Problem 440: K-th Smallest in Lexicographical Order

Problem Information

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

Problem Description

Given two integers

n

and

k

, return

the

k

th

lexicographically smallest integer in the range

[1, n]

.

Example 1:

Input:

n = 13, k = 2

Output:

10

Explanation:

The lexicographical order is [1, 10, 11, 12, 13, 2, 3, 4, 5, 6, 7, 8, 9], so the second smallest number is 10.

Example 2:

Input:

n = 1, k = 1

Output:

1

Constraints:

1 <= k <= n <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int findKthNumber(int n, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function findKthNumber(n: number, k: number): number {

};
```

C#:

```
public class Solution {
public int FindKthNumber(int n, int k) {

}
}
```

C:

```
int findKthNumber(int n, int k) {

}
```

Go:

```
func findKthNumber(n int, k int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int findKthNumber(int n, int k) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: K-th Smallest in Lexicographical Order
* Difficulty: Hard
* Tags: graph
*
* 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 findKthNumber(int n, int k) {

}
};
```

Java Solution:

```
/**
* Problem: K-th Smallest in Lexicographical Order
```

```
* Difficulty: Hard
* Tags: graph
*
* 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 findKthNumber(int n, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: K-th Smallest in Lexicographical Order
Difficulty: Hard
Tags: graph

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 findKthNumber(self, n: int, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: K-th Smallest in Lexicographical Order
* Difficulty: Hard
* Tags: graph
*
* 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
* @param {number} k
* @return {number}
*/
var findKthNumber = function(n, k) {

};
```

TypeScript Solution:

```
/**
* Problem: K-th Smallest in Lexicographical Order
* Difficulty: Hard
* Tags: graph
*
* 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 findKthNumber(n: number, k: number): number {

};
```

C# Solution:

```
/*
* Problem: K-th Smallest in Lexicographical Order
* Difficulty: Hard
* Tags: graph
```

```
*
* 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 FindKthNumber(int n, int k) {

}
}
```

C Solution:

```
/*
* Problem: K-th Smallest in Lexicographical Order
* Difficulty: Hard
* Tags: graph
*
* 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
*/

int findKthNumber(int n, int k) {

}
```

Go Solution:

```
// Problem: K-th Smallest in Lexicographical Order
// Difficulty: Hard
// Tags: graph
//
// 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 findKthNumber(n int, k int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: K-th Smallest in Lexicographical Order
// Difficulty: Hard
// Tags: graph
//
// 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 find_kth_number(n: i32, k: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int findKthNumber(int n, int k) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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