Problem 2992: Number of Self-Divisible Permutations

Problem Information

Difficulty: Medium
Acceptance Rate: 72.02%
Paid Only: Yes
Tags: Array, Math, Dynamic Programming, Backtracking, Bit Manipulation, Number Theory, Bitmask

Problem Description

Given an integer `n`, return _the number of**permutations** of the **1-indexed** array_ `nums = [1, 2, ..., n]`_, such that it 's **self- divisible**_.

A **1-indexed** array `a` of length `n` is **self-divisible** if for every `1 <= i <= n`, `gcd(a[i], i) == 1`.

A **permutation** of an array is a rearrangement of the elements of that array, for example here are all of the permutations of the array `[1, 2, 3]`:

* `[1, 2, 3]` * `[1, 3, 2]` * `[2, 1, 3]` * `[2, 3, 1]` * `[3, 1, 2]` * `[3, 2, 1]`

Example 1:

Input: n = 1 **Output:** 1 **Explanation:** The array [1] has only 1 permutation which is self-divisible.

Example 2:

Input: n = 2 **Output:** 1 **Explanation:** The array [1,2] has 2 permutations and only one of them is self-divisible: nums = [1,2]: This is not self-divisible since gcd(nums[2], 2) != 1. nums = [2,1]: This is self-divisible since gcd(nums[1], 1) == 1 and gcd(nums[2], 2) == 1.

Example 3:

Input: n = 3 **Output:** 3 **Explanation:** The array [1,2,3] has 3 self-divisble permutations: [1,3,2], [3,1,2], [2,3,1]. It can be shown that the other 3 permutations are not self-divisible. Hence the answer is 3.

Constraints:

* `1 <= n <= 12`

Code Snippets

C++:

```
class Solution {
public:
int selfDivisiblePermutationCount(int n) {

}
};
```

Java:

```
class Solution {
public int selfDivisiblePermutationCount(int n) {

}
}
```

Python3:

```
class Solution:
def selfDivisiblePermutationCount(self, n: int) -> int:
```