# Sliding Windows

## Max Consecutive Sum

Given an array of positive, negative integers, return a consecutive seq that sums to largest amount.

### $O(n^3)$

For every $(i, j)$, $i < j$, compute $arr(i) + \ldots + arr(j)$

Return seq with max sum

$O(n^2)$ pairs, each sum computation for a pair $(i, j)$ takes $O(n)$ time, so $O(n^3)$

### $O(n^2)$

→ Precompute sum_till_[i] = arr[0] + ... + arr[i]
= sum_till_[i-1] + arr[i]

($O(n)$ precomputation time)

→ For every pair $(i, j)$ : compute
$i < j$
sum_till_[j] − sum_till_[i-1]

$O(1) \times O(n^2) = O(n^2)$

→ Find max $O(n)$

→ So overall $O(n^2)$

## O(n) time

Best_sum_ending_at [j]

= max ( arr [j] , Best_sum_ending_at [j-1] + arr [j] )

→ At each index i, you can either extend the ongoing subarray, or start afresh

arr = [-2 1 -3 4 -1 2 1 -5 4]

| | | |
|---|---|---|
| | | Best sum till |
| s↓ -2 ↓e | | -2 |
| -2, 1 | -1 = -2+1 < | 1 |
| -2, 1, -3 | -2 = 4 + -3 > | -3 |
| -2, 1, -3, 4 | -2 + 4 < | 4 |
| -2, 1, -3, 4, -1 | 3 = 4 + -1 > | -1 |
| -2, 1, -3, 4, -1, 2 | 5 = 3 + 2 > | 2 |
| -2, 1, -3, 4, -1, 2, 1 | 6 = 5 + 1 > | 1 |
| -2, 1, -3, 4, -1, 2, 1, -5, | -1 = 6 + -5 > | -5 |
| -2, 1, -3, 4, -1, 2, 1, -5, 4 | -1 + 4 < | 4 |

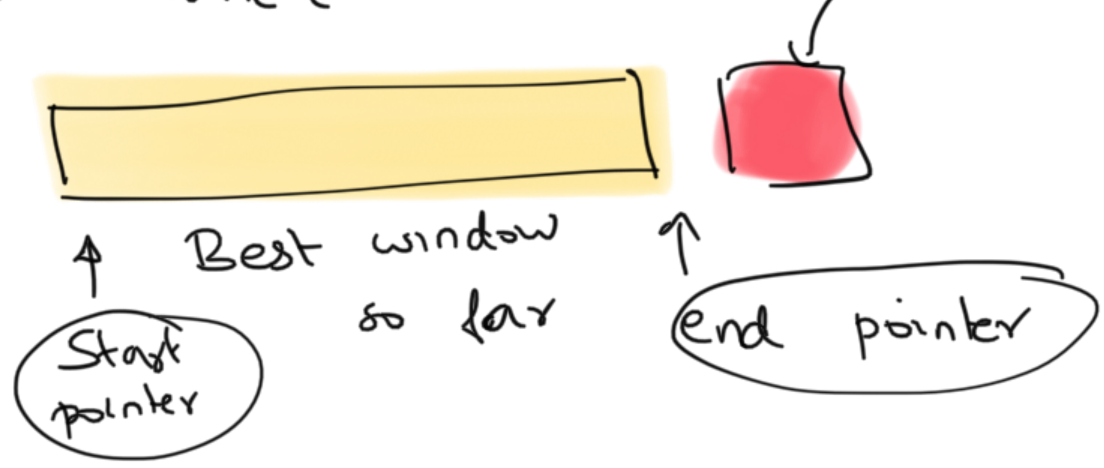
max ( -2, 1, -2, 4, 3, 5, 6, -1, 4 )

