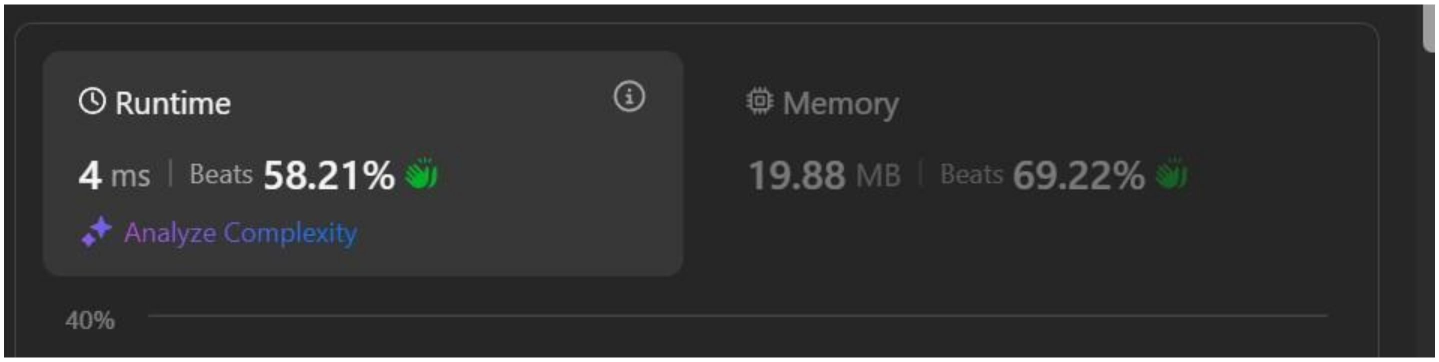
NAME- Vansh Namdev| UID- 22BCS10714 | SECTION- 601/A 1

Max Units on a Truck

```
bool cmp(vector<int> &a , vector<int> &b){
   return a[1] > b[1];
}

class Solution {
public:
   int maximumUnits(vector<vector<int>>&
boxTypes, int truckSize) {
      sort(boxTypes.begin() , boxTypes.end() , cmp);
      int profit = 0;
      for(int i=0 ; i < boxTypes.size() ; i++){
         if(boxTypes[i][0] <= truckSize){
            profit += boxTypes[i][0]*boxTypes[i][1];
            truckSize -= boxTypes[i][0];
         }
         else{
            profit += truckSize*boxTypes[i][1];
            truckSize = 0;
         }
         if(truckSize == 0) break;
      }
      return profit;
   }
};
```


