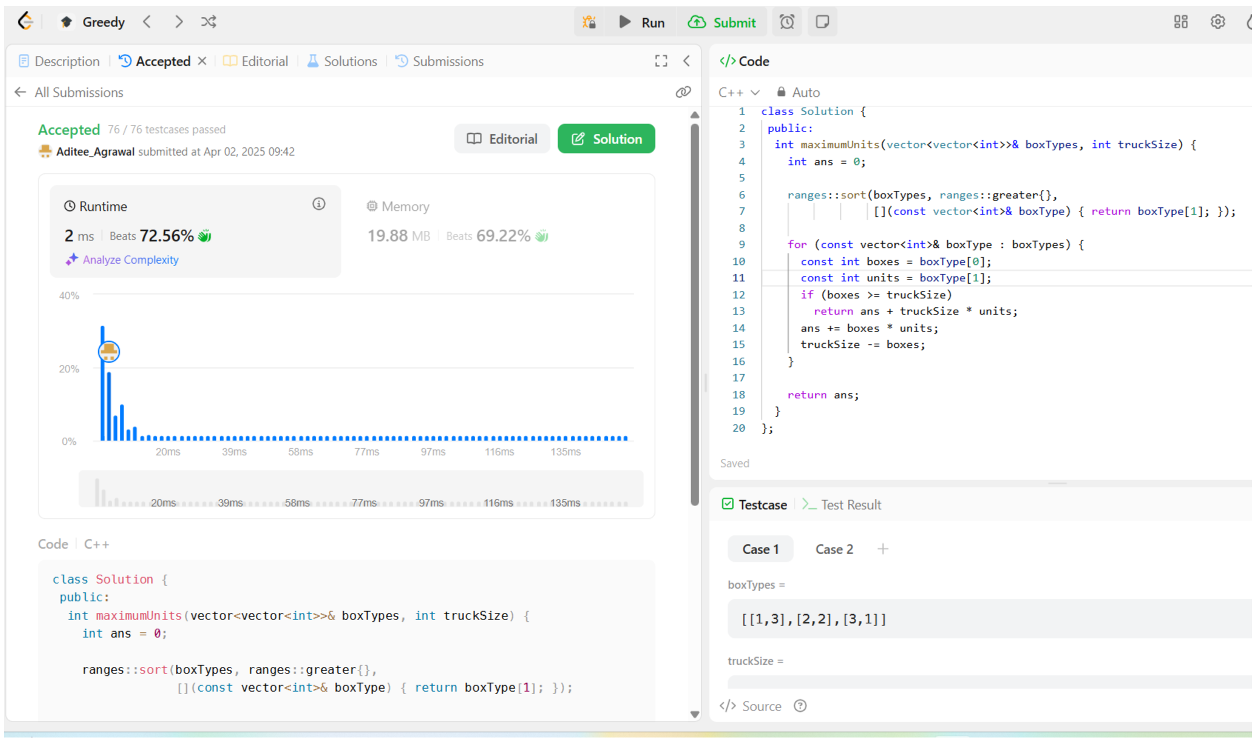
Name: Aaditee Agrawal
Uid: 22bcs11305
Section: FL_Iot 601 ‘A’

1. Max Units on a Truck

```
class Solution {
 public:
  int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
    int ans = 0;

    ranges::sort(boxTypes, ranges::greater{},
                 [](const vector<int>& boxType) { return boxType[1]; });

    for (const vector<int>& boxType : boxTypes) {
      const int boxes = boxType[0];
      const int units = boxType[1];
      if (boxes >= truckSize)
        return ans + truckSize * units;
      ans += boxes * units;
      truckSize -= boxes;
    }

    return ans;
  }
};
```



Accepted 76 / 76 testcases passed

Aditee_Agrawal submitted at Apr 02, 2025 09:42

Runtime 2 ms Beats 72.56%

Memory 19.88 MB Beats 69.22%

```cpp
class Solution {
 public:
  int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
    int ans = 0;

    ranges::sort(boxTypes, ranges::greater{},
                 [](const vector<int>& boxType) { return boxType[1]; });

    for (const vector<int>& boxType : boxTypes) {
      const int boxes = boxType[0];
      const int units = boxType[1];
      if (boxes >= truckSize)
        return ans + truckSize * units;
      ans += boxes * units;
      truckSize -= boxes;
    }

    return ans;
  }
};
```

