
Problem Statement:

Given an array of stock prices (where each element represents the price on a given day), determine the maximum profit achievable by buying on one day and selling on a later day. If no profit is possible, return 0.

Solution 1

```python
def max_profit(prices: list[int]) -> int:
    if not prices or len(prices) < 2:
        return 0 # Empty list

    lowest = prices[0]
    max_profit = 0

    for price in prices[1:]:
        max_profit = max(max_profit, price - lowest)
        if price < lowest:
            lowest = price

    return max_profit
```

Step-by-Step Breakdown

1. **Input:**
    - prices: list of integers representing daily stock prices.
      Example: [7, 1, 5, 3, 6, 4]
2. **Intermittent Step 1: Validate Input**
    - Check if the list is empty or contains fewer than 2 elements.
        - If true, return 0 - at least two prices are needed for a buy-sell transaction.
3. **Intermittent Step 2: Initialization**
    - Set lowest to the first price (prices[0]). This will track the minimum price encountered so far.
    - Initialize max_profit to 0. This will track the maximum profit computed.
4. **Intermittent Step 3: Iteration and Update**
    - Iterate over each price in prices[1:] (starting from the second element).
        - For each price:
            - Compute Profit:
            - Calculate the potential profit as price - lowest. b. Update Maximum Profit:

- Set max_profit to the maximum of the current max_profit and the computed profit. c. Update Lowest Price:
- If the current price is lower than lowest, update lowest to this new value.

5. **Output:**
   - Return the value of max_profit, which represents the maximum achievable profit.
6. **Efficiency:**
   - **Time Complexity: O(n)**
     - The algorithm processes the list in a single pass.
   - **Space Complexity: O(1)**

Visual Flow Diagram