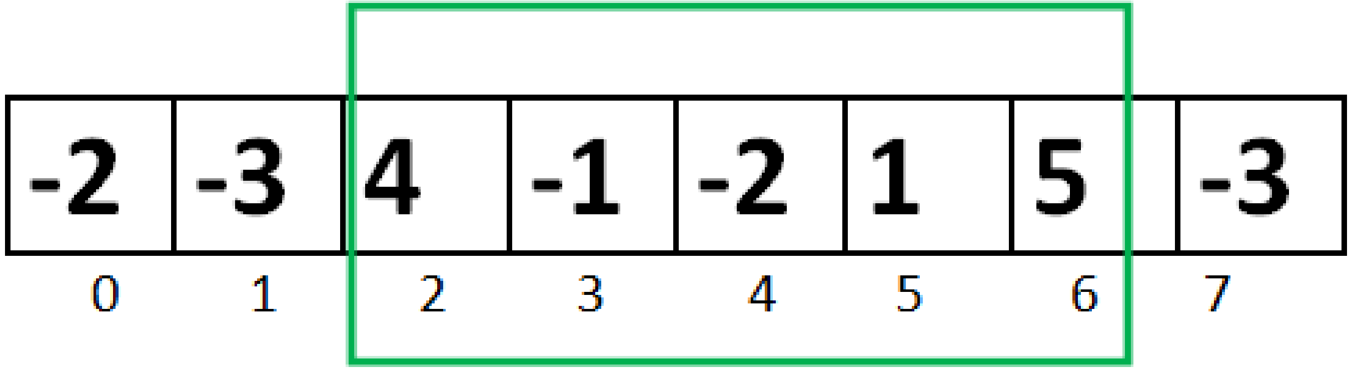
Largest Sum Contiguous Subarray

Problem

- Given N numbers in A
  - Find the subarray with max sum

Largest Subarray Sum Problem

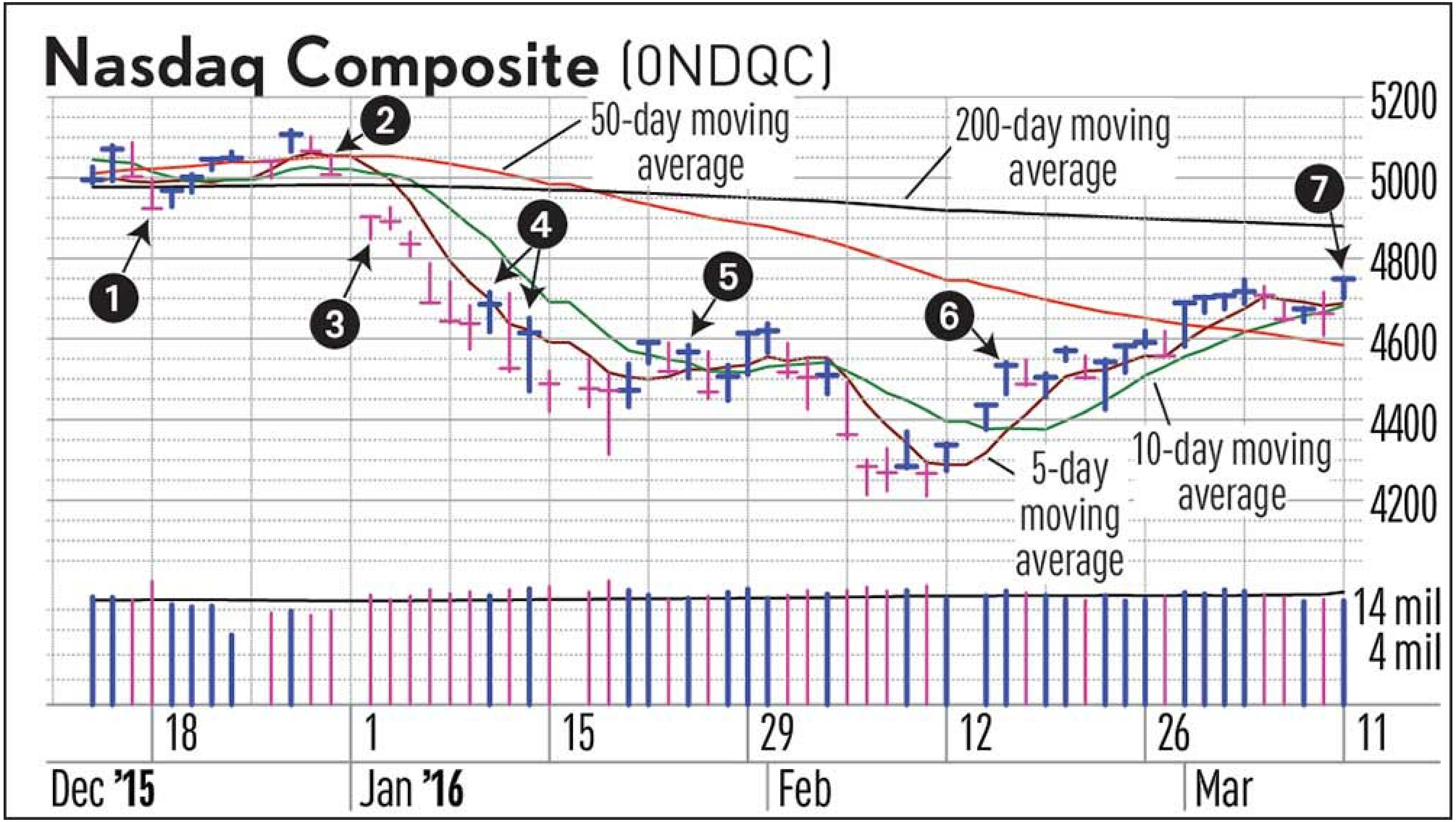


-2	-3	4	-1	-2	1	5	-3
0	1	2	3	4	5	6	7

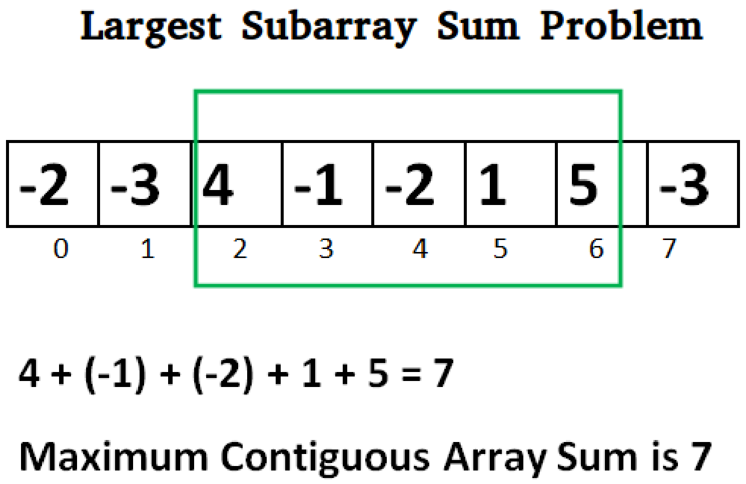
4 + (-1) + (-2) + 1 + 5 = 7

Maximum Contiguous Array Sum is 7

Application



Naïve Solution?



