Best Time to Buy and Sell Stock

Created	@2025年5月10日 下午11:55
Question Type	Sliding Window
Difficulty	Easy
LeetCode Question Link	https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/

1. Question Self-understanding:

1.1 Description:

This question asks us to maximize profit by determining when to buy and when to sell a single share of stock.

1.2 Input:

A list of integers where each integer represents the value of a stock on a given day.

1.3 Input Assumption

- All prices are non-negative (at least 0).
- The list of prices is not empty.

1.4 Output:

Return an integer that represents the maximum profit possible.

1.5 Example:

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

1.6 Other Q&A:

- None for this question.

2. Attempt 1:

2.1 Thought:

- A straightforward idea is to note that to maximize profit, you want to buy at a low price and sell at a higher price that appears later in the list. One way to conceptualize this is to look from the right side and keep track of the highest stock price seen so far (this would be your best potential selling price). Then, for each day when you consider buying, you can calculate the profit if you sold at that recorded best price on the right. You update the maximum profit whenever you see a better opportunity.

2.2 Pseudo-Code: (Ignore this part. It’s a draft for brainstorming.)

```
MAX-PROFIT(prices)
1   curr_max ← 0
2   curr_max_profit ← 0
3   for i ← length(prices) - 1 downto 0
4       if (curr_max - prices[i] > curr_max_profit)
5           curr_max_profit ← curr_max - prices[i]
6       if (prices[i] > curr_max)
```

```
        curr_max ← prices[i]
return curr_max_profit
```

2.3 Implementation through python:

```
class Solution:
    def maxProfit(self, prices: List[int]) → int:
        curr_max      = 0    # best “sell” price seen so far (to our right)
        curr_max_profit = 0  # best profit seen so far
        i = len(prices) - 1

        while i >= 0:

            # “If we bought at price[i] and sold later at curr_max,
            #  how much would we make?”
            if curr_max - prices[i] > curr_max_profit:
                curr_max_profit = curr_max - prices[i]

            # “Is today’s price a better sell-point than any we’ve seen?”
            if prices[i] > curr_max:
                curr_max = prices[i]

            i -= 1

        return curr_max_profit
```

2.4 Time Complexity and Space Complexity

2.4.1 Time Complexity:

- $O(n)$. We traverse the price list ***once*** from right to left.

2.4.2 Space Complexity:

- $O(1)$. We only keep a few variables (`curr_max`, `curr_max_profit`, and index `i`).