Problem 1805: Number of Different Integers in a String

Problem Information

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

Problem Description

You are given a string

word

that consists of digits and lowercase English letters.

You will replace every non-digit character with a space. For example,

"a123bc34d8ef34"

will become

" 123 34 8 34"

. Notice that you are left with some integers that are separated by at least one space:

"123"

,

"34"

,

"8"

, and

"34"

.

Return

the number of

different

integers after performing the replacement operations on

word

.

Two integers are considered different if their decimal representations

without any leading zeros

are different.

Example 1:

Input:

word = "a

123

bc

34

d

8

ef

34

"

Output:

3

Explanation:

The three different integers are "123", "34", and "8". Notice that "34" is only counted once.

Example 2:

Input:

word = "leet

1234

code

234

"

Output:

2

Example 3:

Input:

word = "a

1

b

01

c

001

"

Output:

1

Explanation:

The three integers "1", "01", and "001" all represent the same integer because the leading zeros are ignored when comparing their decimal values.

Constraints:

1 <= word.length <= 1000

word

consists of digits and lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int numDifferentIntegers(string word) {

}
```

```
};
```

Java:

```
class Solution {
public int numDifferentIntegers(String word) {

}
}
```

Python3:

```
class Solution:
def numDifferentIntegers(self, word: str) -> int:
```

Python:

```
class Solution(object):
def numDifferentIntegers(self, word):
"""
:type word: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} word
* @return {number}
*/
var numDifferentIntegers = function(word) {

};
```

TypeScript:

```
function numDifferentIntegers(word: string): number {

};
```

C#:

```
public class Solution {
public int NumDifferentIntegers(string word) {

}
}
```

C:

```
int numDifferentIntegers(char* word) {

}
```

Go:

```
func numDifferentIntegers(word string) int {

}
```

Kotlin:

```
class Solution {
fun numDifferentIntegers(word: String): Int {

}
}
```

Swift:

```
class Solution {
func numDifferentIntegers(_ word: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn num_different_integers(word: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} word
# @return {Integer}
def num_different_integers(word)

end
```

PHP:

```
class Solution {

/**
* @param String $word
* @return Integer
*/
function numDifferentIntegers($word) {

}
}
```

Dart:

```
class Solution {
int numDifferentIntegers(String word) {

}
}
```

Scala:

```
object Solution {
def numDifferentIntegers(word: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec num_different_integers(word :: String.t) :: integer
def num_different_integers(word) do

end
end
```

Erlang:

```
-spec num_different_integers(Word :: unicode:unicode_binary()) -> integer().
num_different_integers(Word) ->
.
```

Racket:

```
(define/contract (num-different-integers word)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Different Integers in a String
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int numDifferentIntegers(string word) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Different Integers in a String
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int numDifferentIntegers(String word) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Different Integers in a String
Difficulty: Easy
Tags: string, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def numDifferentIntegers(self, word: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numDifferentIntegers(self, word):
"""
:type word: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Different Integers in a String
* Difficulty: Easy
```

```
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} word
* @return {number}
*/
var numDifferentIntegers = function(word) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Different Integers in a String
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function numDifferentIntegers(word: string): number {

};
```

C# Solution:

```
/*
* Problem: Number of Different Integers in a String
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

public class Solution {
public int NumDifferentIntegers(string word) {

}
}
```

C Solution:

```
/*
* Problem: Number of Different Integers in a String
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int numDifferentIntegers(char* word) {

}
```

Go Solution:

```
// Problem: Number of Different Integers in a String
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func numDifferentIntegers(word string) int {

}
```

Kotlin Solution:

```
class Solution {
fun numDifferentIntegers(word: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func numDifferentIntegers(_ word: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Different Integers in a String
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn num_different_integers(word: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} word
# @return {Integer}
def num_different_integers(word)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String $word
* @return Integer
*/
function numDifferentIntegers($word) {

}
}
```

Dart Solution:

```
class Solution {
int numDifferentIntegers(String word) {

}
}
```

Scala Solution:

```
object Solution {
def numDifferentIntegers(word: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec num_different_integers(word :: String.t) :: integer
def num_different_integers(word) do

end
end
```

Erlang Solution:

```
-spec num_different_integers(Word :: unicode:unicode_binary()) -> integer().
num_different_integers(Word) ->
.
```

Racket Solution:

```
(define/contract (num-different-integers word)
(-> string? exact-integer?)
)
```