

Name: Pranjal Singh
Sec: FL_IOT-601/ A

UID: 22BCS13041
Sub: AP Lab -II

Max Units on a Truck

```
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) {
        sort(boxTypes.begin(),boxTypes.end(),myfunction);
        int ans=0;
        for(auto box: boxTypes){
            int x=min(box[0],truckSize);
            ans+=(x*box[1]);
            truckSize-=x;
            if(!truckSize) break;
        }
        return ans;
    }
};
```

The screenshot displays a submission page for the problem 'Max Units on a Truck'. On the left, the 'Runtime' section shows a time of 7 ms, which beats 42.15% of other submissions. The 'Memory' section shows a usage of 20.79 MB, beating 32.88%. Below these, a bar chart visualizes the performance distribution. The right side of the interface shows the C++ code used for the solution, which is identical to the code provided in the text block. At the bottom right, the 'Testcase' section is visible, showing 'Case 1' with the input boxTypes = [[1,3], [2,2], [3,1]].

Min Operations to Make Array Increasing

```
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;
    }
};

```

The screenshot displays a LeetCode submission for the problem "Remove Stones to Maximize Total". The submission is accepted, with 94/94 test cases passed. The runtime is 424 ms, which is 5.00% faster than other solutions. The memory usage is 19.60 MB, which is 65.13% better than other solutions. A bar chart shows the runtime distribution, with a peak at 11ms. The code editor on the right shows the C++ solution. The test case section shows "Case 1" with input "[1, 1, 1]".

Runtime: 424 ms | Beats 5.00%

Memory: 19.60 MB | Beats 65.13%

Testcase: Case 1

nums = [1, 1, 1]

