122. Best Time to Buy and Sell Stock II

Easy 3508 1902 Add to List Share

Say you have an array `prices` for which the i^{th} element is the price of a given stock on day *i*.

Design an algorithm to find the maximum profit. 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 at the same time (i.e., you must sell the stock before you buy again).

Example 1:

```
Input: [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: [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: [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 * 10 ^ 4`
- `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 = +Infinity

there isn't any difference between k and k-1

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

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

There are two unknow variables each day:

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][0] - price[i])

```cpp
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        // base case
        int t_i_k_0 = 0;    // T[-1][k][0] = 0
        int t_i_k_1 = INT_MIN;  // T[-1][k][1] = -Infinity

        // recurrence
        for (auto price : prices) {
            int t_i_k_0_temp = t_i_k_0;
            // T[i][k][0] = max(T[i-1][k][0], T[i-1][k][1] + prices[i])
            t_i_k_0 = max(t_i_k_0_temp, t_i_k_1 + price);
            // T[i][k][1] = max(T[i-1][k][1], T[i-1][k][0] - prices[i])
            t_i_k_1 = max(t_i_k_1, t_i_k_0_temp - price);
        }

        return t_i_k_0;
    }
};
```