Experiment 7 A

Student Name: PARDEEP SINGH **UID: 22BCS16692**

Branch: CSE **Section/Group: Ntpp 602-A**

Semester: 6TH **Date of Performance:06/03/25**

Subject Name: AP Lab-2 **Subject Code: 22CSH-352**

1. TITLE:

Max Units on a Truck

2. AIM:

You are assigned to put some amount of boxes onto one truck. You are given a 2D array boxTypes, where boxTypes[i] = [numberOfBoxesi, numberOfUnitsPerBoxi]:

numberOfBoxesi is the number of boxes of type i.

numberOfUnitsPerBoxi is the number of units in each box of the type i.

3. Algorithm

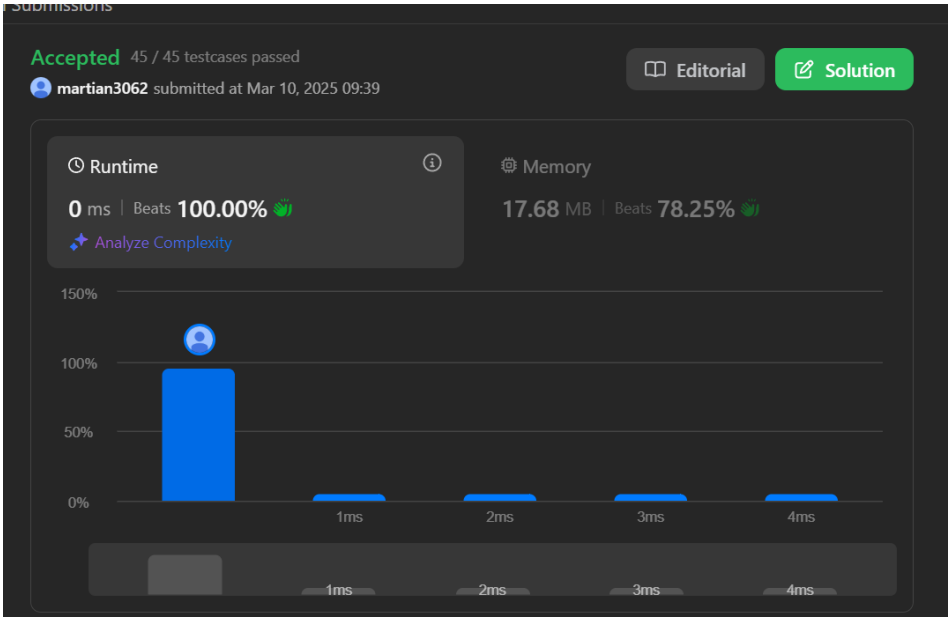
- Sort box types by the number of units in descending order.
- Initialize a counter to track the total units.
- Loop through sorted boxes and add them to the truck if space allows.
- Add boxes to the truck while updating the total units.
- Reduce the truck size as boxes are added, stop when the truck is full.

Implemetation/Code

```
class Solution:
    def maximumUnits(self, boxTypes: List[List[int]], truckSize: int) -> int:
        b = sorted(boxTypes, key = lambda x:x[1], reverse=True )
        c = 0
        for box, n in b:
```

```
    if truckSize ==0:
        return c
    boxes = min(box, truckSize)
    c += boxes * n
    truckSize -= boxes
return c
```

Output



Time Complexity : O(n.logn)

Space Complexity : O(n)

Learning Outcomes:-

- Learn how greedy algorithms make optimal choices for immediate benefits.
- Understand how to optimize limited resources for maximum output

Experiment 7 B

Student Name: PARDEEP SINGH **UID: 22BCS16692**

Branch: CSE **Section/Group: Ntpp 602-A**

Semester: 6TH **Date of Performance:06/03/25**

Subject Name: AP Lab-2 **Subject Code: 22CSH-352**

1. TITLE:

Max Score from removing substrings

2. AIM:

You are given a string s and two integers x and y. You can perform two types of operations any number of times.

Remove substring "ab" and gain x points.

For example, when removing "ab" from "cabxbae" it becomes "cxbae".

Remove substring "ba" and gain y points.

For example, when removing "ba" from "cabxbae" it becomes "cabxe".

Return the maximum points you can gain after applying the above operations on s.

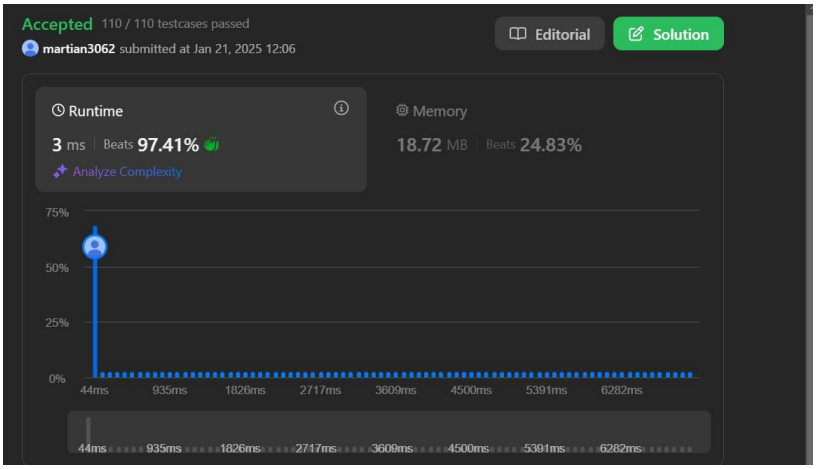
3. Algorithm

1. Determine which pair (ab or ba) gives a higher score by comparing x and y.
2. Initialize counters for 'a' and 'b' and a variable for the total score.
3. Loop through the string and process 'a' and 'b' characters.
4. Add points when an 'a' is followed by a 'b' or vice versa, adjusting counters.
5. Add final score for any remaining unmatched 'a' and 'b' pairs.

Implemetation/Code:

```
class Solution:
    def maximumGain(self, s: str, x: int, y: int) -> int:
        a, b = "a", "b"
        if x < y:
            x, y = y, x
            a, b = b, a
        ans = cnt0 = cnt1 = 0
        for c in s:
            if c not in "ab":
                ans += min(cnt0, cnt1) * y
                cnt0 = cnt1 = 0
            elif c == b:
                if cnt0:
                    cnt0 -= 1
                    ans += x
                else: cnt1 += 1
            else: cnt0 += 1
        return ans + min(cnt0, cnt1) * y
```

Output



Time Complexity : O(n)

Space Complexity : O(1)

Learning Outcomes:-

- Learn how to make optimal choices in sequential problems.
- Understand how to process strings efficiently with minimal resources.