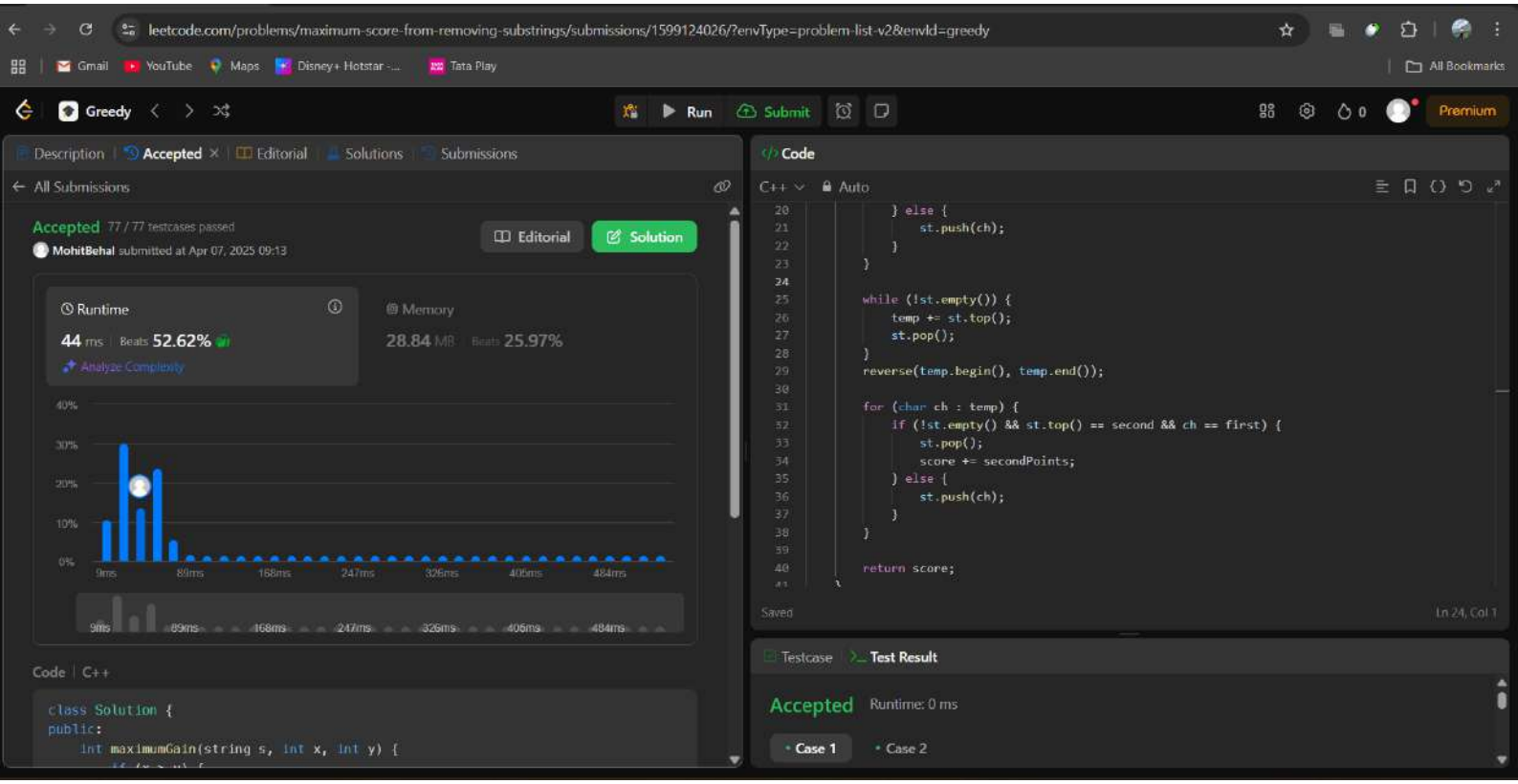
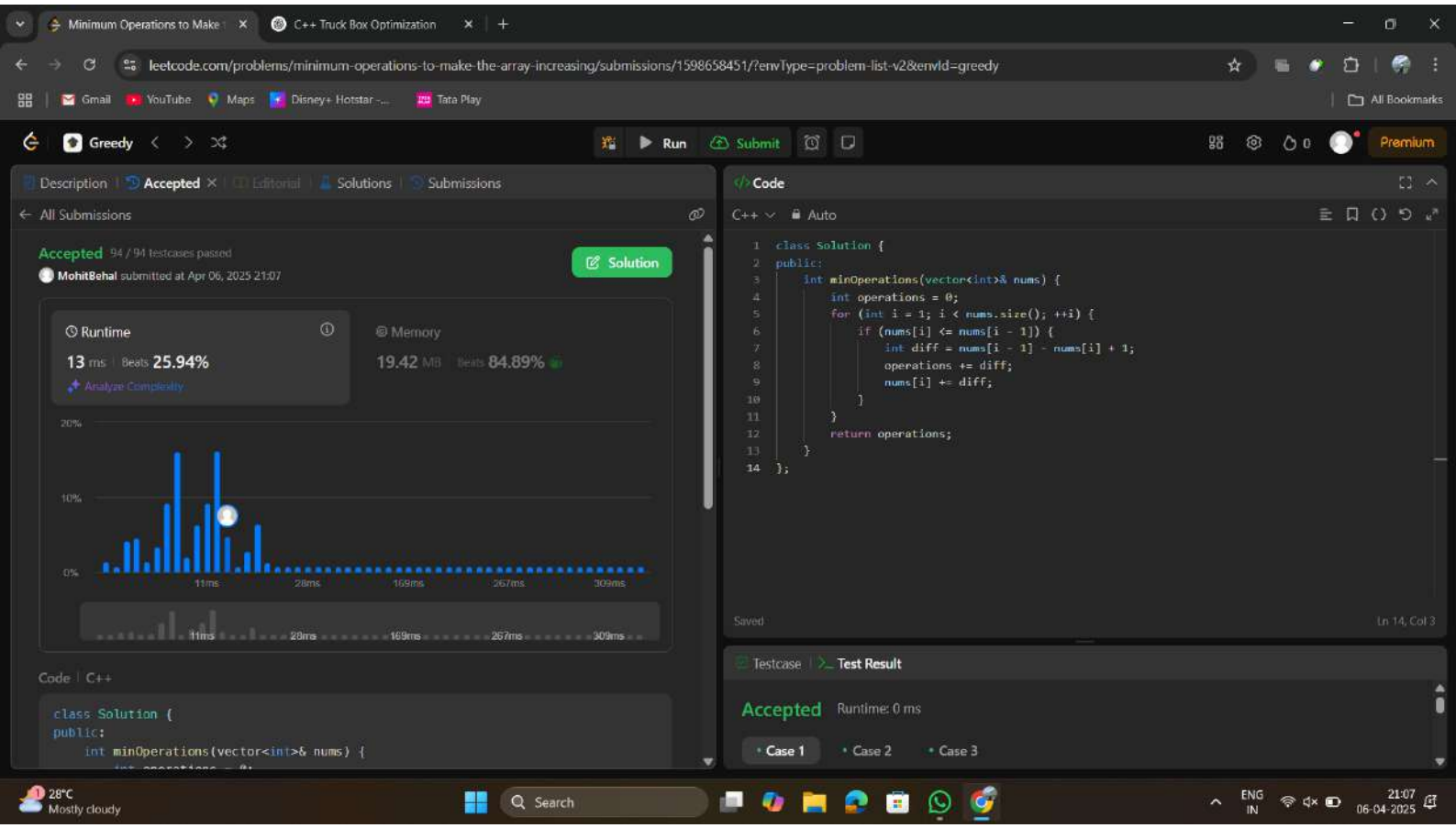