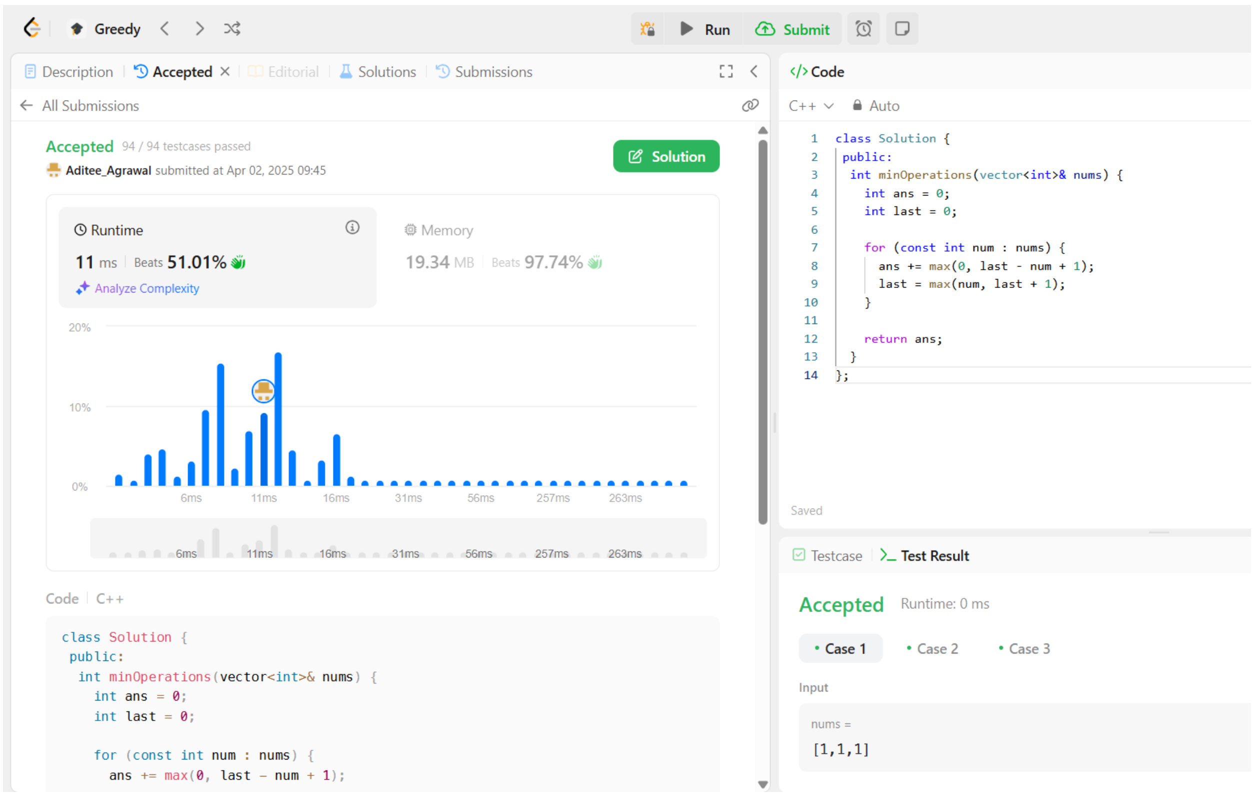
boxTypes =

```
[[1,3],[2,2],[3,1]]
```

truckSize =

2. Minimum Operations to Make the Array Increasing

```cpp
class Solution {
 public:
  int minOperations(vector<int>& nums) {
    int ans = 0;
    int last = 0;

    for (const int num : nums) {
      ans += max(0, last - num + 1);
      last = max(num, last + 1);
    }
    return ans;
  }
};
```



Accepted 94 / 94 testcases passed
Aditee_Agrawal submitted at Apr 02, 2025 09:45

Runtime
11 ms | Beats 51.01%

Memory
19.34 MB | Beats 97.74%

```cpp
class Solution {
 public:
  int minOperations(vector<int>& nums) {
    int ans = 0;
    int last = 0;

    for (const int num : nums) {
      ans += max(0, last - num + 1);
      last = max(num, last + 1);
    }

    return ans;
  }
};
```

