Problem 1317: Convert Integer to the Sum of Two No-Zero Integers

Problem Information

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

Problem Description

No-Zero integer

is a positive integer that

does not contain any

0

in its decimal representation.

Given an integer

n

, return

a list of two integers

[a, b]

where

:

a

and

b

are

No-Zero integers

.

a + b = n

The test cases are generated so that there is at least one valid solution. If there are many valid solutions, you can return any of them.

Example 1:

Input:

n = 2

Output:

[1,1]

Explanation:

Let a = 1 and b = 1. Both a and b are no-zero integers, and a + b = 2 = n.

Example 2:

Input:

n = 11

Output:

[2,9]

Explanation:

Let a = 2 and b = 9. Both a and b are no-zero integers, and a + b = 11 = n. Note that there are other valid answers as [8, 3] that can be accepted.

Constraints:

2 <= n <= 10

4

Code Snippets

C++:

```
class Solution {
public:
vector<int> getNoZeroIntegers(int n) {

}
};
```

Java:

```
class Solution {
public int[] getNoZeroIntegers(int n) {

}
}
```

Python3:

```
class Solution:
def getNoZeroIntegers(self, n: int) -> List[int]:
```

Python:

```
class Solution(object):
def getNoZeroIntegers(self, n):
"""
```

```
:type n: int
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {number[]}
*/
var getNoZeroIntegers = function(n) {

};
```

TypeScript:

```
function getNoZeroIntegers(n: number): number[] {

};
```

C#:

```
public class Solution {
public int[] GetNoZeroIntegers(int n) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* getNoZeroIntegers(int n, int* returnSize) {

}
```

Go:

```
func getNoZeroIntegers(n int) []int {

}
```

Kotlin:

```
class Solution {
fun getNoZeroIntegers(n: Int): IntArray {

}
}
```

Swift:

```
class Solution {
func getNoZeroIntegers(_ n: Int) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn get_no_zero_integers(n: i32) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {Integer[]}
def get_no_zero_integers(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer[]
*/
function getNoZeroIntegers($n) {

}
```

```
}
```

Dart:

```
class Solution {
List<int> getNoZeroIntegers(int n) {

}
}
```

Scala:

```
object Solution {
def getNoZeroIntegers(n: Int): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_no_zero_integers(n :: integer) :: [integer]
def get_no_zero_integers(n) do

end
end
```

Erlang:

```
-spec get_no_zero_integers(N :: integer()) -> [integer()].
get_no_zero_integers(N) ->
.
```

Racket:

```
(define/contract (get-no-zero-integers n)
(-> exact-integer? (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Convert Integer to the Sum of Two No-Zero Integers
* Difficulty: Easy
* 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:
vector<int> getNoZeroIntegers(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Convert Integer to the Sum of Two No-Zero Integers
* Difficulty: Easy
* 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[] getNoZeroIntegers(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Convert Integer to the Sum of Two No-Zero Integers
Difficulty: Easy
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 getNoZeroIntegers(self, n: int) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getNoZeroIntegers(self, n):
"""
:type n: int
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Convert Integer to the Sum of Two No-Zero Integers
* Difficulty: Easy
* 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 getNoZeroIntegers = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Convert Integer to the Sum of Two No-Zero Integers
* Difficulty: Easy
* 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 getNoZeroIntegers(n: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Convert Integer to the Sum of Two No-Zero Integers
* Difficulty: Easy
* 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[] GetNoZeroIntegers(int n) {

}
}
```

C Solution:

```
/*
* Problem: Convert Integer to the Sum of Two No-Zero Integers
* Difficulty: Easy
* 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
```

```
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* getNoZeroIntegers(int n, int* returnSize) {

}
```

Go Solution:

```
// Problem: Convert Integer to the Sum of Two No-Zero Integers
// Difficulty: Easy
// 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 getNoZeroIntegers(n int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun getNoZeroIntegers(n: Int): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func getNoZeroIntegers(_ n: Int) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Convert Integer to the Sum of Two No-Zero Integers
// Difficulty: Easy
// 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 get_no_zero_integers(n: i32) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Integer[]}
def get_no_zero_integers(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Integer[]
*/
function getNoZeroIntegers($n) {

}
}
```

Dart Solution:

```
class Solution {
List<int> getNoZeroIntegers(int n) {

}
}
```

Scala Solution:

```
object Solution {
def getNoZeroIntegers(n: Int): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_no_zero_integers(n :: integer) :: [integer]
def get_no_zero_integers(n) do

end
end
```

Erlang Solution:

```
-spec get_no_zero_integers(N :: integer()) -> [integer()].
get_no_zero_integers(N) ->
.
```

Racket Solution:

```
(define/contract (get-no-zero-integers n)
(-> exact-integer? (listof exact-integer?))
)
```