

AP ASSIGNMENT - 5

Name : Naman Kumar

UID : 22ICS10001

Class : FL_IOT_604 (A)

389. Find the Difference

```
class Solution:
    def findTheDifference(self, s: str, t: str) -> str:
        res = 0
        for i in (s+t):
            res ^= ord(i)
        return chr(res)
```

The screenshot shows a web browser displaying a LeetCode submission for problem 389, "Find the Difference". The URL is <https://leetcode.com/problems/find-the-difference/submissions/1568228068/>. The submission is marked as "Accepted" with 54/54 test cases passed, submitted on Mar 09, 2025 at 21:00. The runtime is 3 ms, beating 53.47% of submissions, and the memory usage is 17.95 MB, beating 24.68%. A bar chart shows the runtime distribution. The code is written in Python3 and is as follows:

```
class Solution:
    def findTheDifference(self, s: str, t: str) -> str:
        res = 0
        for i in (s+t):
            res ^= ord(i)
        return chr(res)
```

The test result for Case 1 shows the input `s = "abcd"` and the output is "Accepted" with a runtime of 0 ms.

976. Largest Perimeter Triangle

```
from typing import List

class Solution:
    def largestPerimeter(self, nums: List[int]) -> int:
        nums.sort()
        for i in range(len(nums) - 1, 1, -1):
            if nums[i - 1] + nums[i - 2] > nums[i]:
                return nums[i - 1] + nums[i - 2] + nums[i]
        return 0
```

leetcod.com/problems/largest-perimeter-triangle/

Problem List < > >

Description Accepted x Editorial Solution

All Submissions

Accepted 84 / 84 testcases passed
submitted at Mar 09, 2025 21: [Solution](#)

Runtime 11 ms Beats 83.20% [Analyze Complexity](#)

Memory 18.71 MB Beats 46.00%

20%
10%
0%
2ms 12ms 23ms 33ms

Code Python3

```
from typing import List

class Solution:
```

Python3 Auto

```
1 from typing import List
2
3 class Solution:
4     def largestPerimeter(self, nums: List[int]) -> int:
5         nums.sort()
6         for i in range(len(nums) - 1, 1, -1):
7             if nums[i - 1] + nums[i - 2] > nums[i]:
8                 return nums[i - 1] + nums[i - 2] + nums[i]
9         return 0
10
```

Saved Ln 10, Col 1

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
nums =
[2,1,2]
```

414. Third Maximum Number

```
class Solution:
    def thirdMax(self, nums: List[int]) -> int:
        n = list(set(nums))
        if len(n) == 1: return n[0]
        if len(n) == 2: return max(n)
```

```

small = min(n)
first = small
second = small
third = small
for num in n:
    if num >= first:
        third = second
        second = first
        first = num
    elif num >= second:
        third = second
        second = num
    elif num > third:
        third = num
return third

```

leetcode.com/problems/third-maximum-number/submissions/1568230257

Gmail YouTube CUIIMS Outlook yes movies DSA Telegram

Problem List Run Submit

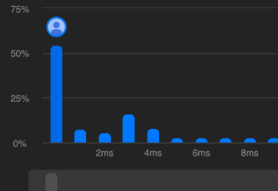
Description Accepted x Editorial Solution

All Submissions

Accepted 34/34 testcases passed
submitted at Mar 09, 2025 21: [Solution](#)

Runtime
0 ms | Beats 100.00% [Analyze Complexity](#)

Memory
18.99 MB | Beats 61.54%



Code | Python3

```

class Solution:
    def thirdMax(self, nums: List[int]) -> int:
        n = list(set(nums))

```

Python3 Auto

```

1 small = min(n)
2 first = small
3 second = small
4 third = small
5 for num in n:
6     if num >= first:
7         third = second
8         second = first
9         first = num
10    elif num >= second:
11        third = second
12        second = num
13    elif num > third:
14        third = num
15 return third

```

Saved Ln 21, Col 21

Testcase Test Result

Case 1 Case 2 Case 3 +

nums =

[3,2,1]

</> Source

451. Sort Characters By Frequency

```
from collections import Counter

class Solution:
    def frequencySort(self, s: str) -> str:
        char_frequency = Counter(s)
        sorted_characters = sorted(char_frequency.items(), key=lambda item:
- item[1])
        return ''.join(character * frequency for character, frequency in
sorted_characters)
```

The screenshot shows the LeetCode interface for problem 451. The code is in Python3 and uses Counter to sort characters by frequency. The runtime is 3 ms, beating 97.56% of solutions. The memory is 18.91 MB, beating 63.07% of solutions. The test result is Accepted.

Code:

```
1 from collections import Counter
2
3 class Solution:
4     def frequencySort(self, s: str) -> str:
5         char_frequency = Counter(s)
6         sorted_characters = sorted(char_frequency.items(), key=lambda item: -item[1])
7         return ''.join(character * frequency for character, frequency in sorted_characters)
```

Runtime: 3 ms | Beats 97.56%

Memory: 18.91 MB | Beats 63.07%

Test Result: Accepted Runtime: 0 ms

Case 1: Input: s = "tree"

452. Minimum Number of Arrows to Burst Balloons