Code | C++

```cpp
class Solution {
 public:
  int minOperations(vector<int>& nums) {
    int ans = 0;
    int last = 0;

    for (const int num : nums) {
      ans += max(0, last - num + 1);
```

Accepted Runtime: 0 ms

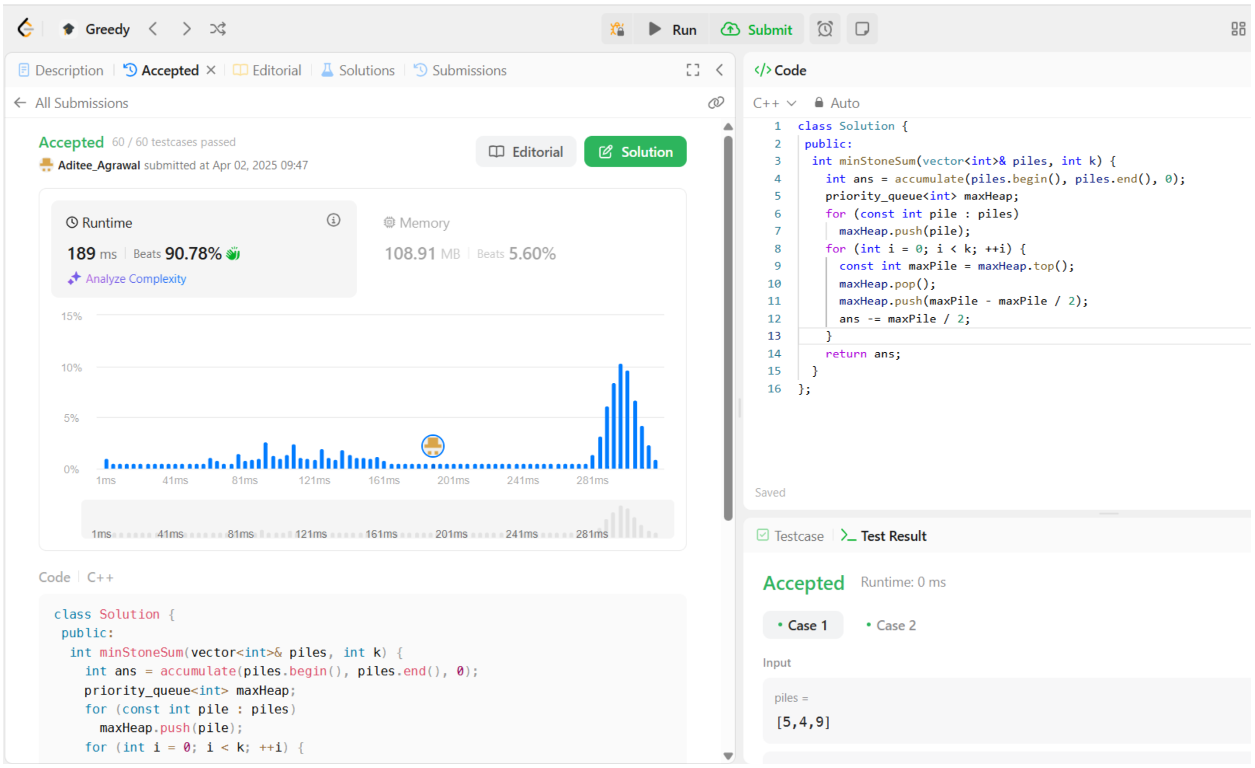
Case 1 Case 2 Case 3

Input

nums =
[1,1,1]

3. Remove Stones to Minimize the Total

```cpp
class Solution {
 public:
  int minStoneSum(vector<int>& piles, int k) {
    int ans = accumulate(piles.begin(), piles.end(), 0);
    priority_queue<int> maxHeap;

    for (const int pile : piles)
      maxHeap.push(pile);

    for (int i = 0; i < k; ++i) {
      const int maxPile = maxHeap.top();
      maxHeap.pop();
      maxHeap.push(maxPile - maxPile / 2);
      ans -= maxPile / 2;
    }
    return ans;
  }
};
```



