Solution Review: Buy and Sell Stock

This lesson contains the solution review for the challenge of finding the maximum profit generated by buying and selling stocks!

WE'LL COVER THE FOLLOWING

In this lesson, we will review the solution to the problem in the last exercise. Let's go over the problem statement once again:

Given an array of numbers consisting of daily stock prices, calculate the maximum amount of profit that can be made from buying on one day and selling on another day.

We consider two approaches to this problem. In the first, we consider a brute force approach that solves the problem in $O(N^2)$, where N is the size of the array of numbers. We then improve upon this solution to take our solution to a time complexity of $O(N)$.

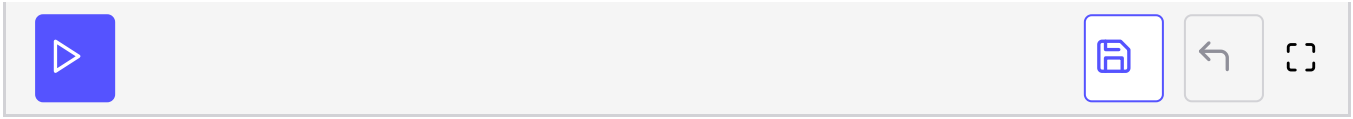
Solution 1: Brute Force

Implementation

```
A = [310, 315, 275, 295, 260, 270, 290, 230, 255, 250]
# Time Complexity: O(n^2)
# Space Complexity: O(1)
def buy_and_sell_once(A):
  max_profit = 0
  for i in range(len(A)-1):
```

```
    for j in range(i+1, len(A)):
      if A[j] - A[i] > max_profit:
          max_profit = A[j] - A[i]
  return max_profit

print(buy_and_sell_once(A))
```



Explanation

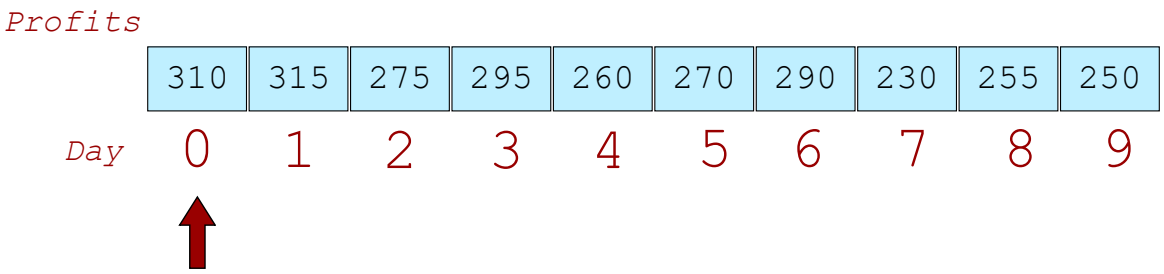
This brute force solution is an intuitive approach. We set `max_profit` equal to `0` on **line 5**. The for loop on **line 6** iterates over every element in `A`. Now for every element of `A`, we iterate over all the elements in `A` which are at a greater position than that element in `A` using the for loop on **line 7**. In this way, we calculate all possible profits by selling with all possible prices after we buy the stocks. Now if the calculated profit, i.e., `A[j] - A[i]` is greater than the `max_profit`, we update `max_profit` to its new greater value on **line 9**. Finally, we return `max_profit` on **line 10** which will be the maximum possible profit that we got by buying stocks on each day and selling them on all possible days after the day we buy them. Therefore, this solution has a time complexity of $O(n^2)$ as we generate all the possible buying and selling combinations using nested for loops. Now we need to think of a better solution. Let's check out Solution 2.

