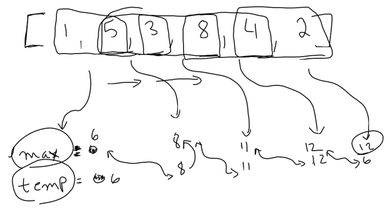
Sliding Window

- $n^2 \rightarrow n$
- used against array or string (1)
- used to find longest, shortest, max, min

maxSubarray(arr, num)
- arr
- window size

arr = [1, 5, 3, 8, 4, 2]

n = 2



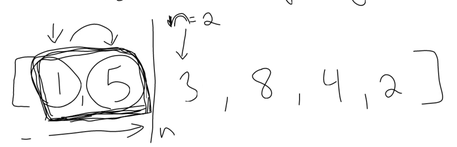

① initialize max and temp
- max: hold our ever max value
- temp: hold our next "potential" max value

let maxSum = 0

let tempSum = 0

② (!array.length) → array is empty
Window Size > array.length } edge cases

③




n = 2

- calculate our initial window Sum

for(let i=0; i<n; i++) {

maxSum += arr[i]

}

i=0 → 1 → ①

i=1 → 5 = ⑥

3

④ tempSum = maxSum

⑤ Sliding window

for(let i=n; i<arr.length; i++) {

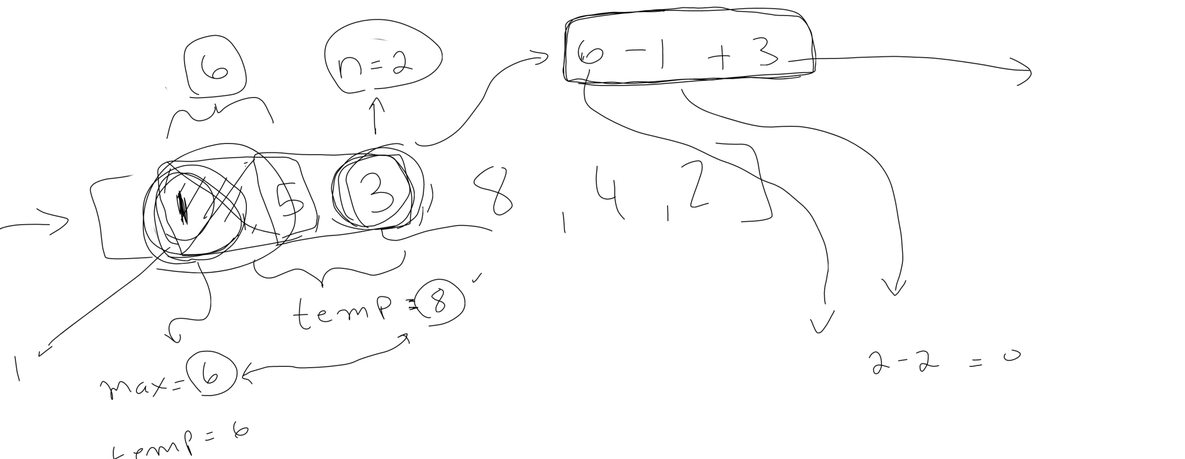
}

Math.floor(9.50)

Math.ceil(9.50)

Math.max([9,8,9]) OR Math.max(9,8,7) → 9

Math.min(5,6,9) → 5




Math.max(max, temp)

temp = temp - arr[i] + arr[n]

6 - 1 + 3 = 8 ✓