Accepted 60 / 60 testcases passed
Aditee_Agrawal submitted at Apr 02, 2025 09:47

Runtime
189 ms | Beats 90.78%

Memory
108.91 MB | Beats 5.60%

```
class Solution {
public:
  int minStoneSum(vector<int>& piles, int k) {
    int ans = accumulate(piles.begin(), piles.end(), 0);
    priority_queue<int> maxHeap;
    for (const int pile : piles)
      maxHeap.push(pile);
    for (int i = 0; i < k; ++i) {
      const int maxPile = maxHeap.top();
      maxHeap.pop();
      maxHeap.push(maxPile - maxPile / 2);
      ans -= maxPile / 2;
    }
    return ans;
  }
};
```

Code | C++

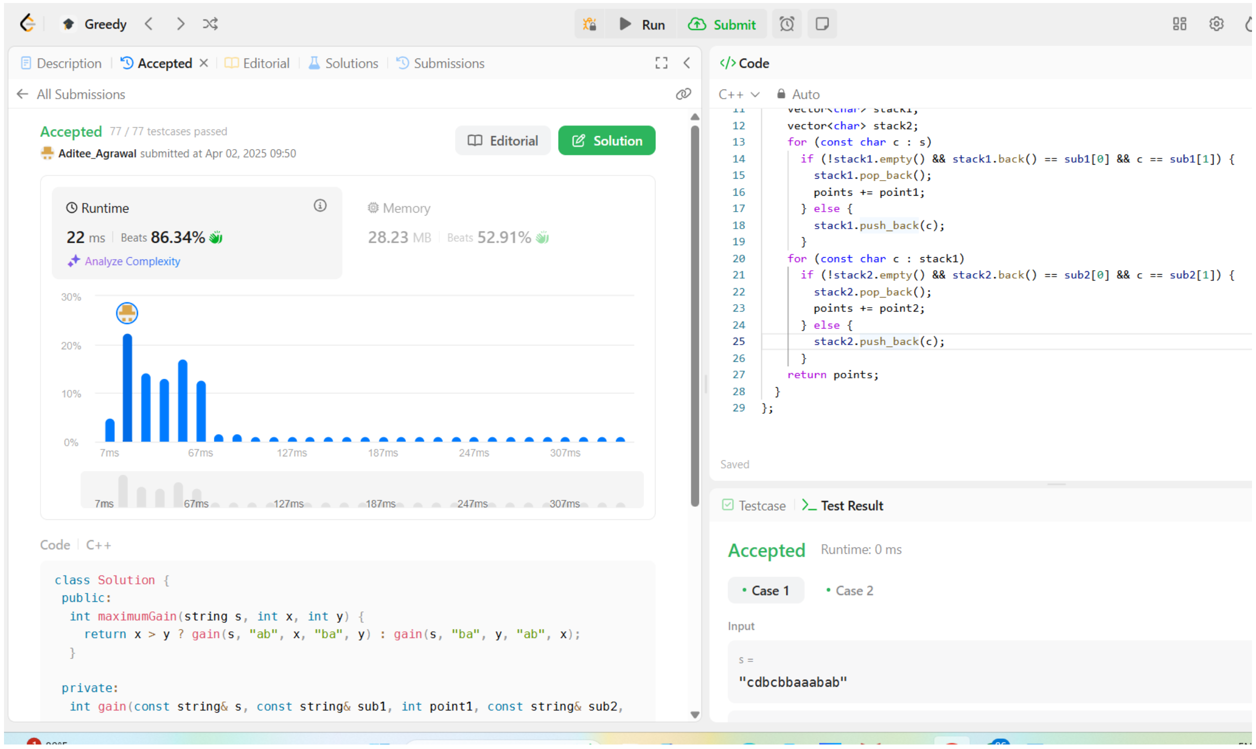
```
class Solution {
 public:
  int minStoneSum(vector<int>& piles, int k) {
    int ans = accumulate(piles.begin(), piles.end(), 0);
    priority_queue<int> maxHeap;
    for (const int pile : piles)
      maxHeap.push(pile);
    for (int i = 0; i < k; ++i) {
```

Accepted Runtime: 0 ms

piles =
[5,4,9]