Runtime

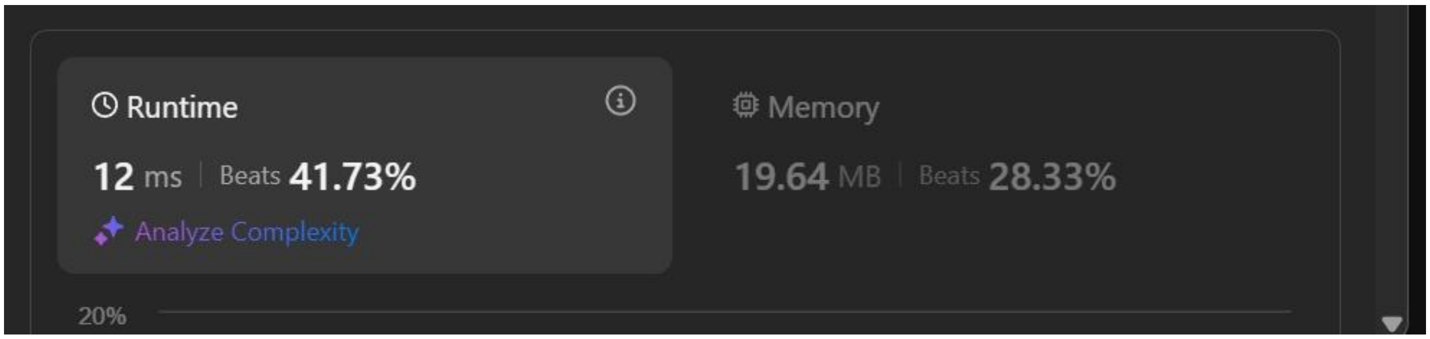
4 ms | Beats 58.21%

Memory

19.88 MB | Beats 69.22%

2. MIN Operations to Make Array Increasing

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


        return count;
    }
};
```



Runtime

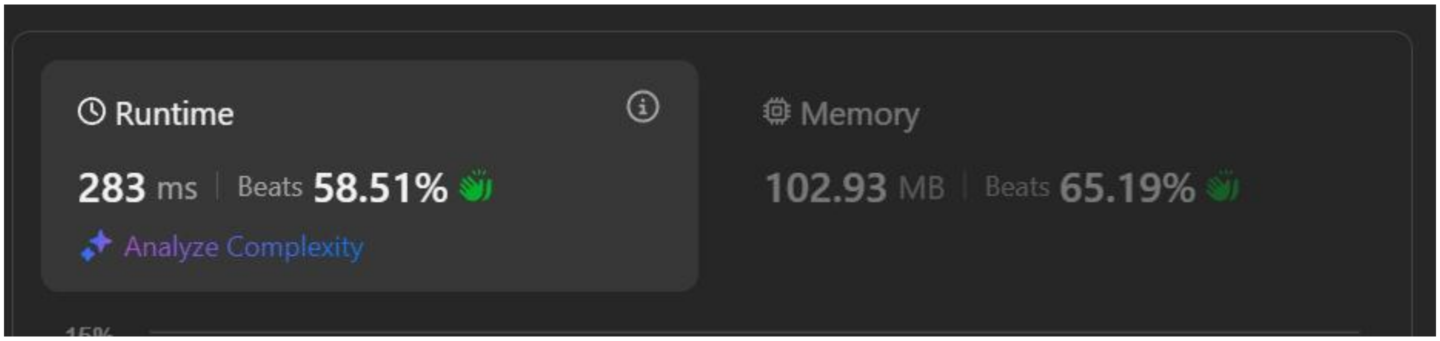
12 ms | Beats 41.73%

Memory

19.64 MB | Beats 28.33%

3. Remove Stones to Maximize Total

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

      while (k--) {
         int maxElement = maxHeap.top();
         maxHeap.pop();
         maxElement -= floor(maxElement / 2);
         maxHeap.push(maxElement);
      }

      int sum = 0;
      while (!maxHeap.empty()) {
         sum += maxHeap.top();
         maxHeap.pop();
      }
      return sum;
   }
};
```


Runtime

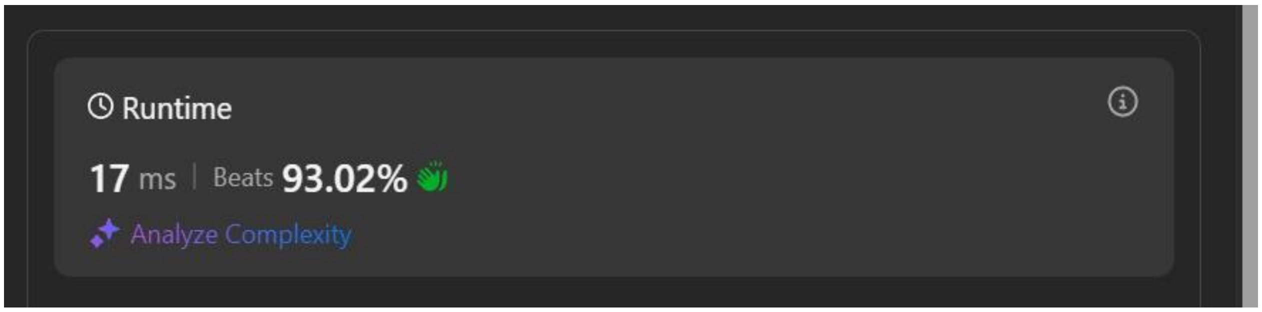
283 ms | Beats 58.51%

Memory

102.93 MB | Beats 65.19%

4. Max Score from Removing Substrings

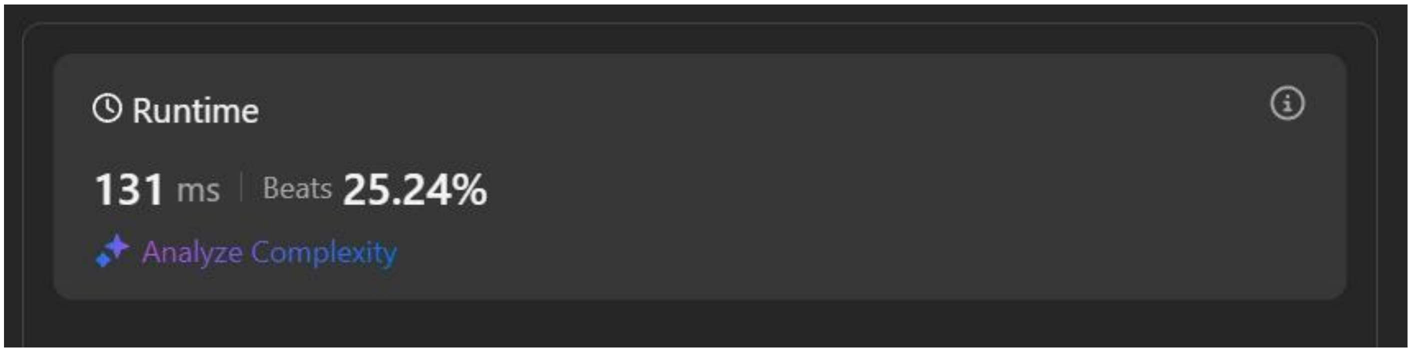
```
class Solution {
public:
   int maximumGain(string s, int x, int y) {
      int aCount = 0;
      int bCount = 0;
      int lesser = min(x, y);
      int result = 0;

      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;
   }
};
```


