Experiment 8

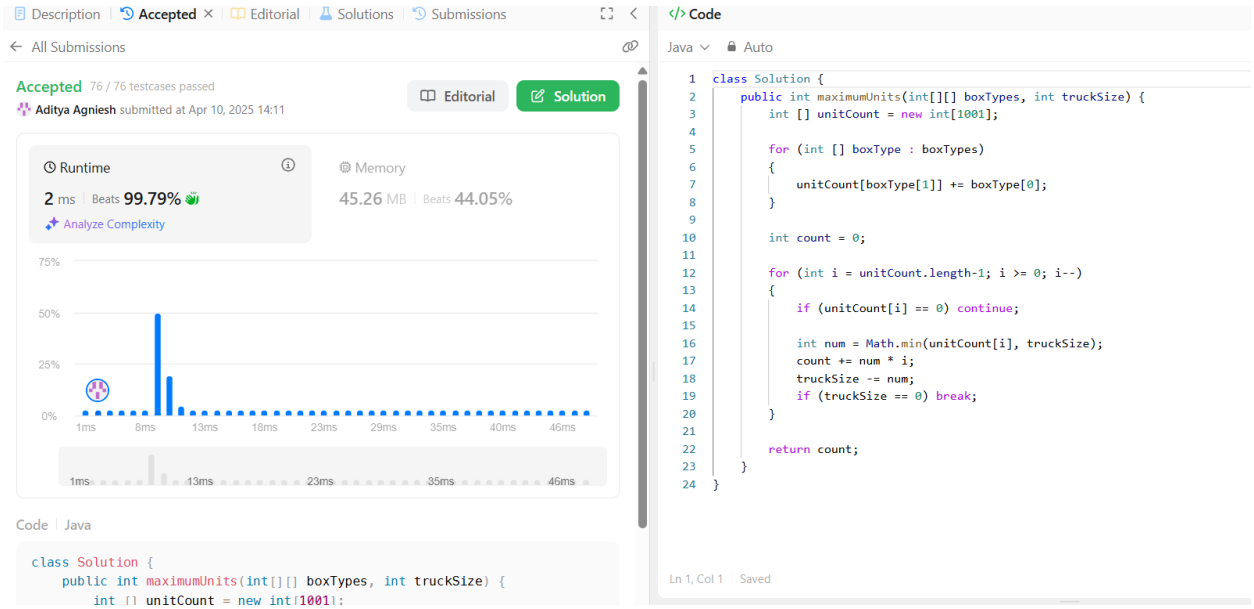
Student Name: Aditya Agniesh **UID:22BCS14644**
Branch: CSE **Section/Group:614_B**
Semester: 6th **Date of Performance:03/04/25**
Subject Name: AP lAB **Subject Code: 22CSP-351**

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;
    }
}
```

Output



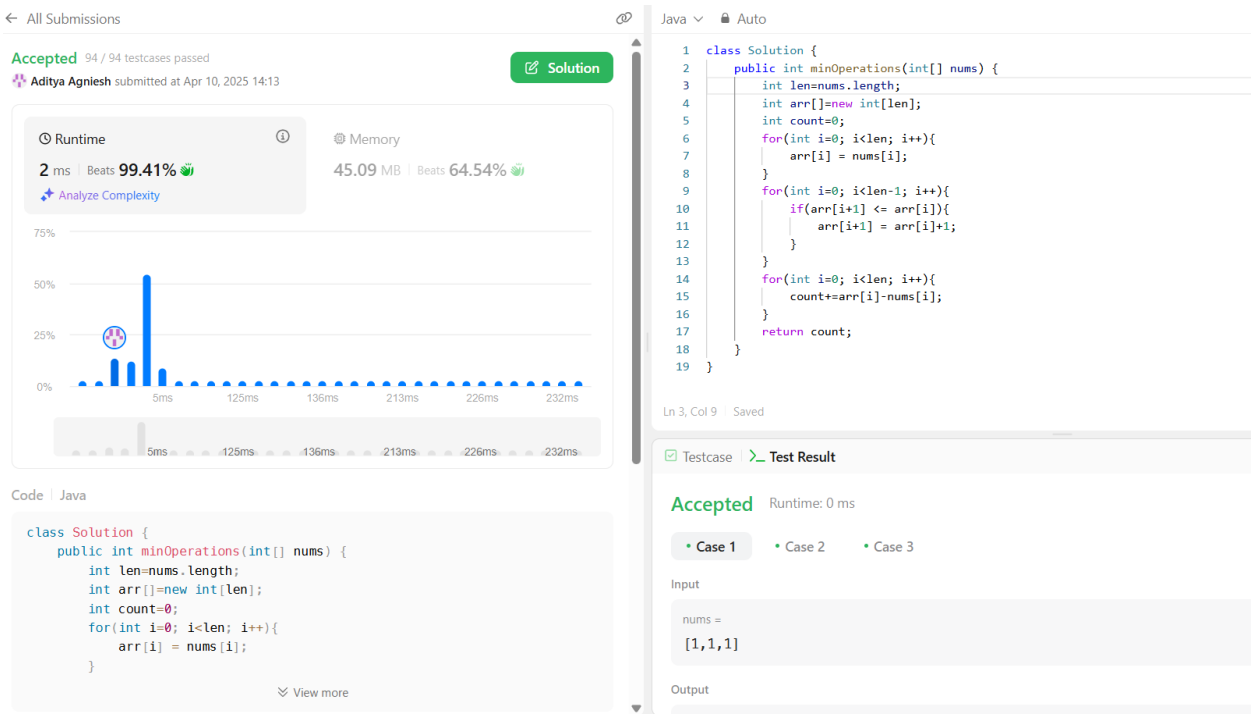
Ques2:Min Operations to Make Array Increasing

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]){
```

```
            arr[i+1] = arr[i]+1;
        }
    }
    for(int i=0; i<len; i++){
        count+=arr[i]-nums[i];
    }
    return count;
  }
}
```

