Problem Link:

https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/

Problem Description:

You are given an array of prices where `prices[i]` is the price of a given stock on the i^{th} day.

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return `0`.

Problem approach: Two Pointers

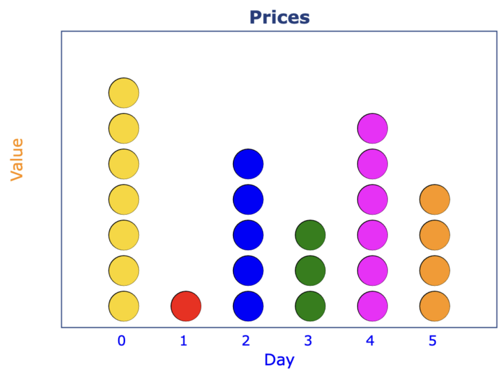
We take two pointers, namely: buyPointer and sellPointer. As the name suggests, one will track the buy price, the other will track the selling price.

Sample Test Case:

Input: prices = [7,1,5,3,6,4]
Output: 5

Solution:

The solution is pretty simple, we initialize the buyPointer as the first day, and the sellPointer as the second day, as it makes sense. We cannot get any profit if we buy and sell on the same day.



We follow a simple algorithm. We need to iterate through the whole array, buy at the least amount, and sell at the maximum amount after that. Hence, the iterable must be the sellPointer. Whenever we find that the price at the sellPointer index is lesser than the buyPointer's price, we replace the buyPointer with the newly found sellPointer's index. If that's not the case, then we calculate the current profit, which is the difference between sellPointer and buyPointer's prices. We maintain the maximumProfit, to keep track of the maximumProfit achieved till now. When the sellPointer reaches the end of the array, we'll have the maximumProfit, and we simply return it, and that is our answer!

Code (Python):

```python
def maxProfit(self, prices: List[int]) -> int:
    buyPrice = 0
    maxProfit = 0
    for sellPrice in range(1,len(prices)):
        if prices[sellPrice] < prices[buyPrice]:
            buyPrice = sellPrice
        else:
            currentProfit = prices[sellPrice] - prices[buyPrice]
            if currentProfit > maxProfit:
                maxProfit = currentProfit
    return maxProfit
```