714. Best Time to Buy and Sell Stock with Transaction Fee

Medium 1991 61 Add to List Share

Your are given an array of integers `prices`, for which the `i`-th element is the price of a given stock on day `i`; and a non-negative integer `fee` representing a transaction fee.

You may complete as many transactions as you like, but you need to pay the transaction fee for each transaction. You may not buy more than 1 share of a stock at a time (ie. you must sell the stock share before you buy again.)

Return the maximum profit you can make.

Example 1:

```
Input: prices = [1, 3, 2, 8, 4, 9], fee = 2
Output: 8
Explanation: The maximum profit can be achieved by:
• Buying at prices[0] = 1
• Selling at prices[3] = 8
• Buying at prices[4] = 4
• Selling at prices[5] = 9
The total profit is ((8 - 1) - 2) + ((9 - 4) - 2) = 8.
```

Note:

- `0 < prices.length <= 50000`.
- `0 < prices[i] < 50000`.
- `0 <= fee < 50000`.

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 but with transaction fee

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

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

Time complexity: O(n)

Space complexity: O(1)

```cpp
class Solution {
public:
    int maxProfit(vector<int>& prices, int fee) {
        // 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] - fee)
            t_i_k_1 = max(t_i_k_1, t_i_k_0_temp - price - fee);
        }

        return t_i_k_0;
    }
};
```