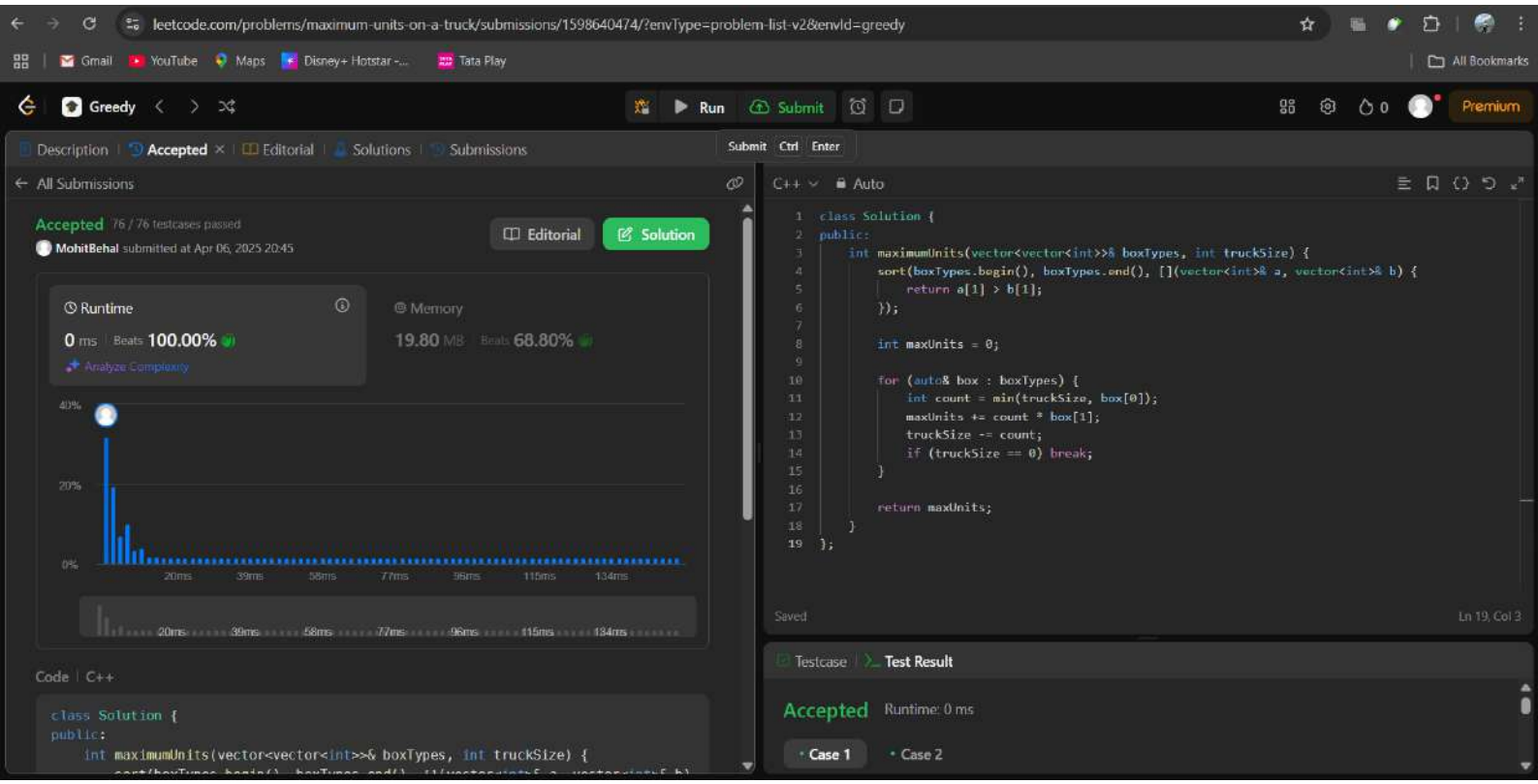
