Problem 1016: Binary String With Substrings Representing 1 To N

Problem Information

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

Problem Description

Given a binary string

s

and a positive integer

n

, return

true

if the binary representation of all the integers in the range

[1, n]

are

substrings

of

s

, or

false

otherwise

.

A

substring

is a contiguous sequence of characters within a string.

Example 1:

Input:

s = "0110", n = 3

Output:

true

Example 2:

Input:

s = "0110", n = 4

Output:

false

Constraints:

1 <= s.length <= 1000

s[i]

is either

'0'

or

'1'

.

1 <= n <= 10

9

Code Snippets

C++:

```
class Solution {
public:
bool queryString(string s, int n) {

}
};
```

Java:

```
class Solution {
public boolean queryString(String s, int n) {

}
}
```

Python3:

```
class Solution:
def queryString(self, s: str, n: int) -> bool:
```

Python:

```
class Solution(object):
def queryString(self, s, n):
"""
:type s: str
:type n: int
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {number} n
* @return {boolean}
*/
var queryString = function(s, n) {

};
```

TypeScript:

```
function queryString(s: string, n: number): boolean {

};
```

C#:

```
public class Solution {
public bool QueryString(string s, int n) {

}
}
```

C:

```
bool queryString(char* s, int n) {

}
```

Go:

```
func queryString(s string, n int) bool {
```

```
}
```

Kotlin:

```
class Solution {
fun queryString(s: String, n: Int): Boolean {

}
}
```

Swift:

```
class Solution {
func queryString(_ s: String, _ n: Int) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn query_string(s: String, n: i32) -> bool {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} n
# @return {Boolean}
def query_string(s, n)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $n
```

```
* @return Boolean
*/
function queryString($s, $n) {

}
}
```

Dart:

```
class Solution {
bool queryString(String s, int n) {

}
}
```

Scala:

```
object Solution {
def queryString(s: String, n: Int): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec query_string(s :: String.t, n :: integer) :: boolean
def query_string(s, n) do

end
end
```

Erlang:

```
-spec query_string(S :: unicode:unicode_binary(), N :: integer()) ->
boolean().
query_string(S, N) ->
.
```

Racket:

```
(define/contract (query-string s n)
(-> string? exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Binary String With Substrings Representing 1 To N
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
bool queryString(string s, int n) {

}
};
```

Java Solution:

```
/**
* Problem: Binary String With Substrings Representing 1 To N
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public boolean queryString(String s, int n) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Binary String With Substrings Representing 1 To N
Difficulty: Medium
Tags: array, string, tree, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

```
class Solution(object):
def queryString(self, s, n):
"""
:type s: str
:type n: int
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Binary String With Substrings Representing 1 To N
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
/**
* @param {string} s
* @param {number} n
* @return {boolean}
*/
var queryString = function(s, n) {

};
```

TypeScript Solution:

```
/**
* Problem: Binary String With Substrings Representing 1 To N
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function queryString(s: string, n: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Binary String With Substrings Representing 1 To N
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public bool QueryString(string s, int n) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Binary String With Substrings Representing 1 To N
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

bool queryString(char* s, int n) {

}
```

Go Solution:

```
// Problem: Binary String With Substrings Representing 1 To N
// Difficulty: Medium
// Tags: array, string, tree, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func queryString(s string, n int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun queryString(s: String, n: Int): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func queryString(_ s: String, _ n: Int) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Binary String With Substrings Representing 1 To N
// Difficulty: Medium
// Tags: array, string, tree, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn query_string(s: String, n: i32) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} n
# @return {Boolean}
def query_string(s, n)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $n
* @return Boolean
*/
function queryString($s, $n) {
```

```
}
}
```

Dart Solution:

```
class Solution {
bool queryString(String s, int n) {

}
}
```

Scala Solution:

```
object Solution {
def queryString(s: String, n: Int): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec query_string(s :: String.t, n :: integer) :: boolean
def query_string(s, n) do

end
end
```

Erlang Solution:

```
-spec query_string(S :: unicode:unicode_binary(), N :: integer()) ->
boolean().
query_string(S, N) ->
.
```

Racket Solution:

```
(define/contract (query-string s n)
(-> string? exact-integer? boolean?)
)
```