Solution 2: Tracking Min Price

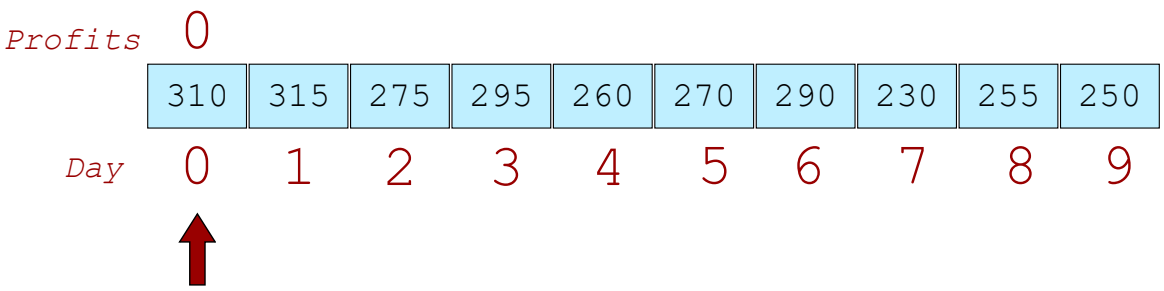
In solution 2, for each index, we calculate the most profit we can make by selling at that time. So, for a given day, the maximum profit can be made if the stock were bought at the minimum price on an earlier day. Therefore, we maintain the minimum price seen so far and subtract it from every future price to keep track of the maximum profit.

We iterate the array once and keep track of the minimum buying price and the maximum profit. We calculate the profit at each iteration by subtracting the minimum buying price seen so far from the current price in each iteration and updating the maximum profit accordingly.

Have a look at the slides below to understand the algorithm:

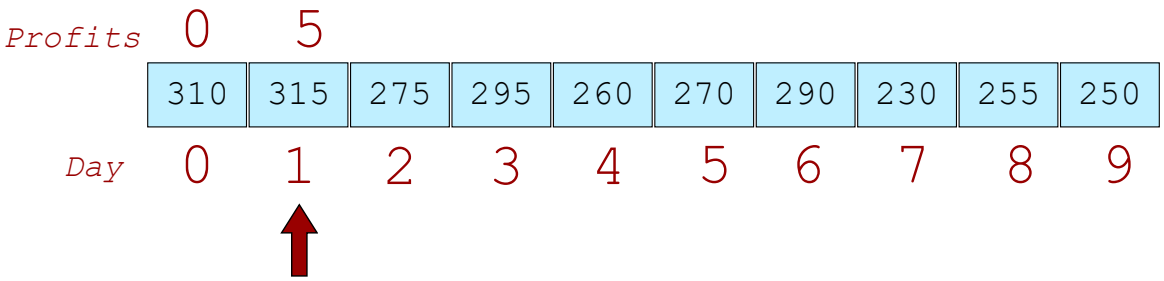


Minimum Price = 310
Maximum Profit = 0



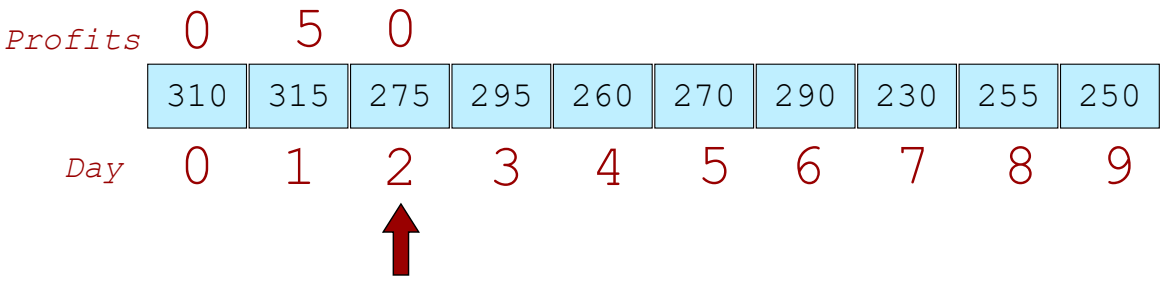
Minimum Buying Price = 310
Selling Price = 310

