Problem 2571: Minimum Operations to Reduce an Integer to 0

Problem Information

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

Problem Description

You are given a positive integer

n

, you can do the following operation

any

number of times:

Add or subtract a

power

of

2

from

n

.

Return

the

minimum

number of operations to make

n

equal to

0

.

A number

x

is power of

2

if

x == 2

i

where

i >= 0

.

Example 1:

Input:

n = 39

Output:

3

Explanation:

We can do the following operations: - Add 2

0

= 1 to n, so now n = 40. - Subtract 2

3

= 8 from n, so now n = 32. - Subtract 2

5

= 32 from n, so now n = 0. It can be shown that 3 is the minimum number of operations we need to make n equal to 0.

Example 2:

Input:

n = 54

Output:

3

Explanation:

We can do the following operations: - Add 2

1

= 2 to n, so now n = 56. - Add 2

3

= 8 to n, so now n = 64. - Subtract 2

6

= 64 from n, so now n = 0. So the minimum number of operations is 3.

Constraints:

1 <= n <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int minOperations(int n) {

}
```

Go:

```
func minOperations(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (min-operations n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Operations to Reduce an Integer to 0
* Difficulty: Medium
* Tags: dp, greedy
*
* 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 minOperations(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Operations to Reduce an Integer to 0
* Difficulty: Medium
* Tags: dp, greedy
*
* 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 minOperations(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Operations to Reduce an Integer to 0
Difficulty: Medium
Tags: dp, greedy

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 minOperations(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Operations to Reduce an Integer to 0
* Difficulty: Medium
* Tags: dp, greedy
*
* 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 minOperations = function(n) {
};
```

TypeScript Solution:

```
/**
* Problem: Minimum Operations to Reduce an Integer to 0
* Difficulty: Medium
* Tags: dp, greedy
*
* 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 minOperations(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Operations to Reduce an Integer to 0
* Difficulty: Medium
* Tags: dp, greedy
*
* 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 MinOperations(int n) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Operations to Reduce an Integer to 0
* Difficulty: Medium
* Tags: dp, greedy
*
* 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 minOperations(int n) {

}
```

Go Solution:

```
// Problem: Minimum Operations to Reduce an Integer to 0
// Difficulty: Medium
// Tags: dp, greedy
//
// 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 minOperations(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Operations to Reduce an Integer to 0
// Difficulty: Medium
// Tags: dp, greedy
//
// 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 min_operations(n: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (min-operations n)
(-> exact-integer? exact-integer?)
)
```