Problem 1663: Smallest String With A Given Numeric Value

Problem Information

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

Problem Description

The

numeric value

of a

lowercase character

is defined as its position

(1-indexed)

in the alphabet, so the numeric value of

a

is

1

, the numeric value of

b

is

2

, the numeric value of

c

is

3

, and so on.

The

numeric value

of a

string

consisting of lowercase characters is defined as the sum of its characters' numeric values. For example, the numeric value of the string

"abe"

is equal to

1 + 2 + 5 = 8

.

You are given two integers

n

and

k

. Return

the

lexicographically smallest string

with

length

equal to

n

and

numeric value

equal to

k

.

Note that a string

x

is lexicographically smaller than string

y

if

x

comes before

y

in dictionary order, that is, either

x

is a prefix of

y

, or if

i

is the first position such that

x[i] != y[i]

, then

x[i]

comes before

y[i]

in alphabetic order.

Example 1:

Input:

n = 3, k = 27

Output:

"aay"

Explanation:

The numeric value of the string is 1 + 1 + 25 = 27, and it is the smallest string with such a value and length equal to 3.

Example 2:

Input:

n = 5, k = 73

Output:

"aaszz"

Constraints:

1 <= n <= 10

5

n <= k <= 26 * n

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public String getSmallestString(int n, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public string GetSmallestString(int n, int k) {

}
}
```

C:

```
char* getSmallestString(int n, int k) {

}
```

Go:

```
func getSmallestString(n int, k int) string {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec get_smallest_string(N :: integer(), K :: integer()) ->
unicode:unicode_binary().
```

```
get_smallest_string(N, K) ->
.
```

Racket:

```
(define/contract (get-smallest-string n k)
(-> exact-integer? exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Smallest String With A Given Numeric Value
* Difficulty: Medium
* Tags: string, graph, greedy
*
* 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:
string getSmallestString(int n, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Smallest String With A Given Numeric Value
* Difficulty: Medium
* Tags: string, graph, greedy
*
* 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 String getSmallestString(int n, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest String With A Given Numeric Value
Difficulty: Medium
Tags: string, graph, greedy

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Smallest String With A Given Numeric Value
* Difficulty: Medium
* Tags: string, graph, greedy
```

```
*
* 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 {string}
*/
var getSmallestString = function(n, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest String With A Given Numeric Value
* Difficulty: Medium
* Tags: string, graph, greedy
*
* 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 getSmallestString(n: number, k: number): string {

};
```

C# Solution:

```
/*
* Problem: Smallest String With A Given Numeric Value
* Difficulty: Medium
* Tags: string, graph, greedy
*
* 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 string GetSmallestString(int n, int k) {

}
}
```

C Solution:

```
/*
* Problem: Smallest String With A Given Numeric Value
* Difficulty: Medium
* Tags: string, graph, greedy
*
* 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* getSmallestString(int n, int k) {

}
```

Go Solution:

```
// Problem: Smallest String With A Given Numeric Value
// Difficulty: Medium
// Tags: string, graph, greedy
//
// 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 getSmallestString(n int, k int) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Smallest String With A Given Numeric Value
// Difficulty: Medium
// Tags: string, graph, greedy
//
// 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 get_smallest_string(n: i32, k: i32) -> String {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec get_smallest_string(N :: integer(), K :: integer()) ->
unicode:unicode_binary().
get_smallest_string(N, K) ->
```

```
.
```

Racket Solution:

```
(define/contract (get-smallest-string n k)
(-> exact-integer? exact-integer? string?)
)
```