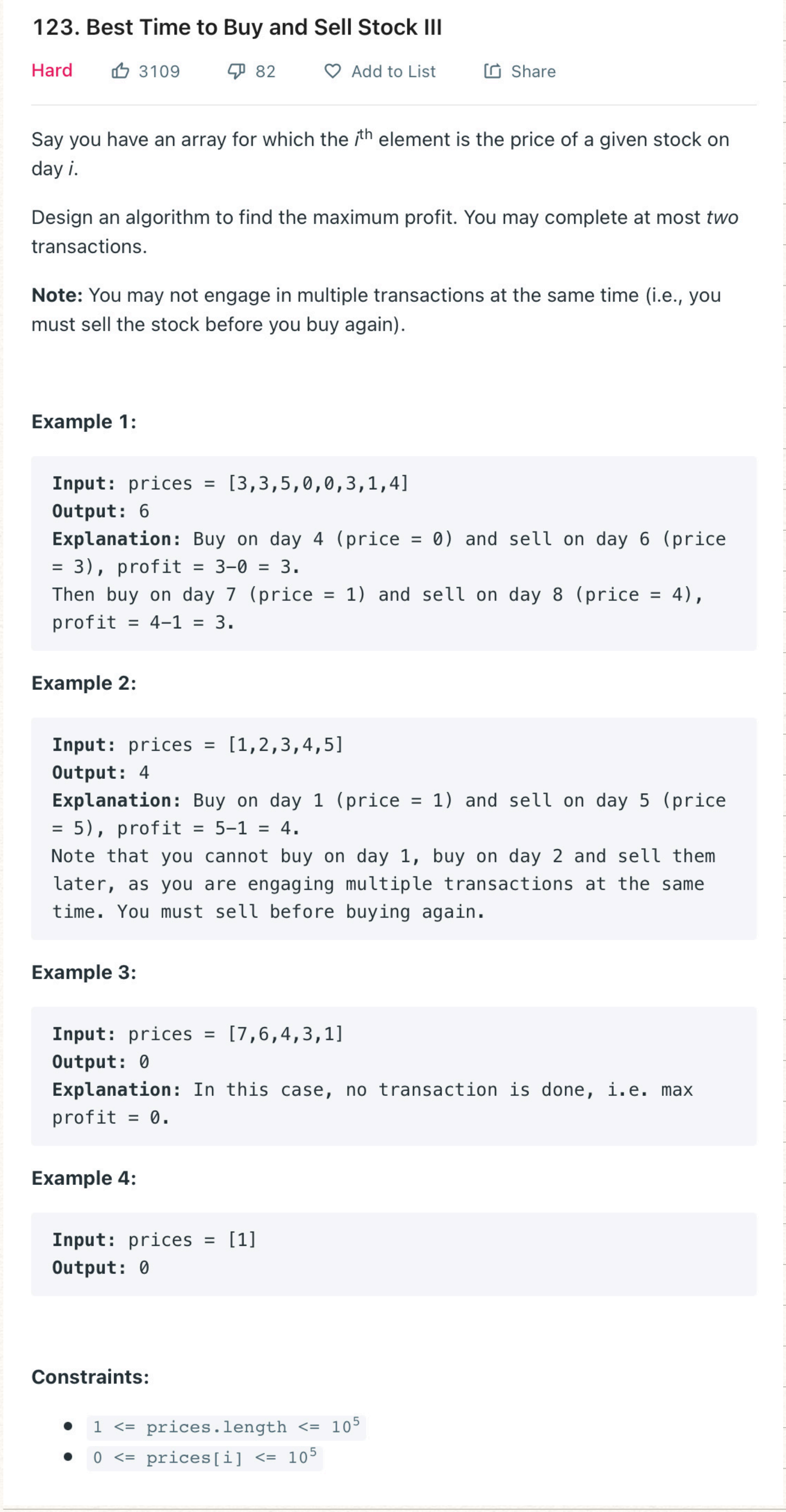


123. Best Time to Buy and Sell Stock III

Hard 3109 82 Add to List Share

Say you have an array 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 at most *two* transactions.

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: prices = [3,3,5,0,0,3,1,4]
Output: 6
Explanation: Buy on day 4 (price = 0) and sell on day 6 (price
= 3), profit = 3-0 = 3.
Then buy on day 7 (price = 1) and sell on day 8 (price = 4),
profit = 4-1 = 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.
```

Example 4:

```
Input: prices = [1]
Output: 0
```

Constraints:

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

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 = 2:

There are four unknown variables each day

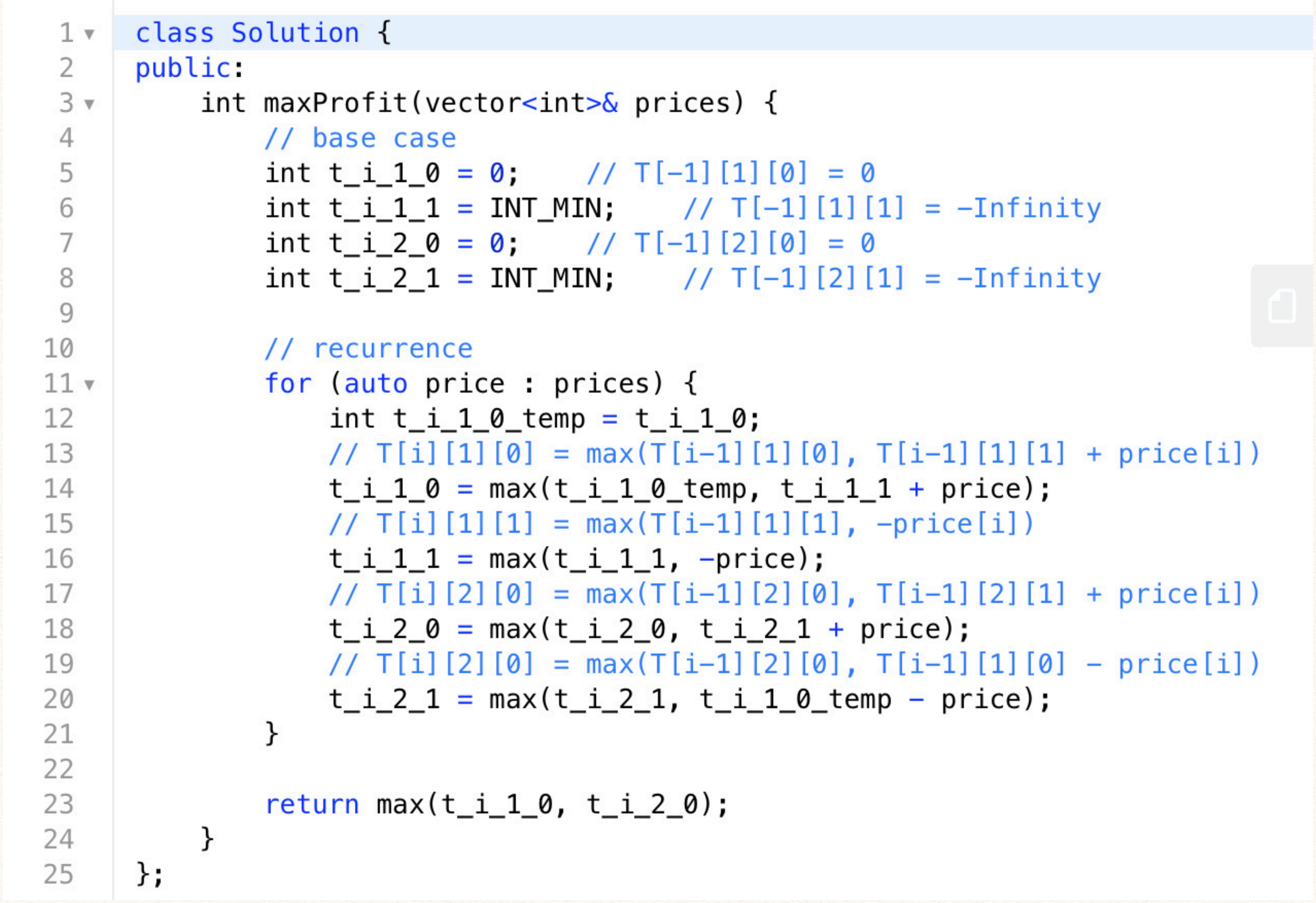
T[i][1][0] = max(T[i-1][1][0], T[i-1][1][1] + prices[i])

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

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

T[i][2][0] = max(T[i-1][2][0], T[i-1][2][1] + prices[i])

T[i][2][1] = max(T[i-1][2][1], T[i-1][1][0] - prices[i])



```cpp
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        // base case
        int t_i_1_0 = 0;    // T[-1][1][0] = 0
        int t_i_1_1 = INT_MIN;    // T[-1][1][1] = -Infinity
        int t_i_2_0 = 0;    // T[-1][2][0] = 0
        int t_i_2_1 = INT_MIN;    // T[-1][2][1] = -Infinity

        // recurrence
        for (auto price : prices) {
            int t_i_1_0_temp = t_i_1_0;
            // T[i][1][0] = max(T[i-1][1][0], T[i-1][1][1] + price[i])
            t_i_1_0 = max(t_i_1_0_temp, t_i_1_1 + price);
            // T[i][1][1] = max(T[i-1][1][1], -price[i])
            t_i_1_1 = max(t_i_1_1, -price);
            // T[i][2][0] = max(T[i-1][2][0], T[i-1][2][1] + price[i])
            t_i_2_0 = max(t_i_2_0, t_i_2_1 + price);
            // T[i][2][0] = max(T[i-1][2][0], T[i-1][1][0] - price[i])
            t_i_2_1 = max(t_i_2_1, t_i_1_0_temp - price);
        }

        return max(t_i_1_0, t_i_2_0);
    }
};
```