

Assignment - 8

Student Name: Nikhil Kumar Tiwari
Branch :BE-CSE
Semester: 6 th
Subject Name: Advanced Programming Lab- 2

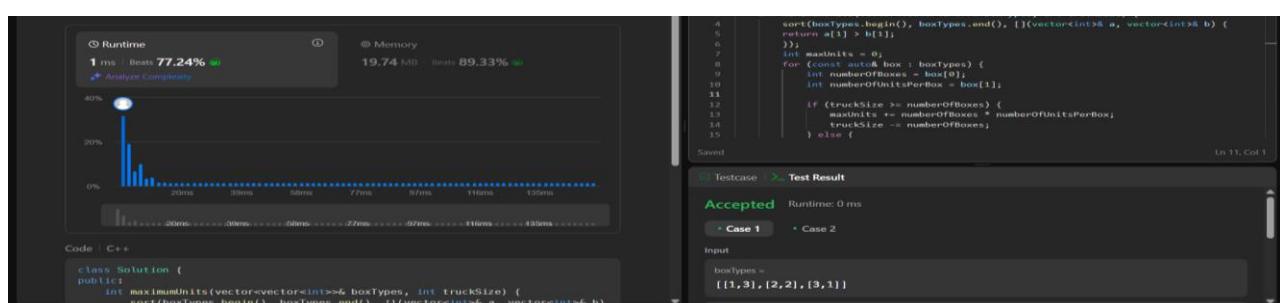
UID: 22BCS10471
Section/Group:22BCS-IOT-FL-601 A
Subject Code: 22CSP-351

Problem 1: Maximum Units on a Truck (<https://leetcode.com/problems/maximum-units-on-a-truck/>)

Code:

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

Screenshot:



The screenshot shows a code editor interface with the following details:

- Runtime:** 1 ms | Beats 77.24% (Green)
- Memory:** 19.74 MB | Beats 89.33% (Green)
- Analyze Complexity:** (button)
- Code (C++):**

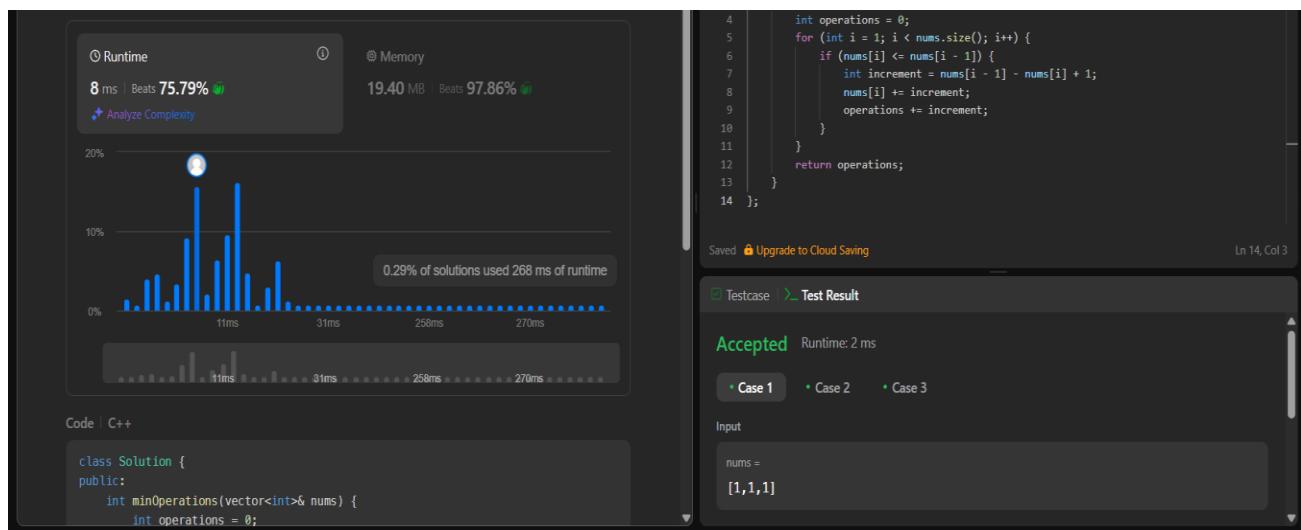
```
4     sort(boxTypes.begin(), boxTypes.end(), [](vector<int>& a, vector<int>& b) {  
5         return a[1] > b[1];  
6     });  
7     int maxUnits = 0;  
8     for (const auto& box : boxTypes) {  
9         int numberOfBoxes = box[0];  
10        int numberOfUnitsPerBox = box[1];  
11        if (truckSize >= numberOfBoxes) {  
12            maxUnits += numberOfBoxes * numberOfUnitsPerBox;  
13            truckSize -= numberOfBoxes;  
14        } else {  
15            maxUnits += truckSize * numberOfUnitsPerBox;  
16            break;  
17        }  
18    }  
19    return maxUnits;  
20};
```
- Testcase:** (button)
- Test Result:** Accepted | Runtime: 0 ms
- Input:** boxTypes = [[1,3],[2,2],[3,1]]

