Problem 634: Find the Derangement of An Array

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

In combinatorial mathematics, a

derangement

is a permutation of the elements of a set, such that no element appears in its original position.

You are given an integer

n

. There is originally an array consisting of

n

integers from

1

to

n

in ascending order, return

the number of

derangements

it can generate

. Since the answer may be huge, return it

modulo

10

9

+ 7

.

Example 1:

Input:

n = 3

Output:

2

Explanation:

The original array is [1,2,3]. The two derangements are [2,3,1] and [3,1,2].

Example 2:

Input:

n = 2

Output:

1

Constraints:

1 <= n <= 10

6

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

```
class Solution(object):
def findDerangement(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {number}
*/
var findDerangement = function(n) {

};
```

TypeScript:

```
function findDerangement(n: number): number {

};
```

C#:

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

}
}
```

C:

```
int findDerangement(int n) {

}
```

Go:

```
func findDerangement(n int) int {

}
```

Kotlin:

```
class Solution {
fun findDerangement(n: Int): Int {

}
}
```

Swift:

```
class Solution {
func findDerangement(_ n: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn find_derangement(n: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {Integer}
def find_derangement(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function findDerangement($n) {

}
}
```

Dart:

```
class Solution {
int findDerangement(int n) {

}
}
```

Scala:

```
object Solution {
def findDerangement(n: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_derangement(n :: integer) :: integer
def find_derangement(n) do

end
end
```

Erlang:

```
-spec find_derangement(N :: integer()) -> integer().
find_derangement(N) ->
.
```

Racket:

```
(define/contract (find-derangement n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Derangement of An Array
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

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

}
};
```

Java Solution:

```
/**
* Problem: Find the Derangement of An Array
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int findDerangement(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Derangement of An Array
Difficulty: Medium
Tags: array, dp, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def findDerangement(self, n: int) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def findDerangement(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Derangement of An Array
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number} n
* @return {number}
*/
var findDerangement = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Derangement of An Array
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

function findDerangement(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Derangement of An Array
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int FindDerangement(int n) {

}
}
```

C Solution:

```
/*
* Problem: Find the Derangement of An Array
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int findDerangement(int n) {

}
```

Go Solution:

```
// Problem: Find the Derangement of An Array
// Difficulty: Medium
// Tags: array, dp, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func findDerangement(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun findDerangement(n: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func findDerangement(_ n: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Derangement of An Array
// Difficulty: Medium
// Tags: array, dp, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn find_derangement(n: i32) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Integer}
def find_derangement(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function findDerangement($n) {

}
}
```

Dart Solution:

```
class Solution {
int findDerangement(int n) {

}
}
```

Scala Solution:

```
object Solution {
def findDerangement(n: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_derangement(n :: integer) :: integer
def find_derangement(n) do

end
end
```

Erlang Solution:

```
-spec find_derangement(N :: integer()) -> integer().
find_derangement(N) ->
.
```

Racket Solution:

```
(define/contract (find-derangement n)
(-> exact-integer? exact-integer?)
)
```