Problem 3577: Count the Number of Computer Unlocking Permutations

Problem Information

Difficulty: Medium
Acceptance Rate: 39.37%
Paid Only: No
Tags: Array, Math, Brainteaser, Combinatorics

Problem Description

You are given an array `complexity` of length `n`.

There are `n` **locked** computers in a room with labels from 0 to `n - 1`, each with its own **unique** password. The password of the computer `i` has a complexity `complexity[i]`.

The password for the computer labeled 0 is **already** decrypted and serves as the root. All other computers must be unlocked using it or another previously unlocked computer, following this information:

* You can decrypt the password for the computer `i` using the password for computer `j`, where `j` is **any** integer less than `i` with a lower complexity. (i.e. `j < i` and `complexity[j] < complexity[i]`) * To decrypt the password for computer `i`, you must have already unlocked a computer `j` such that `j < i` and `complexity[j] < complexity[i]`.

Find the number of permutations of `[0, 1, 2, ..., (n - 1)]` that represent a valid order in which the computers can be unlocked, starting from computer 0 as the only initially unlocked one.

Since the answer may be large, return it **modulo** 109 \+ 7.

Note that the password for the computer **with label** 0 is decrypted, and _not_ the computer with the first position in the permutation.

Example 1:

Input: complexity = [1,2,3]

Output: 2

Explanation:

The valid permutations are:

* [0, 1, 2] * Unlock computer 0 first with root password. * Unlock computer 1 with password of computer 0 since `complexity[0] < complexity[1]`. * Unlock computer 2 with password of computer 1 since `complexity[1] < complexity[2]`. * [0, 2, 1] * Unlock computer 0 first with root password. * Unlock computer 2 with password of computer 0 since `complexity[0] < complexity[2]`. * Unlock computer 1 with password of computer 0 since `complexity[0] < complexity[1]`.

Example 2:

Input: complexity = [3,3,3,4,4,4]

Output: 0

Explanation:

There are no possible permutations which can unlock all computers.

Constraints:

* `2 <= complexity.length <= 105` * `1 <= complexity[i] <= 109`

Code Snippets

C++:

```
class Solution {
public:
int countPermutations(vector<int>& complexity) {

}
};
```

Java:

```
class Solution {
public int countPermutations(int[] complexity) {

}
}
```

Python3:

```
class Solution:
def countPermutations(self, complexity: List[int]) -> int:
```