Runtime

17 ms | Beats 93.02%

5. Min Operations to Make a Subsequence

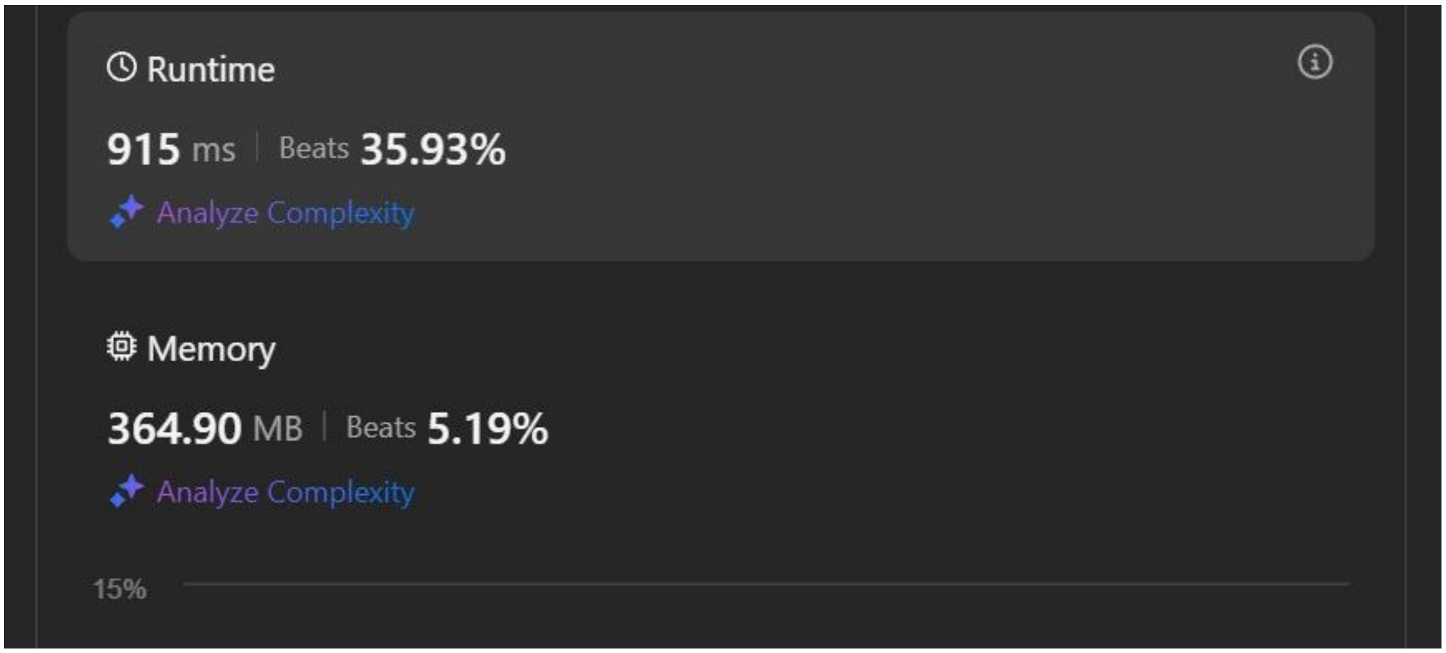
```
   class Solution {
public:
   int minOperations(vector<int>& target, vector<int>& arr) {
      unordered_map<int, int> mapping;
      int i = 0;
      for (auto& num : target)
         mapping[num] = ++i;

      vector<int> A;
      for (int& num : arr)
         if (mapping.find(num) != mapping.end())
            A.push_back(mapping[num]);
      return target.size() - lengthOfLIS(A);
   }
private:
   int lengthOfLIS(vector<int>& nums) {
      if (nums.empty()) return 0;
      vector<int> piles;
      for(int i=0; i<nums.size(); i++) {
         auto it = std::lower_bound(piles.begin(), piles.end(), nums[i]);
         if (it == piles.end())
            piles.push_back(nums[i]);
         else
            *it = nums[i];
      }
      return piles.size();
}

};
```


Runtime

131 ms | Beats 25.24%

6. Max Number of Tasks You Can Assign

```
class Solution {
public:
    int check(vector<int> &tasks, int take, map<int,int> count, int pills, int power) {
        while (take >= 1 && count.size()) {
            auto it = count.end(); --it;

            if (tasks[take - 1] <= it->first) {}
            else if (pills) {
                it = count.lower_bound(tasks[take - 1] - power);
                if (it == count.end()) return 0;
                --pills;
            }
            else return 0;

            --take;
            (it->second)--;
            if (it->second == 0)
                count.erase(it);
        }
        return take == 0;
    }

    int maxTaskAssign(vector<int>& t, vector<int>& w, int p, int s) {
        int n = t.size();
        int m = w.size();
        sort(t.begin(), t.end());
        map<int,int> Count;
        for (auto &strength : w) Count[strength]++;

        int l = 0, r = n, ans = 0;

        while (l <= r) {
            int mid = l + (r - l) / 2;
            int chk = check(t, mid, Count, p, s);
            if (chk) {
                ans = mid;
                l = mid + 1;
            }
            else {
                r = mid - 1;
            }
        }
        return ans;
    }
};
```



Runtime

915 ms | Beats 35.93%

Memory

364.90 MB | Beats 5.19%