KADANE'S ALGORITHM

✫ In this problem, we are supposed to find the maximum sum subArray.

Brute force solution is to carry out the sum of each sub array and find maximum one. Original time complexity is $O(N^3)$. Better solution is to carry sums for previous sub arrays in the brute force to get a time complexity of $O(N^2)$.

Optimal solution is given by the Kadane's Algorithm. We iterate through the array and add each element to the score unless its less than 0, in which case we reset it to 0. Maximum sum obtained throughout the process is recorded.

arr[] = {-2, -3, 4, -1, -2, 1, 5, -3}

MAX : ~~0~~ ~~0~~ ~~4~~ $\boxed{7}$ → Final Ans

SUM : $\cancel{-2}$ → 0 $\cancel{-3}$ → 0 4 (-1) → 3 (-2) → 1 (1) → 2 (5) → 7 (-3) → 4

Pseudocode :

```
maximumSubArraySum(arr, len) {
    sum = 0
    maxS = - infinity
    for (i = 0 -> N) {
        sum + = arr[i]
        if (sum < 0)   sum = 0
        maxS = max(maxS, sum)
    }
    return  maxS
}
```