Partially Sorted Array

Data Structures and Algorithms, Group I

The values stored in an array may be more or less sorted. In many cases it is useful to know if an array is *more or less* sorted. For example, the *quicksort* algorithm is $O(n^2)$ if the array is sorted, whereas the complexity of the *insertion sort* algorithm for *almost* sorted arrays is *almost* linear.

In this problem, we say an array is *almost sorted* if the maximum element of the right half is $\geq$ all the elements of the left half and the minimum element of the left half is $\leq$ all the elements of the right half. In addition, both the left and the right halves are *almost sorted*.

For a given array of positive integer numbers, write a recursive algorithm that checks whether the array is *partially sorted* or not.

You have to specify the algorithm, code it and calculate its complexity.

Input

The input consists in a series of test cases. Each test case contains the values stored in the array, with a final value of 0 that is not part of the array. The last test case is only a 0 that must not be processed.

The number of values in each test case is a power of 2.

Output

For each test case the program must write SI if the array is *partially sorted* and NO otherwise.

Sample Input	**Sample Output**
2 6 3 8 0	SI
6 12 8 18 10 15 16 40 0	SI
5 5 5 5 0	SI
2 6 1 8 0	NO
1 3 2 5 3 1 3 4 0	NO
0	

Notes

This exercise must be understood in the context of the Data Structures and Algorithms course, FDI-UCM 2017/2018 (prof. Gonzalo Méndez). Therefore, the only valid solutions are those that use the concepts studied in this course. Additional remarks may be provided in class.

The original problem was designed by Isabel Pita.