Problem 1997: First Day Where You Have Been in All the Rooms

Problem Information

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

Problem Description

There are

n

rooms you need to visit, labeled from

0

to

n - 1

. Each day is labeled, starting from

0

. You will go in and visit one room a day.

Initially on day

0

, you visit room

0

. The

order

you visit the rooms for the coming days is determined by the following

rules

and a given

0-indexed

array

nextVisit

of length

n

:

Assuming that on a day, you visit room

i

,

if you have been in room

i

an

odd

number of times (

including

the current visit), on the

next

day you will visit a room with a

lower or equal room number

specified by

nextVisit[i]

where

0 <= nextVisit[i] <= i

;

if you have been in room

i

an

even

number of times (

including

the current visit), on the

next

day you will visit room

(i + 1) mod n

.

Return

the label of the

first

day where you have been in

all

the rooms

. It can be shown that such a day exists. Since the answer may be very large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

nextVisit = [0,0]

Output:

2

Explanation:

- On day 0, you visit room 0. The total times you have been in room 0 is 1, which is odd.   On the next day you will visit room nextVisit[0] = 0 - On day 1, you visit room 0, The total times you have been in room 0 is 2, which is even.   On the next day you will visit room (0 + 1) mod 2 = 1 - On day 2, you visit room 1. This is the first day where you have been in all the rooms.

Example 2:

Input:

nextVisit = [0,0,2]

Output:

6

Explanation:

Your room visiting order for each day is: [0,0,1,0,0,1,2,...]. Day 6 is the first day where you have been in all the rooms.

Example 3:

Input:

nextVisit = [0,1,2,0]

Output:

6

Explanation:

Your room visiting order for each day is: [0,0,1,1,2,2,3,...]. Day 6 is the first day where you have been in all the rooms.

Constraints:

n == nextVisit.length

2 <= n <= 10

5

0 <= nextVisit[i] <= i

Code Snippets

C++:

```
class Solution {
public:
int firstDayBeenInAllRooms(vector<int>& nextVisit) {

}
};
```

Java:

```
class Solution {
public int firstDayBeenInAllRooms(int[] nextVisit) {

}
}
```

Python3:

```
class Solution:
def firstDayBeenInAllRooms(self, nextVisit: List[int]) -> int:
```

Python:

```
class Solution(object):
def firstDayBeenInAllRooms(self, nextVisit):
"""
:type nextVisit: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} nextVisit
* @return {number}
*/
var firstDayBeenInAllRooms = function(nextVisit) {

};
```

TypeScript:

```
function firstDayBeenInAllRooms(nextVisit: number[]): number {

};
```

C#:

```
public class Solution {
public int FirstDayBeenInAllRooms(int[] nextVisit) {

}
}
```

C:

```
int firstDayBeenInAllRooms(int* nextVisit, int nextVisitSize) {

}
```

Go:

```
func firstDayBeenInAllRooms(nextVisit []int) int {

}
```

Kotlin:

```
class Solution {
fun firstDayBeenInAllRooms(nextVisit: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func firstDayBeenInAllRooms(_ nextVisit: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn first_day_been_in_all_rooms(next_visit: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} next_visit
# @return {Integer}
def first_day_been_in_all_rooms(next_visit)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nextVisit
* @return Integer
*/
function firstDayBeenInAllRooms($nextVisit) {

}
}
```

Dart:

```
class Solution {
int firstDayBeenInAllRooms(List<int> nextVisit) {

}
}
```

Scala:

```
object Solution {
def firstDayBeenInAllRooms(nextVisit: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec first_day_been_in_all_rooms(next_visit :: [integer]) :: integer
def first_day_been_in_all_rooms(next_visit) do

end
end
```

Erlang:

```
-spec first_day_been_in_all_rooms(NextVisit :: [integer()]) -> integer().
first_day_been_in_all_rooms(NextVisit) ->
.
```

Racket:

```
(define/contract (first-day-been-in-all-rooms nextVisit)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: First Day Where You Have Been in All the Rooms
* Difficulty: Medium
* Tags: array, dp
*
* 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 firstDayBeenInAllRooms(vector<int>& nextVisit) {

}
};
```

Java Solution:

```
/**
* Problem: First Day Where You Have Been in All the Rooms
* Difficulty: Medium
* Tags: array, dp
*
* 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 firstDayBeenInAllRooms(int[] nextVisit) {

}
}
```

Python3 Solution:

```
"""
Problem: First Day Where You Have Been in All the Rooms
Difficulty: Medium
Tags: array, dp

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 firstDayBeenInAllRooms(self, nextVisit: List[int]) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def firstDayBeenInAllRooms(self, nextVisit):
"""
:type nextVisit: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: First Day Where You Have Been in All the Rooms
* Difficulty: Medium
* Tags: array, dp
*
* 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[]} nextVisit
* @return {number}
*/
var firstDayBeenInAllRooms = function(nextVisit) {

};
```

TypeScript Solution:

```
/**
* Problem: First Day Where You Have Been in All the Rooms
* Difficulty: Medium
* Tags: array, dp
*
* 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 firstDayBeenInAllRooms(nextVisit: number[]): number {

};
```

C# Solution:

```
/*
* Problem: First Day Where You Have Been in All the Rooms
* Difficulty: Medium
* Tags: array, dp
*
* 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 FirstDayBeenInAllRooms(int[] nextVisit) {

}
}
```

C Solution:

```
/*
* Problem: First Day Where You Have Been in All the Rooms
* Difficulty: Medium
* Tags: array, dp
*
* 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 firstDayBeenInAllRooms(int* nextVisit, int nextVisitSize) {

}
```

Go Solution:

```
// Problem: First Day Where You Have Been in All the Rooms
// Difficulty: Medium
// Tags: array, dp
//
// 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 firstDayBeenInAllRooms(nextVisit []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun firstDayBeenInAllRooms(nextVisit: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func firstDayBeenInAllRooms(_ nextVisit: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: First Day Where You Have Been in All the Rooms
// Difficulty: Medium
// Tags: array, dp
//
// 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 first_day_been_in_all_rooms(next_visit: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} next_visit
# @return {Integer}
def first_day_been_in_all_rooms(next_visit)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nextVisit
* @return Integer
*/
function firstDayBeenInAllRooms($nextVisit) {

}
}
```

Dart Solution:

```
class Solution {
int firstDayBeenInAllRooms(List<int> nextVisit) {

}
}
```

Scala Solution:

```
object Solution {
def firstDayBeenInAllRooms(nextVisit: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec first_day_been_in_all_rooms(next_visit :: [integer]) :: integer
def first_day_been_in_all_rooms(next_visit) do

end
end
```

Erlang Solution:

```
-spec first_day_been_in_all_rooms(NextVisit :: [integer()]) -> integer().
first_day_been_in_all_rooms(NextVisit) ->
.
```

Racket Solution:

```
(define/contract (first-day-been-in-all-rooms nextVisit)
(-> (listof exact-integer?) exact-integer?)
)
```