



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 8

Student Name: Aditya Agniesz

Branch: CSE

Semester: 6th

Subject Name: AP IAB

UID:22BCS14644

Section/Group:614_B

Date of Performance:03/04/25

Subject Code: 22CSP-351

Ques1. Max Units on a Truck

Code:

```
class Solution {
    public int maximumUnits(int[][] boxTypes, int truckSize) {
        int [] unitCount = new int[1001];

        for (int [] boxType : boxTypes)
        {
            unitCount[boxType[1]] += boxType[0];
        }

        int count = 0;

        for (int i = unitCount.length-1; i >= 0; i--)
        {
            if (unitCount[i] == 0) continue;

            int num = Math.min(unitCount[i], truckSize);
            count += num * i;
            truckSize -= num;
            if (truckSize == 0) break;
        }

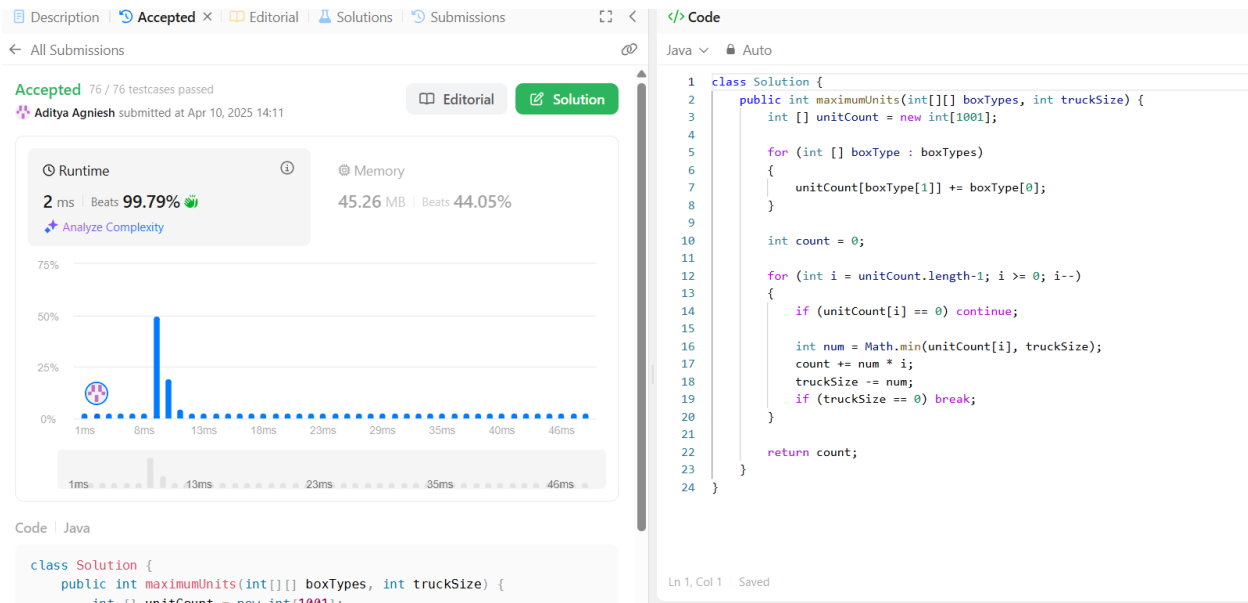
        return count;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Output



Ques2:Min Operations to Make Array Increasing

Code

```
class Solution {

    public int minOperations(int[] nums) {

        int len=nums.length;

        int arr[]=new int[len];

        int count=0;

        for(int i=0; i<len; i++){

            arr[i] = nums[i];

        }

        for(int i=0; i<len-1; i++){

            if(arr[i+1] <= arr[i]){
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        arr[i+1] = arr[i]+1;

    }

}

for(int i=0; i<len; i++){

    count+=arr[i]-nums[i];

}

return count;

}

}
```

Output

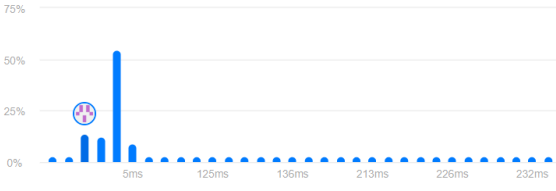
← All Submissions

Accepted 94 / 94 testcases passed
Aditya Agniesz submitted at Apr 10, 2025 14:13

[Solution](#)

Runtime 2 ms | Beats 99.41% 🏆
[Analyze Complexity](#)

Memory 45.09 MB | Beats 64.54% 🏆



Code | Java

