Kick Start 2016 - Round E

Sorting Array

Problem

We are in the process of creating a somehow esoteric sorting algorithm to sort an array A of all integers between 1 and **N**. The integers in A can start in an arbitrary order. Besides the input order, the algorithm depends on two integers **P**(which would be at most 3) and **K**. Here is how the algorithms works:

1. Partition A into **K** disjoint non-empty subarrays A_1, A_2, ..., A_K such that such that concatenating them in order A_1A_2 ... A_K produces A.
2. Sort each subarray individually.
3. Choose up to **P** of the subarrays, and swap any two of them any number of times.

For example, consider A = [1 5 4 3 2] and **P** = 2. A possible partition into **K** = 4 disjoint subarrays is:

```
A1 = [1]
A2 = [5]
A3 = [4]
A4 = [3 2]

After Sorting Each Subarray:

A1 = [1]
A2 = [5]
A3 = [4]
A4 = [2 3]

After swapping A4 and A2:

A1 = [1]
A2 = [2 3]
A3 = [4]
A4 = [5]
```

We want to show the algorithm is good for distributed environments by finding, for a fixed input and value of **P**, the maximum number of partitions **K** such that, choosing the partitions and swaps wisely, we can achieve a sorting of the original order. Can you help us to calculate that **K**?

Input

The first line of the input gives the number of test cases, **T**.
T test cases follow. Each test case consists of two lines. The first line contains two integers **N** and **P**, as described above.
The second line of the test case contains **N** integers $\mathbf{X_1}$, $\mathbf{X_2}$, ..., $\mathbf{X_N}$ represting array A.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the maximum possible value for the parameter **K**.

Limits

1 ≤ **T** ≤ 100.
Time limit: 40 seconds per test set.
Memory limit: 1GB.
1 ≤ **N** ≤ 5000.
1 ≤ $\mathbf{X}_i$ ≤ **N**, for all i.
$\mathbf{X}_i$ ≠ $\mathbf{X}_j$ for all i ≠ j.

Small dataset (Test set 1 - Visible)

P = 2.

Large dataset (Test set 2 - Hidden)

P = 3.

Sample

Sample Input

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

Sample Output

```
Case #1: 4
Case #2: 2
Case #3: 3
Case #4: 3
Case #5: 6
```

Case #1:
Same as walk through in the statement.

Case #2:
[4 5] [1 2 3]
Swap the 2 blocks: [1 2 3] [4 5]

Case #3:
[6] [3 5 2 4] [1]
Sort [3 5 2 4], then swap [6] and [1], we get: [1] [2 3 4 5] [6]

Case #4:
[4 5] [1] [2 3]
Swap [4 5] and [1], then swap [2 3] and [4 5]: [1] [2 3] [4 5]

Case #5:
[1] [2] [6] [4] [5] [3]
Swap [6] and [3]: [1] [2] [3] [4] [5] [6]

Note: First 3 sample cases would not appear in the Large dataset and the last 2 sample cases would not appear in the Small dataset.