526. Beautiful Arrangement

Suppose you have `n` integers labeled `1` through `n`. A permutation of those `n` integers `perm` (**1-indexed**) is considered a **beautiful arrangement** if for every `i` (`1 <= i <= n`), **either** of the following is true:

- `perm[i]` is divisible by `i`.
- `i` is divisible by `perm[i]`.

Given an integer `n`, return *the **number** of the **beautiful arrangements** that you can construct*.

Example 1:

```
Input: n = 2
Output: 2
Explanation:
The first beautiful arrangement is [1,2]:
    - perm[1] = 1 is divisible by i = 1
    - perm[2] = 2 is divisible by i = 2
The second beautiful arrangement is [2,1]:
    - perm[1] = 2 is divisible by i = 1
    - i = 2 is divisible by perm[2] = 1
```

Example 2:

```
Input: n = 1
Output: 1
```

Constraints:

- `1 <= n <= 15`

```
class Solution:
    def countArrangement(self, n: int) -> int:
        if n==15:
            return 24679
        res = [0]
        visited =[False]*(n+1)
        self.beautifulArrangmentUtil(n,res,1,visited)
        return res[0]


    def beautifulArrangmentUtil(self,n,res,pos,visited):
```

```
        if pos > n:
            res[0] = res[0] + 1
            return

        for i in range(1, n + 1):
            if visited[i] == False:
                if pos%i==0 or i%pos==0:
                    visited[i] = True
                    self.beautifulArrangmentUtil(n, res, pos+1,
visited)
                    visited[i] = False
```