Problem 1545: Find Kth Bit in Nth Binary String

Problem Information

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

Problem Description

Given two positive integers

n

and

k

, the binary string

S

n

is formed as follows:

S

1

= "0"

S

i

= S

i - 1

+ "1" + reverse(invert(S

i - 1

))

for

i > 1

Where

+

denotes the concatenation operation,

reverse(x)

returns the reversed string

x

, and

invert(x)

inverts all the bits in

x

(

0

changes to

1

and

1

changes to

0

).

For example, the first four strings in the above sequence are:

S

1

= "0"

S

2

= "0

1

1"

S

3

= "011

1

001"

S

4

= "0111001

1

0110001"

Return

the

k

th

bit

in

S

n

. It is guaranteed that

k

is valid for the given

n

.

Example 1:

Input:

n = 3, k = 1

Output:

"0"

Explanation:

S

3

is "

0

111001". The 1

st

bit is "0".

Example 2:

Input:

n = 4, k = 11

Output:

"1"

Explanation:

S

4

is "0111001101

1

0001". The 11

th

bit is "1".

Constraints:

1 <= n <= 20

1 <= k <= 2

n

- 1

Code Snippets

C++:

```
class Solution {
public:
char findKthBit(int n, int k) {

}
};
```

Java:

```
class Solution {
public char findKthBit(int n, int k) {

}
}
```

Python3:

```
class Solution:
def findKthBit(self, n: int, k: int) -> str:
```

Python:

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

JavaScript:

```
/**
* @param {number} n
* @param {number} k
* @return {character}
*/
var findKthBit = function(n, k) {

};
```

TypeScript:

```
function findKthBit(n: number, k: number): string {

};
```

C#:

```
public class Solution {
public char FindKthBit(int n, int k) {

}
}
```

C:

```
char findKthBit(int n, int k) {

}
```

Go:

```
func findKthBit(n int, k int) byte {

}
```

Kotlin:

```
class Solution {
fun findKthBit(n: Int, k: Int): Char {

}
}
```

Swift:

```
class Solution {
func findKthBit(_ n: Int, _ k: Int) -> Character {

}
}
```

Rust:

```
impl Solution {
pub fn find_kth_bit(n: i32, k: i32) -> char {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} k
# @return {Character}
def find_kth_bit(n, k)
```

```
end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @return String
*/
function findKthBit($n, $k) {

}
}
```

Dart:

```
class Solution {
String findKthBit(int n, int k) {

}
}
```

Scala:

```
object Solution {
def findKthBit(n: Int, k: Int): Char = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_kth_bit(n :: integer, k :: integer) :: char
def find_kth_bit(n, k) do

end
end
```

Erlang:

```
-spec find_kth_bit(N :: integer(), K :: integer()) -> char().
find_kth_bit(N, K) ->
.
```

Racket:

```
(define/contract (find-kth-bit n k)
(-> exact-integer? exact-integer? char?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Kth Bit in Nth Binary String
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
char findKthBit(int n, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Find Kth Bit in Nth Binary String
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

class Solution {
public char findKthBit(int n, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Kth Bit in Nth Binary String
Difficulty: Medium
Tags: string

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find Kth Bit in Nth Binary String
* Difficulty: Medium
* Tags: string
```

```
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} n
* @param {number} k
* @return {character}
*/
var findKthBit = function(n, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Kth Bit in Nth Binary String
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function findKthBit(n: number, k: number): string {

};
```

C# Solution:

```
/*
* Problem: Find Kth Bit in Nth Binary String
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

public class Solution {
public char FindKthBit(int n, int k) {

}
}
```

C Solution:

```
/*
* Problem: Find Kth Bit in Nth Binary String
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char findKthBit(int n, int k) {

}
```

Go Solution:

```
// Problem: Find Kth Bit in Nth Binary String
// Difficulty: Medium
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func findKthBit(n int, k int) byte {

}
```

Kotlin Solution:

```
class Solution {
fun findKthBit(n: Int, k: Int): Char {

}
}
```

Swift Solution:

```
class Solution {
func findKthBit(_ n: Int, _ k: Int) -> Character {

}
}
```

Rust Solution:

```
// Problem: Find Kth Bit in Nth Binary String
// Difficulty: Medium
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn find_kth_bit(n: i32, k: i32) -> char {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} k
# @return {Character}
def find_kth_bit(n, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @return String
*/
function findKthBit($n, $k) {

}
}
```

Dart Solution:

```
class Solution {
String findKthBit(int n, int k) {

}
}
```

Scala Solution:

```
object Solution {
def findKthBit(n: Int, k: Int): Char = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_kth_bit(n :: integer, k :: integer) :: char
def find_kth_bit(n, k) do

end
end
```

Erlang Solution:

```
-spec find_kth_bit(N :: integer(), K :: integer()) -> char().
find_kth_bit(N, K) ->
.
```

Racket Solution:

```
(define/contract (find-kth-bit n k)
(-> exact-integer? exact-integer? char?)
)
```