Problem 1969: Minimum Non-Zero Product of the Array Elements

Problem Information

Difficulty: Medium
Acceptance Rate: 36.83%
Paid Only: No
Tags: Math, Greedy, Recursion

Problem Description

You are given a positive integer `p`. Consider an array `nums` (**1-indexed**) that consists of the integers in the **inclusive** range `[1, 2p - 1]` in their binary representations. You are allowed to do the following operation **any** number of times:

* Choose two elements `x` and `y` from `nums`. * Choose a bit in `x` and swap it with its corresponding bit in `y`. Corresponding bit refers to the bit that is in the **same position** in the other integer.

For example, if `x = 11 _0_ 1` and `y = 00 _1_ 1`, after swapping the `2nd` bit from the right, we have `x = 11 _1_ 1` and `y = 00 _0_ 1`.

Find the **minimum non-zero** product of `nums` after performing the above operation **any** number of times. Return _this product_ _**modulo** _`109 + 7`.

Note: The answer should be the minimum product **before** the modulo operation is done.

Example 1:

Input: p = 1 **Output:** 1 **Explanation:** nums = [1]. There is only one element, so the product equals that element.

Example 2:

Input: p = 2 **Output:** 6 **Explanation:** nums = [01, 10, 11]. Any swap would either make the product 0 or stay the same. Thus, the array product of 1 * 2 * 3 = 6 is already minimized.

Example 3:

Input: p = 3 **Output:** 1512 **Explanation:** nums = [001, 010, 011, 100, 101, 110, 111] - In the first operation we can swap the leftmost bit of the second and fifth elements. - The resulting array is [001, _1_ 10, 011, 100, _0_ 01, 110, 111]. - In the second operation we can swap the middle bit of the third and fourth elements. - The resulting array is [001, 110, 0 _0_ 1, 1 _1_ 0, 001, 110, 111]. The array product is 1 * 6 * 1 * 6 * 1 * 6 * 7 = 1512, which is the minimum possible product.

Constraints:

* `1 <= p <= 60`

Code Snippets

C++:

```
class Solution {
public:
int minNonZeroProduct(int p) {

}
};
```

Java:

```
class Solution {
public int minNonZeroProduct(int p) {

}
}
```

Python3:

```
class Solution:
def minNonZeroProduct(self, p: int) -> int:
```