Nearest Smaller Distance Queue

Problem Description

Given an integer array A of length N, for each index i (1-based), define the nearest strictly smaller element as the rightmost index $j < i$ such that $A_j < A_i$. If such j exists, output the distance $i - j$; otherwise, output -1.

Next, take all distance values that are not -1 (in increasing order of their indices) and enqueue them into a queue. Finally, output the contents of this queue. If the queue is empty, output `EMPTY`.

You are expected to compute all distances in linear time using a monotonic increasing stack and perform queue operations in amortized $O(1)$ time. Naïve solutions will time out.

Input Format

The input consists of two lines:

- The first line contains an integer N.
- The second line contains N integers $A_1, A_2, \ldots, A_N$.

Output Format

- Line 1: N space-separated integers —the distance for each position.
- Line 2: the queue contents formed by enqueuing all distances that are not -1 (space-separated). If the queue is empty, print `EMPTY`.

Constraints

- $1 \leq N \leq 2 \times 10^5$ ◦
- $-10^9 \leq A_i \leq 10^9$ ◦
- Time Limit: 1 second ◦
- Memory Limit: 256 MB ◦

Additional Notes

1. **Complexity Target:** Use a monotonic increasing stack to achieve overall $O(N)$ time.

A brute-force scan to the left for every position is $O(N^2)$ and will not pass within the time limit when $N = 2 \times 10^5$.

Example Test Case

Sample Input 1

```
6
2 1 4 3 5 6
```

Sample Output 1

```
-1 -1 1 2 1 1
1 2 1 1
```

Explanation 1

- $i = 1$: no elements to the left $\rightarrow -1$.
- $i = 2$: left side [2] has nothing $< 1 \rightarrow -1$.
- $i = 3$: nearest strictly smaller is $A[2] = 1$, distance $3 - 2 = 1$.
- $i = 4$: nearest strictly smaller is $A[2] = 1$, distance $4 - 2 = 2$.
- $i = 5$: nearest strictly smaller is $A[4] = 3$, distance 1.
- $i = 6$: nearest strictly smaller is $A[5] = 5$, distance 1.

Queue of non-`(-1)` distances: `1 2 1 1`.

Sample Input 2

```
5
1 2 3 4 5
```

Sample Output 2

```
-1 1 1 1 1
1 1 1 1
```

Explanation 2

Strictly increasing sequence: for every position after the first, the nearest strictly smaller element is the immediate predecessor, so each distance is 1.

Sample Input 3

```
5
5 4 4 4 1
```

Sample Output 3

```
-1 -1 -1 -1 -1
EMPTY
```

Explanation 3

We require strictly smaller. For each 4, the left side has 5 (not smaller) or equal 4, so all distances are -1; thus the queue is empty.