The maximum-subarray problem

Joe Song

February 18, 2020

The lecture notes are mostly based on Chapter 4.1 of Cormen, Leiserson, Rivest, and Stein. Introduction to Algorithms. 3rd Ed. 2009. MIT Press. Cambridge, Massachusetts.

Contents

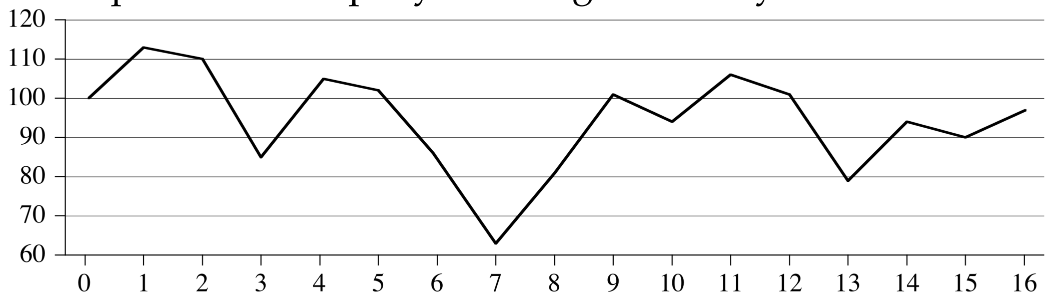
1 An example of trading stocks

Problem: If we magically know the price of a stock in advance, can we write a computer program to buy and sell stocks to maximize the profit?

Solutions:

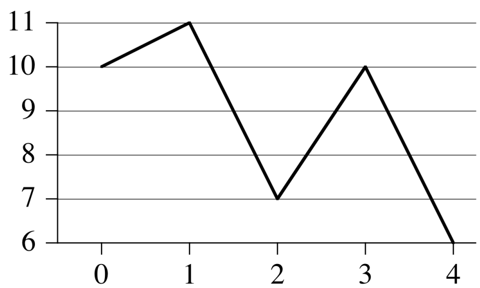
1. *buy low and sell high*. $O(n)$ (Will this always work?) No. The highest may come before the lowest. Counter example below:

   Stock price of a company in a range of 17 days:



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

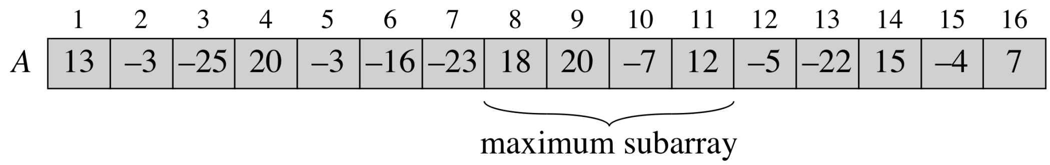
2. *buy low or sell high*. This will not always work. Counter example below:

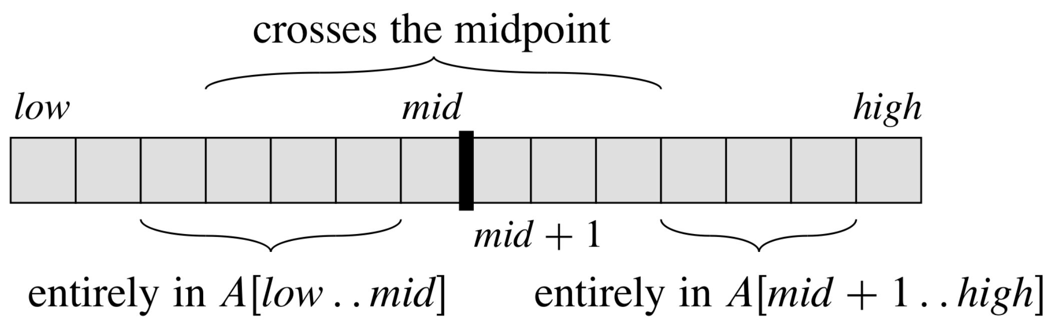


Day	0	1	2	3	4
Price	10	11	7	10	6
Change		1	−4	3	−4

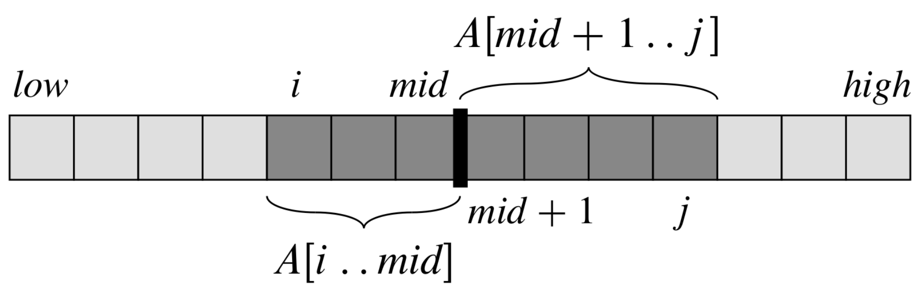
3. Try all possible (buy, sell) days and find the maximum profit. $O(n^2)$. Slow!

4. The maximum subarray concept:





(a)



(b)

2 The algorithm

Input: array A

Output: a maximum subarray in A, whose sum is the largest.

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
// Find a maximum subarray of the form A[i..mid]
left.sum = -∞
sum = 0
for i = mid downto low
    sum = sum + A[i]
    if sum > left.sum
        left.sum = sum
        max.left = i
// Find a maximum subarray of the form A[mid + 1..j]
right.sum = -∞
sum = 0
for j = mid + 1 to high
    sum = sum + A[j]
    if sum > right.sum
        right.sum = sum
        max.right = j
// Return the indices and the sum of the two subarrays
return (max.left, max.right, left.sum + right.sum)
```

```
FIND-MAXIMUM-SUBARRAY(A, low, high)
if high == low
    return (low, high, A[low]) // base case: only one element
else mid = ⌊(low + high)/2⌋
    (left.low, left.high, left.sum)
        = FIND-MAXIMUM-SUBARRAY(A, low, mid)
    (right.low, right.high, right.sum)
        = FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross.low, cross.high, cross.sum)
        = FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left.sum ≥ right.sum and left.sum ≥ cross.sum
        return (left.low, left.high, left.sum)
    elseif right.sum ≥ left.sum and right.sum ≥ cross.sum
        return (right.low, right.high, right.sum)
    else return (cross.low, cross.high, cross.sum)
```

3 Running time

Let $T(n)$ be the runtime of the algorithm with an input array of size n. As

$$T(n) = 2T(n/2) + n$$

by the master theorem, we have

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