<https://leetcode.com/problems/final-prices-with-a-special-discount-in-a-shop/description/>

You are given an integer array prices where prices[i] is the price of the ith item in a shop.

There is a special discount for items in the shop. If you buy the ith item, then you will receive a discount equivalent to prices[j] where j is the minimum index such that j > i and prices[j] <= prices[i]. Otherwise, you will not receive any discount at all.

Return an integer array answer where answer[i] is the final price you will pay for the ith item of the shop, considering the special discount.

**Example 1:**

**Input:** prices = [8,4,6,2,3]

**Output:** [4,2,4,2,3]

**Explanation:**

For item 0 with price[0]=8 you will receive a discount equivalent to prices[1]=4, therefore, the final price you will pay is 8 - 4 = 4.

For item 1 with price[1]=4 you will receive a discount equivalent to prices[3]=2, therefore, the final price you will pay is 4 - 2 = 2.

For item 2 with price[2]=6 you will receive a discount equivalent to prices[3]=2, therefore, the final price you will pay is 6 - 2 = 4.

For items 3 and 4 you will not receive any discount at all.

**Example 2:**

**Input:** prices = [1,2,3,4,5]

**Output:** [1,2,3,4,5]

**Explanation:** In this case, for all items, you will not receive any discount at all.

**Example 3:**

**Input:** prices = [10,1,1,6]

**Output:** [9,0,1,6]

**Constraints:**

1 <= prices.length <= 500

1 <= prices[i] <= 1000

**Attempt 1: 2024-10-23**

**Solution 1: Monotonic Increasing Stack (10 min)**

class Solution {

    public int[] finalPrices(int[] prices) {

        // Intuition 1: Find next smaller element we can relate

        // to Monotonic Increasing Stack, when seen smaller

        // element than current element on Stack peek, we pop

        // out all larger elements on Stack than this element

        // Intuition 2: Store index instead of actual value,

        // because we need index to manipulate on original array

        Stack<Integer> stack = new Stack<>();

        for(int i = 0; i < prices.length; i++) {

            while(!stack.isEmpty() && prices[stack.peek()] >= prices[i]) {

                prices[stack.pop()] -= prices[i];

            }

            stack.push(i);

        }

        return prices;

    }

}

Time Complexity: O(n)

Space Complexity: O(n)

**Refer to**

<https://leetcode.com/problems/final-prices-with-a-special-discount-in-a-shop/solutions/685390/java-c-python-stack-one-pass/>

**Intuition**

Similar to the problem [503. Next Greater Element II](https://leetcode.com/problems/next-greater-element-ii/discuss/98270/JavaC++Python-Loop-Twice)

**Java:**

public int[] finalPrices(int[] A) {

Stack<Integer> stack = new Stack<>();

for (int i = 0; i < A.length; i++) {

while (!stack.isEmpty() && A[stack.peek()] >= A[i])

A[stack.pop()] -= A[i];

stack.push(i);

}

return A;

}

**Refer to**

[L84.Largest Rectangle in Histogram](note://E2FC22D9715540C5A8F06F60FBBE886F)