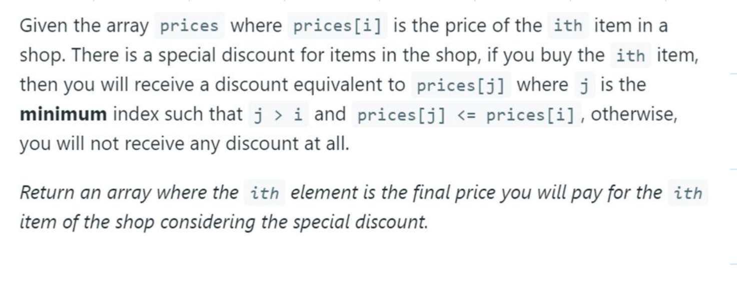
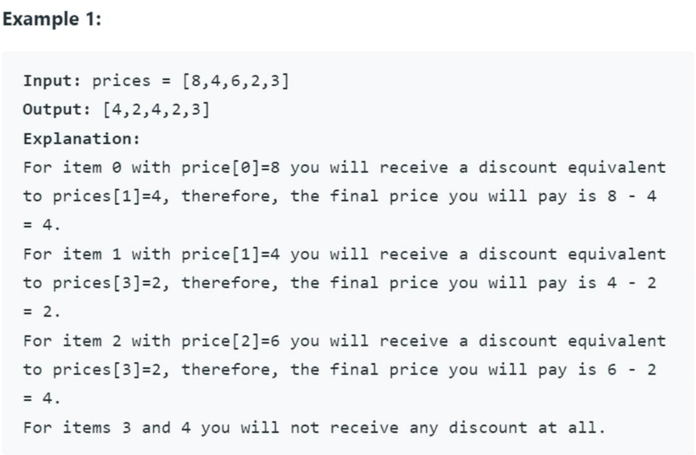
1475. Final Prices With a Special Discount in a Shop

08 February 2022 05:11 PM



Given the 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 array where the `ith` element 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.
```

Monotonic Stack See the concept for that (In the notes)

$$\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \end{array}$$

$[8, 4, 6, 2, 3] \rightarrow$ prices

Discount conditions :

if you buy i^{th} item, then you will receive a discount equivalent to prices[j] when j is the mini index $j > i$ and prices $[j] \leq$ prices$[i]$, otherwise no discount.

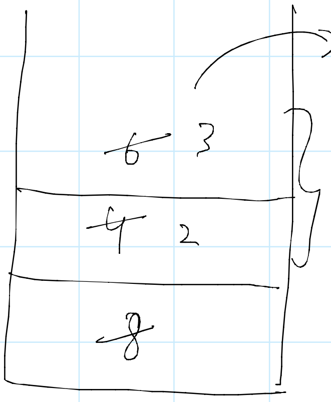
return the final price

$$\begin{array}{ccccc} & i & j & & \\ [8, & 4, & 6, & 2, & 3] \\ 0 & 1 & 2 & 3 & 4 \end{array}$$

$$= [8 - 4 = 4, \underbrace{4 - 2 = 2}_{arr[1] - arr[3]}, \underbrace{6 - 2 = 4}_{arr[2] - arr[3]}, \underbrace{2, 3}_{\text{No discount.}}]$$

follow the condition

8 4 6 2 3



pop out with original price

No discount

Increasing Stack

Ans

8 − 4 = 4

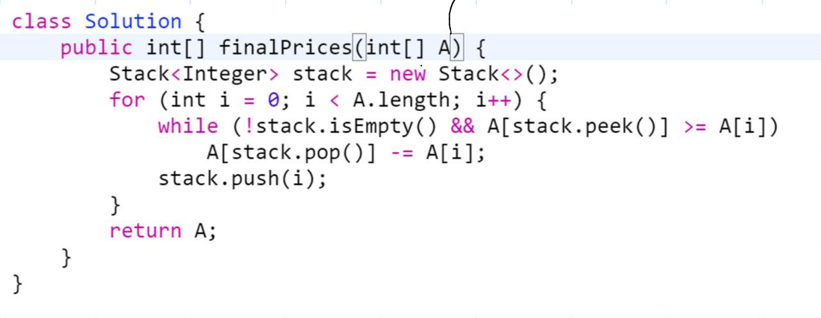
4 − 2 = 2

6 − 2 = 4.

4 2 4 2 3

T.C → O(n)

prices



```
class Solution {
    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;
    }
}
```