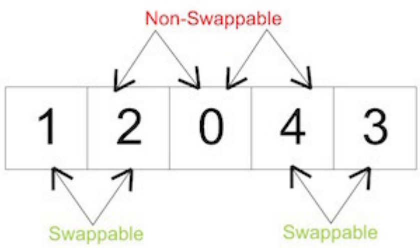
Accurate Sorting

Consider an unsorted array, $A = a_0, a_1, \cdots, a_{n-1}$, of distinct integers from 0 to $n-1$. We can *swap* two adjacent elements in A any number of times as long as the absolute difference between these elements is 1.

For example, the diagram below depicts an array where we can swap adjacent elements 1 and 2 or 4 and 3, but we cannot swap adjacent elements 2 and 0 or 0 and 4:



Answer q queries, where each query consists of some array A. For each query, print Yes on a new line if it's possible to sort the array in ascending order by performing the swap operation defined above; otherwise, print No instead.

Input Format

The first line contains a single integer denoting q. The subsequent lines describe each of the q queries in the following format:

1. The first line contains an integer denoting n.
2. The second line contains n space-separated integers describing the respective values of $a_0, a_1, \cdots, a_{n-1}$.

Constraints

- $1 \le q \le 10$
- $1 \le n \le 10^5$
- The sum of n over all queries doesn't exceed 10^5.

Output Format

For each query, print Yes on a new line if it's possible to sort the array; otherwise, print No instead.

Sample Input 0

```
2
4
1 0 3 2
3
2 1 0
```

Sample Output 0

```
Yes
No
```

Explanation 0

We perform the following $q = 2$ queries:

1. The following sequence of swaps will sort the array in ascending order:

   $A = [1, 0, 3, 2] \rightarrow [\mathbf{0}, \mathbf{1}, 3, 2] \rightarrow [0, 1, \mathbf{2}, \mathbf{3}]$
   Because A is now sorted, we print `Yes` on a new line.

2. Initially, we can perform two possible swaps:

   1. $A = [2, 1, 0] \rightarrow [\mathbf{1}, \mathbf{2}, 0]$
      After performing this swap, no number of additional swaps can move 0 to the front of the array.
   2. $A = [2, 1, 0] \rightarrow [2, \mathbf{0}, \mathbf{1}]$
      After performing this swap, no number of additional swaps can move 2 to the back of the array.

   Because there's no way for us to sort the array, we print `No` on a new line.