Problem 1680: Concatenation of Consecutive Binary Numbers

Problem Information

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

Problem Description

Given an integer

n

, return

the

decimal value

of the binary string formed by concatenating the binary representations of

1

to

n

in order,

modulo

10

9

+ 7

.

Example 1:

Input:

n = 1

Output:

1

Explanation:

"1" in binary corresponds to the decimal value 1.

Example 2:

Input:

n = 3

Output:

27

Explanation:

In binary, 1, 2, and 3 corresponds to "1", "10", and "11". After concatenating them, we have "11011", which corresponds to the decimal value 27.

Example 3:

Input:

n = 12

Output:

505379714

Explanation

: The concatenation results in "110111001011101111000100110101010111100". The decimal value of that is 118505380540. After modulo 10

9

+ 7, the result is 505379714.

Constraints:

1 <= n <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```


int concatenatedBinary(int n){

}
```

Go:

```
func concatenatedBinary(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
```

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

}
}
```

Scala:

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

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Concatenation of Consecutive Binary Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int concatenatedBinary(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Concatenation of Consecutive Binary Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int concatenatedBinary(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Concatenation of Consecutive Binary Numbers
Difficulty: Medium
Tags: string, math

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Concatenation of Consecutive Binary Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Concatenation of Consecutive Binary Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function concatenatedBinary(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Concatenation of Consecutive Binary Numbers
* Difficulty: Medium
* Tags: string, math
*
```

```
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

C Solution:

```
/*
* Problem: Concatenation of Consecutive Binary Numbers
* Difficulty: Medium
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/



int concatenatedBinary(int n){

}
```

Go Solution:

```
// Problem: Concatenation of Consecutive Binary Numbers
// Difficulty: Medium
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func concatenatedBinary(n int) int {
```

```
}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Concatenation of Consecutive Binary Numbers
// Difficulty: Medium
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Scala Solution:

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

}
}
```