Profit = 310 - 310 = 0
Maximum Profit = 0



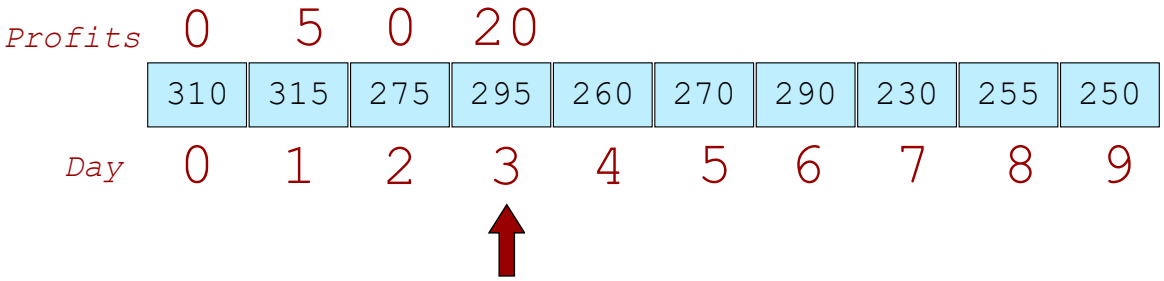
Minimum Buying Price = 310
Selling Price = 315

Profit = 315 - 310 = 5
Maximum Profit = 5



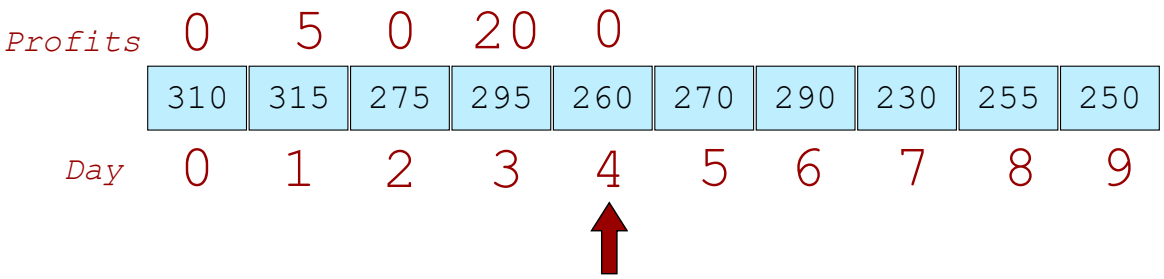
Minimum Buying Price = 275
Selling Price = 275

Profit = 275 - 275 = 0
Maximum Profit = 5



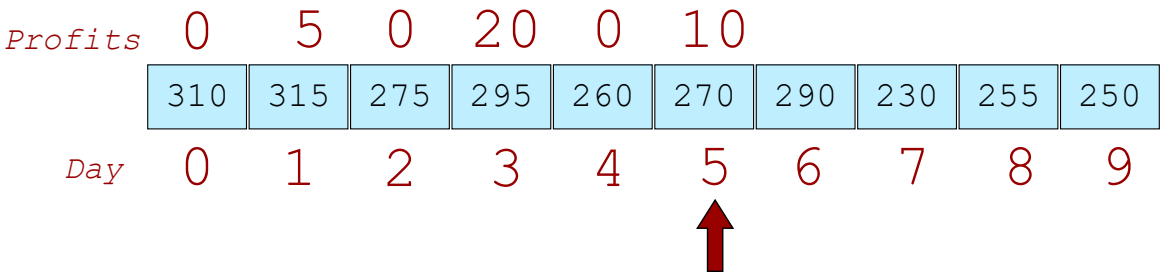
Minimum Buying Price = 275
Selling Price = 295

