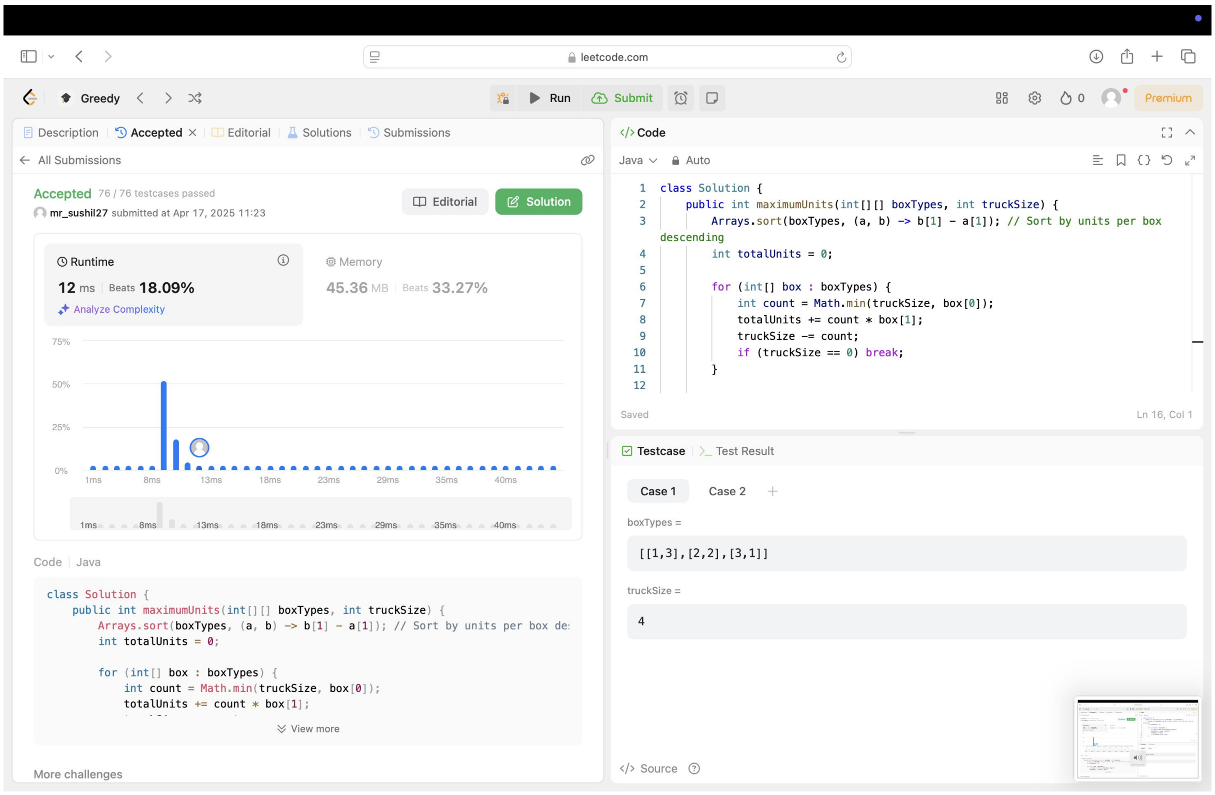
AP Experiment-8

Name: SUSHIL KUMAR   UID: 22BCS16771   Section: 22BCS_614-B

Q1. Maximum Units on a Truck https://leetcode.com/problems/maximum-units-on-a-truck/description/?envType=problem-list-v2&envId=greedy

CODE:

```
class Solution {
    public int maximumUnits(int[][] boxTypes, int truckSize) {
        Arrays.sort(boxTypes, (a, b) -> b[1] - a[1]); // Sort by units per box descending
        int totalUnits = 0;

        for (int[] box : boxTypes) {
            int count = Math.min(truckSize, box[0]);
            totalUnits += count * box[1];
            truckSize -= count;
            if (truckSize == 0) break;
        }

        return totalUnits;
    }
}
```



