Problem 2457: Minimum Addition to Make Integer Beautiful

Problem Information

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

Problem Description

You are given two positive integers

n

and

target

.

An integer is considered

beautiful

if the sum of its digits is less than or equal to

target

.

Return the

minimum

non-negative

integer

x

such that

n + x

is beautiful

. The input will be generated such that it is always possible to make

n

beautiful.

Example 1:

Input:

n = 16, target = 6

Output:

4

Explanation:

Initially n is 16 and its digit sum is 1 + 6 = 7. After adding 4, n becomes 20 and digit sum becomes 2 + 0 = 2. It can be shown that we can not make n beautiful with adding non-negative integer less than 4.

Example 2:

Input:

n = 467, target = 6

Output:

33

Explanation:

Initially n is 467 and its digit sum is 4 + 6 + 7 = 17. After adding 33, n becomes 500 and digit sum becomes 5 + 0 + 0 = 5. It can be shown that we can not make n beautiful with adding non-negative integer less than 33.

Example 3:

Input:

n = 1, target = 1

Output:

0

Explanation:

Initially n is 1 and its digit sum is 1, which is already smaller than or equal to target.

Constraints:

1 <= n <= 10

12

1 <= target <= 150

The input will be generated such that it is always possible to make

n

beautiful.

Code Snippets

C++:

```
class Solution {
public:
long long makeIntegerBeautiful(long long n, int target) {

}
};
```

Java:

```
class Solution {
public long makeIntegerBeautiful(long n, int target) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number} n
* @param {number} target
* @return {number}
*/
var makeIntegerBeautiful = function(n, target) {

};
```

TypeScript:

```
function makeIntegerBeautiful(n: number, target: number): number {

};
```

C#:

```
public class Solution {
public long MakeIntegerBeautiful(long n, int target) {

}
}
```

C:

```
long long makeIntegerBeautiful(long long n, int target) {

}
```

Go:

```
func makeIntegerBeautiful(n int64, target int) int64 {

}
```

Kotlin:

```
class Solution {
fun makeIntegerBeautiful(n: Long, target: Int): Long {

}
}
```

Swift:

```
class Solution {
func makeIntegerBeautiful(_ n: Int, _ target: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn make_integer_beautiful(n: i64, target: i32) -> i64 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} target
# @return {Integer}
def make_integer_beautiful(n, target)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $target
* @return Integer
*/
function makeIntegerBeautiful($n, $target) {

}
}
```

Dart:

```
class Solution {
int makeIntegerBeautiful(int n, int target) {

}
}
```

Scala:

```
object Solution {
def makeIntegerBeautiful(n: Long, target: Int): Long = {

}
```

```
}
```

Elixir:

```
defmodule Solution do
@spec make_integer_beautiful(n :: integer, target :: integer) :: integer
def make_integer_beautiful(n, target) do

end
end
```

Erlang:

```
-spec make_integer_beautiful(N :: integer(), Target :: integer()) ->
integer().
make_integer_beautiful(N, Target) ->
.
```

Racket:

```
(define/contract (make-integer-beautiful n target)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Addition to Make Integer Beautiful
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
```

```
long long makeIntegerBeautiful(long long n, int target) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Addition to Make Integer Beautiful
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public long makeIntegerBeautiful(long n, int target) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Addition to Make Integer Beautiful
Difficulty: Medium
Tags: greedy, math

Approach: Greedy algorithm with local optimal choices
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Addition to Make Integer Beautiful
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} n
* @param {number} target
* @return {number}
*/
var makeIntegerBeautiful = function(n, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Addition to Make Integer Beautiful
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function makeIntegerBeautiful(n: number, target: number): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Minimum Addition to Make Integer Beautiful
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* 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 long MakeIntegerBeautiful(long n, int target) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Addition to Make Integer Beautiful
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

long long makeIntegerBeautiful(long long n, int target) {

}
```

Go Solution:

```
// Problem: Minimum Addition to Make Integer Beautiful
// Difficulty: Medium
```

```
// Tags: greedy, math
//
// Approach: Greedy algorithm with local optimal choices
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func makeIntegerBeautiful(n int64, target int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun makeIntegerBeautiful(n: Long, target: Int): Long {

}
}
```

Swift Solution:

```
class Solution {
func makeIntegerBeautiful(_ n: Int, _ target: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Addition to Make Integer Beautiful
// Difficulty: Medium
// Tags: greedy, math
//
// Approach: Greedy algorithm with local optimal choices
// 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 make_integer_beautiful(n: i64, target: i32) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} target
# @return {Integer}
def make_integer_beautiful(n, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $target
* @return Integer
*/
function makeIntegerBeautiful($n, $target) {

}
}
```

Dart Solution:

```
class Solution {
int makeIntegerBeautiful(int n, int target) {

}
}
```

Scala Solution:

```
object Solution {
def makeIntegerBeautiful(n: Long, target: Int): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec make_integer_beautiful(n :: integer, target :: integer) :: integer
def make_integer_beautiful(n, target) do

end
end
```

Erlang Solution:

```
-spec make_integer_beautiful(N :: integer(), Target :: integer()) ->
integer().
make_integer_beautiful(N, Target) ->
.
```

Racket Solution:

```
(define/contract (make-integer-beautiful n target)
(-> exact-integer? exact-integer? exact-integer?)
)
```