Task 1: Cryptography (`crypto`)

Charles the Cryptographer has been researching novel methods of generating random numbers. In particular, by combining multiple sources of random numbers, he hopes to create a *cryptographically secure pseudorandom number generator (CSPRNG)*.

One algorithm that he has recently invented is as follows:

1. Randomly generate a sequence S of N **distinct** positive integers $S_1, \ldots, S_N$
2. Randomly shuffle S to obtain a permutation[1] P of N elements $P_1, \ldots, P_N$
3. Find the lexicographical order of P
4. As the answer can be very large, output the value modulo[2] 1 000 000 007

The lexicographical order of P is defined as the number of permutations of S that are lexicographically smaller than[3] or equal to P.

Unfortunately, Charles is a Cryptographer and not a Coder. Given the resultant permutation P, help Charles to find its lexicographical order, modulo 1 000 000 007.

Input

Your program must read from standard input.

The first line contains a single integer N.

The second line contains N space-separated integers, $P_1, \ldots, P_N$.

Output

Your program must print to standard output.

The output should contain a single integer on a single line, the lexicographical order of P, modulo 1 000 000 007.

Implementation Note

As the input lengths for subtasks 3, 4, 7, and 8 may be very large, you are recommended to use C++ with fast input routines to solve this problem. The scientific committee does not have a

[1] A permutation P of a sequence S is a rearrangement of the elements of S

[2] The remainder when the value is divided by 1 000 000 007

[3] A permutation $P_1, \ldots, P_N$ is considered lexicographically smaller than another permutation $P_1', \ldots, P_N'$ if there exists $1 \leq k \leq N$ such that $P_k < P_k'$ and $P_i = P_i'$ for $i = 1, \ldots, k-1$.

solution written in Python that can fully solve this problem.

C++ and Java source files containing fast input/output templates have been provided in the attachment. You are strongly recommended to use these templates.

If you are implementing your solution in Java, please name your file `Crypto.java` and place your main function inside `class Crypto`.

Subtasks

The maximum execution time on each instance is 1.0s, and the maximum memory usage on each instance is 1GiB. For all testcases, the input will satisfy the following bounds:

- $1 \leq N \leq 3 \times 10^5$
- $1 \leq P_i \leq 10^9$
- $P_i \neq P_j$ for $i \neq j$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	5	$N = 2$
2	9	$1 \leq N \leq 8$
3	10	P is either strictly increasing or decreasing.
4	11	$P = [k, 1, \ldots, k-1, k+1, \ldots, N]$ where $1 \leq k \leq N$
5	21	$1 \leq N \leq 3 \times 10^3, 1 \leq P_i \leq N$
6	13	$1 \leq N \leq 3 \times 10^3$
7	19	$1 \leq P_i \leq N$
8	12	-

Sample Testcase 1

This testcase is valid for subtasks 2, 6 and 8 only.

Input	Output
3 42 100 1	4

Sample Testcase 1 Explanation

We have the following 6 permutations in lexicographical order:

1. $[1, 42, 100]$
2. $[1, 100, 42]$
3. $[42, 1, 100]$
4. $[42, 100, 1]$
5. $[100, 1, 42]$
6. $[100, 42, 1]$

Hence, the lexicographical order of [42, 100, 1] is 4.

Sample Testcase 2

This testcase is valid for subtasks 2, 5, 6, 7 and 8 only.

Input	Output
5 1 5 2 4 3	20

Sample Testcase 3

This testcase is valid for all subtasks.

Input	Output
2 2 1	2