```
class Solution {
    public int minOperations(int[] nums) {
        int len=nums.length;
        int arr[]=new int[len];
        int count=0;
        for(int i=0; i<len; i++){
            arr[i] = nums[i];
        }
    }
}
```

[View more](#)

Java Auto

```
1 class Solution {
2     public int minOperations(int[] nums) {
3         int len=nums.length;
4         int arr[]=new int[len];
5         int count=0;
6         for(int i=0; i<len; i++){
7             arr[i] = nums[i];
8         }
9         for(int i=0; i<len-1; i++){
10            if(arr[i+1] <= arr[i]){
11                arr[i+1] = arr[i]+1;
12            }
13        }
14        for(int i=0; i<len; i++){
15            count+=arr[i]-nums[i];
16        }
17        return count;
18    }
19 }
```

Ln 3, Col 9 Saved

☒ Testcase [Test Result](#)

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =
[1,1,1]

Output



Ques 3: Remove Stones to Maximize Total

Code

```
class Solution {
    public int maximumUnits(int[][] boxTypes, int truckSize) {
        int [] unitCount = new int[1001];

        for (int [] boxType : boxTypes)
        {
            unitCount[boxType[1]] += boxType[0];
        }

        int count = 0;

        for (int i = unitCount.length-1; i >= 0; i--)
        {
            if (unitCount[i] == 0) continue;

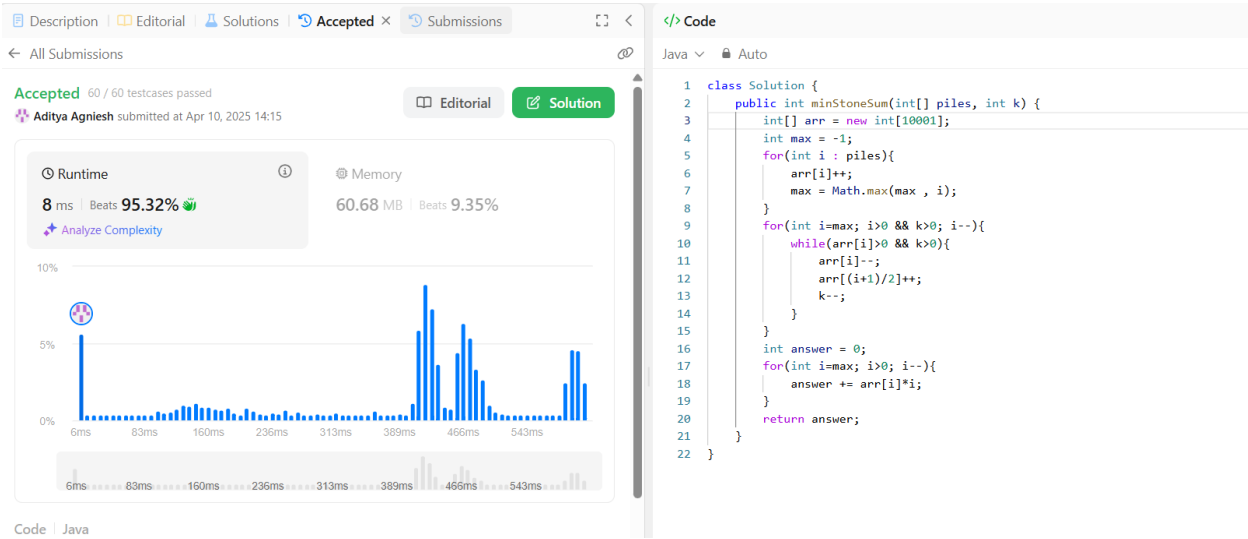
            int num = Math.min(unitCount[i], truckSize);
            count += num * i;
            truckSize -= num;
            if (truckSize == 0) break;
        }

        return count;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.



