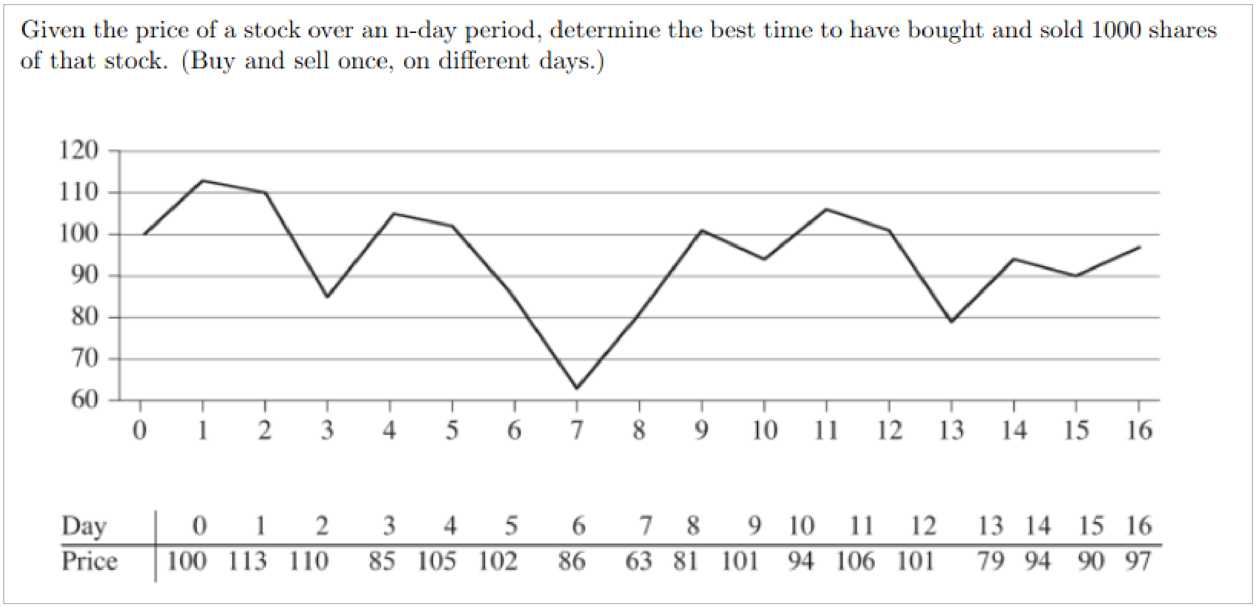

Given the price of a stock over an n-day period, determine the best time to have bought and sold 1000 shares of that stock. (Buy and sell once, on different days.)

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

Establish the problem.

- specifications

Task: determine the buy and sell days (buy < sell) resulting in max profit (or smallest loss if no profit)

Input: daily stock price over *n* days

Output: buy day, sell day

- examples

buy on day 7, sell on day 11 results in max profit of 105-65 = 40

Identify avenues of attack.

- targets

Brute force: check all buy-sell pairs and take max profit/min loss of those → O(n^2)

- approach

Divide-and-conquer.

- paradigms and patterns

Paradigm: divide-and-conquer.

easy split: split stock prices into first half of the days, second half of the days → get back best buy-sell pair in the first half, best busy-sell in the second half → ...

easy merge: n/a

Define the algorithm.

- size

n – the number of stock prices

- generalize / define subproblems

Task: determine the buy and sell days (buy < sell) resulting in max profit (or smallest loss if no profit) within the range (low,high)

Input: daily stock price *S* over *n* days, range low..high (inclusive)

Output: buy day, sell day, min price day, max price day

- base case(s)

n=2 (low = high-1) – (have to buy/sell on different days, so < 2 days doesn’t make sense)

return (low,high) – buy on low, sell on high

- main case

// split array into first half, second half

mid ← (low+high)/2

// find best buy-sell in each half

(buy1,sell1,min1,max1) ← stocks(A,low,mid)

(buy2,sell2,min2,max2) ← stocks(A,mid+1,high)

// determine overall best buy-sell

~~check all buy in the first half, sell in the second half pairs – (n/2) x (n/2) = n^2/4 pairs~~

~~find min in the first half, max in the second half~~

return best of: (buy1,sell1), (buy2,sell2), and (min1,max2) ~~all of the buy-first, sell-second pairs~~ overall min: min(min1,min2) overall max: max(max1,max2)

$T(n) = 2\,T(n/2) + O(1) \rightarrow T(n) = O(n)$

- top level
  - initial subproblem
  - setup
  - wrapup
- special cases
- algorithm

Show termination and correctness.

- termination
  - making progress

    - the end is reached
- correctness
    - establish the base case(s)
    - show the main case
    - final answer

Determine efficiency.

- implementation
- time and space

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

- room for improvement