E. Sorting Permutations

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

We are given a permutation sequence $a_1, a_2, ..., a_n$ of numbers from 1 to n. Let's assume that in one second, we can choose some disjoint pairs $(u_1, v_1), (u_2, v_2), ..., (u_k, v_k)$ and swap all a_{u_i} and a_{v_i} for every i at the same time $(1 \le u_i < v_i \le n)$. The pairs are disjoint if every u_i and v_j are different from each other.

We want to sort the sequence completely in increasing order as fast as possible. Given the initial permutation, calculate the number of ways to achieve this. Two ways are different if and only if there is a time t, such that the set of pairs used for swapping at that time are different as sets (so ordering of pairs doesn't matter). If the given permutation is already sorted, it takes no time to sort, so the number of ways to sort it is 1.

To make the problem more interesting, we have k holes inside the permutation. So exactly k numbers of $a_1, a_2, ..., a_n$ are not yet determined. For every possibility of filling the holes, calculate the number of ways, and print the total sum of these values modulo 1000000007 $(10^9 + 7)$.

Input

The first line contains two integers n $(1 \le n \le 10^5)$ and k $(0 \le k \le 12)$. The second line contains the permutation sequence $a_1, ..., a_n$ $(0 \le a_i \le n)$. If a number is not yet determined, it is denoted as 0. There are exactly k zeroes. All the numbers a_i that aren't equal to zero are distinct.

Output

Print the total sum of the number of ways modulo 1000000007 $(10^9 + 7)$.

Examples

input

```
5 0
1 5 2 4 3
```

output

```
6
```

input

```
5 2
1 0 2 4 0
```

output

```
7
```