**难度: 简单**

给定一个数组，它的第 i 个元素是一支给定股票第 i 天的价格。

如果你最多只允许完成一笔交易（即买入和卖出一支股票一次），设计一个算法来计算你所能获取的最大利润。

注意：你不能在买入股票前卖出股票。

示例 1:

输入: [7,1,5,3,6,4]

输出: 5

解释: 在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大利润 = 6-1 = 5 。

注意利润不能是 7-1 = 6, 因为卖出价格需要大于买入价格；同时，你不能在买入前卖出股票。

示例 2:

输入: [7,6,4,3,1]

输出: 0

解释: 在这种情况下, 没有交易完成, 所以最大利润为 0。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/best-time-to-buy-and-sell-stock

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1(暴力破解):**

1). 创建一个最大值

2). 每次用卖出的价格-买入的价格

3). 然后持续比较

|  |
| --- |
| public int maxProfit(int[] prices) {  int max = 0;  //每次用卖出的价钱 - 买入的价钱  for (int i = 0; i < prices.length; i++) {  for (int j = i+1; j < prices.length; j++) {  if (max < prices[j] - prices[i]) {  max = prices[j] - prices[i];  }  }  }  return max;  } |

**思路2(代码有注释):**

|  |
| --- |
| public int maxProfit2(int[] prices) {  //最小买入  int minprice = Integer.MAX\_VALUE;  //最大卖出  int maxprofit = 0;  for (int i = 0; i < prices.length; i++) {  //当前值小于 最小买入  if (prices[i] < minprice) {  minprice = prices[i];  //比较当前值 减 最大卖出 大于 最大卖出值  }else if (prices[i] - minprice > maxprofit) {  maxprofit = prices[i] - minprice;  }  }  //返回最大卖出值  return maxprofit;  } |