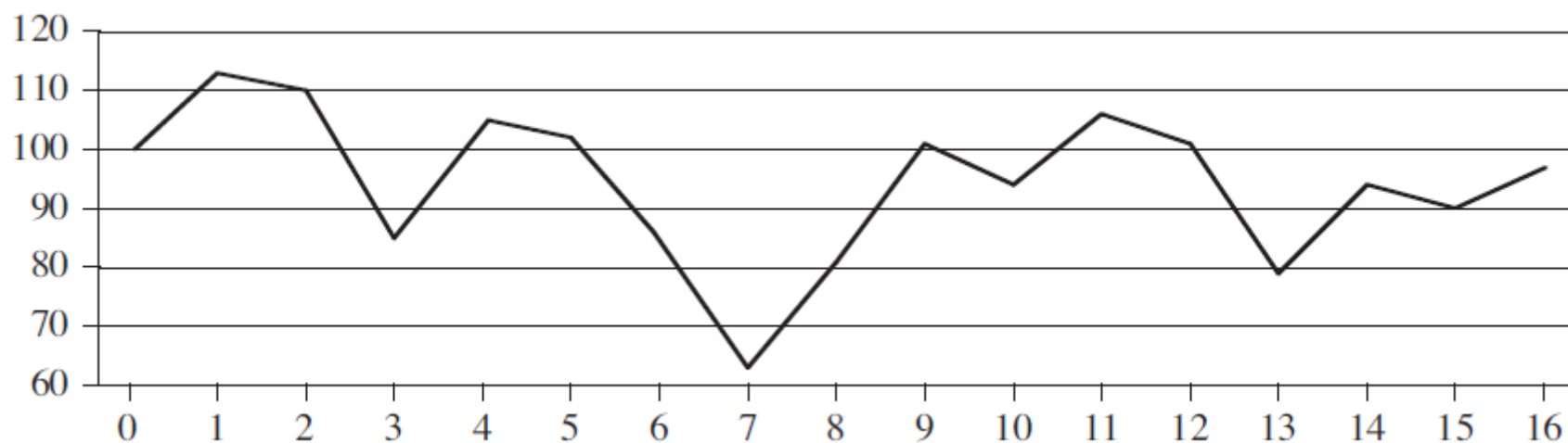


The maximum-subarray problem

- You are allowed to buy one unit of stock only one time and then sell it at a later date, buying and selling after the close of trading for the day. To compensate for this restriction, you are allowed to learn what the price of the stock will be in the future. Your goal is to maximize your profit.



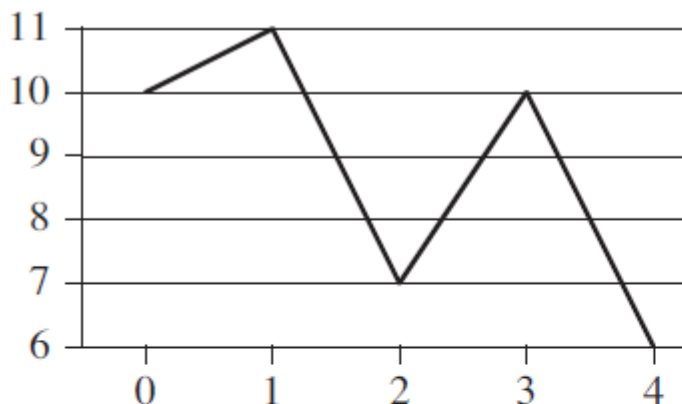


Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Change		13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

Figure 4.1 Information about the price of stock in the Volatile Chemical Corporation after the close of trading over a period of 17 days. The horizontal axis of the chart indicates the day, and the vertical axis shows the price. The bottom row of the table gives the change in price from the previous day.



Naïve idea: By buying at the lowest price or selling at the highest price.



Day	0	1	2	3	4
Price	10	11	7	10	6
Change		1	-4	3	-4

Figure 4.2 An example showing that the maximum profit does not always start at the lowest price or end at the highest price. Again, the horizontal axis indicates the day, and the vertical axis shows the price. Here, the maximum profit of \$3 per share would be earned by buying after day 2 and selling after day 3. The price of \$7 after day 2 is not the lowest price overall, and the price of \$10 after day 3 is not the highest price overall.



A brute-force solution

- Idea: Try every possible pair of buy and sell dates in which the buy date precedes the sell date.
- Time Complexity:

$$f(n) = \Omega(n^2)$$



Transformation

- Change Idea: We want to find a sequence of days over which the net change from the first day to the last is maximum.
- The change on day i is the difference between the prices after day $i-1$ and after day i .
- \rightarrow *Maximum Subarray Problem*



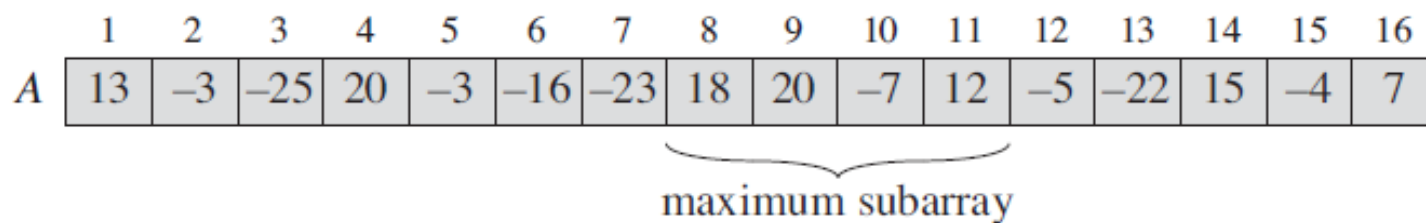


Figure 4.3 The change in stock prices as a maximum-subarray problem. Here, the subarray $A[8..11]$, with sum 43, has the greatest sum of any contiguous subarray of array A .