Accepted 76 / 76 testcases passed
mr_sushil27 submitted at Apr 17, 2025 11:23

Runtime
12 ms | Beats 18.09%
Analyze Complexity

Memory
45.36 MB | Beats 33.27%

Code | Java

```
class Solution {
    public int maximumUnits(int[][] boxTypes, int truckSize) {
        Arrays.sort(boxTypes, (a, b) -> b[1] - a[1]); // Sort by units per box de
        int totalUnits = 0;

        for (int[] box : boxTypes) {
            int count = Math.min(truckSize, box[0]);
            totalUnits += count * box[1];
```

View more

More challenges

```
class Solution {
    public int maximumUnits(int[][] boxTypes, int truckSize) {
        Arrays.sort(boxTypes, (a, b) -> b[1] - a[1]); // Sort by units per box
descending
        int totalUnits = 0;

        for (int[] box : boxTypes) {
            int count = Math.min(truckSize, box[0]);
            totalUnits += count * box[1];
            truckSize -= count;
            if (truckSize == 0) break;
        }
```

Saved   Ln 16, Col 1

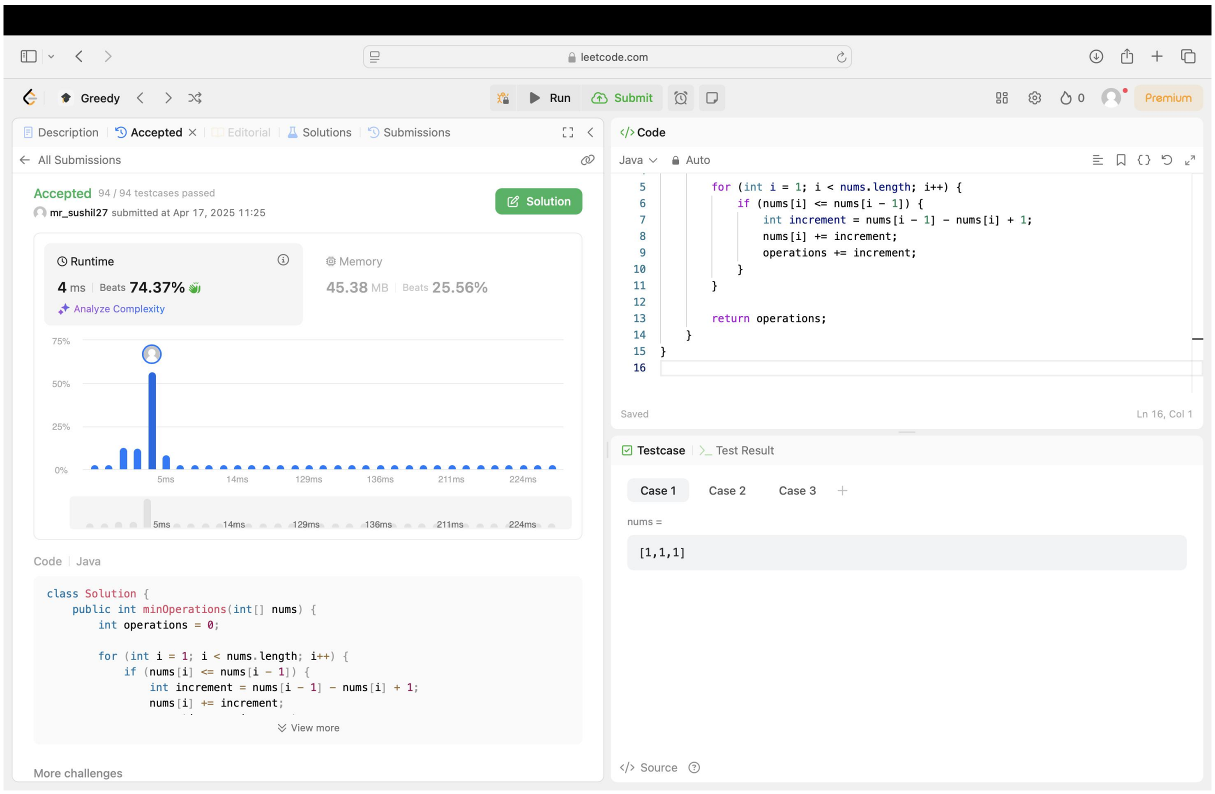
boxTypes =
[[1,3],[2,2],[3,1]]

truckSize =
4

Q2. https://leetcode.com/problems/minimum-operations-to-make-the-array-increasing/description/?envType=problem-list-v2&envId=greedy

