Questions asked 3

Submissions

Diwali lightings

5pt	Not attempted **1615/2160 users** correct (75%)
8pt	Not attempted **1262/1580 users** correct (80%)

Beautiful Numbers

6pt	Not attempted **1429/1592 users** correct (90%)
15pt	Not attempted **211/1189 users** correct (18%)

Partioning Number

9pt	Not attempted **646/851 users** correct (76%)
17pt	Not attempted **193/470 users** correct (41%)

Sorting Array

13pt	Not attempted **5/65 users** correct (8%)
27pt	Not attempted **2/2 users** correct (100%)

Top Scores

AngryBacon	100
LittleBuger	100
wcwswswws	78
legedexinshi	73
TheTerminalGuy	71
Shaon	71
ajs97	65
thonsi	65
john0312	65
rossSJTU	65

Problem D. Sorting Array

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 13 points	Solve D-small
Large input 27 points	Solve D-large

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 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

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{N} \le 5000$.
$1 \le \mathbf{X_i} \le \mathbf{N}$, for all i.
$\mathbf{X_i} \ne \mathbf{X_j}$ for all $i \ne j$.

Small dataset

P = 2.

Large dataset

P = 3.

Sample

Input	Output
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	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.