

# Problem 233: Number of Digit One

## Problem Information

Difficulty: Hard

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

Given an integer

n

, count

the total number of digit

1

appearing in all non-negative integers less than or equal to

n

.

Example 1:

Input:

$n = 13$

Output:

6

**Example 2:**

**Input:**

n = 0

**Output:**

0

**Constraints:**

0 <= n <= 10

9

## Code Snippets

**C++:**

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

**Java:**

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

**Python3:**

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

**Python:**

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

**JavaScript:**

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

};
```

**TypeScript:**

```
function countDigitOne(n: number): number {
}
```

**C#:**

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

**C:**

```
int countDigitOne(int n) {
}
```

**Go:**

```
func countDigitOne(n int) int {
```

```
}
```

### Kotlin:

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

### Swift:

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

### Rust:

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

### Ruby:

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

### PHP:

```
class Solution {  
  
    /**  
     * @param Integer $n  
     * @return Integer  
     */
```

```
function countDigitOne($n) {  
}  
}  
}
```

### Dart:

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

### Scala:

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

### Elixir:

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

### Erlang:

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

### Racket:

```
(define/contract (count-digit-one n)  
  (-> exact-integer? exact-integer?)  
)
```

## Solutions

### C++ Solution:

```
/*
 * Problem: Number of Digit One
 * 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:
    int countDigitOne(int n) {

    }
};
```

### Java Solution:

```
/**
 * Problem: Number of Digit One
 * 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 int countDigitOne(int n) {

    }
}
```

### Python3 Solution:

```

"""
Problem: Number of Digit One
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 countDigitOne(self, n: int) -> int:
    # TODO: Implement optimized solution
    pass

```

## Python Solution:

```

class Solution(object):

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

```

## JavaScript Solution:

```

/**
 * Problem: Number of Digit One
 * 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 countDigitOne = function(n) {

```

```
};
```

### TypeScript Solution:

```
/**  
 * Problem: Number of Digit One  
 * 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 countDigitOne(n: number): number {  
  
};
```

### C# Solution:

```
/*  
 * Problem: Number of Digit One  
 * 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 int CountDigitOne(int n) {  
  
    }  
}
```

### C Solution:

```
/*  
 * Problem: Number of Digit One  
 * 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
*/
int countDigitOne(int n) {
}

```

### Go Solution:

```

// Problem: Number of Digit One
// 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 countDigitOne(n int) int {
}

```

### Kotlin Solution:

```

class Solution {
    fun countDigitOne(n: Int): Int {
    }
}

```

### Swift Solution:

```

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

```

### Rust Solution:

```
// Problem: Number of Digit One
// 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_digit_one(n: i32) -> i32 {
        }

    }
}
```

### Ruby Solution:

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

end
```

### PHP Solution:

```
class Solution {

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

    }
}
```

### Dart Solution:

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

```
}
```

```
}
```

### Scala Solution:

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

### Elixir Solution:

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

### Erlang Solution:

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

### Racket Solution:

```
(define/contract (count-digit-one n)  
  (-> exact-integer? exact-integer?)  
  )
```