4. Maximum Score From Removing Substrings

```
class Solution {
 public:
  int maximumGain(string s, int x, int y) {
    return x > y ? gain(s, "ab", x, "ba", y) : gain(s, "ba", y, "ab", x);
  }

 private:
  int gain(const string& s, const string& sub1, int point1, const string& sub2,
           int point2) {
    int points = 0;
    vector<char> stack1;
    vector<char> stack2;
    for (const char c : s)
      if (!stack1.empty() && stack1.back() == sub1[0] && c == sub1[1]) {
        stack1.pop_back();
        points += point1;
      } else {
        stack1.push_back(c);
      }
    for (const char c : stack1)
      if (!stack2.empty() && stack2.back() == sub2[0] && c == sub2[1]) {
        stack2.pop_back();
        points += point2;
      } else {
        stack2.push_back(c);
      }
    return points;
  }
};
```



Accepted 77 / 77 testcases passed

Aditee_Agrawal submitted at Apr 02, 2025 09:50

Runtime

22 ms | Beats 86.34%

Memory

28.23 MB | Beats 52.91%

Code | C++

```
class Solution {
 public:
  int maximumGain(string s, int x, int y) {
    return x > y ? gain(s, "ab", x, "ba", y) : gain(s, "ba", y, "ab", x);
  }

 private:
  int gain(const string& s, const string& sub1, int point1, const string& sub2,
```

```
    vector<char> stack2;
    for (const char c : s)
      if (!stack1.empty() && stack1.back() == sub1[0] && c == sub1[1]) {
        stack1.pop_back();
        points += point1;
      } else {
        stack1.push_back(c);
      }
    for (const char c : stack1)
      if (!stack2.empty() && stack2.back() == sub2[0] && c == sub2[1]) {
        stack2.pop_back();
        points += point2;
      } else {
        stack2.push_back(c);
      }
    return points;
  }
};
```

Accepted Runtime: 0 ms

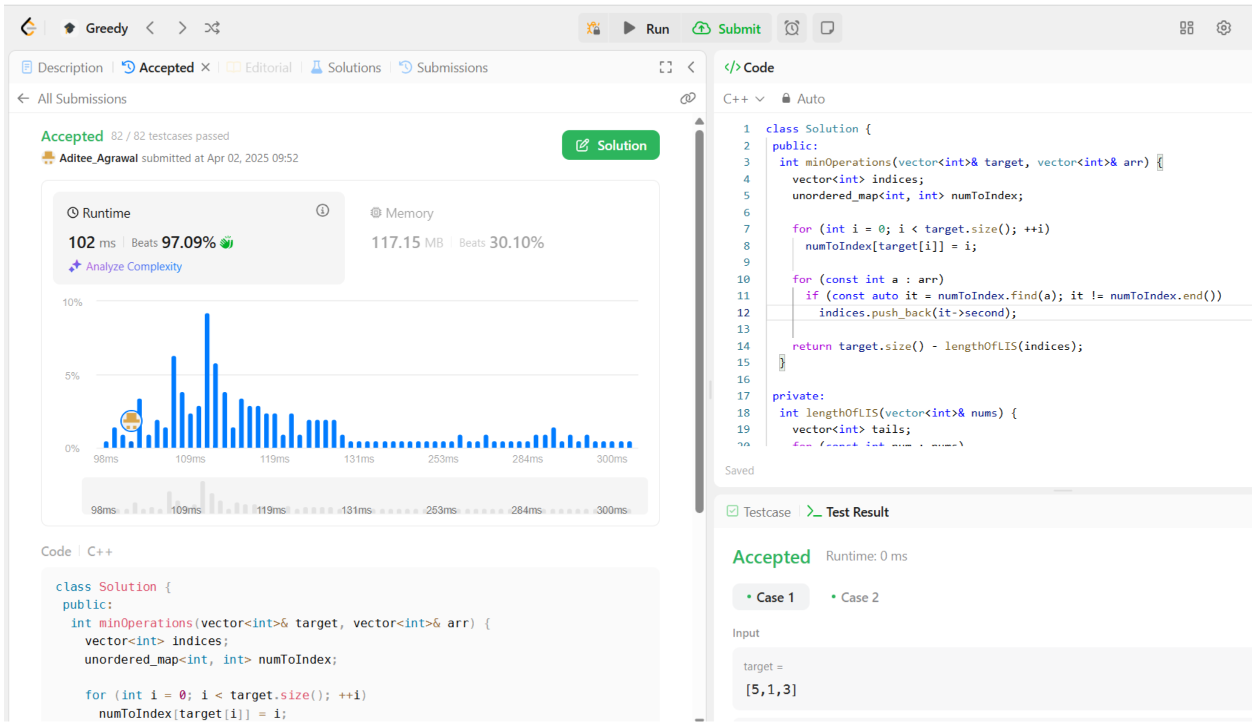
Input

s =

"cdbcbbaaabab"

