# Daily Coding Problem #193

## Problem

This problem was asked by Affirm.

Given a array of numbers representing the stock prices of a company in chronological order, write a function that calculates the maximum profit you could have made from buying and selling that stock. You're also given a number `fee` that represents a transaction fee for each buy and sell transaction.

You must buy before you can sell the stock, but you can make as many transactions as you like.

For example, given [1, 3, 2, 8, 4, 10] and `fee = 2`, you should return 9, since you could buy the stock at 1 dollar, and sell at 8 dollars, and then buy it at 4 dollars and sell it at 10 dollars. Since we did two transactions, there is a 4 dollar fee, so we have 7 + 6 = 13 profit minus 4 dollars of fees.

## Solution

At each step `i`, we keep track of two things:

- `current_max_profit`, the maximum profit we could have made at that point
- `hold`, which is the the maximum profit we could have if we currently own the stock

At each step, we update `current_max_profit`: either we keep the current profit, or calculate how much we would have if we sold the stick using `hold`. We also update `hold`

by updating it as if we bought the stock on that day.

```
def buy_and_sell_with_fee(arr, fee):
    current_max_profit = 0
    hold = -arr[0]
    for price in arr[1:]:
        current_max_profit = max(current_max_profit, hold + price - fee)
        hold = max(hold, current_max_profit - price)
    return current_max_profit
```

This takes O(n) time and O(1) space.

---