Problem 3456: Find Special Substring of Length K

Problem Information

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

Problem Description

You are given a string

s

and an integer

k

.

Determine if there exists a

substring

of length

exactly

k

in

s

that satisfies the following conditions:

The substring consists of

only one distinct character

(e.g.,

"aaa"

or

"bbb"

).

If there is a character

immediately before

the substring, it must be different from the character in the substring.

If there is a character

immediately after

the substring, it must also be different from the character in the substring.

Return

true

if such a substring exists. Otherwise, return

false

.

Example 1:

Input:

s = "aaabaaa", k = 3

Output:

true

Explanation:

The substring

s[4..6] == "aaa"

satisfies the conditions.

It has a length of 3.

All characters are the same.

The character before

"aaa"

is

'b'

, which is different from

'a'

.

There is no character after

"aaa"

.

Example 2:

Input:

s = "abc", k = 2

Output:

false

Explanation:

There is no substring of length 2 that consists of one distinct character and satisfies the conditions.

Constraints:

1 <= k <= s.length <= 100

s

consists of lowercase English letters only.

Code Snippets

C++:

```
class Solution {
public:
bool hasSpecialSubstring(string s, int k) {

}
};
```

Java:

```
class Solution {
public boolean hasSpecialSubstring(String s, int k) {
```

```
}
}
```

Python3:

```
class Solution:
def hasSpecialSubstring(self, s: str, k: int) -> bool:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function hasSpecialSubstring(s: string, k: number): boolean {

};
```

C#:

```
public class Solution {
public bool HasSpecialSubstring(string s, int k) {

}
}
```

C:

```
bool hasSpecialSubstring(char* s, int k) {

}
```

Go:

```
func hasSpecialSubstring(s string, k int) bool {

}
```

Kotlin:

```
class Solution {
fun hasSpecialSubstring(s: String, k: Int): Boolean {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} k
# @return {Boolean}
def has_special_substring(s, k)
```

```
end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return Boolean
*/
function hasSpecialSubstring($s, $k) {

}
}
```

Dart:

```
class Solution {
bool hasSpecialSubstring(String s, int k) {

}
}
```

Scala:

```
object Solution {
def hasSpecialSubstring(s: String, k: Int): Boolean = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec has_special_substring(S :: unicode:unicode_binary(), K :: integer()) ->
boolean().
has_special_substring(S, K) ->
.
```

Racket:

```
(define/contract (has-special-substring s k)
(-> string? exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Special Substring of Length K
* Difficulty: Easy
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
bool hasSpecialSubstring(string s, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Find Special Substring of Length K
* Difficulty: Easy
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
```

```
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public boolean hasSpecialSubstring(String s, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Special Substring of Length K
Difficulty: Easy
Tags: string, tree

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find Special Substring of Length K
* Difficulty: Easy
```

```
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* 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} k
* @return {boolean}
*/
var hasSpecialSubstring = function(s, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Special Substring of Length K
* Difficulty: Easy
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function hasSpecialSubstring(s: string, k: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Find Special Substring of Length K
* Difficulty: Easy
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* 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 HasSpecialSubstring(string s, int k) {

}
}
```

C Solution:

```
/*
* Problem: Find Special Substring of Length K
* Difficulty: Easy
* Tags: string, tree
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

bool hasSpecialSubstring(char* s, int k) {

}
```

Go Solution:

```
// Problem: Find Special Substring of Length K
// Difficulty: Easy
// Tags: string, tree
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func hasSpecialSubstring(s string, k int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun hasSpecialSubstring(s: String, k: Int): Boolean {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find Special Substring of Length K
// Difficulty: Easy
// Tags: string, tree
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn has_special_substring(s: String, k: i32) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} k
# @return {Boolean}
def has_special_substring(s, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return Boolean
*/
function hasSpecialSubstring($s, $k) {

}
}
```

Dart Solution:

```
class Solution {
bool hasSpecialSubstring(String s, int k) {

}
}
```

Scala Solution:

```
object Solution {
def hasSpecialSubstring(s: String, k: Int): Boolean = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec has_special_substring(S :: unicode:unicode_binary(), K :: integer()) ->
boolean().
has_special_substring(S, K) ->
```

```
.
```

Racket Solution:

```
(define/contract (has-special-substring s k)
(-> string? exact-integer? boolean?)
)
```