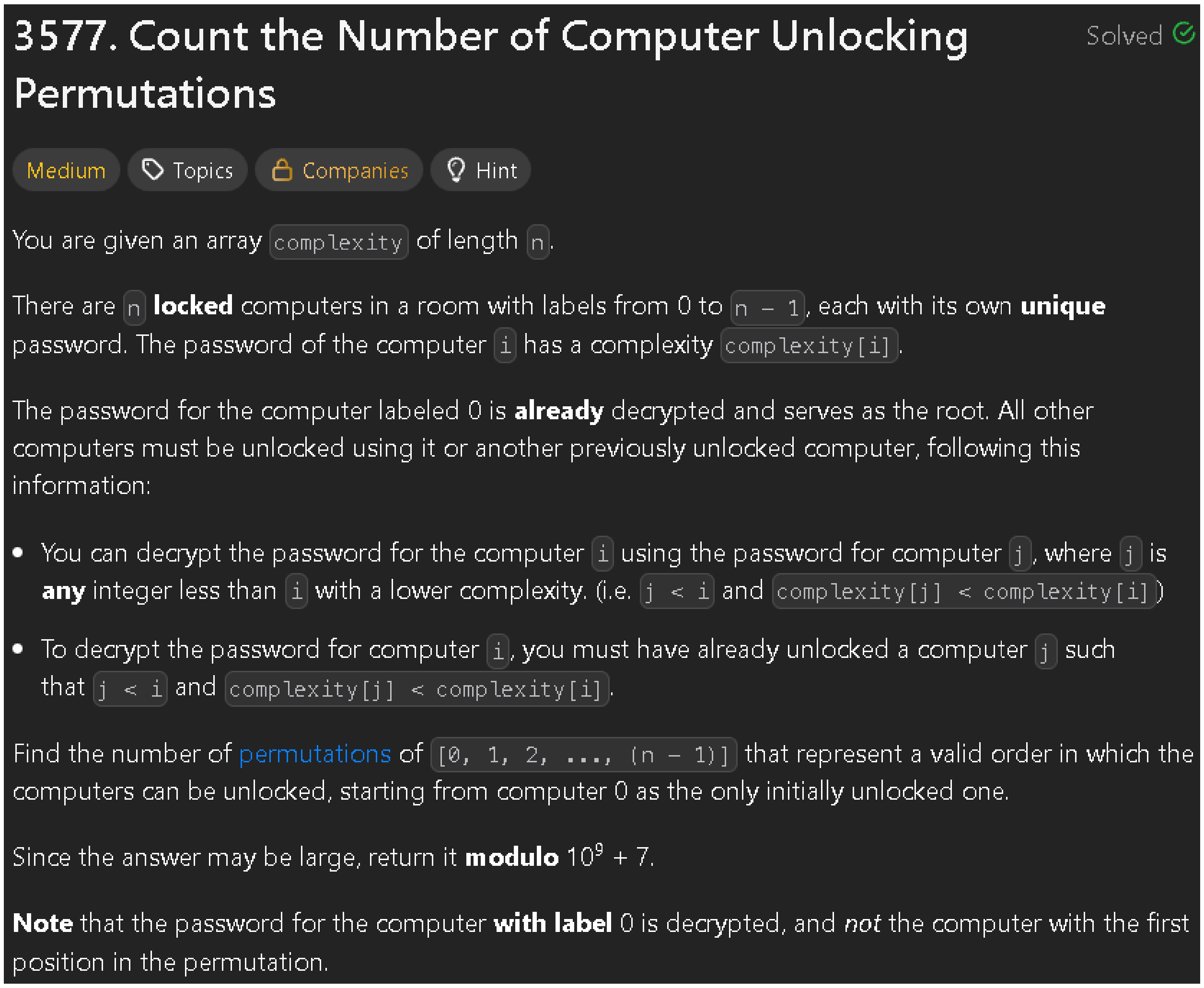


3577. Count the Number of Computer Unlocking Permutations

Solved ✓

Medium · Topics · Companies · Hint

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]`.