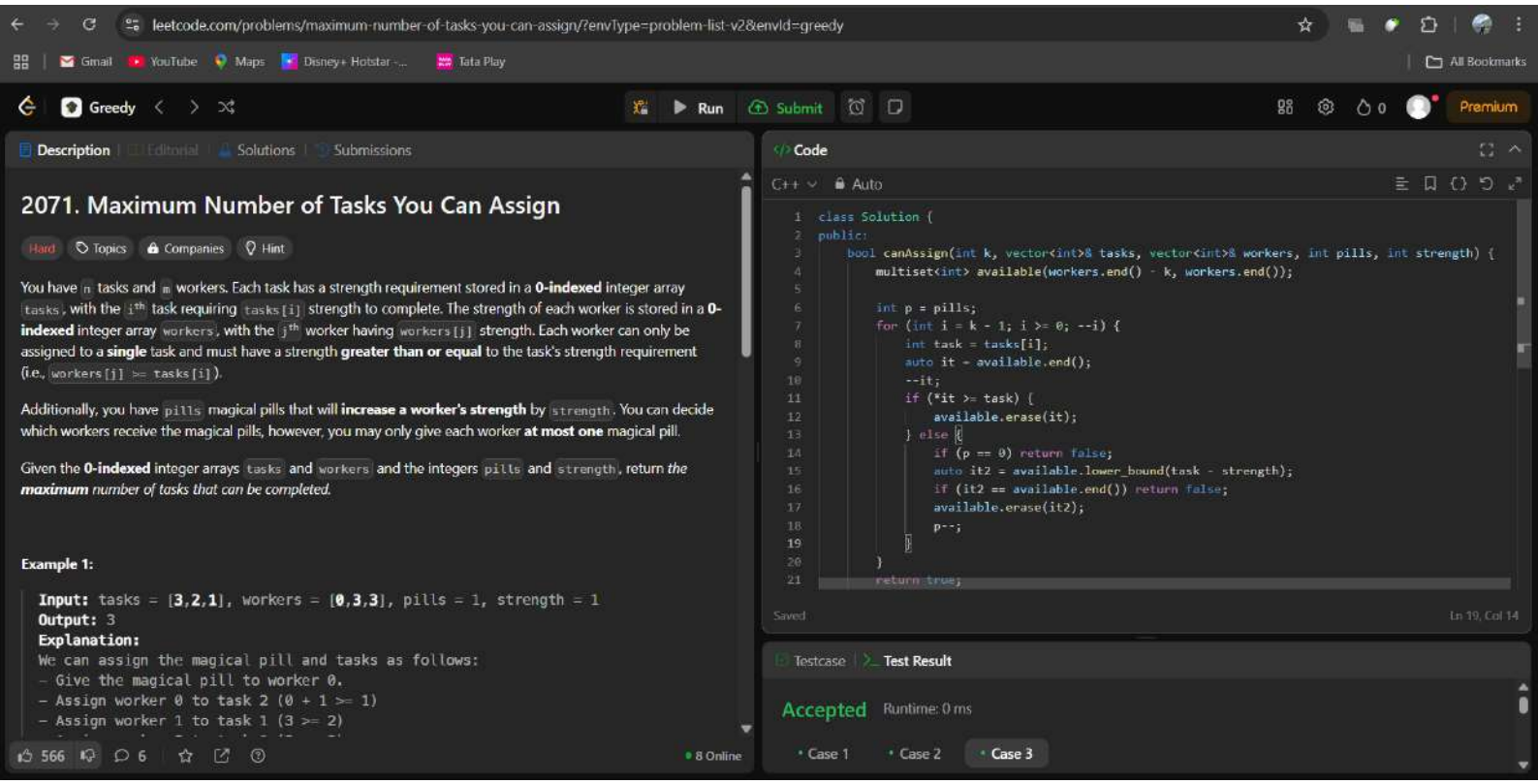
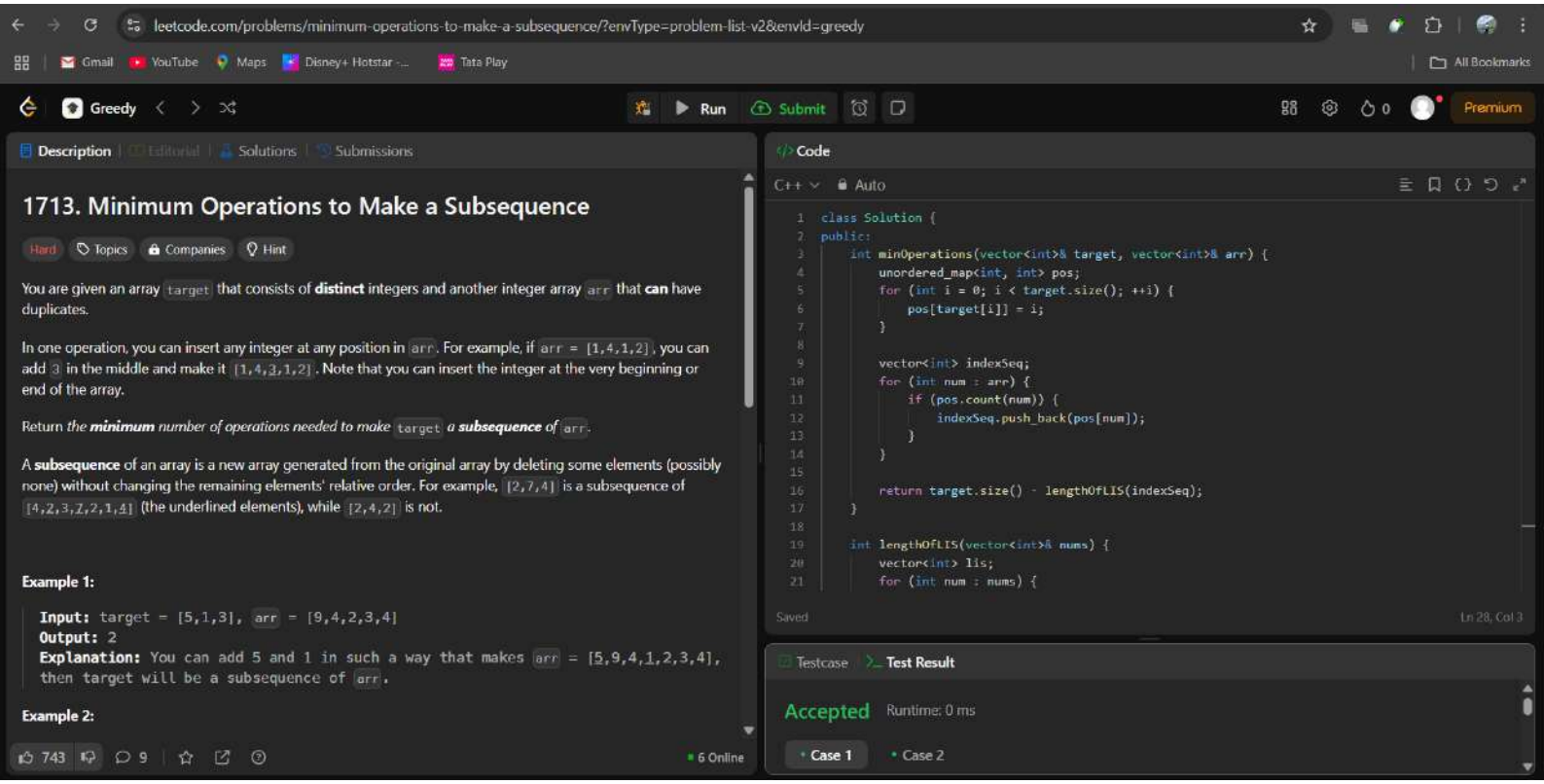
