. Max Score from Removing Substrings

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 = "cdbcbbbaaab"`, `x = 4`, `y = 5`
Output: 19
Explanation:
- Remove the `"ba"` underlined in `"cdbcbbbaaab"`. Now, `s = "cdbcbbbaab"` and 5 points are added to the score.
- Remove the `"ab"` underlined in `"cdbcbbbaab"`. Now, `s = "cdbcbbbaa"` and 4 points are added to the score.
- Remove the `"ba"` underlined in `"cdbcbbbaa"`. Now, `s = "cdbcbaa"` and 5 points are added to the score.
- Remove the `"ba"` underlined in `"cdbcbaa"`. Now, `s = "cdbc"` and 5 points are added to the score.

```
C++  
int maximumScore(string s, int x, int y) {  
    int result = 0;  
    int aCount = 0, bCount = 0;  
    for (int i = 0; i < s.size(); i++)  
    {  
        if (s[i] == 'a') aCount++;  
        else if (s[i] == 'b') bCount++;  
        else if (s[i] == 'x' && aCount > 0)  
        {  
            aCount--;  
            result += x;  
        }  
        else if (s[i] == 'y' && bCount > 0)  
        {  
            bCount--;  
            result += y;  
        }  
    }  
    result += min(aCount, bCount) * lesser;  
    return result;  
}
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

`s = "cdbcbbbaaab"`