

AP 8TH EXPERIMENT

Q1. Max Units on a Truck

Code:

```
class Solution {
public:
    int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
        sort(boxTypes.begin(), boxTypes.end(), [](auto &a, auto &b) {
            return a[1] > b[1];
        });
        int totalUnits = 0;
        for (auto& box : boxTypes) {
            int countOfBoxes = min(truckSize, box[0]);
            totalUnits += countOfBoxes * box[1];
            truckSize -= countOfBoxes;
            if (truckSize == 0) break;
        }
        return totalUnits;
    }
};
```

Output:

The screenshot displays a coding platform interface with two main panels. The left panel shows the submission status as 'Accepted' with 76/76 testcases passed, submitted by 'bansalch' on Apr 10, 2025 at 22:43. It includes tabs for 'Editorial' and 'Solution'. The 'Runtime' section shows 0 ms and 100.00% efficiency. The 'Memory' section shows 19.75 MB and 89.01% efficiency. A bar chart at the bottom shows the user's performance relative to others. The right panel shows the test case details for 'Case 1'. The input is 'boxTypes = [[1,3],[2,2],[3,1]]' and 'truckSize = 4'. The output is '8', which matches the expected result '8'. A 'Contribute a testcase' link is at the bottom.

Accepted 76 / 76 testcases passed
bansalch submitted at Apr 10, 2025 22:43

Editorial Solution

Runtime
0 ms | Beats 100.00%
Analyze Complexity

Memory
19.75 MB | Beats 89.01%

40%
20%

Accepted runtime: 0 ms

Case 1 Case 2

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

truckSize =
4

Output
8

Expected
8

Contribute a testcase

Q2. Min Operations to Make Array Increasing

Code:

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

Output:

The screenshot shows a submission on a coding platform. On the left, the 'Runtime' section indicates 6 ms and 88.57% beats, while the 'Memory' section shows 19.62 MB and 26.96% beats. A bar chart at the bottom left compares the user's performance with other users. On the right, the 'Accepted' status is shown with a runtime of 3 ms. Below this, three test cases are listed: Case 1, Case 2, and Case 3. Case 1 is selected, showing an input array [1, 1, 1] and an output of 3. The expected output is also 3. A 'Contribute a testcase' link is visible at the bottom right.

Accepted 94 / 94 testcases passed
bansalch submitted at Apr 10, 2025 22:44

Runtime
6 ms | Beats 88.57%
[Analyze Complexity](#)

Memory
19.62 MB | Beats 26.96%

Accepted runtime: 3 ms

Case 1 Case 2 Case 3

Input
nums =
[1,1,1]

Output
3

Expected
3

[Contribute a testcase](#)

Q3. Remove Stones to Maximize Total

Code:

```
class Solution {
public:
    int minStoneSum(vector<int>& piles, int k) {
        priority_queue<int> maxHeap(piles.begin(), piles.end());
        while (k-- > 0) {
            int largest = maxHeap.top();
            maxHeap.pop();
            maxHeap.push(largest - largest / 2);
        }
        int total = 0;
        while (!maxHeap.empty()) {
            total += maxHeap.top();
            maxHeap.pop();
        }
        return total;
    }
};
```

Output:

Accepted 60 / 60 testcases passed
bansalch submitted at Apr 10, 2025 22:46

Editorial Solution

Runtime
281 ms | Beats: 59.43%
[Analyze Complexity](#)

Memory
102.68 MB | Beats: 95.86%

Accepted Runtime: 0 ms

Case 1 Case 2

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

Output
12

Expected
12

[Contribute a testcase](#)

Q4. Max Score from Removing Substrings

Code:

```
int maximumGain(string s, int x, int y) {
    if (x < y) {
        swap(x, y);
        for (char &c : s) {
            if (c == 'a') c = 'b';
            else if (c == 'b') c = 'a';
        }
    }
    int total = 0;
    stack<char> st;
    for (char c : s) {
        if (!st.empty() && st.top() == 'a' && c == 'b') {
            st.pop();
            total += x;
        } else {
            st.push(c);
        }
    }
    string remaining;
    while (!st.empty()) {
        remaining += st.top();
        st.pop();
    }
    reverse(remaining.begin(), remaining.end());
    for (char c : remaining) {
        if (!st.empty() && st.top() == 'b' && c == 'a') {
            st.pop();
            total += y;
        } else {
            st.push(c);
        }
    }
    return total;
}
```