Largest Subarray Sum Problem

-2	-3	4	-1	-2	1	5	-3
0	1	2	3	4	5	6	7

4 + (-1) + (-2) + 1 + 5 = 7

Maximum Contiguous Array Sum is 7

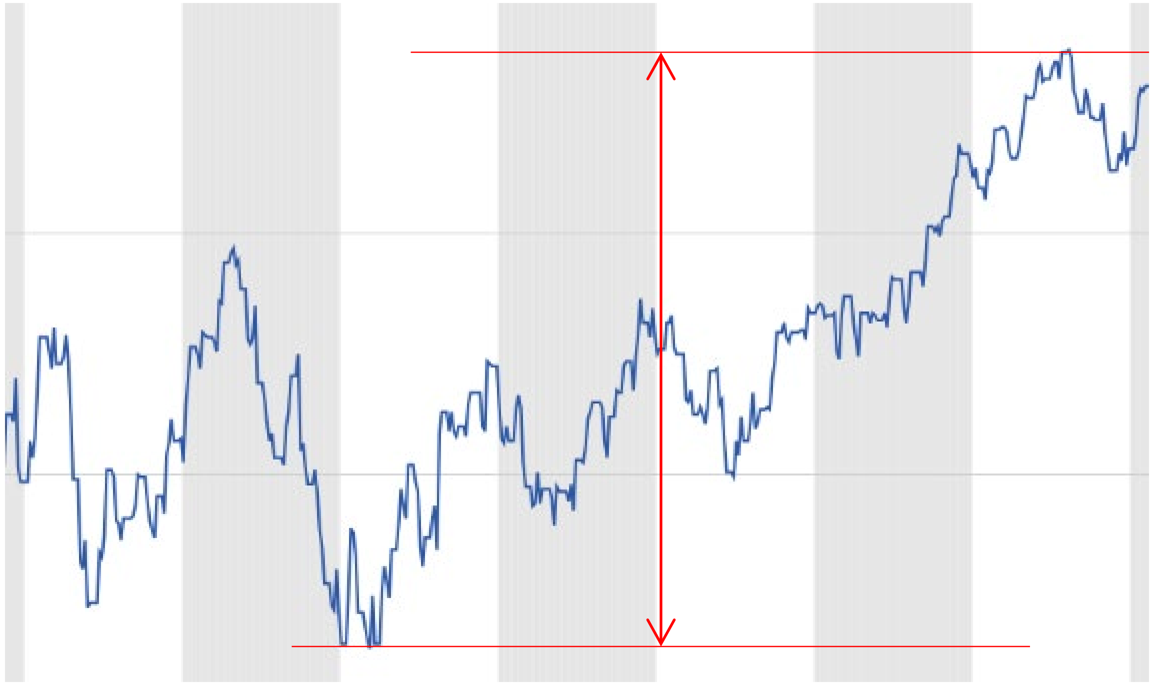
- Compute the sum for every sub array of A
    - Each subarray starts with an index i from 0 to N-1
    - And ends with j from i to N-1
- Algorithm

```
for i from 0 to N-1
  for j from i to N-1
    Compute the sum of the array from i to j
```

