121. Best Time to Buy and Sell Stock

Easy 7274 323 Add to List Share

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

Constraints:

- `1 <= prices.length <= 10^5`
- `0 <= prices[i] <= 10^4`

Base case:

T[-1][k][0] = T[i][0][0] = 0

T[-1][k][1] = T[i][0][1] = -Infinity

Recurrence relations:

T[i][k][0] = max(T[i-1][k][0], T[i-1][k][1] + price[i])

T[i][k][1] = max(T[i-1][k][1], T[i-1][k-1][0] - price[i])

k = 1:

Two unknown variables each day

T[i][1][0] = max(T[i-1][1][0], T[i-1][1][1] + price[i])

T[i][1][1] = max(T[i-1][1][1], T[i-1][0][1] + price[i])

= max(T[i-1][1][1], -price[i])

```cpp
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        // base case
        int t_i_1_0 = 0; // T[-1][1][0]
        int t_i_1_1 = INT_MIN; // T[-1][1][1]

        // recurrence
        for (int price : prices) {
            // T[i][1][0] = max(T[i-1][1][0], T[i-1][1][1] + prices[i])
            t_i_1_0 = max(t_i_1_0, t_i_1_1 + price);
            // T[i][1][1] = max(T[i-1][1][1], -prices[i])
            t_i_1_1 = max(t_i_1_1, -price);
        }

        return t_i_1_0;
    }
};
```