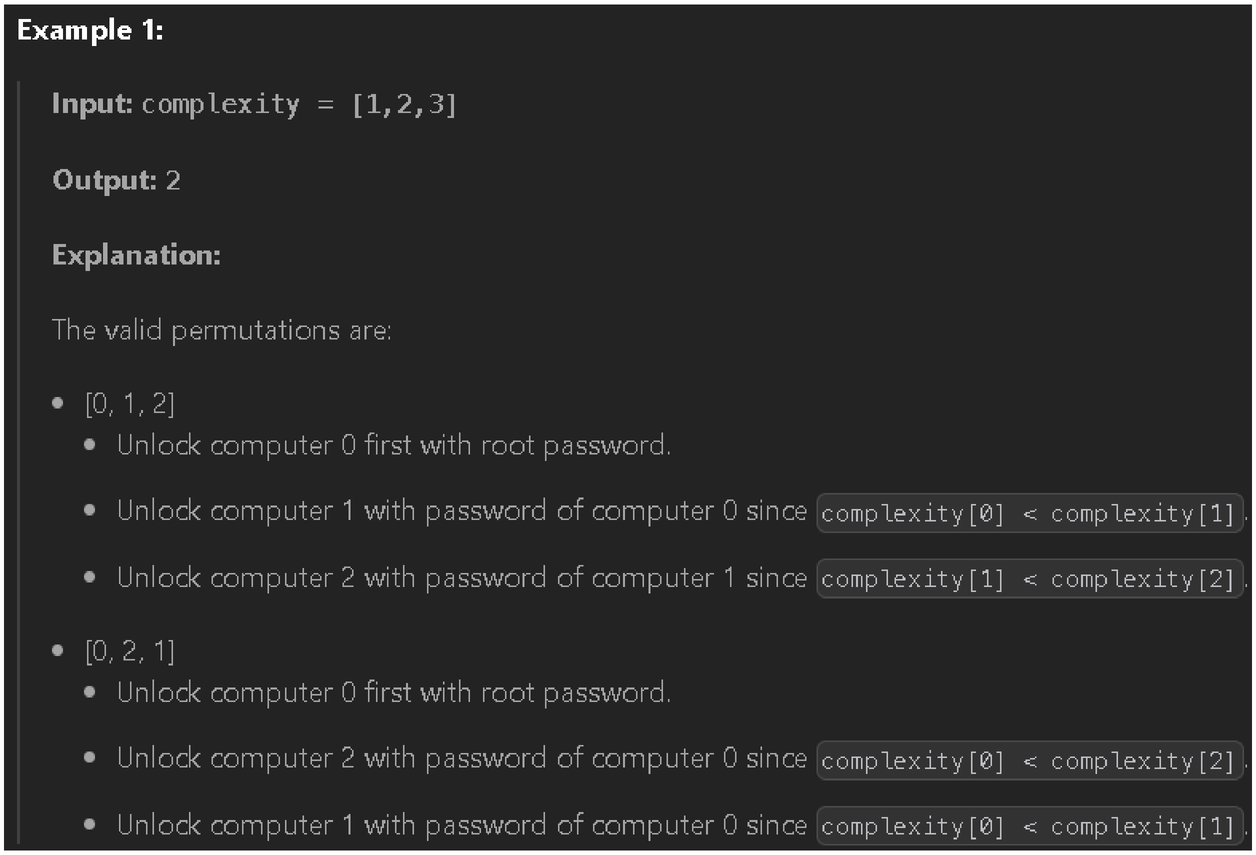
The password for the computer labeled 0 is **already** decrypted and serves as the root. All other computers must be unlocked using it or another previously unlocked computer, following this information:

- 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.

