Problem 1: **Best Time to Buy and Sell Stock**

<https://www.youtube.com/watch?v=34WE6kwq49U>

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.

**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.

**Constraints:**

* 1 <= prices.length <= 105
* 0 <= prices[i] <= 104

class Solution {

public:

int maxProfit(vector<int>& prices) {

int maxProfit = 0;

int min = prices[0];

for(int i=1;i<prices.size();i++){

if(maxProfit < prices[i] - min){

maxProfit = prices[i] - min;

}

if(min > prices[i]){

min = prices[i];

}

}

return maxProfit;

}

};

Problem 2: **Best Time to Buy and Sell Stock II**

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

Find the maximum profit you can achieve. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times).

**Note:** You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

**Example 1:**

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

**Output:** 7

**Explanation:** Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4.

Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.

**Example 2:**

**Input:** prices = [1,2,3,4,5]

**Output:** 4

**Explanation:** Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.

Note that you cannot buy on day 1, buy on day 2 and sell them later, as you are engaging multiple transactions at the same time. You must sell before buying again.

**Example 3:**

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

**Output:** 0

**Explanation:** In this case, no transaction is done, i.e., max profit = 0.

**Constraints:**

* 1 <= prices.length <= 3 \* 104
* 0 <= prices[i] <= 104

class Solution {

public:

int maxProfit(vector<int>& prices) {

int profit = 0;

for(int i=1;i<prices.size();i++){

if(prices[i] > prices[i-1]){

profit += (prices[i] - prices[i-1]);

}

}

return profit;

}

};