Remove Stones to Maximize Total

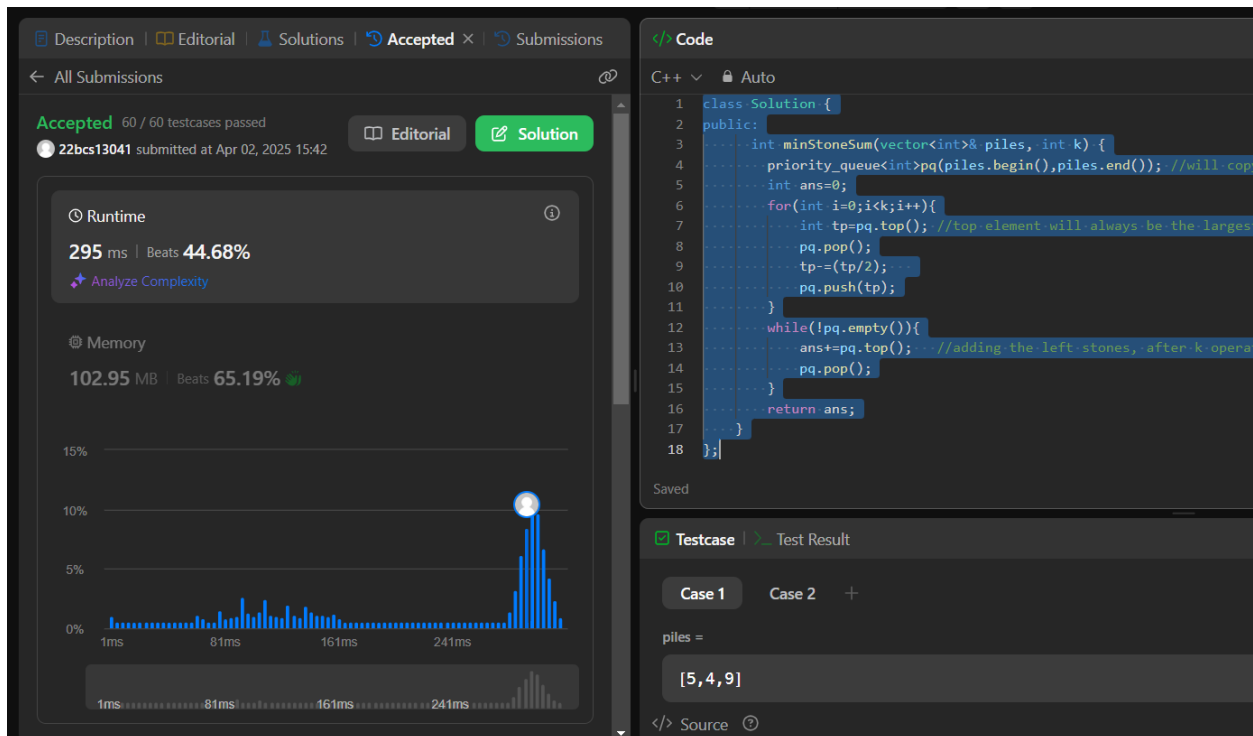
```

class Solution {

```

public:

```
int minStoneSum(vector<int>& piles, int k) {  
    priority_queue<int> pq(piles.begin(), piles.end()); //will copy the vector to the priority queue  
    int ans=0;  
    for(int i=0; i<k; i++){  
        int tp=pq.top(); //top element will always be the largest element  
        pq.pop();  
        tp=(tp/2);  
        pq.push(tp);  
    }  
    while(!pq.empty()){  
        ans+=pq.top(); //adding the left stones, after k operations  
        pq.pop();  
    }  
    return ans;  
}  
};
```



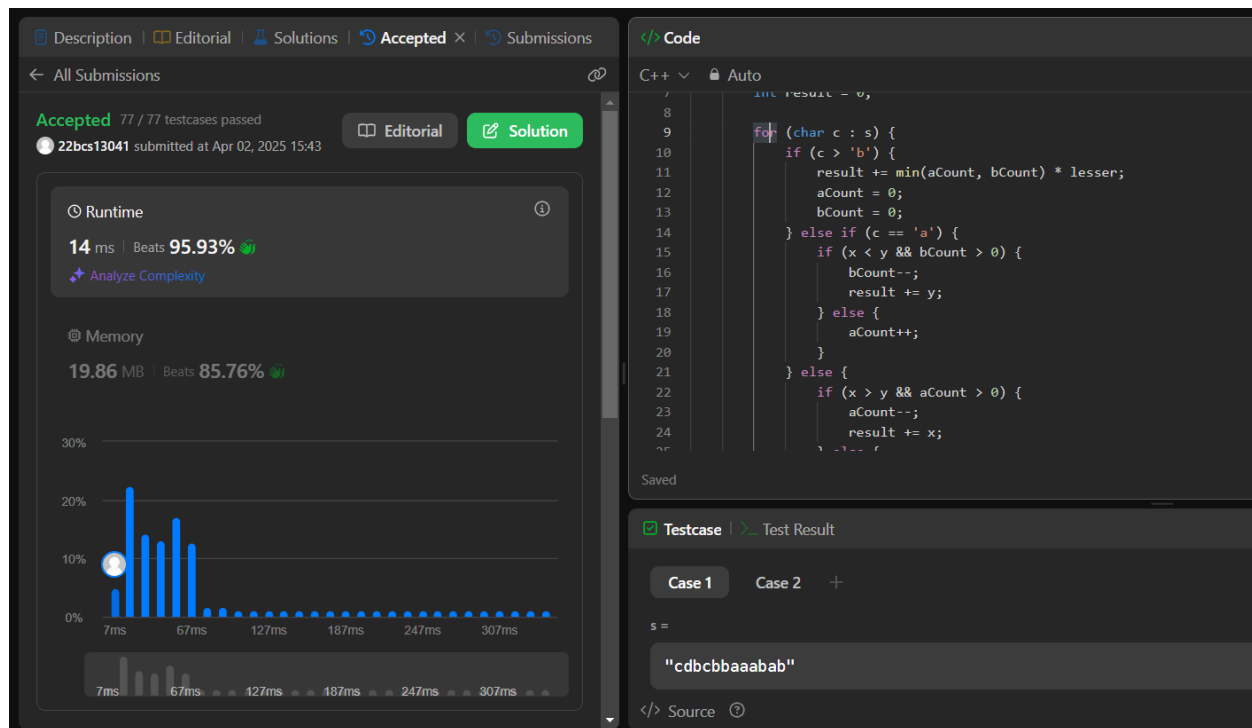
Max Score from Removing Substrings

```
class Solution {
```

```
public:
int maximumGain(string s, int x, int y) {
    int aCount = 0;
    int bCount = 0;
    int lesser = min(x, y);
    int result = 0;

    for (char c : s) {
        if (c > 'b') {
            result += min(aCount, bCount) * lesser;
            aCount = 0;
            bCount = 0;
        } else if (c == 'a') {
            if (x < y && bCount > 0) {
                bCount--;
                result += y;
            } else {
                aCount++;
            }
        } else {
            if (x > y && aCount > 0) {
                aCount--;
                result += x;
            } else {
                bCount++;
            }
        }
    }

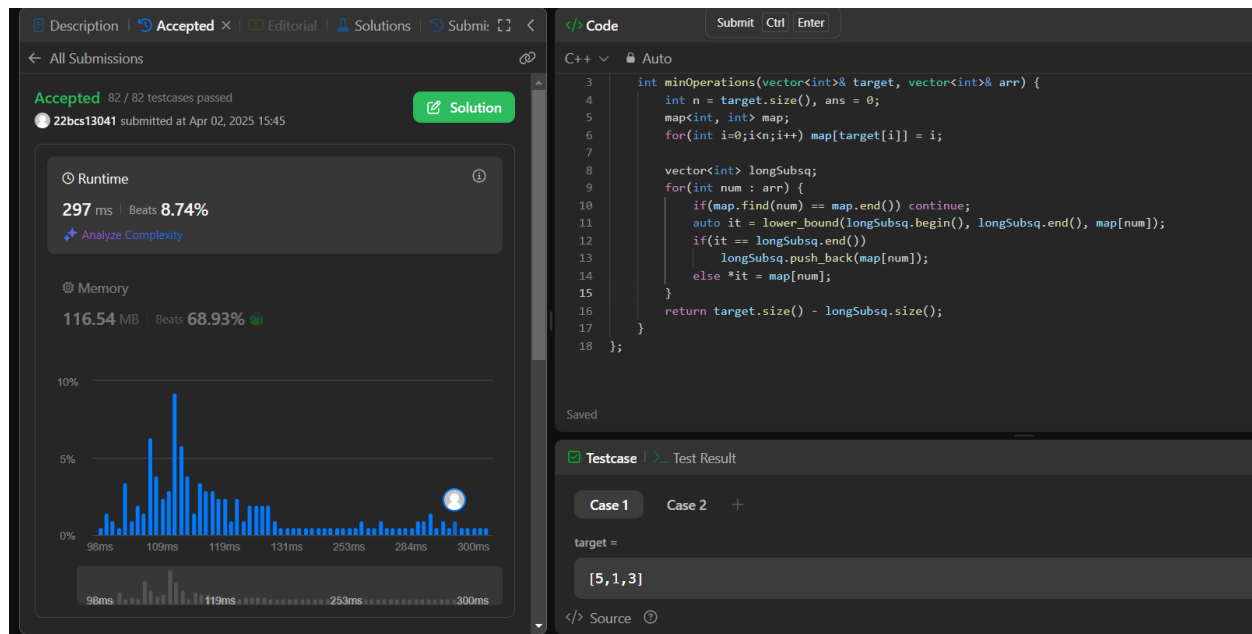
    result += min(aCount, bCount) * lesser;
    return result;
}
};
```