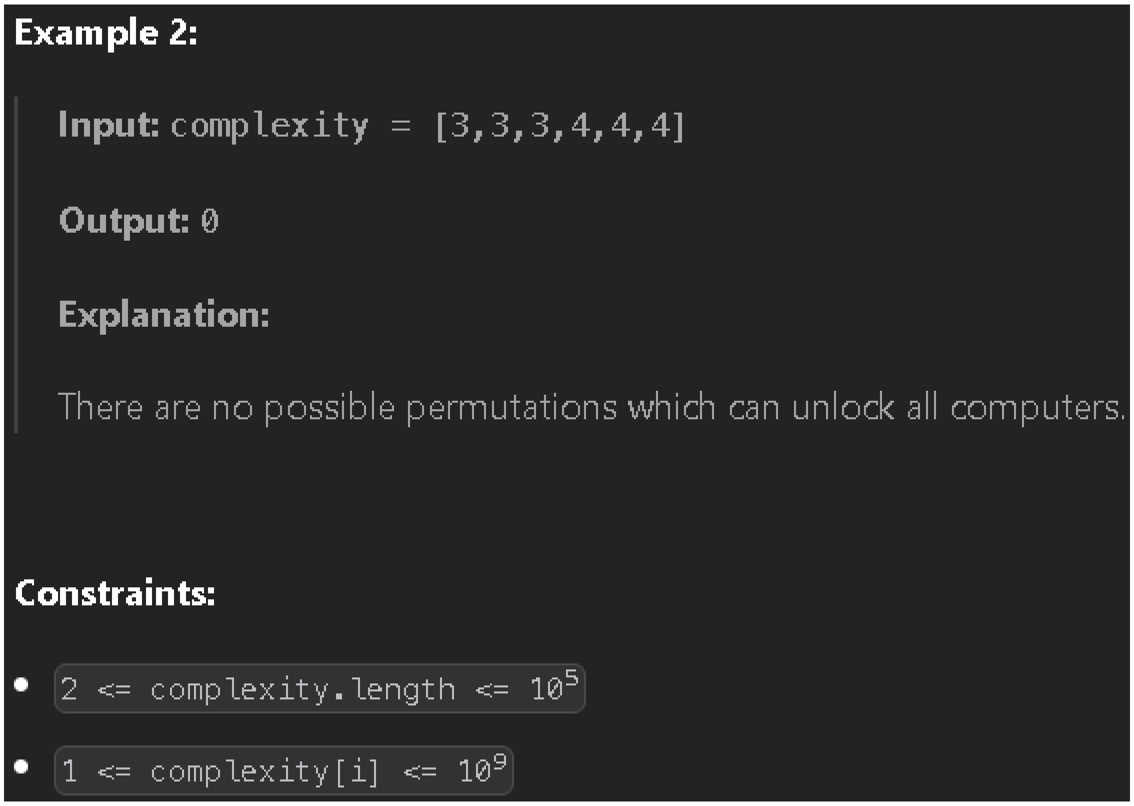
Since the answer may be large, return it **modulo** $10^9 + 7$.

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 <=` 10^5
- `1 <= complexity[i] <=` 10^9

Python:

```
class Solution(object):
    MOD = 10**9 + 7

    def countPermutations(self, comp):
        """
        :type complexity: List[int]
        :rtype: int
        """

        n = len(comp)
        ans = 1

        for i in range(1, n):
            if comp[i] <= comp[0]:
                return 0
            ans *= i
            ans %= self.MOD

        return ans
```

JavaScript:

```
countPermutations = function (complexity) {
    const n = complexity.length;

    for (let i = 1; i < n; i++) {
        if (complexity[i] <= complexity[0]) return 0;
    }

    let ans = 1;
    const MOD = 10 ** 9 + 7;

    for (let i = n - 1; i >= 1; i--) {
        ans = (ans * i) % MOD;
    }

    return ans;
}
```

Java:

```
class Solution {
    static final int MOD = 1_000_000_007;

    public int countPermutations(int[] comp) {
        int n = comp.length;
        int first = comp[0];

        // Check that first is the unique minimum
        for (int i = 1; i < n; i++) {
            if (comp[i] <= first) return 0;
        }

        // Compute factorial (n-1)!
        long fact = 1;
        for (int i = 2; i < n; i++) {
            fact = (fact * i) % MOD;
        }

        return (int) fact;
    }
}
```