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

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return 0.

**Example 1:**

**Input:** prices = [7,1,5,3,6,4]

**Output:** 5

**Explanation:** Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

**Example 2:**

**Input:** prices = [7,6,4,3,1]

**Output:** 0

**Explanation:** In this case, no transactions are done and the max profit = 0.

**Constraints:**

- 1 <= prices.length <= 105
- 0 <= prices[i] <= 104

**Approach:**
The above code calculates the maximum profit from stock prices by iterating through the prices array, tracking the minimum price up to each day and comparing it with the current price to update the maximum profit, using dynamic programming.

**Code:**

```
class Solution {
```

```
    public int maxProfit(int[] prices) {
        if(prices.length<=1)  return 0;
        int max=Integer.MIN_VALUE,n=prices.length;
        int[] dp=new int[n];
        dp[0]=prices[0];
        for(int i=1;i<n;i++){
            if(prices[i]-dp[i-1]<=0)   max=Math.max(0,max);
            else    max=Math.max(max,prices[i]-dp[i-1]);
            dp[i]=Math.min(prices[i],dp[i-1]);
        }
        return max;
    }
}
```