Google Kickstart '17 Round F Problem A - Kicksort

Time limit: 1.0s **Memory limit:** 64M

Here at Kickstart, we are fans of the well-known Quicksort algorithm, which chooses a pivot value from a list, moves each other value into one of two new lists depending on how it compares with the pivot value, and then recursively sorts each of those new lists. However, the algorithm might choose a pivot that causes all of the other values to end up in only one of the two new lists, which defeats the purpose of the divide-and-conquer strategy. We call such a pivot a *worst-case pivot*.

To try to avoid this problem, we have created our own variant, Kicksort. Someone told us that it is good to use a value in the middle as a pivot, so our algorithm works as follows:

```
Kicksort(A): // A is a 0-indexed array with E elements
  If E ≤ 1, return A.
  Otherwise:
    Create empty new lists B and C.
    Choose A[floor((E-1)/2)] as the pivot P.
    For i = 0 to E-1, except for i = floor((E-1)/2):
      If A[i] ≤ P, append it to B.
      Otherwise, append it to C.
  Return the list Kicksort(B) + P + Kicksort(C).
```

For practice, we are trying Kicksort out on lists that are permutations of the numbers 1 through N. Unfortunately, it looks like Kicksort still has the same problem as Quicksort: it is possible for every pivot to be a worst-case pivot!

For example, consider the list `1 4 3 2`. Kicksort will choose `4` as a pivot, and all of the other values `1 3 2` will end up in one of the two new lists. Then, when Kicksort is called on that list `1 3 2`, it will choose `3`, and once again, all of the other values will end up in one of the two new lists. Finally, it will choose `1` from the list `1 2`, and the other value `2` will of course end up in only one of the two new lists. In every case, the algorithm will choose a worst-case pivot. (Notice that when Kicksort is called on a list with 0 or 1 elements, it does not choose a pivot at all.)

Please help us investigate this further! Given a permutation of the numbers 1 through N, determine whether Kicksort will choose only worst-case pivots.

Input Specification

The first line of the input gives the number of test cases, T. T test cases follow; each consists of two lines. The first line has one integer N: the number of elements in the permutation. The second line contains N integers A_i, which are a permutation of the values from 1 through N.

Output Specification

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is `YES` if Kicksort will choose only worst-case pivots when sorting this list, or `NO` otherwise.

Limits

The values A_i are a permutation of the values from 1 to N.

Small dataset

$1 \leq T \leq 32$

$2 \leq N \leq 4$

Large dataset

$1 \leq T \leq 100$

$2 \leq N \leq 10\,000$

Sample Input

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

Sample Output

```
Case #1: YES
Case #2: NO
Case #3: YES
Case #4: NO
```

Sample Case #1 is the one described in the problem statement.

In Sample Case #2, our first pivot will be `1`, which is a worst-case pivot, because it causes all of the other values `2 3 4` to end up in one of the two new lists. However, the Kicksort call on the list `2 3 4` will choose `3` as a pivot. This is not a worst-case pivot, because it puts `2` in one of the new lists, and `4` in the other.

In Sample Case #3, Kicksort will start by choosing the worst-case pivot `2`, and then it has no other pivot choices to make.

In Sample Case #4, Kicksort will start by choosing `2`, which is not a worst-case pivot.