Min Operations to Make a Subsequence

```
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();
    }
};
```



Max Number of Tasks You Can Assign

```

class Solution {
public:
    bool check(vector<int>& tasks, vector<int>& workers, int pills, int strength, int index)
    {
        multiset<int> st;
        for(auto it:workers)
        {
            st.insert(it);
        }
        for(int i=index-1; i>=0; i--)
        {
            auto it=st.lower_bound(tasks[i]);
            if(it!=st.end())
            {
                st.erase(it);
            }
            else
            {
                if(pills<=0)
                {

```

```

        return false;
    }
    else
    {
        it=st.lower_bound(tasks[i]-strength);
        if(it!=st.end())
        {
            st.erase(it);
            pills--;
        }
        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;
    int high=min(workers.size(),tasks.size());
    while(low<high)
    {
        int mid=(low+high+1)/2;
        if(check(tasks,workers,pills,strength,mid)==true)
        {
            low=mid;
        }
        else
        {
            high=mid-1;
        }
    }
    return high;
}

```

};

Description | Editorial | Solutions | Accepted x | Submissions

← All Submissions

Accepted 49 / 49 testcases passed

22bcs13041 submitted at Apr 02, 2025 15:46

Solution

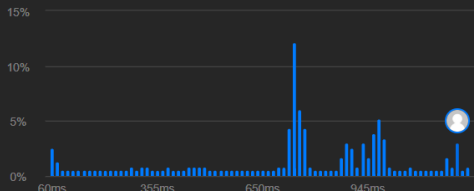
Runtime

1198 ms | Beats 8.23%

Analyze Complexity

Memory

338.76 MB | Beats 38.10%



60ms 355ms 650ms 945ms

Code

C++ Auto

```
43 int low=0;
44 int high=min(workers.size(),tasks.size());
45 while(low<high)
46 {
47     int mid=(low+high+1)/2;
48     if(check(tasks,workers,pills,strength,mid)==true)
49     {
50         low=mid;
51     }
52     else
53     {
54         high=mid-1;
55     }
56 }
57 return high;
58 }
59 ;
```

Saved

Testcase | Test Result

Case 1 Case 2 Case 3 +

tasks =

[3,2,1]

</> Source ⓘ