How to divide-and-conquer?

Target: to find a maximum subarray of $A[\text{low}..\text{high}]$.

Divide: We divide the subarray into two subarrays of as equal size as possible. So we find the midpoint mid and $A[\text{low}..\text{high}] \rightarrow A[\text{low}..\text{mid}] + A[\text{mid}..\text{high}]$.

Conquer:

Combine: $\max(A[\text{low}..\text{high}])$ must be either:

- entirely in the subarray $A[\text{low}..\text{mid}]$, so that $\text{low} \leq i \leq j \leq \text{mid}$,
- entirely in the subarray $A[\text{mid} + 1..\text{high}]$, so that $\text{mid} < i \leq j \leq \text{high}$, or
- crossing the midpoint, so that $\text{low} \leq i \leq \text{mid} < j \leq \text{high}$.

To find a maximum subarray that crosses the midpoint



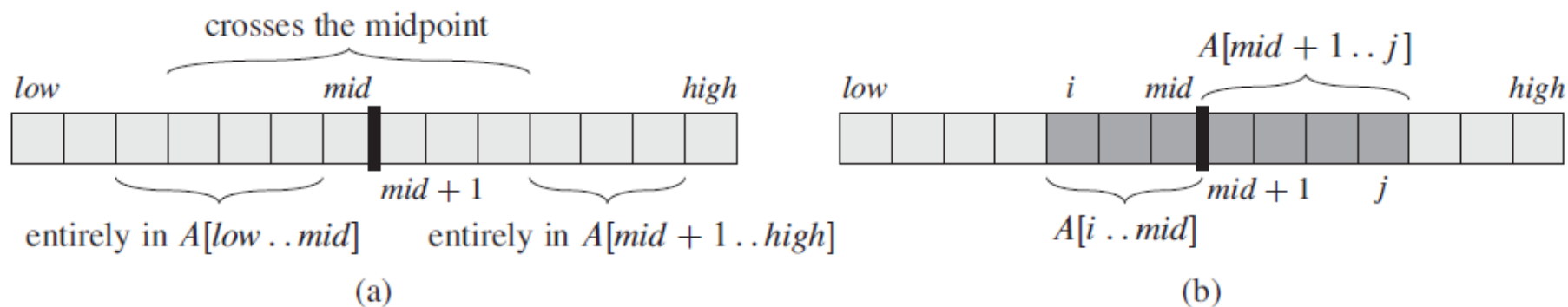


Figure 4.4 (a) Possible locations of subarrays of $A[low \dots high]$: entirely in $A[low \dots mid]$, entirely in $A[mid + 1 \dots high]$, or crossing the midpoint mid . (b) Any subarray of $A[low \dots high]$ crossing the midpoint comprises two subarrays $A[i \dots mid]$ and $A[mid + 1 \dots j]$, where $low \leq i \leq mid$ and $mid < j \leq high$.



How to find a maximum subarray crossing the midpoint? In linear time?

FIND-MAX-CROSSING-SUBARRAY(*A, low, mid, high*)

```
1  left-sum =  $-\infty$ 
2  sum = 0
3  for i = mid downto low
4      sum = sum + A[i]
5      if sum > left-sum
6          left-sum = sum
7          max-left = i
8  right-sum =  $-\infty$ 
9  sum = 0
10 for j = mid + 1 to high
11     sum = sum + A[j]
12     if sum > right-sum
13         right-sum = sum
14         max-right = j
15 return (max-left, max-right, left-sum + right-sum)
```



Algorithm for the maximum-subarray problem

FIND-MAXIMUM-SUBARRAY($A, low, high$)

```
1  if  $high == low$ 
2      return ( $low, high, A[low]$ )           // base case: only one element
3  else  $mid = \lfloor (low + high) / 2 \rfloor$ 
4      ( $left-low, left-high, left-sum$ ) =
          FIND-MAXIMUM-SUBARRAY( $A, low, mid$ )
5      ( $right-low, right-high, right-sum$ ) =
          FIND-MAXIMUM-SUBARRAY( $A, mid + 1, high$ )
6      ( $cross-low, cross-high, cross-sum$ ) =
          FIND-MAX-CROSSING-SUBARRAY( $A, low, mid, high$ )
7      if  $left-sum \geq right-sum$  and  $left-sum \geq cross-sum$ 
8          return ( $left-low, left-high, left-sum$ )
9      elseif  $right-sum \geq left-sum$  and  $right-sum \geq cross-sum$ 
10         return ( $right-low, right-high, right-sum$ )
11     else return ( $cross-low, cross-high, cross-sum$ )
```



Time complexity

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ 2T(n/2) + \Theta(n) & \text{if } n > 1. \end{cases}$$

Master Theorem Case 2:

$$T(n) = \Theta(n \lg n)$$

