# Problem 3704: Count No-Zero Pairs That Sum to N

## Problem Information

**Difficulty:** Hard
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

A

no-zero

integer is a

positive

integer that

does not contain the digit

0 in its decimal representation.

Given an integer

n

, count the number of pairs

(a, b)

where:

a

and

b

are

no-zero

integers.

a + b = n

Return an integer denoting the number of such pairs.

Example 1:

Input:

n = 2

Output:

1

Explanation:

The only pair is

(1, 1)

.

Example 2:

Input:

n = 3

Output:

2

Explanation:

The pairs are

(1, 2)

and

(2, 1)

.

Example 3:

Input:

n = 11

Output:

8

Explanation:

The pairs are

(2, 9)

,

(3, 8)

,

(4, 7)

,

(5, 6)

,

(6, 5)

,

(7, 4)

,

(8, 3)

, and

(9, 2)

. Note that

(1, 10)

and

(10, 1)

do not satisfy the conditions because 10 contains 0 in its decimal representation.

Constraints:

2 <= n <= 10

15

## Code Snippets

**C++:**

```
class Solution {
public:
long long countNoZeroPairs(long long n) {

}
};
```

**Java:**

```
class Solution {
public long countNoZeroPairs(long n) {

}
}
```

**Python3:**

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

**Python:**

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

**JavaScript:**

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

};
```

**TypeScript:**

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

};
```

**C#:**

```
public class Solution {
public long CountNoZeroPairs(long n) {

}
}
```

**C:**

```
long long countNoZeroPairs(long long n) {

}
```

**Go:**

```
func countNoZeroPairs(n int64) int64 {

}
```

**Kotlin:**

```
class Solution {
fun countNoZeroPairs(n: Long): Long {

}
}
```

**Swift:**

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

}
}
```

**Rust:**

```
impl Solution {
pub fn count_no_zero_pairs(n: i64) -> i64 {

}
}
```

**Ruby:**

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

end
```

**PHP:**

```
class Solution {

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

}
}
```

**Dart:**

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

}
}
```

**Scala:**

```
object Solution {
def countNoZeroPairs(n: Long): Long = {

}
}
```

### Elixir:

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

end
end
```

### Erlang:

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

### Racket:

```
(define/contract (count-no-zero-pairs n)
(-> exact-integer? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Count No-Zero Pairs That Sum to N
* Difficulty: Hard
* 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:
long long countNoZeroPairs(long long n) {

}
};
```

### Java Solution:

```
/**
* Problem: Count No-Zero Pairs That Sum to N
* Difficulty: Hard
* 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 long countNoZeroPairs(long n) {

}
}
```

### Python3 Solution:

```
"""
Problem: Count No-Zero Pairs That Sum to N
Difficulty: Hard
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 countNoZeroPairs(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

### Python Solution:

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

```
"""
```

**JavaScript Solution:**

```
/**
* Problem: Count No-Zero Pairs That Sum to N
* Difficulty: Hard
* 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 countNoZeroPairs = function(n) {

};
```

**TypeScript Solution:**

```
/**
* Problem: Count No-Zero Pairs That Sum to N
* Difficulty: Hard
* 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 countNoZeroPairs(n: number): number {

};
```

**C# Solution:**

```
/*
* Problem: Count No-Zero Pairs That Sum to N
* Difficulty: Hard
* 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 long CountNoZeroPairs(long n) {

}
}
```

**C Solution:**

```
/*
* Problem: Count No-Zero Pairs That Sum to N
* Difficulty: Hard
* 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
*/

long long countNoZeroPairs(long long n) {

}
```

**Go Solution:**

```
// Problem: Count No-Zero Pairs That Sum to N
// Difficulty: Hard
// 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 countNoZeroPairs(n int64) int64 {

}
```

### Kotlin Solution:

```
class Solution {
fun countNoZeroPairs(n: Long): Long {

}
}
```

### Swift Solution:

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

}
}
```

### Rust Solution:

```
// Problem: Count No-Zero Pairs That Sum to N
// Difficulty: Hard
// 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 count_no_zero_pairs(n: i64) -> i64 {

}
}
```

### Ruby Solution:

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

```
end
```

### PHP Solution:

```
class Solution {

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

}
}
```

### Dart Solution:

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

}
}
```

### Scala Solution:

```
object Solution {
def countNoZeroPairs(n: Long): Long = {

}
}
```

### Elixir Solution:

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

end
end
```

## Erlang Solution:

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

## Racket Solution:

```
(define/contract (count-no-zero-pairs n)
(-> exact-integer? exact-integer?)
)
```