Profit = 295 - 275 = 20
Maximum Profit = 20



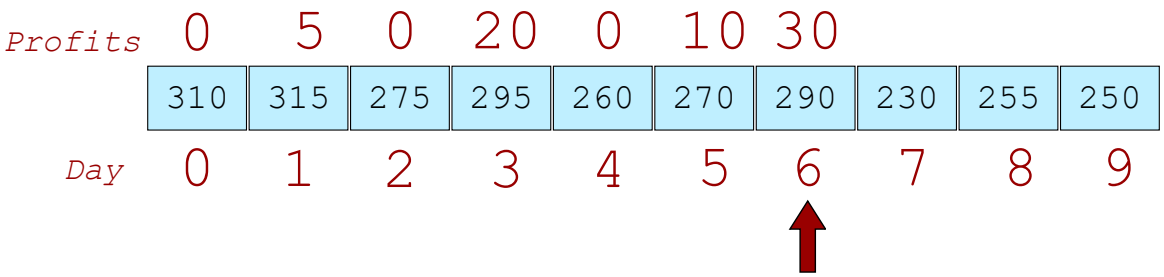
Minimum Buying Price = 260
Selling Price = 260

Profit = 260 - 260 = 0
Maximum Profit = 20



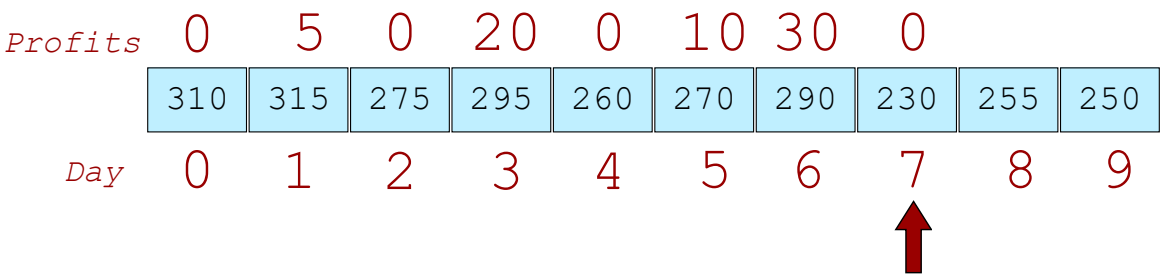
Minimum Buying Price = 260
Selling Price = 270

Profit = 270 - 260 = 10
Maximum Profit = 20



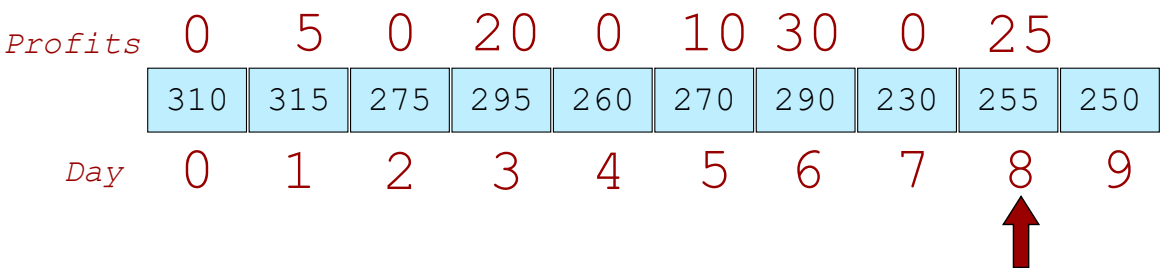
Minimum Buying Price = 260
Selling Price = 290

