Problem 3352: Count K-Reducible Numbers Less Than N

Problem Information

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

Problem Description

You are given a

binary

string

s

representing a number

n

in its binary form.

You are also given an integer

k

.

An integer

x

is called

k-reducible

if performing the following operation

at most

k

times reduces it to 1:

Replace

x

with the

count

of

set bits

in its binary representation.

For example, the binary representation of 6 is

"110"

. Applying the operation once reduces it to 2 (since

"110"

has two set bits). Applying the operation again to 2 (binary

"10"

) reduces it to 1 (since

"10"

has one set bit).

Return an integer denoting the number of positive integers

less

than

n

that are

k-reducible

.

Since the answer may be too large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

s = "111", k = 1

Output:

3

Explanation:

n = 7

. The 1-reducible integers less than 7 are 1, 2, and 4.

Example 2:

Input:

s = "1000", k = 2

Output:

6

Explanation:

n = 8

. The 2-reducible integers less than 8 are 1, 2, 3, 4, 5, and 6.

Example 3:

Input:

s = "1", k = 3

Output:

0

Explanation:

There are no positive integers less than

n = 1

, so the answer is 0.

Constraints:

1 <= s.length <= 800

s

has no leading zeros.

s

consists only of the characters

'0'

and

'1'

.

1 <= k <= 5

Code Snippets

C++:

```
class Solution {
public:
int countKReducibleNumbers(string s, int k) {

}
};
```

Java:

```
class Solution {
public int countKReducibleNumbers(String s, int k) {
}
}
```

Python3:

```
class Solution:
def countKReducibleNumbers(self, s: str, k: int) -> int:
```

Python:

```
class Solution(object):
def countKReducibleNumbers(self, s, k):
"""
:type s: str
:type k: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {number} k
* @return {number}
*/
var countKReducibleNumbers = function(s, k) {

};
```

TypeScript:

```
function countKReducibleNumbers(s: string, k: number): number {

};
```

C#:

```
public class Solution {
public int CountKReducibleNumbers(string s, int k) {

}

}
```

C:

```
int countKReducibleNumbers(char* s, int k) {

}
```

Go:

```
func countKReducibleNumbers(s string, k int) int {

}
```

Kotlin:

```
class Solution {
fun countKReducibleNumbers(s: String, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func countKReducibleNumbers(_ s: String, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_k_reducible_numbers(s: String, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} k
# @return {Integer}
def count_k_reducible_numbers(s, k)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return Integer
*/
function countKReducibleNumbers($s, $k) {

}
}
```

Dart:

```
class Solution {
int countKReducibleNumbers(String s, int k) {

}
}
```

Scala:

```
object Solution {
def countKReducibleNumbers(s: String, k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_k_reducible_numbers(s :: String.t, k :: integer) :: integer
def count_k_reducible_numbers(s, k) do

end
```

```
end
```

Erlang:

```
-spec count_k_reducible_numbers(S :: unicode:unicode_binary(), K ::
integer()) -> integer().
count_k_reducible_numbers(S, K) ->
.
```

Racket:

```
(define/contract (count-k-reducible-numbers s k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count K-Reducible Numbers Less Than N
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int countKReducibleNumbers(string s, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Count K-Reducible Numbers Less Than N
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int countKReducibleNumbers(String s, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Count K-Reducible Numbers Less Than N
Difficulty: Hard
Tags: string, dp, math

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

class Solution:
def countKReducibleNumbers(self, s: str, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countKReducibleNumbers(self, s, k):
"""
:type s: str
:type k: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count K-Reducible Numbers Less Than N
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} s
* @param {number} k
* @return {number}
*/
var countKReducibleNumbers = function(s, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Count K-Reducible Numbers Less Than N
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function countKReducibleNumbers(s: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Count K-Reducible Numbers Less Than N
* Difficulty: Hard
* Tags: string, dp, math
```

```
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int CountKReducibleNumbers(string s, int k) {

}
}
```

C Solution:

```
/*
* Problem: Count K-Reducible Numbers Less Than N
* Difficulty: Hard
* Tags: string, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int countKReducibleNumbers(char* s, int k) {

}
```

Go Solution:

```
// Problem: Count K-Reducible Numbers Less Than N
// Difficulty: Hard
// Tags: string, dp, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func countKReducibleNumbers(s string, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countKReducibleNumbers(s: String, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func countKReducibleNumbers(_ s: String, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count K-Reducible Numbers Less Than N
// Difficulty: Hard
// Tags: string, dp, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn count_k_reducible_numbers(s: String, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} k
# @return {Integer}
def count_k_reducible_numbers(s, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return Integer
*/
function countKReducibleNumbers($s, $k) {

}
}
```

Dart Solution:

```
class Solution {
int countKReducibleNumbers(String s, int k) {

}
}
```

Scala Solution:

```
object Solution {
def countKReducibleNumbers(s: String, k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_k_reducible_numbers(s :: String.t, k :: integer) :: integer
def count_k_reducible_numbers(s, k) do

end
end
```

Erlang Solution:

```
-spec count_k_reducible_numbers(S :: unicode:unicode_binary(), K ::
integer()) -> integer().
count_k_reducible_numbers(S, K) ->
.
```

Racket Solution:

```
(define/contract (count-k-reducible-numbers s k)
(-> string? exact-integer? exact-integer?)
)
```