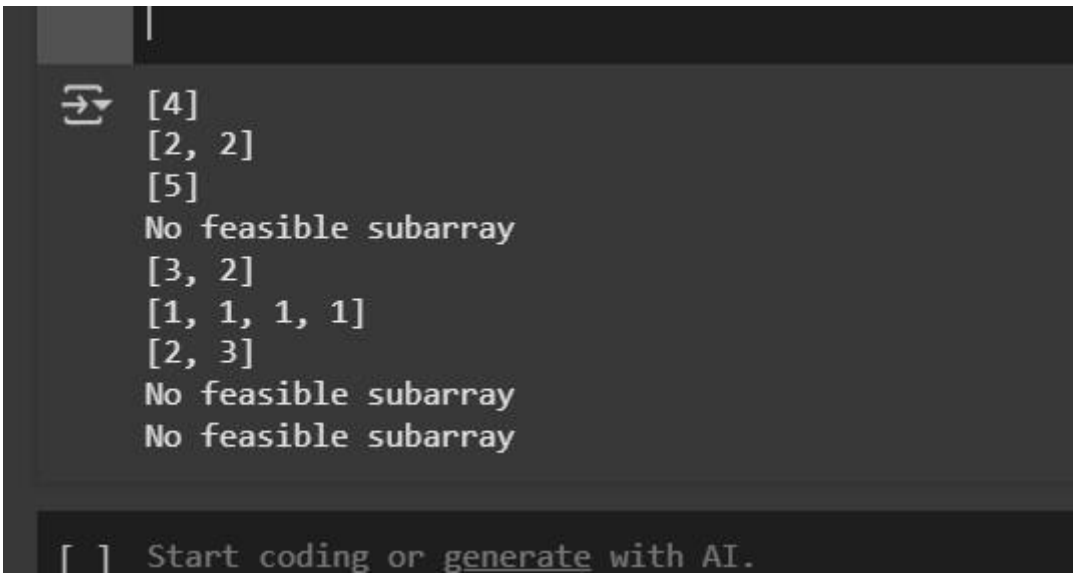
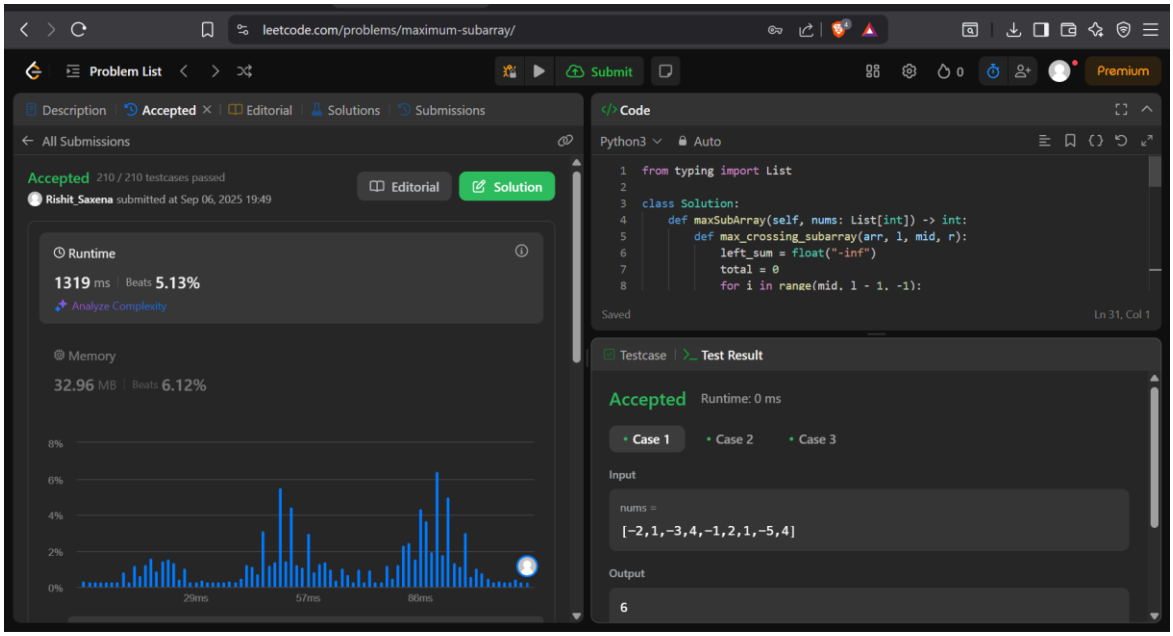
```
RISHIT SAXENA A5 B2 29 def
divide_array(arr, constraint):
if not arr or constraint <= 0:
        return "No feasible subarray"
    if len(arr) == 1:
        return (arr if arr[0] <= constraint else "No feasible subarray")

    mid = len(arr) // 2
left_half = arr[:mid]
right_half = arr[mid:]

    best_left = divide_array(left_half, constraint)
best_right = divide_array(right_half, constraint)

    max_cross = []     total = 0
left_best = []     for i in
range(mid - 1, -1, -1):
        total += arr[i]
        if total <= constraint and total > sum(left_best):
            left_best = arr[i:mid]

    total = 0     right_best = []
for j in range(mid, len(arr)):
        total += arr[j]
        if total + sum(left_best) <= constraint and total + sum(left_best) > sum(left_best + right_best):
            right_best = arr[mid:j+1]

    max_cross = left_best + right_best

    candidates = []     if isinstance(best_left, list):
candidates.append(best_left)     if isinstance(best_right, list):
```

```
candidates.append(best_right)     if max_cross:
candidates.append(max_cross)


    if not candidates:
        return "No feasible subarray"


    return max(candidates, key=sum)

print(divide_array([2, 1, 3, 4], 5))
print(divide_array([2, 2, 2, 2], 4))
print(divide_array([1, 5, 2, 3], 5))
print(divide_array([6, 7, 8], 5))
print(divide_array([1, 2, 3, 2, 1], 5))
print(divide_array([1, 1, 1, 1, 1], 4))
print(divide_array([4, 2, 3, 1], 5))
print(divide_array([], 10)) print(divide_array([1,
2, 3], 0))
```




leetcode.com/problems/maximum-subarray/
Problem List
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 210 / 210 testcases passed
Rishit_Saxena submitted at Sep 06, 2025 19:49
Editorial
Solution
Runtime
1319 ms | Beats 5.13%
Analyze Complexity
Memory
32.96 MB | Beats 6.12%
8%
6%
4%
2%
0%
29ms
57ms
86ms
Code
Python3
Auto
1 from typing import List
2
3 class Solution:
4 def maxSubArray(self, nums: List[int]) -> int:
5 def max_crossing_subarray(arr, l, mid, r):
6 left_sum = float("-inf")
7 total = 0
8 for i in range(mid. l - 1. -1):
Saved
Ln 31, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
nums =
[-2,1,-3,4,-1,2,1,-5,4]
Output
6