WAVELABS

To find the maximum profit, we can use the concept of Dynamic Programming. We will create two arrays, one that will keep track of the maximum profit that can be made by buying and selling only once, and the other that will keep track of the maximum profit that can be made by buying and selling twice.

To calculate the maximum profit that can be made by buying and selling only once, we will traverse the array from left to right, keeping track of the minimum price so far and the maximum profit that can be made by selling at the current price.

To calculate the maximum profit that can be made by buying and selling twice, we will traverse the array from right to left, keeping track of the maximum price so far and the maximum profit that can be made by buying at the current price and selling at the maximum price found so far.

Finally, we will traverse the array again to find the maximum of the two maximum profits calculated.

CODE:-

```
#include <bits/stdc++.h>

#include <iostream>

using namespace std;

int maxProfit(vector<int>& prices) {

    int n = prices.size();

    if (n <= 1) {

        return 0;

    }

   // calculate the maximum profit of one transaction from left to right

    vector<int> left_profit(n, 0);

    int left_min = prices[0];

    for (int i = 1; i < n; i++) {

        left_profit[i] = max(left_profit[i-1], prices[i] - left_min);

        left_min = min(left_min, prices[i]);

    }

  // calculate the maximum profit of one transaction from right to left

    vector<int> right_profit(n, 0);

    int right_max = prices[n-1];

    for (int i = n-2; i >= 0; i--) {

        right_profit[i] = max(right_profit[i+1], right_max - prices[i]);
        right_max = max(right_max, prices[i]);
    }


    // calculate the maximum profit of two transactions
    int max_profit = 0;
    for (int i = 0; i < n; i++) {
        max_profit = max(max_profit, left_profit[i] + right_profit[i]);
    }


    return max_profit;
}
int main(){
    vector<int> prices_month1 = {3,3,5,0,0,3,1,4};
    /* Answer : 6 */
    cout << maxProfit(prices_month1) << endl;
   vector<int> prices_month2 = {1,2,3,4,5};
    /* Answer : 4 */
    cout << maxProfit(prices_month2) << endl;
   vector<int> prices_month3 = {7,6,4,3,1};
    /* Answer : 0 */
    cout << maxProfit(prices_month3) << endl;


    return 0;
}
```

Explanation:

The solution uses dynamic programming to calculate the maximum profit of one transaction from left to right and from right to left. Then, it calculates the maximum profit of two transactions by adding the left and right profits at each day and finding the maximum sum.

Code:-