Problem 2: Minimum Operations to Make the Array (<https://leetcode.com/problems/minimum-operations-to-make-the-array-increasing/>)

Code:

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

Screenshot:

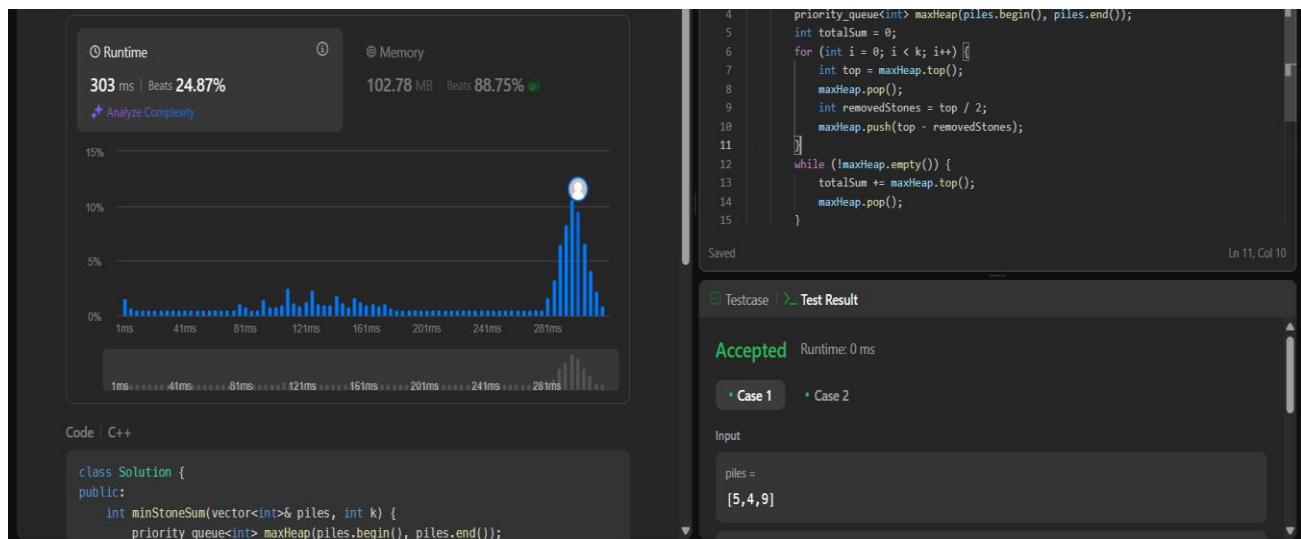


Problem 3: Remove Stones to Minimize the Total (<https://leetcode.com/problems/remove-stones-to-minimize-the-total/>)

Code:

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

Screenshot:



Problem 4: Maximum Score from Removing Substrings

(<https://leetcode.com/problems/maximum-score-from-removing-substrings/>)

Code:

```
class Solution {  
  
public:  
  
    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 points = 0;  
  
        stack<char> st;  
  
        for (char c : s){  
  
            if (!st.empty() && st.top() == 'a' && c == 'b') {  
  
                st.pop();  
  
                points += 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();  
  
                points += y;  
  
            }  
  
        }  
  
        return points;  
    }  
};
```

```

st.pop();

points += y;

} else {

    st.push(c);

}

return points;

};


```

Screenshot:

The screenshot shows a development environment interface with the following sections:

- Runtime & Memory:** Displays performance metrics. Runtime: 70 ms (Beats 18.34%) and Memory: 28.67 MB (Beats 38.19%).
- Code Editor:** Shows the C++ code for the solution.
- Test Results:** Shows the test case "Testcase" was accepted with a runtime of 0 ms. It includes two test cases: Case 1 and Case 2, both of which passed.

```

4     if (x < y) {
5         swap(x, y);
6         for (char& c : s) {
7             if (c == 'a') c = 'b';
8             else if (c == 'b') c = 'a';
9         }
10    }
11    int points = 0;
12    stackchar> st;
13    for (char c : s) {
14        if (!st.empty() && st.top() == 'a' && c == 'b') {
15            st.pop();

```

Ln 10, Col 10

Accepted Runtime: 0 ms

Case 1 Case 2

Input

s =
"cdbcbbaaabab"

Problem 5: Minimum Operations to make a Subsequence

(<https://leetcode.com/problems/minimum-operations-to-make-a-subsequence/>)

Code:

```
class Solution {  
  
public:  
  
    int minOperations(vector<int>& target, vector<int>& arr) {  
  
        int n = target.size(), ans = 0;  
  
        map<int, int> map; //  
  
        for(int i=0;i<n;i++) map[target[i]] = i;  
  
        vector<int> longSubsq;  
  
        for(int num : arr) {  
  
            if(map.find(num) == map.end()) continue;  
  
            auto it = lower_bound(longSubsq.begin(), longSubsq.end(), map[num]);  
  
            if(it == longSubsq.end())  
  
                longSubsq.push_back(map[num]);  
  
            else *it = map[num];  
  
        }  
  
        return target.size() - longSubsq.size();  
    }  
};
```

Screenshot:

