Problem 1553: Minimum Number of Days to Eat N Oranges

Problem Information

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

Problem Description

There are

n

oranges in the kitchen and you decided to eat some of these oranges every day as follows:

Eat one orange.

If the number of remaining oranges

n

is divisible by

2

then you can eat

n / 2

oranges.

If the number of remaining oranges

n

is divisible by

3

then you can eat

2 * (n / 3)

oranges.

You can only choose one of the actions per day.

Given the integer

n

, return

the minimum number of days to eat

n

oranges

.

Example 1:

Input:

n = 10

Output:

4

Explanation:

You have 10 oranges. Day 1: Eat 1 orange, 10 - 1 = 9. Day 2: Eat 6 oranges, 9 - 2*(9/3) = 9 - 6 = 3. (Since 9 is divisible by 3) Day 3: Eat 2 oranges, 3 - 2*(3/3) = 3 - 2 = 1. Day 4: Eat the last orange 1 - 1 = 0. You need at least 4 days to eat the 10 oranges.

Example 2:

Input:

n = 6

Output:

3

Explanation:

You have 6 oranges. Day 1: Eat 3 oranges, 6 - 6/2 = 6 - 3 = 3. (Since 6 is divisible by 2). Day 2: Eat 2 oranges, 3 - 2*(3/3) = 3 - 2 = 1. (Since 3 is divisible by 3) Day 3: Eat the last orange 1 - 1 = 0. You need at least 3 days to eat the 6 oranges.

Constraints:

1 <= n <= 2 * 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int minDays(int n) {

}
```

Go:

```
func minDays(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Days to Eat N Oranges
* 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 minDays(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Days to Eat N Oranges
* 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 minDays(int n) {
}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Days to Eat N Oranges
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 minDays(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Number of Days to Eat N Oranges
* 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 minDays = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Days to Eat N Oranges
* 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 minDays(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Days to Eat N Oranges
* 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 MinDays(int n) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Days to Eat N Oranges
* 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 minDays(int n) {

}
```

Go Solution:

```
// Problem: Minimum Number of Days to Eat N Oranges
// 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 minDays(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```rust
// Problem: Minimum Number of Days to Eat N Oranges
// 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 min_days(n: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```php
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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