~~[-2, 1, -3, 4, -1, 2, 1, -5, 4]~~

✗ -- 5 | -3 | 8

✓ -- 8 | -3 | 5

INDEX	0	1	2	3 (start)	4	5 (i)	6	7	8
Array	-2	1	-3	4	-1	2	1	-5	4
Cum-sum	-2	-1	-4	0	-1	1	2	-3	1
max-sum (curr-element + max-sum from previous; check if included[i-1] = Yes)	-2	+1 max(1, -2, +1-2)	+1 m(-3, -3+1) ~~This can't read, marker~~ M(-3, +1, -3+1) if included[i-1] = No	4 m(4, -3+4, -1, 4+(-3+1)) curr + cum-sum[i-1] - cum-sum[start-1]	m(4, -1, 4+(-1)) =4 These 3 decide if included = Yes or No & end & start index we change or not	m(4, 2, 4+(-1), 2+(-1-(-4))) end-i = 5 =5	m(5, 1, 1+5) end-i = i = 6 =6	m(6, -5, -5+6) =6 if all test case fail = included[i] = No	m(6, 4, 4+(-5), 4+(-3-(-4))) =6
elements	-2	1	1	4	4	4, -1, 2	4, -1, 2, 1	4, -1, 2, 1	4, -1, 2, 1
included	Yes (1)	Yes (1)	No (0)	Yes (1)	No (0)	Yes (1)	Yes (1)	No (0)	No (0)
start index	0	1	1	3	3	3	3	3	3
end index	0	1	1	3	3	5	6	6	6

1. We would need "included" arrays, "cum-sum" arrays
2. I think there is no need for start & end Arrays. Variables are good.