CODE:

```
class Solution {
public int minOperations(int[] nums) {
int operations = 0;

for (int i = 1; i < nums.length; i++) {
if (nums[i] <= nums[i - 1]) {
int increment = nums[i - 1] - nums[i] + 1;
nums[i] += increment;
operations += increment;
}
}

return operations;
}
}
```



Accepted 94 / 94 testcases passed
mr_sushil27 submitted at Apr 17, 2025 11:25

Runtime
4 ms | Beats 74.37%
Analyze Complexity

Memory
45.38 MB | Beats 25.56%

Code | Java

```
class Solution {
    public int minOperations(int[] nums) {
        int operations = 0;

        for (int i = 1; i < nums.length; i++) {
            if (nums[i] <= nums[i - 1]) {
                int increment = nums[i - 1] - nums[i] + 1;
                nums[i] += increment;
```

View more

More challenges

Code

Java Auto

```
        for (int i = 1; i < nums.length; i++) {
            if (nums[i] <= nums[i - 1]) {
                int increment = nums[i - 1] - nums[i] + 1;
                nums[i] += increment;
                operations += increment;
            }
        }

        return operations;
    }
}

```

Saved   Ln 16, Col 1

Testcase | Test Result

Case 1   Case 2   Case 3

nums =

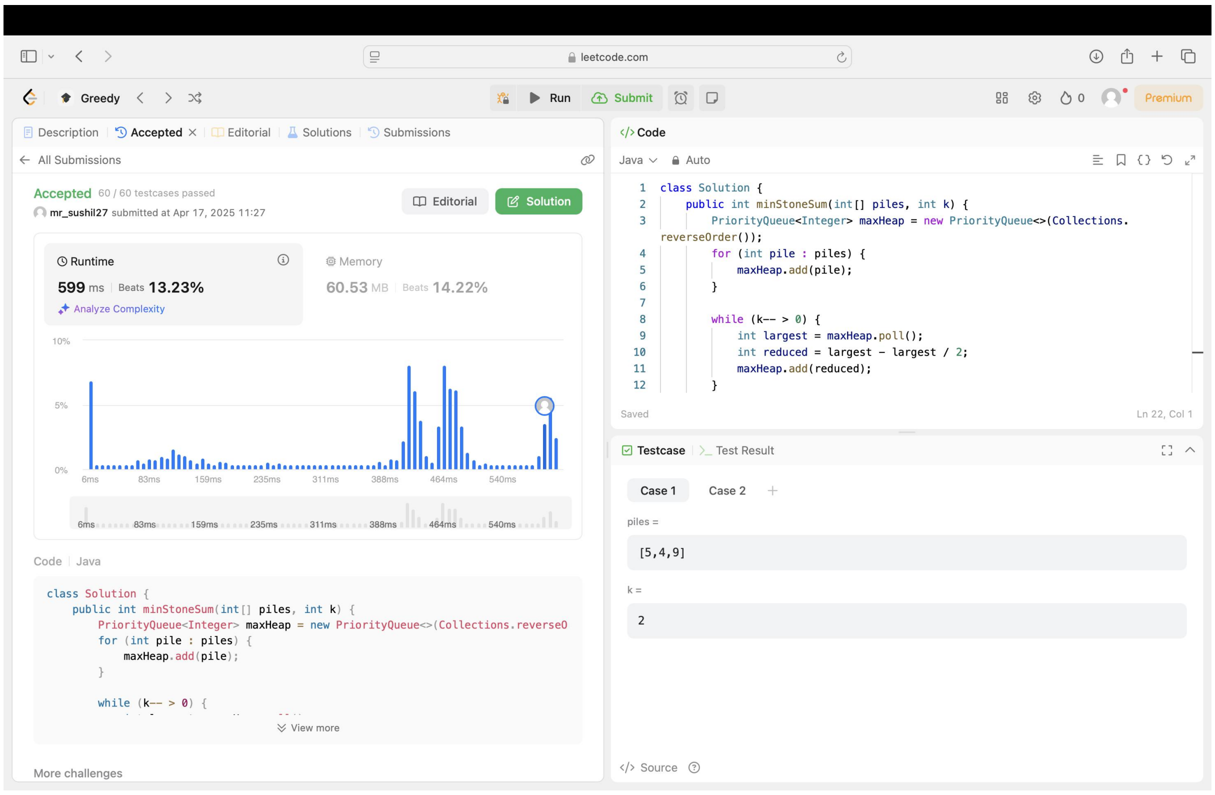
```
[1,1,1]
```

Source

Q3. https://leetcode.com/problems/remove-stones-to-minimize-the-total/description/?envType=problem-list-v2&envId=greedy

CODE:

```
class Solution {
public int minStoneSum(int[] piles, int k) {
PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Collections.reverseOrder());
for (int pile : piles) {
maxHeap.add(pile);
}

while (k-- > 0) {
int largest = maxHeap.poll();
int reduced = largest - largest / 2;
maxHeap.add(reduced);
}

int total = 0;
while (!maxHeap.isEmpty()) {
total += maxHeap.poll();
}

return total;
}
}
```



Accepted 60 / 60 testcases passed
mr_sushil27 submitted at Apr 17, 2025 11:27

Runtime
599 ms | Beats 13.23%

Memory
60.53 MB | Beats 14.22%

10%
5%
0%
6ms 83ms 159ms 235ms 311ms 388ms 464ms 540ms

Code | Java

```
class Solution {
    public int minStoneSum(int[] piles, int k) {
        PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Collections.reverseO
        for (int pile : piles) {
            maxHeap.add(pile);
        }

        while (k-- > 0) {
```

```
class Solution {
    public int minStoneSum(int[] piles, int k) {
        PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Collections.
reverseOrder());
        for (int pile : piles) {
            maxHeap.add(pile);
        }

        while (k-- > 0) {
            int largest = maxHeap.poll();
            int reduced = largest - largest / 2;
            maxHeap.add(reduced);
        }
```

Saved   Ln 22, Col 1

Case 1   Case 2

piles =

```
[5,4,9]
```

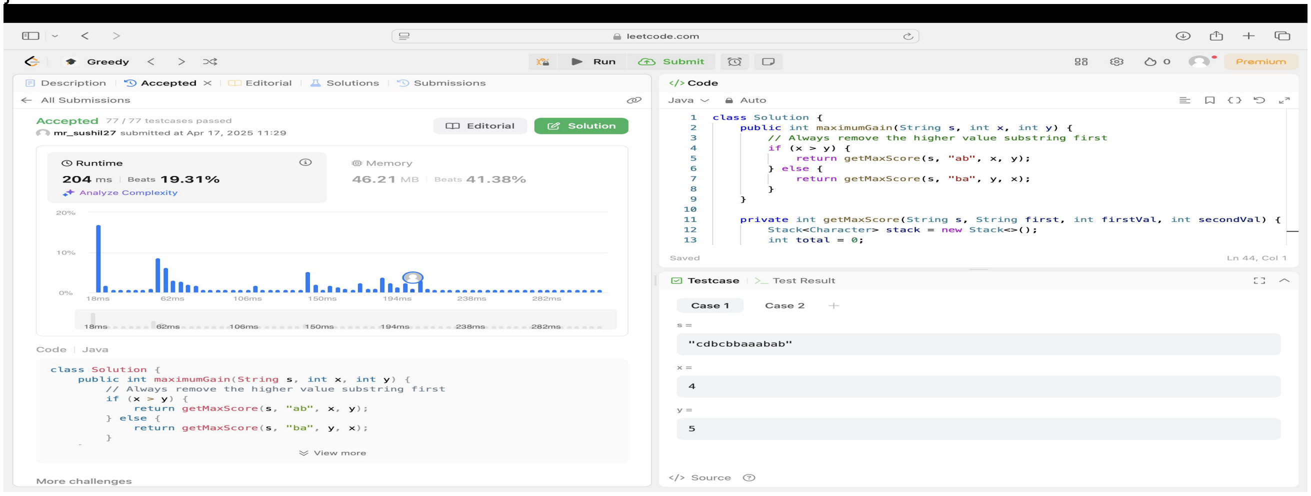
k =

```
2
```

Q4. https://leetcode.com/problems/maximum-score-from-removing-substrings/description/?envType=problem-list-v2&envId=greedy

