Maximum Sub-Array Sum [Kadane's Algorithm]

↪ take forward the sum which adds value to the next sum; i.e. is +ve.

→ if sum > 0 ⇒ sum += a[i]

→ max = math.max(max, sum);

~~return~~ max. (to be returned)

→ if sum < 0 ⇒ assign sum = 0;

Eg. a = { -2, -3, 4, -1, -2, 1, 5, -3 }

sum = -2	max = -2
→ sum = 0	max = 0
→ sum = -3	max = [illegible]
→ sum = 0	max = 0
→ sum = 4	max = 4
→ sum = 3	
→ sum = 1	
→ sum = 2	
→ sum = 7	max = 7 ✓
→ sum = 4	

Pseudo Code :

```
int max = Int_min;
int sum = 0;
for (i=0; i<n; i++){
    if (sum >= 0) { sum += a[i] };
      max = math.max(max, sum);
    else { sum = 0 } }
```

Finding max^m sum

Finding the subarray with max^m sum

```
int max = INT_MIN;
int sum = 0; (sStart, sEnd = -1)
for (i=0; i<n; i++){
        sum += a[i];
        if (sum > max){
                max = sum;
                sStart = start;
                sEnd = i;
        }

        if (sum < 0) { sum = 0;
                       start = i;
                     }
```

✓ Optimal sol^n

for subarray return

(sStart, sEnd)

Time Complexity :

⇒ O(N) ✓

Space ⇒ O(1) ✓

for both cases