Problem 3307: Find the K-th Character in String Game II

Problem Information

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

Problem Description

Alice and Bob are playing a game. Initially, Alice has a string

word = "a"

.

You are given a

positive

integer

k

. You are also given an integer array

operations

, where

operations[i]

represents the

type

of the

i

th

operation.

Now Bob will ask Alice to perform

all

operations in sequence:

If

operations[i] == 0

,

append

a copy of

word

to itself.

If

operations[i] == 1

, generate a new string by

changing

each character in

word

to its

next

character in the English alphabet, and

append

it to the

original

word

. For example, performing the operation on

"c"

generates

"cd"

and performing the operation on

"zb"

generates

"zbac"

.

Return the value of the

k

th

character in

word

after performing all the operations.

Note

that the character

'z'

can be changed to

'a'

in the second type of operation.

Example 1:

Input:

k = 5, operations = [0,0,0]

Output:

"a"

Explanation:

Initially,

word == "a"

. Alice performs the three operations as follows:

Appends

"a"

to

"a"

,

word

becomes

"aa"

.

Appends

"aa"

to

"aa"

,

word

becomes

"aaaa"

.

Appends

"aaaa"

to

"aaaa"

,

word

becomes

"aaaaaaaa"

.

Example 2:

Input:

k = 10, operations = [0,1,0,1]

Output:

"b"

Explanation:

Initially,

word == "a"

. Alice performs the four operations as follows:

Appends

"a"

to

"a"

,

word

becomes

"aa"

.

Appends

"bb"

to

"aa"

,

word

becomes

"aabb"

.

Appends

"aabb"

to

"aabb"

,

word

becomes

"aabbaabb"

.

Appends

"bbccbbcc"

to

"aabbaabb"

,

word

becomes

"aabbaabbbbccbbcc"

.

Constraints:

1 <= k <= 10

14

1 <= operations.length <= 100

operations[i]

is either 0 or 1.

The input is generated such that

word

has

at least

k

characters after all operations.

Code Snippets

C++:

```
class Solution {
public:
char kthCharacter(long long k, vector<int>& operations) {

}
};
```

Java:

```
class Solution {
public char kthCharacter(long k, int[] operations) {

}
}
```

Python3:

```
class Solution:
def kthCharacter(self, k: int, operations: List[int]) -> str:
```

Python:

```
class Solution(object):
def kthCharacter(self, k, operations):
"""
:type k: int
```

```
:type operations: List[int]
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number} k
* @param {number[]} operations
* @return {character}
*/
var kthCharacter = function(k, operations) {

};
```

TypeScript:

```
function kthCharacter(k: number, operations: number[]): string {

};
```

C#:

```
public class Solution {
public char KthCharacter(long k, int[] operations) {

}
}
```

C:

```
char kthCharacter(long long k, int* operations, int operationsSize) {

}
```

Go:

```
func kthCharacter(k int64, operations []int) byte {

}
```

Kotlin:

```
class Solution {
fun kthCharacter(k: Long, operations: IntArray): Char {

}
}
```

Swift:

```
class Solution {
func kthCharacter(_ k: Int, _ operations: [Int]) -> Character {

}
}
```

Rust:

```
impl Solution {
pub fn kth_character(k: i64, operations: Vec<i32>) -> char {

}
}
```

Ruby:

```
# @param {Integer} k
# @param {Integer[]} operations
# @return {Character}
def kth_character(k, operations)

end
```

PHP:

```
class Solution {

/**
* @param Integer $k
* @param Integer[] $operations
* @return String
*/
function kthCharacter($k, $operations) {

}
```

```
}
```

Dart:

```
class Solution {
String kthCharacter(int k, List<int> operations) {

}
}
```

Scala:

```
object Solution {
def kthCharacter(k: Long, operations: Array[Int]): Char = {

}
}
```

Elixir:

```
defmodule Solution do
@spec kth_character(k :: integer, operations :: [integer]) :: char
def kth_character(k, operations) do

end
end
```

Erlang:

```
-spec kth_character(K :: integer(), Operations :: [integer()]) -> char().
kth_character(K, Operations) ->
.
```

Racket:

```
(define/contract (kth-character k operations)
(-> exact-integer? (listof exact-integer?) char?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the K-th Character in String Game II
* Difficulty: Hard
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
char kthCharacter(long long k, vector<int>& operations) {

}
};
```

Java Solution:

```
/**
* Problem: Find the K-th Character in String Game II
* Difficulty: Hard
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public char kthCharacter(long k, int[] operations) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the K-th Character in String Game II
Difficulty: Hard
Tags: array, string, math
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def kthCharacter(self, k: int, operations: List[int]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def kthCharacter(self, k, operations):
"""
:type k: int
:type operations: List[int]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Find the K-th Character in String Game II
* Difficulty: Hard
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} k
* @param {number[]} operations
* @return {character}
*/
var kthCharacter = function(k, operations) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the K-th Character in String Game II
* Difficulty: Hard
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function kthCharacter(k: number, operations: number[]): string {

};
```

C# Solution:

```
/*
* Problem: Find the K-th Character in String Game II
* Difficulty: Hard
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public char KthCharacter(long k, int[] operations) {

}
}
```

C Solution:

```
/*
* Problem: Find the K-th Character in String Game II
* Difficulty: Hard
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char kthCharacter(long long k, int* operations, int operationsSize) {

}
```

Go Solution:

```
// Problem: Find the K-th Character in String Game II
// Difficulty: Hard
// Tags: array, string, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func kthCharacter(k int64, operations []int) byte {

}
```

Kotlin Solution:

```
class Solution {
fun kthCharacter(k: Long, operations: IntArray): Char {

}
}
```

Swift Solution:

```
class Solution {
func kthCharacter(_ k: Int, _ operations: [Int]) -> Character {

}
}
```

Rust Solution:

```
// Problem: Find the K-th Character in String Game II
// Difficulty: Hard
// Tags: array, string, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn kth_character(k: i64, operations: Vec<i32>) -> char {

}
}
```

Ruby Solution:

```
# @param {Integer} k
# @param {Integer[]} operations
# @return {Character}
def kth_character(k, operations)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $k
* @param Integer[] $operations
* @return String
*/
function kthCharacter($k, $operations) {

}
}
```

Dart Solution:

```
class Solution {
String kthCharacter(int k, List<int> operations) {
```

```
}
}
```

Scala Solution:

```
object Solution {
def kthCharacter(k: Long, operations: Array[Int]): Char = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec kth_character(k :: integer, operations :: [integer]) :: char
def kth_character(k, operations) do

end
end
```

Erlang Solution:

```
-spec kth_character(K :: integer(), Operations :: [integer()]) -> char().
kth_character(K, Operations) ->
.
```

Racket Solution:

```
(define/contract (kth-character k operations)
(-> exact-integer? (listof exact-integer?) char?)
)
```