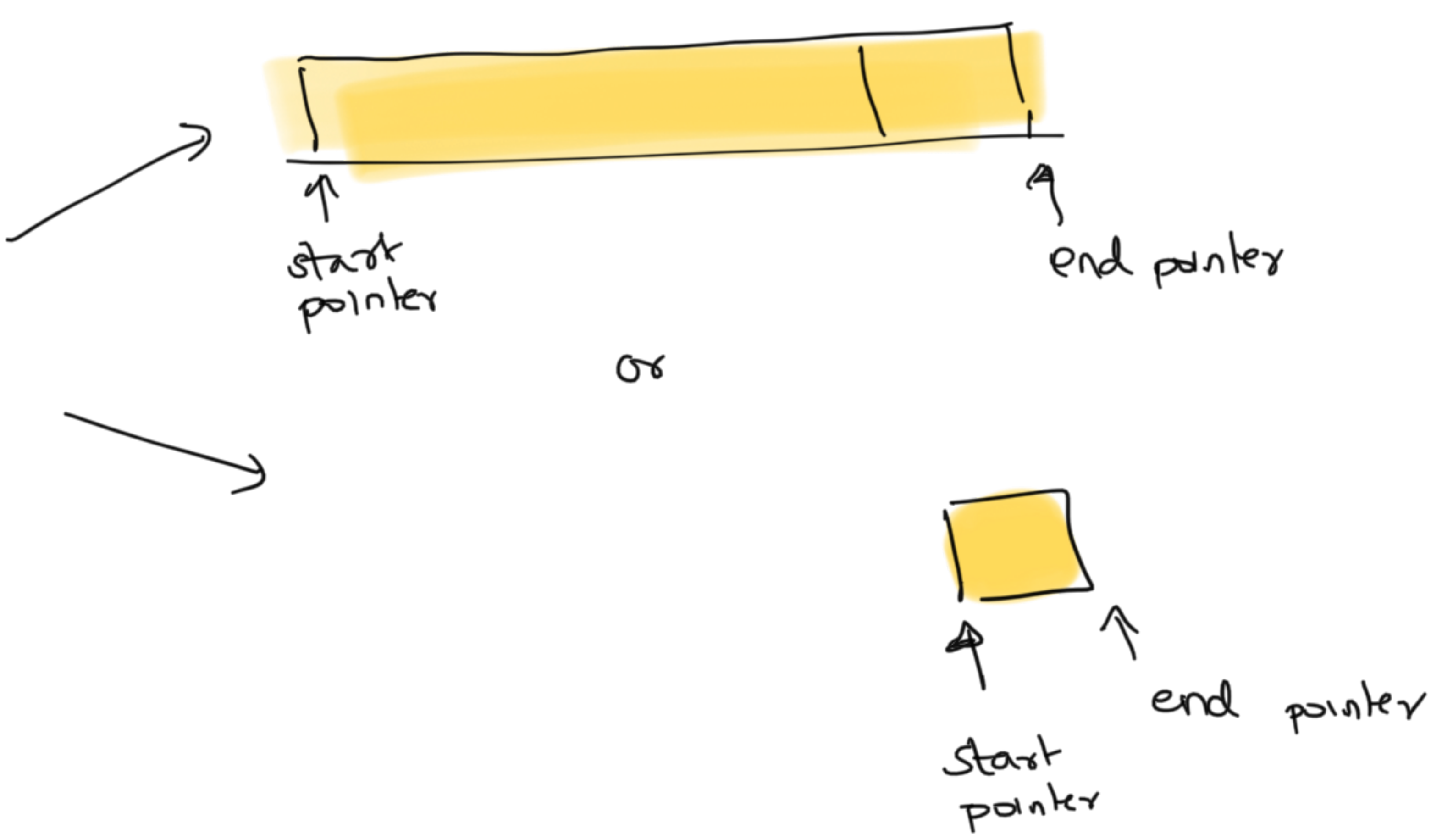
= 6

So answer is

$-2, 1, -3, \underbrace{4, -1, 2, 1}_{\text{sum} = 6}, -5, 4$

you can think about this as a sliding window where




( end pointer moves ahead by 1
start pointer stays where it is / moves to end pointer ! )

start pointer = " catch - up - pointer "

end pointer = " end - pointer "