Profit = 290 - 260 = 30
Maximum Profit = 30



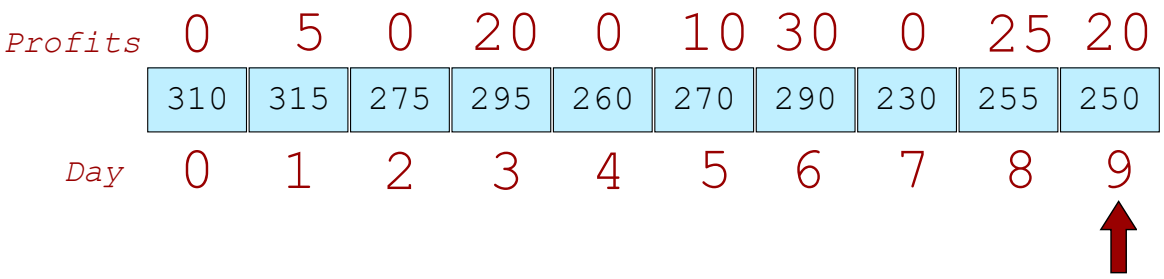
Minimum Buying Price = 230
Selling Price = 230

Profit = 230 - 230 = 0
Maximum Profit = 30



Minimum Buying Price = 230
Selling Price = 255

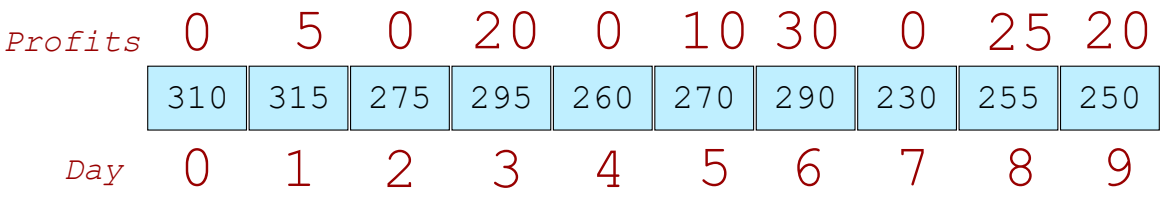
Profit = 255 - 230 = 25
Maximum Profit = 30



Minimum Buying Price = 230
Selling Price = 250

Profit = 250 - 230 = 20
Maximum Profit = 30

11 of 12



Minimum Buying Price = 230
Selling Price = 250

Profit = 250 - 230 = 20
Maximum Profit = 30

12 of 12

Implementation

Let's jump to the implementation in Python of the algorithm illustrated above:

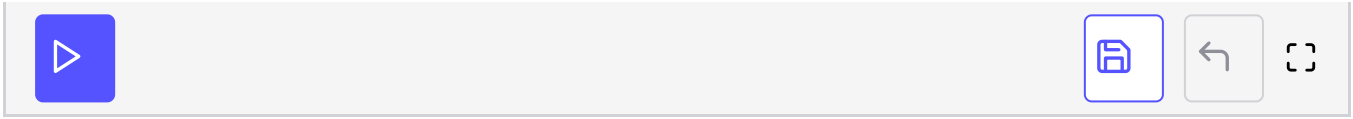
```
A = [310, 315, 275, 295, 260, 270, 290, 230, 255, 250]

# Time Complexity: O(n)
# Space Complexity: O(1)
```

```
def buy_and_sell_once(prices):
  max_profit = 0.0
  min_price = float('inf')

  for price in prices:
    min_price = min(min_price, price)
    compare_profit = price - min_price
    max_profit = max(max_profit, compare_profit)
  return max_profit

print(buy_and_sell_once(A))
```



Explanation

We set `max_profit` and `min_price` to `0` and `infinity` respectively (**lines 6-7**). The list `prices` is iterated using a for loop on **line 8**. As we are supposed to keep track of the minimum price, we update `min_price` using the `min` function on **line 9** where `min_price` is the minimum amount between `min_price` and `price` of the current iteration. In the next line, we store the calculated profit (`price - min_price`) in `compare_profit`. As we also need to keep a check on the `max_profit`, we update `max_profit` to the maximum value between `max_profit` and `compare_profit` on **line 11**. After we are done with the iteration of `prices`, we return `max_profit` on **line 12**.

I hope everything is clear so far and you were able to enjoy the exciting problems regarding arrays in Python in this chapter.

Up next we have **Binary Trees**. Stay tuned!