

**DAA**  
**Name: Aayush Patle**  
**Batch: A5-B4-58**  
**Practical - 4**

## Task A:

```
def max_subarray(arr, constraint):  
    def max_crossing_sum(low, mid, high):  
        left_sum = float('-inf')  
        total = 0  
        best_l = mid  
        for i in range(mid, low - 1, -1):  
            total += arr[i]  
            if total <= constraint and total > left_sum:  
                left_sum = total  
                best_l = i  
  
        right_sum = float('-inf')  
        total = 0  
        best_r = mid  
        for j in range(mid + 1, high + 1):  
            total += arr[j]  
            if total <= constraint and total > right_sum:  
                right_sum = total  
                best_r = j  
  
        if left_sum != float('-inf') and right_sum != float('-inf'):  
            if left_sum + right_sum <= constraint:
```

```

        return (left_sum + right_sum, best_l, best_r)

    if left_sum != float('-inf') and left_sum <= constraint:
        return (left_sum, best_l, mid)

    if right_sum != float('-inf') and right_sum <= constraint:
        return (right_sum, mid + 1, best_r)

    return (0, -1, -1)

def solve(low, high):
    if low == high:
        if arr[low] <= constraint:
            return (arr[low], low, high)
        else:
            return (0, -1, -1)

    mid = (low + high) // 2
    left_ans = solve(low, mid)
    right_ans = solve(mid + 1, high)
    cross_ans = max_crossing_sum(low, mid, high)

    candidates = [ans for ans in (left_ans, right_ans, cross_ans) if ans[1] != -1]
    if not candidates:
        return (0, -1, -1)

    return max(candidates, key=lambda x: x[0])

if not arr or constraint <= 0:
    return (0, -1, -1)

return solve(0, len(arr) - 1)

```

```

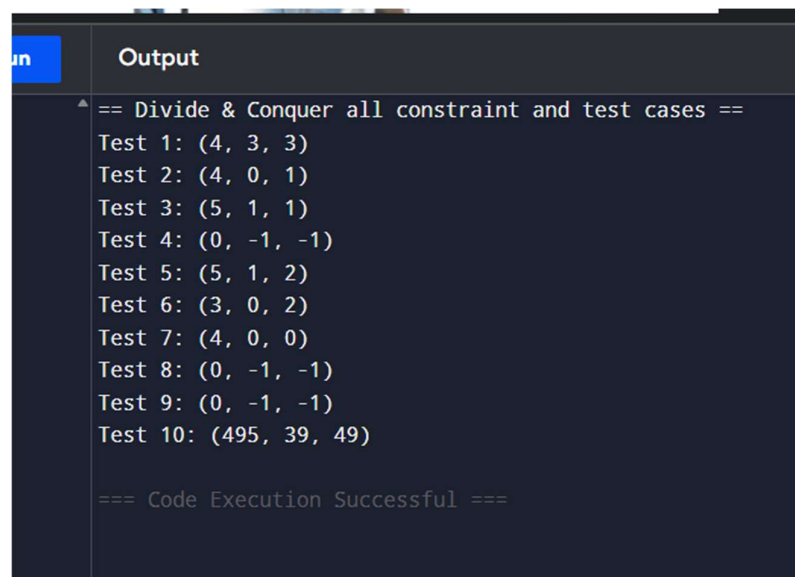
tests = [
    ([2, 1, 3, 4], 5),
    ([2, 2, 2, 2], 4),
    ([1, 5, 2, 3], 5),
    ([6, 7, 8], 5),
    ([1, 2, 3, 2, 1], 5),
    ([1, 1, 1, 1, 1], 4),
    ([4, 2, 3, 1], 5),
    ([], 10),
    ([1, 2, 3], 0),
    (list(range(1, 101)), 500)
]

```

```

print("== Divide & Conquer all constraint and test cases ==")
for i in range(len(tests)):
    arr, cons = tests[i]
    ans = max_subarray(arr, cons)
    print(f"Test {i + 1}: {ans}")

```



The screenshot shows a dark-themed output window with a blue tab on the left. The output text is as follows:

```

Output
^ == Divide & Conquer all constraint and test cases ==
Test 1: (4, 3, 3)
Test 2: (4, 0, 1)
Test 3: (5, 1, 1)
Test 4: (0, -1, -1)
Test 5: (5, 1, 2)
Test 6: (3, 0, 2)
Test 7: (4, 0, 0)
Test 8: (0, -1, -1)
Test 9: (0, -1, -1)
Test 10: (495, 39, 49)

=== Code Execution Successful ===

```

Task 2

LEETCODE:

Accepted 210 / 210 testcases passed

P5uW7xlB74 submitted at Sep 07, 2025 19:36

EditorialSolution


Runtime

1297 ms | Beats 5.14%

Analyze Complexity

Memory

45.75 MB | Beats 6.14%



Code | Python3

```
from typing import List

class Solution:
    def maxSubArray(self, nums: List[int]) -> int:
        def max_crossing_sum(low, mid, high):
            left_sum = float("-inf")
```

23

24

25

26

27

28

29

30

31

32

```
        mid = (low + high) // 2
        left = solve(low, mid)
        right = solve(mid + 1, high)
        cross = max_crossing_sum(low, mid, high)

        return max(left, right, cross)

return solve(0, len(nums) - 1)
```

Saved

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

Expected

