Gnome Sort (gnomesort)

There are N gnomes, each wearing a hat with a unique integer from 0 to $N-1$. The gnomes stand in a line in some arbitrary order. Your task is to sort them in increasing order from left to right.

Figure 1: Gnomes to be sorted.

In one operation, you can select any subset of gnomes and give them the following instruction:

"Rearrange yourselves so that everyone who was originally to your left is now to your right."

The selected gnomes can only swap positions among themselves, and after rearranging, they must occupy the same set of positions as before. The chosen gnomes do **not** need to be adjacent. It can be proven that there is always exactly one way to execute this instruction correctly.

However, the gnomes are both lazy and stubborn – they refuse to follow orders more than once. This means that each gnome can be included in **at most one** operation.

Your task:

1. Determine whether it is possible to sort the gnomes.
2. If sorting is possible, compute the minimum number of operations required.
3. Provide a sequence of operations that achieves sorting and has minimum length.

☞ Among the attachments of this task you may find a template file `gnomesort.*` with a sample incomplete implementation.

Input

The first line of the input contains a single integer N, the number of gnomes. The second line contains N integers $P_0, P_1, \ldots P_{N-1}$, the numbers on the gnomes' hats from left to right.

Output

If it is possible to sort the gnomes, print `YES` on the first line. Otherwise, print `NO`.

If sorting is possible, print an integer M (the minimum number of operations required) on the second line. Each of the next M lines should describe one operation:

- The first number on the line is the integer K, the number of gnomes selected.
- This is followed by K integers representing the numbers on the hats of the selected gnomes (in any order).

If multiple solutions exist, you may print any valid one.

Constraints

- $1 \leq N \leq 500\,000$.
- $0 \leq P_i < N$ for each $i = 0 \ldots N-1$.
- $P_i \neq P_j$ for every $0 \leq i < j < N$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points) Examples.

– **Subtask 2** (5 points) $N \leq 2$.

– **Subtask 3** (10 points) $N \leq 3$.

– **Subtask 4** (35 points) $N \leq 5000$.

– **Subtask 5** (50 points) No additional limitations.

In this task you can get **partial scores** in every subtask.

1. If you correctly determine whether it is possible to sort the gnomes in every test case of a subtask, you will receive 20% of the points.
2. If the first line is correct, and you correctly calculate the minimum number of operations needed whenever the answer is `YES`, you will get an additional 40% of the points (60% in total).

3. If the first two lines are correct, and you print a valid sequence of operations, you will get an additional 40% of the points (100% in total).

Examples

input	output
6 3 4 2 0 1 5	YES 2 3 1 2 4 2 0 3
5 2 4 0 1 3	NO

Explanation

In the **first sample case**, the initial order of the gnomes is:

$$3, 4, 2, 0, 1, 5$$

The gnomes can be sorted in two operations:

- Select gnomes 1, 2 and 4 to perform the instruction. After they rearrange themselves, the order becomes:

$$3, 1, 2, 0, 4, 5$$

- Select gnomes 0 and 3 to perform the instruction. After they rearrange, the gnomes are fully sorted:

$$0, 1, 2, 3, 4, 5$$

In the **second sample case**, sorting the gnomes is impossible because at least one gnome would need to follow more than one instruction.