Output:

Accepted 77 / 77 testcases passed
bansalch submitted at Apr 10, 2025 22:48

Editorial Solution

Runtime
61 ms | Beats: 31.67%
[Analyze Complexity](#)

Memory
28.80 MB | Beats: 30.83%

40%
20%

Input
s =
"cdbcbbaaabab"
x =
4
y =
5

Output
19

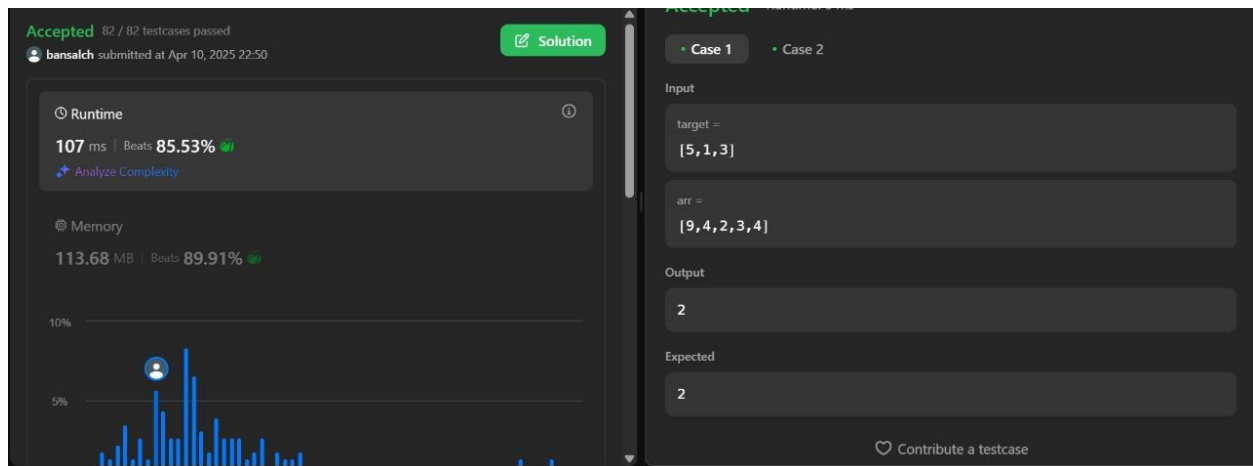
Expected
19

Q5. Min Operations to Make a Subsequence

Code:

```
class Solution {
public:
    int minOperations(vector<int>& target, vector<int>& arr) {
        unordered_map<int, int> indexMap;
        for (int i = 0; i < target.size(); ++i) {
            indexMap[target[i]] = i;
        }
        vector<int> seq;
        for (int num : arr) {
            if (indexMap.find(num) != indexMap.end()) {
                int idx = indexMap[num];
                auto it = lower_bound(seq.begin(), seq.end(), idx);
                if (it == seq.end()) {
                    seq.push_back(idx);
                } else {
                    *it = idx;
                }
            }
        }
        return target.size() - seq.size();
    }
};
```

Output:



Q6. Max Number of Tasks You Can Assign

Code:

```
    pills, int strength) {
        deque<int> dq(workers.end() - k, workers.end());
        int pillCount = pills;
        for (int i = k - 1; i >= 0; --i) {
            int task = tasks[i];
            if (!dq.empty() && dq.back() >= task) {
                dq.pop_back();
            } else if (!dq.empty() && pillCount > 0 && dq.front() + strength
>= task) {
                dq.pop_front();
                --pillCount;
            } else {
                return false;
            }
        }
        return true;
    }

    int maxTaskAssign(vector<int>& tasks, vector<int>& workers, int pills,
int strength) {
        sort(tasks.begin(), tasks.end());
        sort(workers.begin(), workers.end());
        int low = 0, high = min((int)tasks.size(), (int)workers.size());
        int result = 0;
        while (low <= high) {
            int mid = (low + high) / 2;
            if (canAssign(mid, tasks, workers, pills, strength)) {
                result = mid;
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return result;
    }
}
```

Output:

