38) Best time to buy and sell stock II

09 December 2024 03:03

You are given an integer array prices where prices[i] is the price of a given stock on the ith day.
On each day, you may decide to buy and/or sell the stock. You can only hold **at most one** share of the stock at any time. However, you can buy it then immediately sell it on the **same day**.
Find and return *the **maximum** profit you can achieve*.

From <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/description/>

```
    int helper(int day, int buy, vector<int>& prices, int &n,  vector<vector<int>>
&dp){
        if(day == n) return 0;
        if(dp[day][buy] != -1) return dp[day][buy];

        int profit = 0;
        if(buy){
            profit = max(-prices[day] + helper(day+1, 0, prices, n, dp), (0 +
helper(day+1, 1, prices, n, dp)));
        }
        else{
            profit = max(prices[day] + helper(day+1, 1, prices, n, dp), (0 +
helper(day+1, 0, prices, n, dp)));
        }
        return dp[day][buy] = profit;
    }
    int maxProfit(vector<int>& prices) {
        int n = prices.size();
        vector<vector<int>> dp(n, vector<int>(2, -1));
        return helper(0, 1, prices, n, dp);
    }
```

39) Best time to buy and sell stock III

09 December 2024 03:15

You are given an array prices where prices[i] is the price of a given stock on the i^{th} day.
Find the maximum profit you can achieve. You may complete **at most two transactions**.
Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

From <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/description/>

```
    int helper(int day, int buy, vector<int>& prices, int
&n,  vector<vector<vector<int>>>& dp, int cap){
        if(cap == 0) return 0;
        if(day == n) return 0;
        if(dp[day][buy][cap] != -1) return dp[day][buy][cap];

        int profit = 0;
        if(buy){
            profit = max(-prices[day] + helper(day+1, 0, prices, n, dp, cap), (0 +
helper(day+1, 1, prices, n, dp, cap)));
        }
        else{
            profit = max(prices[day] + helper(day+1, 1, prices, n, dp, cap-1), (0 +
helper(day+1, 0, prices, n, dp, cap)));
        }
        return dp[day][buy][cap] = profit;
    }
    int maxProfit(vector<int>& prices) {
        int n = prices.size();
        int cap = 2;
        vector<vector<vector<int>>> dp(n, vector<vector<int>>(2, vector<int>
(3, -1)));
        return helper(0, 1, prices, n, dp, cap);
    }
```

40) Best time to buy and sell stock IV

09 December 2024 03:16

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

Find the maximum profit you can achieve. You may complete at most k transactions: i.e. you may buy at most k times and sell at most k times.

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

From <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iv/description/>

```
    int helper(int day, int buy, vector<int>& prices, int
&n,  vector<vector<vector<int>>>& dp, int cap){
        if(cap == 0) return 0;
        if(day == n) return 0;
        if(dp[day][buy][cap] != -1) return dp[day][buy][cap];

        int profit = 0;
        if(buy){
            profit = max(-prices[day] + helper(day+1, 0, prices, n, dp, cap), (0 +
helper(day+1, 1, prices, n, dp, cap)));
        }
        else{
            profit = max(prices[day] + helper(day+1, 1, prices, n, dp, cap-1), (0 +
helper(day+1, 0, prices, n, dp, cap)));
        }
        return dp[day][buy][cap] = profit;
    }
    int maxProfit(int k, vector<int>& prices) {
        int n = prices.size();
        int cap = k;
        vector<vector<vector<int>>> dp(n, vector<vector<int>>(2, vector<int>(k+
1, -1)));
        return helper(0, 1, prices, n, dp, cap);
    }
```

41) Best time to buy and sell stock with cooldown

09 December 2024 03:22

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) with the following restrictions:

- After you sell your stock, you cannot buy stock on the next day (i.e., cooldown one day).

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

From <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-cooldown/description/>

```
    int helper(int day, int buy, vector<int>& prices, int &n,  vector<vector<int>> 
&dp){
        if(day >= n) return 0;
        if(dp[day][buy] != -1) return dp[day][buy];

        int profit = 0;
        if(buy){
            profit = max(-prices[day] + helper(day+1, 0, prices, n, dp), (0 + 
helper(day+1, 1, prices, n, dp)));
        }
        else{
            profit = max(prices[day] + helper(day+2, 1, prices, n, dp), (0 + 
helper(day+1, 0, prices, n, dp)));
        }
        return dp[day][buy] = profit;
    }
    int maxProfit(vector<int>& prices) {
        int n = prices.size();
        vector<vector<int>> dp(n, vector<int>(2, -1));
        return helper(0, 1, prices, n, dp);
    }
```

42) Best Time to Buy and Sell Stock with Transaction Fee

09 December 2024 03:27

You are given an array prices where prices[i] is the price of a given stock on the ith day, and an integer fee representing a transaction fee.
Find the maximum profit you can achieve. You may complete as many transactions as you like, but you need to pay the transaction fee for each transaction.

From <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-transaction-fee/description/>

```
    int helper(int day, int buy, vector<int>& prices, int &n,  vector<vector<int>>
&dp, int &fee){
        if(day == n) return 0;
        if(dp[day][buy] != -1) return dp[day][buy];

        int profit = 0;
        if(buy){
            profit = max(-prices[day] + helper(day+1, 0, prices, n, dp, fee), (0 +
helper(day+1, 1, prices, n, dp, fee)));
        }
        else{
            profit = max(prices[day] - fee + helper(day+1, 1, prices, n, dp, fee),
(0 + helper(day+1, 0, prices, n, dp, fee)));
        }
        return dp[day][buy] = profit;
    }
    int maxProfit(vector<int>& prices, int fee) {
        int n = prices.size();
        vector<vector<int>> dp(n, vector<int>(2, -1));
        return helper(0, 1, prices, n, dp, fee);
    }
```