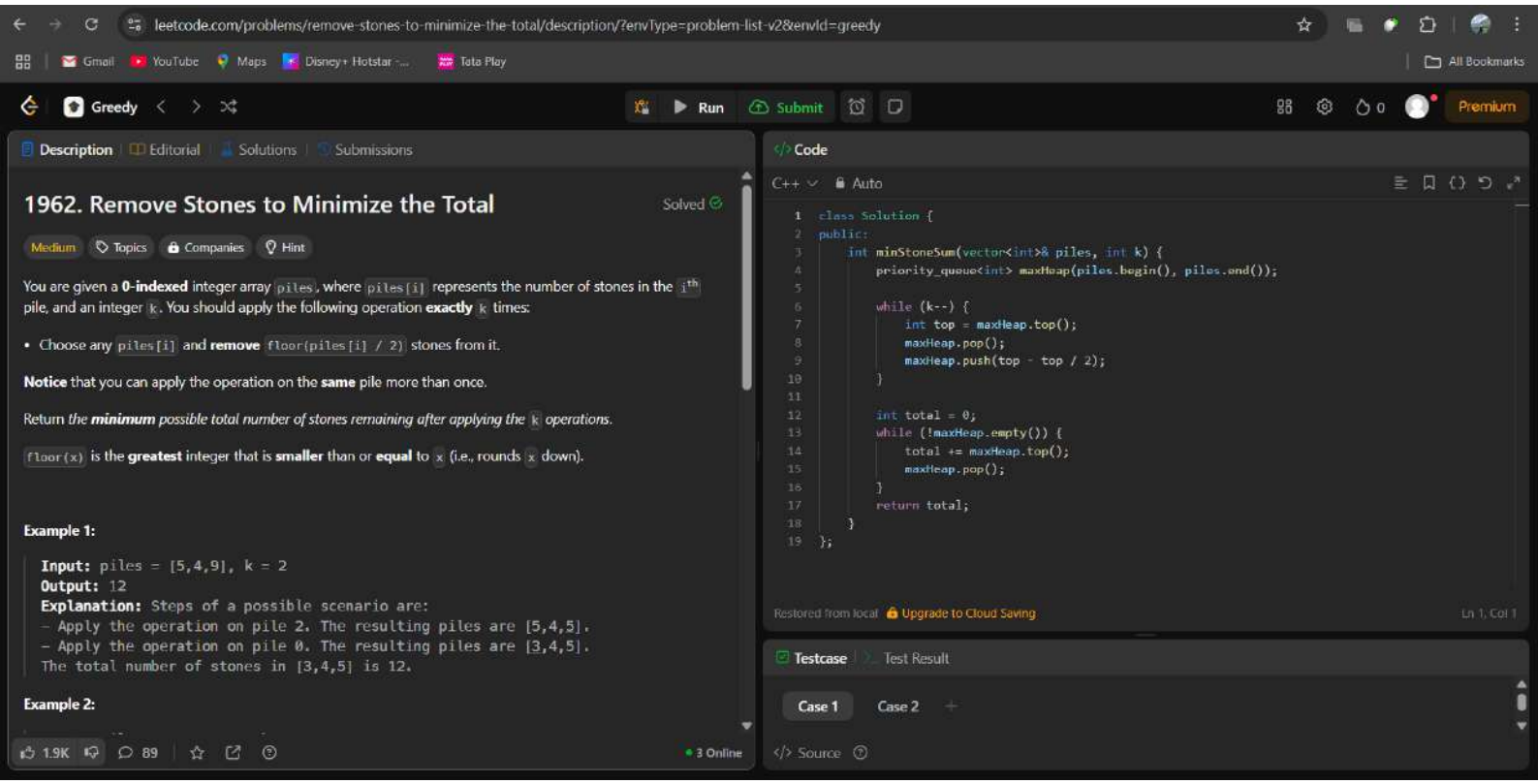

leetcode.com/problems/maximum-score-from-removing-substrings/submissions/1599124026/?envType=problem-list-v2&envId=greedy
Gmail
YouTube
Maps
Disney+ Hotstar -...