- Time complexity?
    - O(N^3)

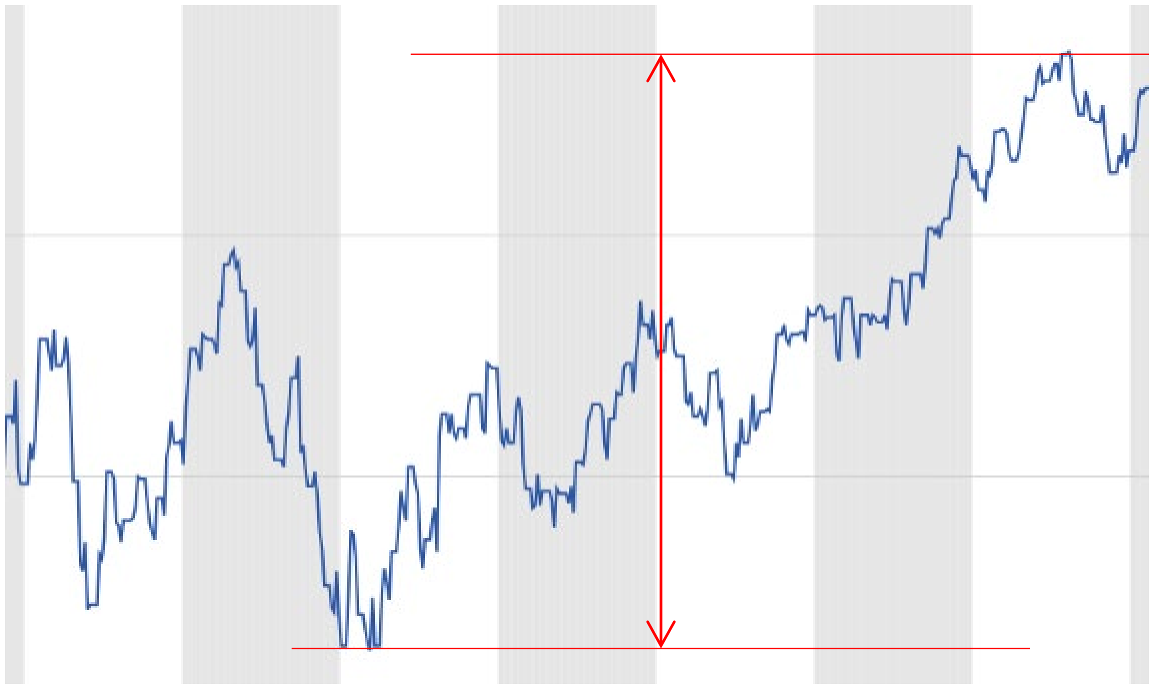
Divide-and-conquer?

- Compute the accumulative sum array B of A
  - B[i] = sum of A[0] to A[i]
- Like stock market, you can think of A is the daily changes of a value
  - B will be the absolute values
- The array with max sum in A
  - Equal to the biggest interval in B



