# PSH: Counting Substitution Cipher Keys

In a substitution cipher, encryption occurs by transforming each symbol (or block of symbols) in an alphabet with another symbol (or block of symbols). For example, the following table below shows a key for a substitution cipher in which the letters are shifted by 3 positions:

```
Plaintext   a b c d e f g h i j k l m n o p q r s t u v w x y z
Ciphertext  d e f g h i j k l m n o p q r s t u v w x y z a b c
```

In this key, every letter "a" would be replaced with a "d", "b" would be replaced, by "e", etc.

Substitution ciphers are bad for security today, because they can be easily cracked. Some keys are worse than bad, though, because some of the symbols are mapped to themselves. For example, consider the key below:

```
Plaintext   a b c d e f g h i j k l m n o p q r s t u v w x y z
Ciphertext  d b f g h i j k l m n o p q r s t u v w x y z a c e
```

With this key the letter "b" would not be changed during the encryption.

In this challenge, you will count the number of possible cipher keys that do not map any letters to themselves.

Notes:

- You should use recursion to solve this problem.
- See below for some hints.

**Input Format**

Input consists of an integer *n* which gives the number of symbols in the alphabet.

**Constraints**

1 ≤ $n$ ≤ 20,000

**Output Format**

Output a number mod 1,000,000,007, which represents the number of keys that do not map any symbols to themselves.

**Hints**

- Consider solving the problem recursively.
- Consider thinking about the problem a bit differently: Each symbol has a unique taboo symbol, to which it cannot be mapped. Initially each symbol's taboo would be the symbol itself.
- Suppose that the first symbol is 'A', and 'A' is mapped to symbol $x$. There are two possibilities for symbol $x$: it is mapped to 'A' or it is mapped to a letter other than 'A'. Do either or both of these become instances of a subproblem that is similar to the original problem to be solved?
- If you are receiving a time-limit exceeded error, are you doing redundant computation? In other words, are you calling your recursive function with the same parameters multiple times? Is there a way to speed up the algorithm for the second and subsequent calls?

**Sample Input 0**

```
2
```

**Sample Output 0**

```
1
```

**Explanation 0**

When there are two symbols, there is only one key that does not map any symbols to themselves. For example, if the symbols were chosen from the set {a,b}, the key would be:

```
Plaintext   a b
Ciphertext  b a
```

**Sample Input 1**

```
3
```

**Sample Output 1**

```
2
```

**Explanation 1**

If the symbols were chosen from the set {a,b,c}, the following keys are valid:

Key 1:

```
Plaintext   a b c
Ciphertext  c a b
```

Key 2:

```
Plaintext   a b c
Ciphertext  b c a
```