Problem I: Games R Us

GamesAreUs.com has just completed its outside audit for this year. One item that was caught was the lack of any business rules for assigning permissions to files on the company's shared file server. The analysts are working on setting up some roles for all employees and what permissions should be given to each role. Your team is to take a look at the existing situation so you can provide some input to the analysts.

Fortunately permission assignment has not been completely random. The most common way to set up a new employee is to ask that they be set up "just like Joe", effectively making an ad-hoc prototype system.

You will be given the access control lists (ACLs) for the top level directories of the shared file server. Using these, your team is to write a program to split the users up into equivalence classes where all members of a class have access to exactly the same directories.

Because of the several departments in GamesAreUs.com, there are multiple access control lists to be processed: one set of lists per department.

Input (from file i.in)

The first line of input to your program is a single integer **n** (**0 < n < 100**), by itself on the line without any white space, giving the number of departments. Following that are the data for those departments.

ACLs associated with one department is a sequence of ACLs ended by "-1" by itself. Each ACL is a line of unsigned integers (`1 <= x <= 2147483647`) separated from each other by a single blank. The first integer on the line is the file id (FID) of the directory. The remaining numbers on the line are the user ids (UIDs) that have access. Each line will have a FID and at least one UID. There will be no duplicate UIDs on the line, but note the UIDs and FIDs are in separate spaces so a UID could be the same integer as a FID. The ACLs, and within an ACL the UIDs, appear in no particular order. In the full list of ACLs, a given FID will appear only once. There are at most 50 top-level directory ACLs and there are at most 100 UIDs. All FIDs and UIDs are greater than 0.

Output (to monitor)

For each department, the first line of output identifies which case is being processed, beginning with 1. That line contain the word "`Case`", one blank, and the integer identifying which case it is.

For every class with at least 2 members, print a line with the number of members in the class with no sign or leading zeros, one space, and the smallest UID in the class with no sign, leading zeros, or trailing spaces. Sort the output by the number of members, descending, and then by the UIDs, ascending. If there are no such classes, print "`no prototypes found`".

Sample input

```
2
100 9 7 2 3 1 6
200 9 6 7 1 2 5 3 8
300 6 3 7 8 5
400 4 5 8
-1
100 1
200 37
-1
```

Sample output

```
Case 1
3 1
3 3
2 5
Case 2
no prototypes found
```