Start

MAX_CROSSING_SUB_ARRAY ROUTINE (Array, start, mid, end)

	start	mid	mid+1	end.
A:	-2	+3	5	-1

```
left_sum = -∞
sum = 0
for i = mid to start
    sum = sum + A[i]
    if sum > left_sum
        left_sum = sum
        max_left = i

right_sum = -∞
sum = 0
for j = mid+1 to end
    sum = sum + A[j]
    if sum > right_sum
        right_sum = sum
        max_right = j
Total_Sum = left sum + right sum
return (max_left, max_right, Total_sum)
```

Analysis:

Its linear

MAX_SUBARRAY_SUM (A, start, end)

$O(n \log n)$

$T(n) = 2T(\frac{n}{2}) + \Theta(n)$

```
// Base condition
if start == end
    return start, end, A[start]
else mid = L(low + high)/2 J
```

3,-2,-5,-1 ① → L.S, L.E, L.MSS = Max_SUBARRAY_SUM(A, start, mid)

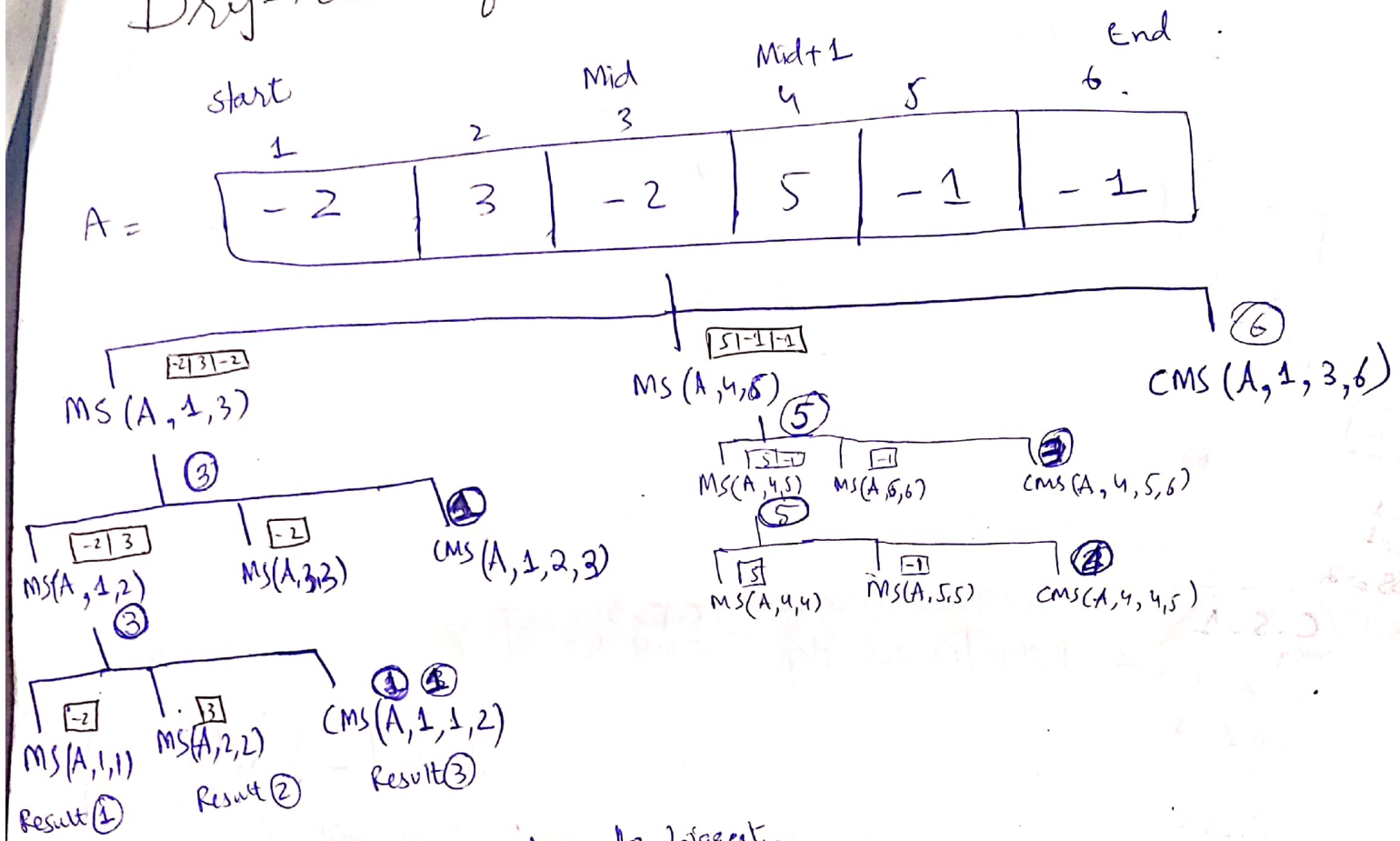
-3,-2,5,-1 ② → R.S, R.E, R.MSS = Max_SUBARRAY_SUM(A, mid+1, end)

3,-2,5,-1 ③ → C.S, C.E, C.MSS = MAX_CROSSING_SUB_ARRAY(A, start, mid, end)

Compare ①, ② & ③ & return the largest.

Dry - run

Dry-run of Divide & Conquer Algo of MSS



Compare three results & return the biggest.