Problem 1137: N-th Tribonacci Number

Problem Information

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

Problem Description

The Tribonacci sequence T

n

is defined as follows:

T

0

= 0, T

1

= 1, T

2

= 1, and T

n+3

= T

n

+ T

n+1

+ T

n+2

for n >= 0.

Given

n

, return the value of T

n

.

Example 1:

Input:

n = 4

Output:

4

Explanation:

T_3 = 0 + 1 + 1 = 2 T_4 = 1 + 1 + 2 = 4

Example 2:

Input:

n = 25

Output:

1389537

Constraints:

0 <= n <= 37

The answer is guaranteed to fit within a 32-bit integer, ie.

answer <= 2^31 - 1

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int tribonacci(int n) {

}
```

Go:

```
func tribonacci(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
```

```
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: N-th Tribonacci Number
* Difficulty: Easy
* Tags: dp, math
*
* 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 tribonacci(int n) {

}
};
```

Java Solution:

```
/**
* Problem: N-th Tribonacci Number
* Difficulty: Easy
* Tags: dp, math
*
* 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 tribonacci(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: N-th Tribonacci Number
Difficulty: Easy
Tags: dp, math
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: N-th Tribonacci Number
* Difficulty: Easy
* Tags: dp, math
*
* 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 tribonacci = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: N-th Tribonacci Number
* Difficulty: Easy
* Tags: dp, math
*
* 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 tribonacci(n: number): number {

};
```

C# Solution:

```
/*
* Problem: N-th Tribonacci Number
* Difficulty: Easy
* Tags: dp, math
*
* 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 Tribonacci(int n) {

}
}
```

C Solution:

```
/*
* Problem: N-th Tribonacci Number
* Difficulty: Easy
* Tags: dp, math
*
* 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 tribonacci(int n) {

}
```

Go Solution:

```
// Problem: N-th Tribonacci Number
// Difficulty: Easy
// Tags: dp, math
//
// 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 tribonacci(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: N-th Tribonacci Number
// Difficulty: Easy
// Tags: dp, math
```

```
//
// 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 tribonacci(n: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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