5. **Minimum Operations to Make a Subsequence**

```
class Solution {
 public:
  int minOperations(vector<int>& target, vector<int>& arr) {
    vector<int> indices;
    unordered_map<int, int> numToIndex;

    for (int i = 0; i < target.size(); ++i)
      numToIndex[target[i]] = i;

    for (const int a : arr)
      if (const auto it = numToIndex.find(a); it != numToIndex.end())
        indices.push_back(it->second);

    return target.size() - lengthOfLIS(indices);
  }

 private:
  int lengthOfLIS(vector<int>& nums) {
    vector<int> tails;
    for (const int num : nums)
      if (tails.empty() || num > tails.back())
        tails.push_back(num);
      else
        tails[firstGreaterEqual(tails, num)] = num;
    return tails.size();
  }

 private:
  int firstGreaterEqual(const vector<int>& arr, int target) {
    return ranges::lower_bound(arr, target) - arr.begin();
  }
};
```



Accepted 82 / 82 testcases passed

Aditee_Agrawal submitted at Apr 02, 2025 09:52

Runtime 102 ms | Beats 97.09%

Memory 117.15 MB | Beats 30.10%

Code | C++

```cpp
class Solution {
 public:
  int minOperations(vector<int>& target, vector<int>& arr) {
    vector<int> indices;
    unordered_map<int, int> numToIndex;

    for (int i = 0; i < target.size(); ++i)
      numToIndex[target[i]] = i;
```

```cpp
class Solution {
 public:
  int minOperations(vector<int>& target, vector<int>& arr) {
    vector<int> indices;
    unordered_map<int, int> numToIndex;

    for (int i = 0; i < target.size(); ++i)
      numToIndex[target[i]] = i;

    for (const int a : arr)
      if (const auto it = numToIndex.find(a); it != numToIndex.end())
        indices.push_back(it->second);

    return target.size() - lengthOfLIS(indices);
  }

 private:
  int lengthOfLIS(vector<int>& nums) {
    vector<int> tails;
```