```
class Solution:
    def findMinArrowShots(self, points: List[List[int]]) -> int:
        num_arrows, last_arrow_pos = 0, float('-inf')
        sorted_points = sorted(points, key=lambda x: x[1])

        for start, end in sorted_points:
            if start > last_arrow_pos:
                num_arrows += 1
                last_arrow_pos = end

        return num_arrows
```

The screenshot shows a LeetCode submission for the problem "452. Minimum Number of Arrows to Burst Balloons". The submission is in Python3 and is marked as "Accepted". The code is as follows:

```
1 class Solution:
2     def findMinArrowShots(self, points: List[List[int]]) -> int:
3         num_arrows, last_arrow_pos = 0, float('-inf')
4         sorted_points = sorted(points, key=lambda x: x[1])
5
6         for start, end in sorted_points:
7             if start > last_arrow_pos:
8                 num_arrows += 1
9                 last_arrow_pos = end
10
11         return num_arrows
12
```

The runtime statistics show 50 ms, which beats 99.72% of other submissions. The memory usage is 51.34 MB, which beats 77.00% of other submissions. The test result is "Accepted" with a runtime of 0 ms. The input for the test case is:

```
points =
[[10,16], [2,8], [1,6], [7,12]]
```

881. Boats to Save People

```
class Solution:
    def numRescueBoats(self, people: List[int], limit: int) -> int:
        people.sort()
        heavy = len(people) - 1
        light = 0
        boats = 0
        while light <= heavy:
            if people[light] + people[heavy] <= limit:
                light += 1
            heavy -= 1
            boats += 1
        return boats
```

The screenshot displays the LeetCode interface for the problem "881. Boats to Save People". The submission is marked as "Accepted" with 78/78 test cases passed. The runtime is 47 ms, which is 70.01% faster than the average. The memory usage is 23.38 MB, which is 21.93% less than the average. A bar chart shows the distribution of runtime times across 78 test cases, with a peak around 42ms. The code editor shows the following Python solution:

```
1 class Solution:
2     def numRescueBoats(self, people: List[int], limit: int) -> int:
3         people.sort()
4         heavy = len(people) - 1
5         light = 0
6         boats = 0
7         while light <= heavy:
8             if people[light] + people[heavy] <= limit:
9                 light += 1
10            heavy -= 1
11            boats += 1
12        return boats
```

The test result section shows "Accepted" with a runtime of 0 ms. The input for the test case is:

```
people = [1,2]
```

973. K Closest Points to Origin

```
class Solution:
    def kClosest(self, points: list[list[int]], k: int) -> list[list[int]]:
        maxHeap = []

        for x, y in points:
            heapq.heappush(maxHeap, (- x * x - y * y, [x, y]))
            if len(maxHeap) > k:
                heapq.heappop(maxHeap)

        return [pair[1] for pair in maxHeap]
```

The screenshot displays the LeetCode submission interface for problem 973. The code is written in Python3 and is shown in the 'Code' tab. The runtime statistics show a runtime of 82 ms, which beats 36.75% of submissions. The memory usage is 22.83 MB, which beats 43.19% of submissions. The test results show that the code is 'Accepted' and passed all test cases.

Runtime: 82 ms | Beats 36.75%

Memory: 22.83 MB | Beats 43.19%

Testcase: Accepted | Runtime: 0 ms

Case 1: Input: points = [[1,3],[-2,2]]

1338. Reduce Array Size to The Half

```
from typing import List
from collections import Counter

class Solution:
    def minSetSize(self, arr: List[int]) -> int:
        frequency_counter = Counter(arr)
        min_set_size = 0
        removed_count = 0

        for _, frequency in frequency_counter.most_common():
            removed_count += frequency
            min_set_size += 1
            if removed_count * 2 >= len(arr):
                break

        return min_set_size
```

The screenshot displays the LeetCode interface for the problem "1338. Reduce Array Size to The Half". The submission is marked as "Accepted" with 33/33 testcases passed. The runtime is 57ms, which is 53.53% faster than the average. The memory usage is 41.08 MB, which is 39.58% less than the average. The code editor shows the following Python code:

```
1 from typing import List
2 from collections import Counter
3
4 class Solution:
5     def minSetSize(self, arr: List[int]) -> int:
6         frequency_counter = Counter(arr)
7         min_set_size = 0
8         removed_count = 0
9
10        for _, frequency in frequency_counter.most_common():
11            removed_count += frequency
12            min_set_size += 1
13            if removed_count * 2 >= len(arr):
14                break
15
16        return min_set_size
17
```

The input array is [3, 3, 3, 3, 5, 5, 2, 2, 7]. The output is 4.