CODE:

```
class Solution {
public int maximumGain(String s, int x, int y) {
// Always remove the higher value substring first
if (x > y) {
return getMaxScore(s, "ab", x, y);
} else {
return getMaxScore(s, "ba", y, x);
}
}

private int getMaxScore(String s, String first, int firstVal, int secondVal) {
Stack<Character> stack = new Stack<>();
int total = 0;

// First, remove the higher value pattern
for (char c : s.toCharArray()) {
if (!stack.isEmpty() && stack.peek() == first.charAt(0) && c == first.charAt(1)) {
stack.pop();
total += firstVal;
} else {
stack.push(c);
}
}

// Now, remove the second pattern from the remaining string
StringBuilder remaining = new StringBuilder();
while (!stack.isEmpty()) {
remaining.append(stack.pop());
}
remaining.reverse();

for (char c : remaining.toString().toCharArray()) {
if (!stack.isEmpty() && stack.peek() == first.charAt(1) && c == first.charAt(0)) {
stack.pop();
total += secondVal;
} else {
stack.push(c);
}
}

return total;
}
}
```



Accepted 77 / 77 testcases passed
mr_sushil27 submitted at Apr 17, 2025 11:29

Runtime
204 ms | Beats 19.31%

Memory
46.21 MB | Beats 41.38%

Code | Java

```
class Solution {
    public int maximumGain(String s, int x, int y) {
        // Always remove the higher value substring first
        if (x > y) {
            return getMaxScore(s, "ab", x, y);
        } else {
            return getMaxScore(s, "ba", y, x);
        }
```

View more

```
class Solution {
    public int maximumGain(String s, int x, int y) {
        // Always remove the higher value substring first
        if (x > y) {
            return getMaxScore(s, "ab", x, y);
        } else {
            return getMaxScore(s, "ba", y, x);
        }
    }

    private int getMaxScore(String s, String first, int firstVal, int secondVal) {
        Stack<Character> stack = new Stack<>();
        int total = 0;
```

s =
"cdbcbbaaabab"

