Problem 2698: Find the Punishment Number of an Integer

Problem Information

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

Problem Description

Given a positive integer

n

, return

the

punishment number

of

n

.

The

punishment number

of

n

is defined as the sum of the squares of all integers

i

such that:

1 <= i <= n

The decimal representation of

i * i

can be partitioned into contiguous substrings such that the sum of the integer values of these substrings equals

i

.

Example 1:

Input:

n = 10

Output:

182

Explanation:

There are exactly 3 integers i in the range [1, 10] that satisfy the conditions in the statement: - 1 since 1 * 1 = 1 - 9 since 9 * 9 = 81 and 81 can be partitioned into 8 and 1 with a sum equal to 8 + 1 == 9. - 10 since 10 * 10 = 100 and 100 can be partitioned into 10 and 0 with a sum equal to 10 + 0 == 10. Hence, the punishment number of 10 is 1 + 81 + 100 = 182

Example 2:

Input:

n = 37

Output:

1478

Explanation:

There are exactly 4 integers i in the range [1, 37] that satisfy the conditions in the statement: - 1 since 1 * 1 = 1. - 9 since 9 * 9 = 81 and 81 can be partitioned into 8 + 1. - 10 since 10 * 10 = 100 and 100 can be partitioned into 10 + 0. - 36 since 36 * 36 = 1296 and 1296 can be partitioned into 1 + 29 + 6. Hence, the punishment number of 37 is 1 + 81 + 100 + 1296 = 1478

Constraints:

1 <= n <= 1000

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int punishmentNumber(int n) {

}
```

Go:

```
func punishmentNumber(n int) int {
```

```
}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

```
function punishmentNumber($n) {

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Find the Punishment Number of an Integer
* Difficulty: Medium
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Find the Punishment Number of an Integer
* Difficulty: Medium
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int punishmentNumber(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Punishment Number of an Integer
Difficulty: Medium
Tags: string, tree, math

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Punishment Number of an Integer
* Difficulty: Medium
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

```
};
```

TypeScript Solution:

```
/**
* Problem: Find the Punishment Number of an Integer
* Difficulty: Medium
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function punishmentNumber(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Punishment Number of an Integer
* Difficulty: Medium
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
}
```

C Solution:

```
/*
* Problem: Find the Punishment Number of an Integer
* Difficulty: Medium
```

```
* Tags: string, tree, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int punishmentNumber(int n) {

}
```

Go Solution:

```
// Problem: Find the Punishment Number of an Integer
// Difficulty: Medium
// Tags: string, tree, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func punishmentNumber(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Punishment Number of an Integer
// Difficulty: Medium
// Tags: string, tree, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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