best-time-to-buy-and-sell-stock

```
package algorithm.greedy;

/**
 * https://leetcode.com/problems/best-time-to-buy-and-sell-stock/
 *
 Say you have an array for which the ith element is the price of a given st

 If you were only permitted to complete at most one transaction (ie, buy or

 * @author xiaobaoqiu  Date: 16-5-21 Time:  下午2:12
 */
public class BestTimeToBuyAndSellStock {

    public static void main(String[] args) {
        int[] price = new int[]{10, 20, 15, 30, 12};
        System.out.println(maxProfit(price));
        System.out.println(maxProfit_1(price));

        price = new int[]{0, 1};
        System.out.println(maxProfit(price));
        System.out.println(maxProfit_1(price));
    }

    /**
     * 思路：遍历数组，找到 prices[i] 买入时候能获得的最大利益
     *
     * 1 ms
     * Your runtime beats 95.55% of java submissions
     */
    public static int maxProfit(int[] prices) {
        if (prices == null || prices.length < 2) return 0;
        int[] profit = new int[prices.length];
        profit[prices.length - 1] = 0;
        int max = prices[prices.length - 1];    //max  表示prices[i]之后的最
        //找到prices[i]买入时候能获得的最大利益
        for (int i = prices.length - 2; i >= 0; i--) {
            profit[i] = max - prices[i];
            if (prices[i] > max) max = prices[i];
        }

        max = 0;
        for (int i = 0; i < prices.length; i++) {
            if (profit[i] > max) max = profit[i];
        }

        return max;
    }

    /**
     * O(1)空间复杂度
     *
     * 3 ms
     * Your runtime beats 13.06% of java submissions.
     */
    public static int maxProfit_1(int[] prices) {
        if (prices == null || prices.length < 2) return 0;

        int max = prices[prices.length - 1], temp;
        prices[prices.length - 1] = 0;
        for (int i = prices.length - 2; i >= 0; i--) {
            temp = prices[i];
            prices[i] = max - prices[i];
            if (temp > max) max = temp;
        }

        max = 0;
        for (int profit : prices) {
            if (profit > max) max = profit;
        }

        return max;
    }
}
```

best-time-to-buy-and-sell-stock-ii

```
package algorithm.greedy;

/**
 * https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/
 *
 Say you have an array for which the ith element is the price of a given st

 Design an algorithm to find the maximum profit.
 You may complete as many transactions as you like (ie, buy one and sell or
 However, you may not engage in multiple transactions at the same time (ie,

 * @author xiaobaoqiu  Date: 16-5-21 Time:  下午2:12
 */
public class BestTimeToBuyAndSellStockII {

    public static void main(String[] args) {
//        int[] price = new int[]{10, 20, 15, 30, 12};    //25
//        int[] price = new int[]{10};
//        int[] price = new int[]{10, 5};
//        int[] price = new int[]{1, 2};
        int[] price = new int[]{2,1,2,0,1}; //2
        System.out.println(maxProfit(price));
//        System.out.println(maxProfit_1(price));
    }

    /**
     * 每次 prices[i] > prices[i+1]  表示之前每的可以卖了
     * 时间：O(n)
     * 空间：O(1)
     *
     * 3 ms
     * Your runtime beats 4.02% of java submissions
     */
    public static int maxProfit(int[] prices) {
        if (prices == null || prices.length == 0) return 0;
        int profit = 0, pos = 0;    //pos 表示这次买入的时间
        for (int i = 1; i < prices.length; i++) {
            if (prices[i] > prices[i-1]) continue;
            profit += (prices[i-1] - prices[pos]);
            pos = i;
        }
        if (pos != prices.length - 1) profit += (prices[prices.length - 1]
        return profit;
    }
}
```