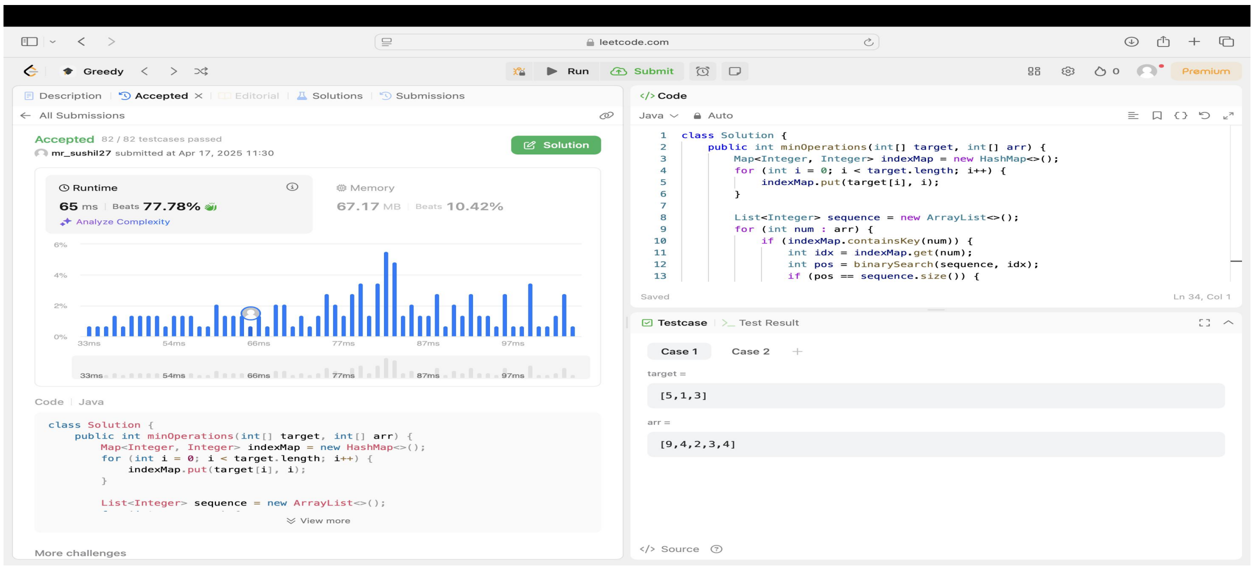
x =
4

y =
5

Q5.https://leetcode.com/problems/minimum-operations-to-make-a-subsequence/description/?envType=problem-list-v2&envId=greedy

CODE:-

```
class Solution {
public int minOperations(int[] target, int[] arr) {
Map<Integer, Integer> indexMap = new HashMap<>();
for (int i = 0; i < target.length; i++) {
indexMap.put(target[i], i);
}

List<Integer> sequence = new ArrayList<>();
for (int num : arr) {
if (indexMap.containsKey(num)) {
int idx = indexMap.get(num);
int pos = binarySearch(sequence, idx);
if (pos == sequence.size()) {
sequence.add(idx);
} else {
sequence.set(pos, idx);
}
}
}

return target.length - sequence.size(); // Minimum insertions
}

private int binarySearch(List<Integer> seq, int target) {
int left = 0, right = seq.size();
while (left < right) {
int mid = (left + right) / 2;
if (seq.get(mid) < target) left = mid + 1;
else right = mid;
}
return left;
}
}
```



Accepted 82 / 82 testcases passed
mr_sushil27 submitted at Apr 17, 2025 11:30

Runtime
65 ms | Beats 77.78%

Memory
67.17 MB | Beats 10.42%

Code | Java

```
class Solution {
    public int minOperations(int[] target, int[] arr) {
        Map<Integer, Integer> indexMap = new HashMap<>();
        for (int i = 0; i < target.length; i++) {
            indexMap.put(target[i], i);
        }

        List<Integer> sequence = new ArrayList<>();
```

```
class Solution {
    public int minOperations(int[] target, int[] arr) {
        Map<Integer, Integer> indexMap = new HashMap<>();
        for (int i = 0; i < target.length; i++) {
            indexMap.put(target[i], i);
        }

        List<Integer> sequence = new ArrayList<>();
        for (int num : arr) {
            if (indexMap.containsKey(num)) {
                int idx = indexMap.get(num);
                int pos = binarySearch(sequence, idx);
                if (pos == sequence.size()) {
```

Saved
Ln 34, Col 1

target =
[5,1,3]