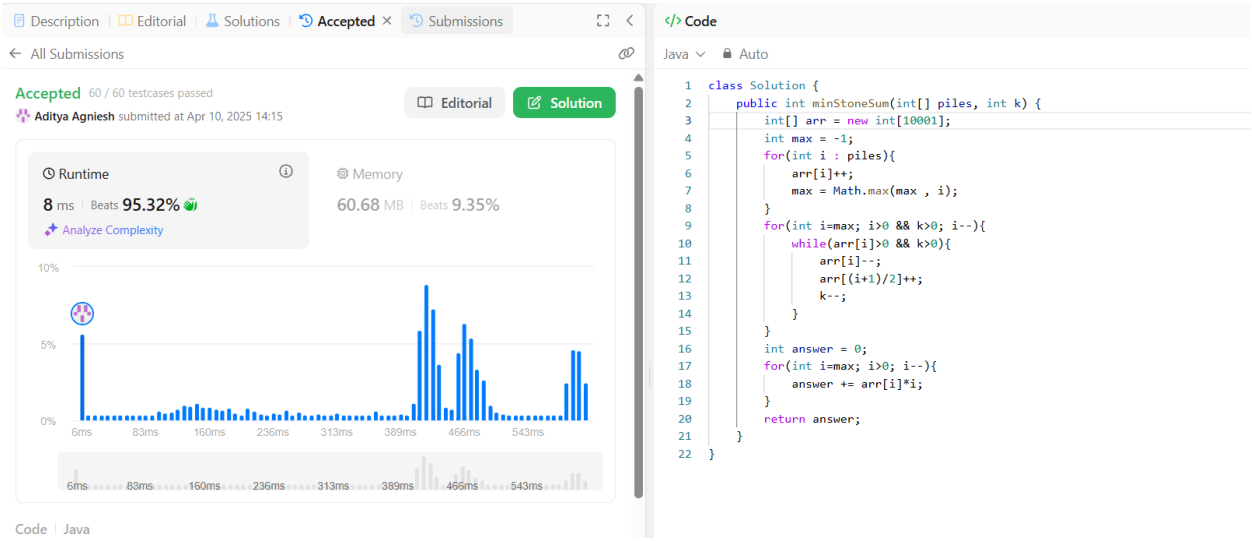
Output



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;
   }
}
```



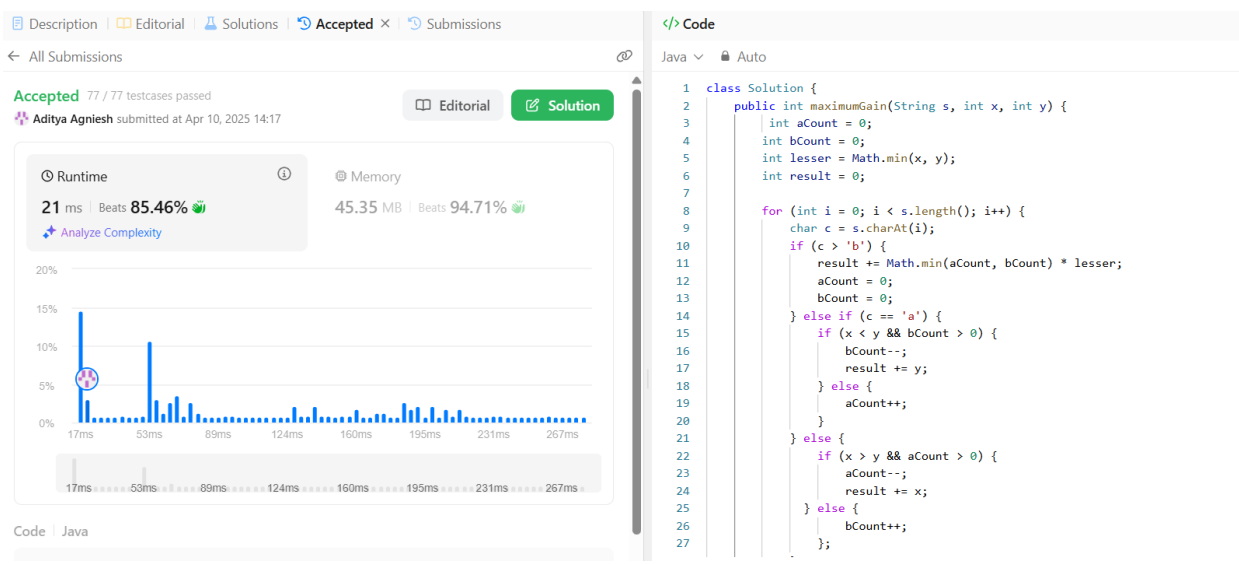
Ques 4: Max Score from Removing Substrings

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:

Description | Editorial | Solutions | Accepted | Submissions

All Submissions

Accepted 82 / 82 testcases passed

Aditya Agniesh 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 {
```

Code

Java | Auto

```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);
```