Ques 4: Max Score from Removing Substrings

Code:

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

        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
            if (c > 'b') {
                result += Math.min(aCount, bCount) * lesser;
                aCount = 0;
                bCount = 0;
            } else if (c == 'a') {
                if (x < y && bCount > 0) {
                    bCount--;
                    result += y;
                } else {
                    aCount++;
                }
            }
        }
    }
}
```

```

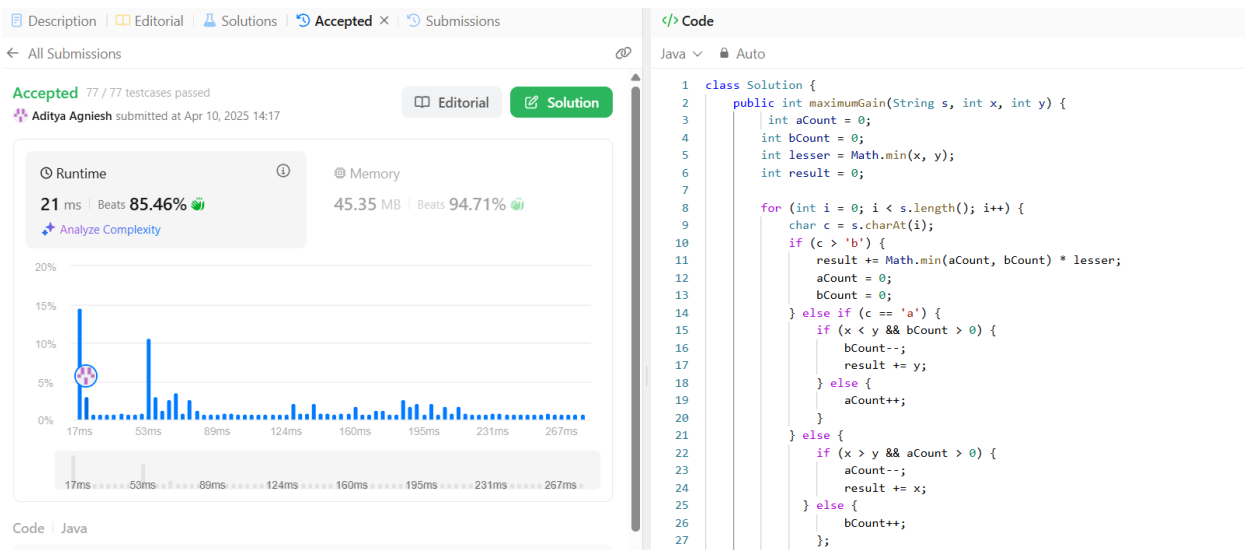
    } else {
        if (x > y && aCount > 0) {
            aCount--;
            result += x;
        } else {
            bCount++;
        }
    }
}

result += Math.min(aCount, bCount) * lesser;

return result;
}
}

```

Output:



Ques 5: Min Operations to Make a Subsequence

Code:

```

class Solution {
    public int minOperations(int[] target, int[] arr) {
        Map<Integer, Integer> map = new HashMap<>();
    }
}

```

```
        for (int i = 0; i < target.length; i++) {
            map.put(target[i], i);
        }

        List<Integer> sequence = new ArrayList<>();
        for (int num : arr) {
            if (map.containsKey(num)) {
                sequence.add(map.get(num));
            }
        }

        int maxSubsequence = lis(sequence);
        return target.length - maxSubsequence;
    }

    private int lis(List<Integer> list) {
        List<Integer> max = new ArrayList<>();
        for (int num : list) {
            if (max.isEmpty() || num > max.get(max.size() - 1)) {
                max.add(num);
            } else {
                int pos = Collections.binarySearch(max, num);
                if (pos >= 0) {
                    max.set(pos, num);
                } else {
                    max.set(-(pos + 1), num);
                }
            }
        }
        return max.size();
    }
}
```

Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Description

Editorial

Solutions

Accepted

Submissions

All Submissions

Accepted 82 / 82 testcases passed

Aditya Agnesh submitted at Apr 10, 2025 14:20

Solution

Runtime

78 ms | Beats 61.47%

Analyze Complexity

Memory

62.52 MB | Beats 66.97%

Code | Java

```
class Solution {  
    public int minOperations(int[] target, int[] arr) {  
        Map<Integer, Integer> map = new HashMap<>();  
        for (int i = 0; i < target.length; i++) {  
            map.put(target[i], i);  
        }  
  
        List<Integer> sequence = new ArrayList<>();  
        for (int num : arr) {  
            if (map.containsKey(num)) {  
                sequence.add(map.get(num));  
            }  
        }  
  
        int maxSubsequence = lis(sequence);  
        return target.length - maxSubsequence;  
    }  
  
    private int lis(List<Integer> list) {  
        List<Integer> max = new ArrayList<>();  
        for (int num : list) {  
            if (max.isEmpty() || num > max.get(max.size() - 1)) {  
                max.add(num);  
            } else {  
                int pos = Collections.binarySearch(max, num);  
                if (pos >= 0) {  
                    max.set(pos, num);  
                }  
            }  
        }  
        return max.size();  
    }  
}
```