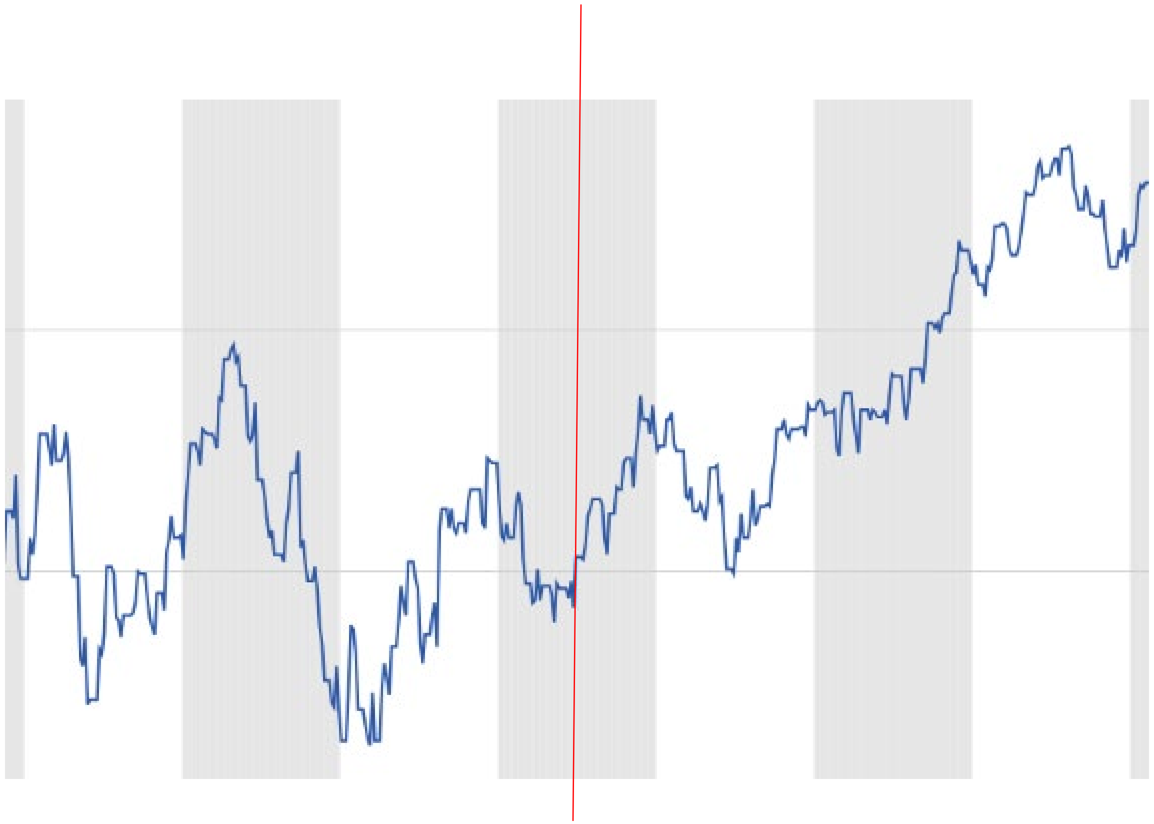
Divide-and-conquer?

- How to find the biggest interval in B?
  - Find min in B and max in B?
  - Only works if the min is on the “left” of the max



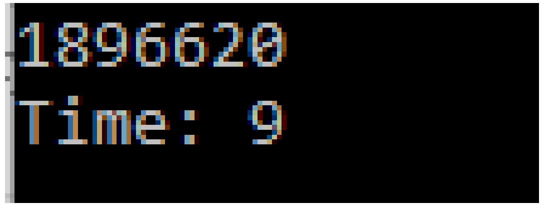
Divide-and-conquer?

- Divide the graph into two halves
  - Possible solution 1
    - Find the min in the left
    - Find the max in the right
  - Possible solution 2
    - The max interval in the left
  - Possible solution 3
    - The max interval in the right
- Final solution
  - The best of the above
- Time complexity:
  - O(N)



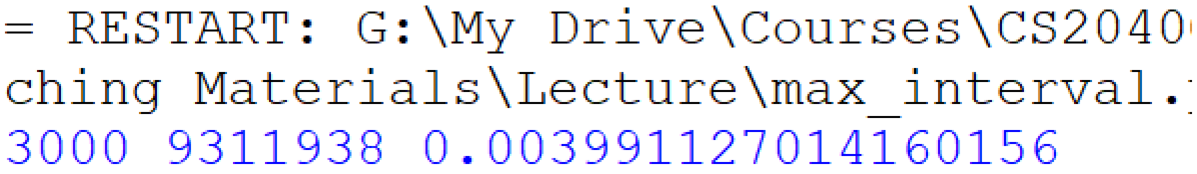
N = 3000

- C++ Naïve Version:



```
1896620
Time: 9
```

- Python Divide-and-conquer verysion:



```
= RESTART: G:\My Drive\Courses\CS2040
ching Materials\Lecture\max_interval.
3000 9311938 0.0039911270141601 56
```

Kadane’s algorithm

```
Initialize:
    max_so_far = INT_MIN
    max_ending_here = 0

Loop for each element of the array

  (a) max_ending_here = max_ending_here + a[i]
  (b) if(max_so_far < max_ending_here)
            max_so_far = max_ending_here
  (c) if(max_ending_here < 0)
            max_ending_here = 0
return max_so_far
```