Problem 1611: Minimum One Bit Operations to Make Integers Zero

Problem Information

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

Problem Description

Given an integer

n

, you must transform it into

0

using the following operations any number of times:

Change the rightmost (

0

th

) bit in the binary representation of

n

.

Change the

i

th

bit in the binary representation of

n

if the

(i-1)

th

bit is set to

1

and the

(i-2)

th

through

0

th

bits are set to

0

.

Return

the minimum number of operations to transform n

into

0

.

Example 1:

Input:

n = 3

Output:

2

Explanation:

The binary representation of 3 is "11". "

1

1" -> "

0

1" with the 2

nd

operation since the 0

th

bit is 1. "0

1

" -> "0

0

" with the 1

st

operation.

Example 2:

Input:

n = 6

Output:

4

Explanation:

The binary representation of 6 is "110". "

1

10" -> "

0

10" with the 2

nd

operation since the 1

st

bit is 1 and 0

th

through 0

th

bits are 0. "01

0

" -> "01

1

" with the 1

st

operation. "0

1

1" -> "0

0

1" with the 2

nd

operation since the 0

th

bit is 1. "00

1

" -> "00

0

" with the 1

st

operation.

Constraints:

0 <= n <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int minimumOneBitOperations(int n) {

}
```

Go:

```
func minimumOneBitOperations(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/

function minimumOneBitOperations($n) {

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (minimum-one-bit-operations n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum One Bit Operations to Make Integers Zero
* Difficulty: Hard
* Tags: dp
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Minimum One Bit Operations to Make Integers Zero
* Difficulty: Hard
* Tags: dp
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int minimumOneBitOperations(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum One Bit Operations to Make Integers Zero
Difficulty: Hard
Tags: dp

Approach: Dynamic programming with memoization or tabulation
Time Complexity: O(n * m) where n and m are problem dimensions
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum One Bit Operations to Make Integers Zero
* Difficulty: Hard
* Tags: dp
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Minimum One Bit Operations to Make Integers Zero
* Difficulty: Hard
* Tags: dp
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

function minimumOneBitOperations(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum One Bit Operations to Make Integers Zero
* Difficulty: Hard
* Tags: dp
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

}
}
```

C Solution:

```
/*
* Problem: Minimum One Bit Operations to Make Integers Zero
* Difficulty: Hard
* Tags: dp
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

int minimumOneBitOperations(int n) {

}
```

Go Solution:

```
// Problem: Minimum One Bit Operations to Make Integers Zero
// Difficulty: Hard
// Tags: dp
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

func minimumOneBitOperations(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum One Bit Operations to Make Integers Zero
// Difficulty: Hard
// Tags: dp
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (minimum-one-bit-operations n)
(-> exact-integer? exact-integer?)
)
```