1. Problem description:

Given an integer array `prices` representing the stock prices for consecutive days, return an array `answer` where `answer[i]` represents the number of days until the next price increase for the `i`-th day. If there is no price increase in the future, replace the value at that position with `0`.

Example 1:

```
Input: prices = [73, 74, 75, 71, 69, 72, 76, 73]
Output: [1, 1, 4, 2, 1, 1, 0, 0]
```

Example 2:

```
Input: prices = [30, 40, 50, 60]
Output: [1, 1, 1, 0]
```

Example 3:

```
Input: prices = [30, 60, 90]
Output: [1, 1, 0]
```

Tips:

- `1 <= prices.length <= 105`
- `30 <= prices[i] <= 100`

2. Problem description:

Given an array `prices` where the `i`-th element `prices[i]` represents the price of a given stock on the `i`-th day.
You can only choose **one day** to buy the stock and choose **a different day in the future** to sell it.
Design an algorithm to calculate the maximum profit you can achieve from this transaction.
Return the maximum profit you can obtain from this transaction. If you cannot make any profit, return `0`.

Example 1:

```
Input: prices = [7, 1, 5, 3, 6, 4]
Output: 5
Explanation: Buy on day 2 (stock price = 1) and sell on day 5 (stock price = 6).
             The maximum profit is 6-1 = 5.
             Note that the profit cannot be 7-1 = 6 because the selling price
             must be greater than the buying price. Also, you cannot sell the
             stock before buying it.
```

Example 2:

```
Input: prices = [7, 6, 4, 3, 1]
Output: 0
Explanation: In this case, no transaction is completed, so the maximum profit is
0.
```

Tips:

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

3. Problem description:

Given an integer array `prices`, where `prices[i]` represents the price of a stock on the `i`-th day. Design an algorithm to calculate the maximum profit. You can complete as many transactions as possible (buying and selling stocks multiple times) under the following constraints:

- After selling a stock, you cannot buy another stock on the next day (there is a cooldown period of 1 day).

Note: You cannot participate in multiple transactions simultaneously (you must sell the previous stock before buying again).

Example 1:

```
Input: prices = [1, 2, 3, 0, 2]
Output: 3
Explanation: The corresponding transaction status is: [Buy, Sell, Cooldown, Buy,
Sell].
```

Example 2:

```
Input: prices = [1]
Output: 0
```

Tips:

- `1 <= prices.length <= 5000`
- `0 <= prices[i] <= 1000`