Best Time to Buy and Sell Stock - Notes

This problem is a classic stock trading problem: you are given an array `prices`, where `prices[i]` is the price of a stock on day `i`. 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. ### Key Idea: - Track the **minimum price** seen so far (best day to buy). - For each day, calculate the profit if you sell on that day (`prices[i] - minimumPrice`). - Update the maximum profit whenever you find a larger profit. - Update the minimum price whenever a new lower price is found. ### Complexity: - Time Complexity: O(n), where n is the number of days. - Space Complexity: O(1). Below is the C++ code implementation:

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int minimumPrice = prices[0];
        int maximumProfit = 0;
        int n = prices.size();

        for(int i = 0; i < n; i++){
            int currentProfit = prices[i] - minimumPrice;
            maximumProfit = max(currentProfit, maximumProfit);
            minimumPrice = min(minimumPrice, prices[i]);
        }

        return maximumProfit;
    }
};
```