Problem 258: Add Digits

Problem Information

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

Problem Description

Given an integer

num

, repeatedly add all its digits until the result has only one digit, and return it.

Example 1:

Input:

num = 38

Output:

2

Explanation:

The process is 38 --> 3 + 8 --> 11 11 --> 1 + 1 --> 2 Since 2 has only one digit, return it.

Example 2:

Input:

num = 0

Output:

0

Constraints:

0 <= num <= 2

31

- 1

Follow up:

Could you do it without any loop/recursion in

O(1)

runtime?

Code Snippets

C++:

```
class Solution {
public:
int addDigits(int num) {

}
};
```

Java:

```
class Solution {
public int addDigits(int num) {

}
}
```

Python3:

```
class Solution:
def addDigits(self, num: int) -> int:
```

Python:

```
class Solution(object):
def addDigits(self, num):
"""
:type num: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} num
* @return {number}
*/
var addDigits = function(num) {

};
```

TypeScript:

```
function addDigits(num: number): number {

};
```

C#:

```
public class Solution {
public int AddDigits(int num) {

}
}
```

C:

```
int addDigits(int num) {

}
```

Go:

```
func addDigits(num int) int {

}
```

Kotlin:

```
class Solution {
fun addDigits(num: Int): Int {

}
}
```

Swift:

```
class Solution {
func addDigits(_ num: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn add_digits(num: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} num
# @return {Integer}
def add_digits(num)

end
```

PHP:

```
class Solution {

/**
* @param Integer $num
* @return Integer
*/
function addDigits($num) {

}
}
```

Dart:

```
class Solution {
int addDigits(int num) {

}
}
```

Scala:

```
object Solution {
def addDigits(num: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec add_digits(num :: integer) :: integer
def add_digits(num) do

end
end
```

Erlang:

```
-spec add_digits(Num :: integer()) -> integer().
add_digits(Num) ->
.
```

Racket:

```
(define/contract (add-digits num)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Add Digits
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int addDigits(int num) {

}
};
```

Java Solution:

```
/**
* Problem: Add Digits
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int addDigits(int num) {

}
}
```

Python3 Solution:

```
"""
Problem: Add Digits
Difficulty: Easy
Tags: math

Approach: Optimized algorithm based on problem constraints
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

```
class Solution(object):
def addDigits(self, num):
"""
:type num: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Add Digits
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
```

```
* @param {number} num
* @return {number}
*/
var addDigits = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Add Digits
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function addDigits(num: number): number {

};
```

C# Solution:

```
/*
* Problem: Add Digits
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int AddDigits(int num) {

}
}
```

C Solution:

```
/*
* Problem: Add Digits
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

int addDigits(int num) {

}
```

Go Solution:

```
// Problem: Add Digits
// Difficulty: Easy
// Tags: math
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func addDigits(num int) int {

}
```

Kotlin Solution:

```
class Solution {
fun addDigits(num: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func addDigits(_ num: Int) -> Int {
}
}
```

Rust Solution:

```
// Problem: Add Digits
// Difficulty: Easy
// Tags: math
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn add_digits(num: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} num
# @return {Integer}
def add_digits(num)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $num
* @return Integer
*/
function addDigits($num) {

}
}
```

Dart Solution:

```
class Solution {
int addDigits(int num) {

}
}
```

Scala Solution:

```
object Solution {
def addDigits(num: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec add_digits(num :: integer) :: integer
def add_digits(num) do

end
end
```

Erlang Solution:

```
-spec add_digits(Num :: integer()) -> integer().
add_digits(Num) ->
.
```

Racket Solution:

```
(define/contract (add-digits num)
(-> exact-integer? exact-integer?)
)
```