0001\_Easy\_BestTimeToBuyAndSellStock\_#121\_Breakdown

Problem:

You are given an array prices where prices[i] is the price of a given stock on the ith 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.

Examples:

**Example 1:**

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

**Output:** 5

**Explanation:** Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

**Example 2:**

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

**Output:** 0

**Explanation:** In this case, no transactions are done and the max profit = 0.

Observations:

* If no profit is achieved return 0.
* The buy day must be before the sell day.

What needs to be true for this problem to work:

* The lowest number comes before the highest number to result in maximum profit.

How would a person solve this problem:

* Start at the beginning of the array, look through the array for the next biggest number, subtract that number with the first number for profit, repeat for each larger number keeping track of the profit. Repeat with the second number in the array and so on.

Brute force:

* Iterate through the array, for each element check each number after that number for a higher number, subtract them to find the profit, repeat, updating the profit if a new higher profit is found.
* Time Complexity: O(n^2), the array is iterated through for each element of the array.
  + Specifically: O(n(n-1)/2), each subsequent iteration is smaller than the previous iteration.
* Space Complexity: O(1), iterations happen in place.

Optimize (BUD, bottleneck, unnecessary code, duplicate code):

* We can utilize variables to reduce the number of iterations by keeping track of the lowest variable and of the highest profit.
* Time Complexity: O(n), the array is iterated through once.
* Space Complexity: O(1), two extra variables are used which is negligible.

Pseudocode:

* Create an integer called low and set it to the first index in the array.
* Create an integer called profit and set it to zero.
* Iterate through the array starting at the second index.
  + If the value of the current index is less than the value of low, replace low with the value of the current index.
  + Else if the value of the current index is greater than the value of low, subtract the value of the current index with the value of low and compare to the value of the current profit, updating current profit with the higher value.
* Return profit.

Note:

* This is also a simple version of Kadane’s algorithm.
* A simpler version of the code uses the min function and the max function.