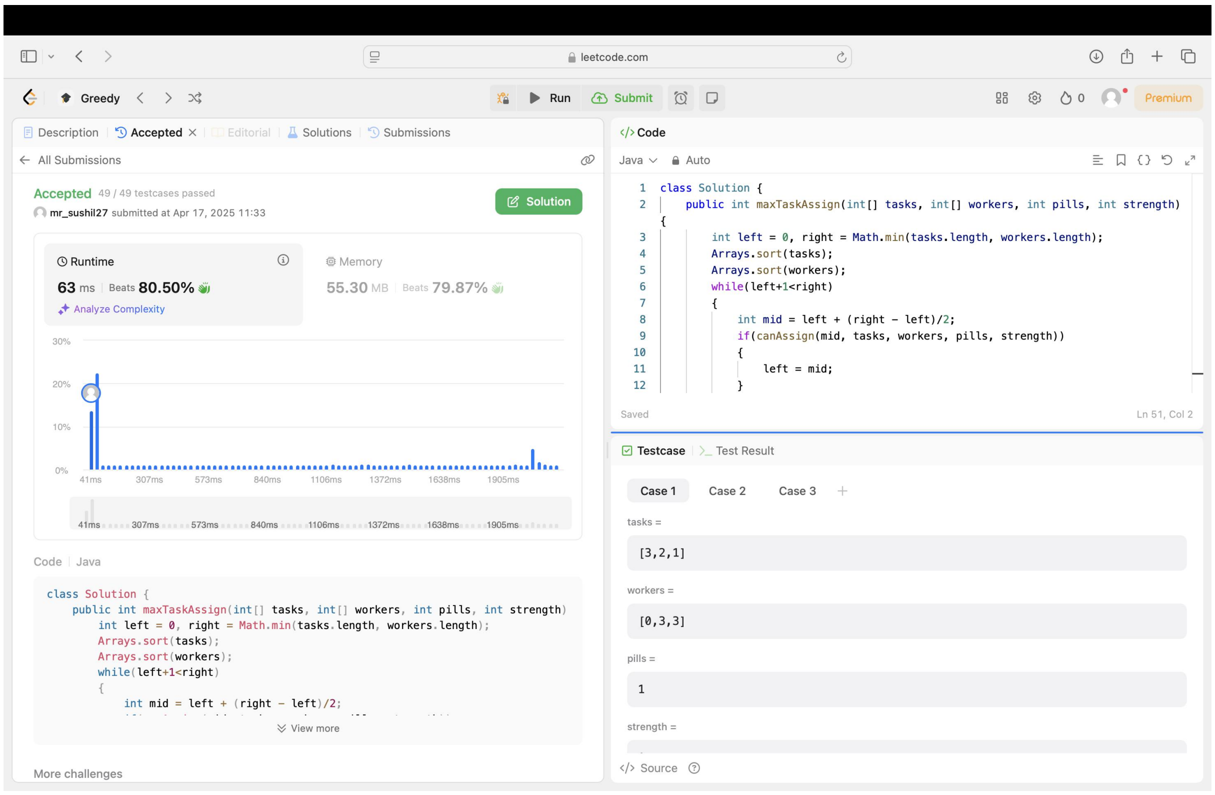
arr =
[9,4,2,3,4]

Q6. https://leetcode.com/problems/maximum-number-of-tasks-you-can-assign/description/?envType=problem-list-v2&envId=greedy

CODE:-

```
class Solution {
public int maxTaskAssign(int[] tasks, int[] workers, int pills, int strength) {
int left = 0, right = Math.min(tasks.length, workers.length);
Arrays.sort(tasks);
Arrays.sort(workers);
while(left+1<right)
{
int mid = left + (right - left)/2;
if(canAssign(mid, tasks, workers, pills, strength))
{
left = mid;
}
else
{
right = mid;
}
}
if(canAssign(right, tasks, workers, pills, strength))
{
return right;
}
else return left;
}
public boolean canAssign(int count, int[] tasks, int[] workers, int pills, int strength){
Deque<Integer> dq = new ArrayDeque<>();
int ind = workers.length - 1;
for (int i = count - 1; i >= 0; i--) {
while(ind>=workers.length-count && workers[ind]+strength>=tasks[i])
{
dq.offerLast(workers[ind]);
ind--;
}
if(dq.isEmpty())return false;
if(dq.peekFirst()>=tasks[i])
{
dq.pollFirst();
}
else
{
```

```
dq.pollLast();
pills--;
if(pills<0)return false;
}
}
return true;
}
}
```



Accepted 49 / 49 testcases passed

mr_sushil27 submitted at Apr 17, 2025 11:33

Runtime

63 ms | Beats 80.50%

Analyze Complexity

Memory

55.30 MB | Beats 79.87%

Code | Java

```
class Solution {
    public int maxTaskAssign(int[] tasks, int[] workers, int pills, int strength)
        int left = 0, right = Math.min(tasks.length, workers.length);
        Arrays.sort(tasks);
        Arrays.sort(workers);
        while(left+1<right)
        {
            int mid = left + (right - left)/2;
```

Java | Auto

```
class Solution {
    public int maxTaskAssign(int[] tasks, int[] workers, int pills, int strength)
{
        int left = 0, right = Math.min(tasks.length, workers.length);
        Arrays.sort(tasks);
        Arrays.sort(workers);
        while(left+1<right)
        {
            int mid = left + (right - left)/2;
            if(canAssign(mid, tasks, workers, pills, strength))
            {
                left = mid;
            }
```

Saved

Ln 51, Col 2

tasks =

[3,2,1]

workers =

[0,3,3]

pills =

1

strength =