

# Problem 1088: Confusing Number II

## Problem Information

**Difficulty:** Hard

**Acceptance Rate:** 0.00%

**Paid Only:** No

## Problem Description

A

confusing number

is a number that when rotated

180

degrees becomes a different number with

each digit valid

We can rotate digits of a number by

180

degrees to form new digits.

When

0

,

1

,

6

,

8

, and

9

are rotated

180

degrees, they become

0

,

1

,

9

,

8

, and

6

respectively.

When

2

,

3

,

4

,

5

, and

7

are rotated

180

degrees, they become

invalid

.

Note that after rotating a number, we can ignore leading zeros.

For example, after rotating

8000

, we have

0008

which is considered as just

8

Given an integer

n

, return

the number of

confusing numbers

in the inclusive range

[1, n]

Example 1:

Input:

n = 20

Output:

6

Explanation:

The confusing numbers are [6,9,10,16,18,19]. 6 converts to 9. 9 converts to 6. 10 converts to 01 which is just 1. 16 converts to 91. 18 converts to 81. 19 converts to 61.

Example 2:

Input:

n = 100

Output:

19

Explanation:

The confusing numbers are [6,9,10,16,18,19,60,61,66,68,80,81,86,89,90,91,98,99,100].

Constraints:

1 <= n <= 10

9

## Code Snippets

C++:

```
class Solution {  
public:  
    int confusingNumberII(int n) {  
  
    }  
};
```

Java:

```
class Solution {  
public int confusingNumberII(int n) {  
  
}  
}
```

**Python3:**

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

**Python:**

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

**JavaScript:**

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

**TypeScript:**

```
function confusingNumberII(n: number): number {  
  
};
```

**C#:**

```
public class Solution {  
    public int ConfusingNumberII(int n) {  
  
    }  
}
```

**C:**

```
int confusingNumberII(int n) {  
  
}
```

**Go:**

```
func confusingNumberII(n int) int {  
}  
}
```

**Kotlin:**

```
class Solution {  
    fun confusingNumberII(n: Int): Int {  
        }  
        }  
}
```

**Swift:**

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

**Rust:**

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

**Ruby:**

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

**PHP:**

```
class Solution {  
  
    /**
```

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

}
}
```

### Dart:

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

}
}
```

### Scala:

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

}
}
```

### Elixir:

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

end
end
```

### Erlang:

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

### Racket:

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

## Solutions

### C++ Solution:

```
/*
 * Problem: Confusing Number II
 * Difficulty: Hard
 * 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 confusingNumberII(int n) {
        }
};
```

### Java Solution:

```
/**
 * Problem: Confusing Number II
 * Difficulty: Hard
 * 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 confusingNumberII(int n) {
        }
}
```

```
}
```

### Python3 Solution:

```
"""
Problem: Confusing Number II
Difficulty: Hard
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 confusingNumberII(self, n: int) -> int:
        # TODO: Implement optimized solution
        pass
```

### Python Solution:

```
class Solution(object):

    def confusingNumberII(self, n):
        """
        :type n: int
        :rtype: int
        """


```

### JavaScript Solution:

```
/**
 * Problem: Confusing Number II
 * Difficulty: Hard
 * 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} n
* @return {number}
*/
var confusingNumberII = function(n) {

};
```

### TypeScript Solution:

```
/** 
 * Problem: Confusing Number II
 * Difficulty: Hard
 * 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 confusingNumberII(n: number): number {

};
```

### C# Solution:

```
/*
 * Problem: Confusing Number II
 * Difficulty: Hard
 * 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 ConfusingNumberII(int n) {
        return 0;
    }
}
```

## C Solution:

```
/*
 * Problem: Confusing Number II
 * Difficulty: Hard
 * 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 confusingNumberII(int n) {

}
```

## Go Solution:

```
// Problem: Confusing Number II
// Difficulty: Hard
// 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 confusingNumberII(n int) int {

}
```

## Kotlin Solution:

```
class Solution {
    fun confusingNumberII(n: Int): Int {
        return 0
    }
}
```

## Swift Solution:

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

```
}
```

```
}
```

### Rust Solution:

```
// Problem: Confusing Number II
// Difficulty: Hard
// 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 confusing_number_ii(n: i32) -> i32 {
        ...
    }
}
```

### Ruby Solution:

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

end
```

### PHP Solution:

```
class Solution {

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

    }
}
```

### Dart Solution:

```
class Solution {  
    int confusingNumberII(int n) {  
  
    }  
}
```

### Scala Solution:

```
object Solution {  
    def confusingNumberII(n: Int) = {  
  
    }  
}
```

### Elixir Solution:

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

### Erlang Solution:

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

### Racket Solution:

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