Tata Play
All Bookmarks
Greedy
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 77 / 77 testcases passed
MohitBehal submitted at Apr 07, 2025 09:13
Editorial
Solution
Runtime
44 ms | Beats 52.62%
Analyze Complexity
Memory
28.84 MB | Beats 25.97%
40%
30%
20%
10%
0%
9ms
89ms
168ms
247ms
326ms
405ms
484ms
Code | C++
class Solution {
public:
int maximumGain(string s, int x, int y) {
Code
C++
Auto
} else {
st.push(ch);
}
}
while (!st.empty()) {
temp += st.top();
st.pop();
}
reverse(temp.begin(), temp.end());
for (char ch : temp) {
if (!st.empty() && st.top() == second && ch == first) {
st.pop();
score += secondPoints;
} else {
st.push(ch);
}
}
return score;
}
Saved
Ln 24, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2


Minimum Operations to Make
C++ Truck Box Optimization
leetcode.com/problems/minimum-operations-to-make-the-array-increasing/submissions/1598658451/?envType=problem-list-v2&envId=greedy
Gmail
YouTube
Maps
Disney+ Hotstar -...
Tata Play
All Bookmarks
Greedy
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 94 / 94 testcases passed
MohitBehal submitted at Apr 06, 2025 21:07
Solution
Runtime
13 ms | Beats 25.94%
Analyze Complexity
Memory
19.42 MB | Beats 84.89%
20%
10%
0%
11ms
28ms
169ms
267ms
309ms
Code | C++
class Solution {
public:
int minOperations(vector<int>& nums) {
Code
C++
Auto
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 diff = nums[i - 1] - nums[i] + 1;
operations += diff;
nums[i] += diff;
}
}
return operations;
}
};
Saved
Ln 14, Col 3
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
28°C
Mostly cloudy
Search
ENG
IN
21:07
06-04-2025


leetcode.com/problems/maximum-units-on-a-truck/submissions/1598640474/?envType=problem-list-v2&envId=greedy
Gmail
YouTube
Maps
Disney+ Hotstar -...
Tata Play
All Bookmarks
Greedy
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
Submit Ctrl Enter
All Submissions
Accepted 76 / 76 testcases passed
MohitBehal submitted at Apr 06, 2025 20:45
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
19.80 MB | Beats 68.80%
40%
20%
0%
20ms
39ms
58ms
77ms
96ms
115ms
134ms
Code | C++
class Solution {
public:
int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
C++
Auto
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 (auto& box : boxTypes) {
int count = min(truckSize, box[0]);
maxUnits += count * box[1];
truckSize -= count;
if (truckSize == 0) break;
}
return maxUnits;
}
};
Saved
Ln 19, Col 3
Testcase | Test Result
Accepted Runtime: 0 ms
Case 1
Case 2


leetcode.com/problems/maximum-number-of-tasks-you-can-assign/?envType=problem-list-v2&envId=greedy
Gmail
YouTube
Maps
Disney+ Hotstar -...
Tata Play
All Bookmarks
Greedy
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
2071. Maximum Number of Tasks You Can Assign
Hard
Topics
Companies
Hint
You have n tasks and m workers. Each task has a strength requirement stored in a 0-indexed integer array tasks, with the i^th task requiring tasks[i] strength to complete. The strength of each worker is stored in a 0-indexed integer array workers, with the j^th worker having workers[j] strength. Each worker can only be assigned to a single task and must have a strength greater than or equal to the task's strength requirement (i.e., workers[j] >= tasks[i]).
Additionally, you have pills magical pills that will increase a worker's strength by strength. You can decide which workers receive the magical pills, however, you may only give each worker at most one magical pill.
Given the 0-indexed integer arrays tasks and workers and the integers pills and strength, return the maximum number of tasks that can be completed.
