

# ARRAY

# Arrays

- Hold values in a **contiguous** block of memory
- The size of the array is **fixed**. However, high-level languages have different implementations, making it dynamic

# Arrays – Kadane's algorithm

Arrays – Kadane's algorithm

# Problem - Best Time to Buy and Sell Stock

Easy

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

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

## 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.

# Solution - Best Time to Buy and Sell Stock

Easy

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

```
int maxProfit(vector<int>& prices) {  
    int profit = 0;  
    int buy = prices[0];  
    for (auto i = 1; i < prices.size(); i++) {  
        if (prices[i] < buy) {  
            buy = prices[i];  
        } else if (prices[i] - buy > profit) {  
            profit = prices[i] - buy;  
        }  
    }  
    return profit;  
}
```