Accepted Runtime: 0 ms

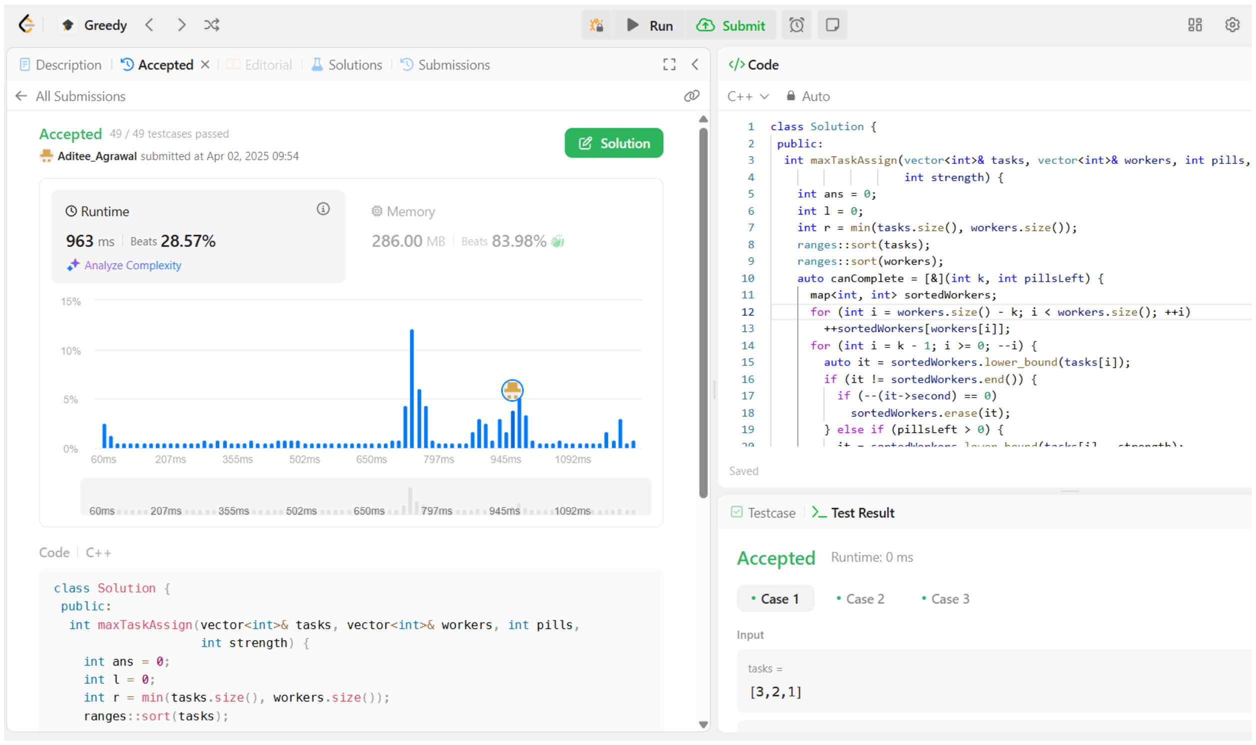
Input

target =

[5,1,3]

6. Maximum Number of Tasks You Can Assign

```cpp
class Solution {
 public:
  int maxTaskAssign(vector<int>& tasks, vector<int>& workers, int pills,
                    int strength) {
    int ans = 0;
    int l = 0;
    int r = min(tasks.size(), workers.size());
    ranges::sort(tasks);
    ranges::sort(workers);
    auto canComplete = [&](int k, int pillsLeft) {
      map<int, int> sortedWorkers;
      for (int i = workers.size() - k; i < workers.size(); ++i)
        ++sortedWorkers[workers[i]];
      for (int i = k - 1; i >= 0; --i) {
        auto it = sortedWorkers.lower_bound(tasks[i]);
        if (it != sortedWorkers.end()) {
          if (--(it->second) == 0)
            sortedWorkers.erase(it);
        } else if (pillsLeft > 0) {
          it = sortedWorkers.lower_bound(tasks[i] - strength);
          if (it != sortedWorkers.end()) {
            if (--(it->second) == 0)
              sortedWorkers.erase(it);
            --pillsLeft;
          } else {
            return false;
          }
        } else {
          return false;
        }
      }

      return true;
    };

    while (l <= r) {
      const int m = (l + r) / 2;
      if (canComplete(m, pills)) {
        ans = m;
        l = m + 1;
      } else {
        r = m - 1;
      }
    }

    return ans;
  }
};
```



Accepted 49 / 49 testcases passed

Aditee_Agrawal submitted at Apr 02, 2025 09:54

Runtime: 963 ms | Beats 28.57%

Memory: 286.00 MB | Beats 83.98%

Code | C++

```cpp
class Solution {
 public:
  int maxTaskAssign(vector<int>& tasks, vector<int>& workers, int pills,
                    int strength) {
    int ans = 0;
    int l = 0;
    int r = min(tasks.size(), workers.size());
    ranges::sort(tasks);
```

```cpp
class Solution {
 public:
  int maxTaskAssign(vector<int>& tasks, vector<int>& workers, int pills,
                    int strength) {
    int ans = 0;
    int l = 0;
    int r = min(tasks.size(), workers.size());
    ranges::sort(tasks);
    ranges::sort(workers);
    auto canComplete = [&](int k, int pillsLeft) {
      map<int, int> sortedWorkers;
      for (int i = workers.size() - k; i < workers.size(); ++i)
        ++sortedWorkers[workers[i]];
      for (int i = k - 1; i >= 0; --i) {
        auto it = sortedWorkers.lower_bound(tasks[i]);
        if (it != sortedWorkers.end()) {
          if (--(it->second) == 0)
            sortedWorkers.erase(it);
        } else if (pillsLeft > 0) {
```

Accepted Runtime: 0 ms

Input

tasks =

[3,2,1]