Example 1:
Input: tasks = [3,2,1], workers = [0,3,3], pills = 1, strength = 1
Output: 3
Explanation:
We can assign the magical pill and tasks as follows:
- Give the magical pill to worker 0.
- Assign worker 0 to task 2 (0 + 1 >= 1)
- Assign worker 1 to task 1 (3 >= 2)
566
6
8 Online
Code
C++
Auto
class Solution {
public:
bool canAssign(int k, vector<int>& tasks, vector<int>& workers, int pills, int strength) {
multiset<int> available(workers.end() - k, workers.end());
int p = pills;
for (int i = k - 1; i >= 0; --i) {
int task = tasks[i];
auto it = available.end();
--it;
if (*it >= task) {
available.erase(it);
} else {
if (p == 0) return false;
auto it2 = available.lower_bound(task - strength);
if (it2 == available.end()) return false;
available.erase(it2);
p--;
}
}
return true;
Saved
Ln 19, Col 14
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3


leetcode.com/problems/minimum-operations-to-make-a-subsequence/?envType=problem-list-v2&envId=greedy
Gmail
YouTube
Maps
Disney+ Hotstar -...
Tata Play
All Bookmarks
Greedy
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
1713. Minimum Operations to Make a Subsequence
Hard
Topics
Companies
Hint
You are given an array target that consists of distinct integers and another integer array arr that can have duplicates.
In one operation, you can insert any integer at any position in arr. For example, if arr = [1,4,1,2], you can add 3 in the middle and make it [1,4,3,1,2]. Note that you can insert the integer at the very beginning or end of the array.
Return the minimum number of operations needed to make target a subsequence of arr.
A subsequence of an array is a new array generated from the original array by deleting some elements (possibly none) without changing the remaining elements' relative order. For example, [2,7,4] is a subsequence of [4,2,3,7,2,1,4] (the underlined elements), while [2,4,2] is not.
Example 1:
Input: target = [5,1,3], arr = [9,4,2,3,4]
Output: 2
Explanation: You can add 5 and 1 in such a way that makes arr = [5,9,4,1,2,3,4], then target will be a subsequence of arr.
Example 2:
743
9
6 Online
Code
C++
Auto
class Solution {
public:
int minOperations(vector<int>& target, vector<int>& arr) {
unordered_map<int, int> pos;
for (int i = 0; i < target.size(); ++i) {
pos[target[i]] = i;
}
vector<int> indexSeq;
for (int num : arr) {
if (pos.count(num)) {
indexSeq.push_back(pos[num]);
}
}
return target.size() - lengthOfLIS(indexSeq);
}
int lengthOfLIS(vector<int>& nums) {
vector<int> lis;
for (int num : nums) {
Saved
Ln 28, Col 3
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2


leetcode.com/problems/remove-stones-to-minimize-the-total/description/?envType=problem-list-v2&envId=greedy
Gmail
YouTube
Maps
Disney+ Hotstar -...
Tata Play
All Bookmarks
Greedy
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
1962. Remove Stones to Minimize the Total
Solved
Medium
Topics
Companies
Hint
You are given a 0-indexed integer array piles, where piles[i] represents the number of stones in the ith pile, and an integer k. You should apply the following operation exactly k times:
Choose any piles[i] and remove floor(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.
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:
1.9K
89
3 Online
Code
C++
Auto
class Solution {
public:
int minStoneSum(vector<int>& piles, int k) {
priority_queue<int> maxHeap(piles.begin(), piles.end());
while (k--) {
int top = maxHeap.top();
maxHeap.pop();
maxHeap.push(top - top / 2);
}
int total = 0;
while (!maxHeap.empty()) {
total += maxHeap.top();
maxHeap.pop();
}
return total;
}
};
Restored from local
Upgrade to Cloud Saving
Ln 1, Col 1
Testcase
Test Result
Case 1
Case 2
Source