Problem 1163: Last Substring in Lexicographical Order

Problem Information

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

Problem Description

Given a string

s

, return

the last substring of

s

in lexicographical order

.

Example 1:

Input:

s = "abab"

Output:

"bab"

Explanation:

The substrings are ["a", "ab", "aba", "abab", "b", "ba", "bab"]. The lexicographically maximum substring is "bab".

Example 2:

Input:

s = "leetcode"

Output:

"tcode"

Constraints:

1 <= s.length <= 4 * 10

5

s

contains only lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
string lastSubstring(string s) {

}
};
```

Java:

```
class Solution {
public String lastSubstring(String s) {
```

```
}
}
```

Python3:

```
class Solution:
def lastSubstring(self, s: str) -> str:
```

Python:

```
class Solution(object):
def lastSubstring(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {string}
*/
var lastSubstring = function(s) {

};
```

TypeScript:

```
function lastSubstring(s: string): string {

};
```

C#:

```
public class Solution {
public string LastSubstring(string s) {

}
}
```

C:

```
char* lastSubstring(char* s) {

}
```

Go:

```
func lastSubstring(s string) string {

}
```

Kotlin:

```
class Solution {
fun lastSubstring(s: String): String {

}
}
```

Swift:

```
class Solution {
func lastSubstring(_ s: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn last_substring(s: String) -> String {

}
}
```

Ruby:

```
# @param {String} s
# @return {String}
def last_substring(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return String
*/
function lastSubstring($s) {

}
}
```

Dart:

```
class Solution {
String lastSubstring(String s) {

}
}
```

Scala:

```
object Solution {
def lastSubstring(s: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec last_substring(s :: String.t) :: String.t
def last_substring(s) do

end
end
```

Erlang:

```
-spec last_substring(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
last_substring(S) ->
.
```

Racket:

```
(define/contract (last-substring s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Last Substring in Lexicographical Order
* Difficulty: Hard
* Tags: array, string, tree, graph
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
string lastSubstring(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Last Substring in Lexicographical Order
* Difficulty: Hard
* Tags: array, string, tree, graph
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public String lastSubstring(String s) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Last Substring in Lexicographical Order
Difficulty: Hard
Tags: array, string, tree, graph

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
def lastSubstring(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def lastSubstring(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Last Substring in Lexicographical Order
* Difficulty: Hard
* Tags: array, string, tree, graph
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
/**
* @param {string} s
* @return {string}
*/
var lastSubstring = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Last Substring in Lexicographical Order
* Difficulty: Hard
* Tags: array, string, tree, graph
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function lastSubstring(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Last Substring in Lexicographical Order
* Difficulty: Hard
* Tags: array, string, tree, graph
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public string LastSubstring(string s) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Last Substring in Lexicographical Order
* Difficulty: Hard
* Tags: array, string, tree, graph
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

char* lastSubstring(char* s) {

}
```

Go Solution:

```
// Problem: Last Substring in Lexicographical Order
// Difficulty: Hard
// Tags: array, string, tree, graph
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func lastSubstring(s string) string {

}
```

Kotlin Solution:

```
class Solution {
fun lastSubstring(s: String): String {

}
}
```

Swift Solution:

```
class Solution {
func lastSubstring(_ s: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Last Substring in Lexicographical Order
// Difficulty: Hard
// Tags: array, string, tree, graph
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn last_substring(s: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {String}
def last_substring(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return String
*/
function lastSubstring($s) {

}
}
```

Dart Solution:

```
class Solution {
String lastSubstring(String s) {

}
}
```

Scala Solution:

```
object Solution {
def lastSubstring(s: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec last_substring(s :: String.t) :: String.t
def last_substring(s) do

end
end
```

Erlang Solution:

```
-spec last_substring(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
last_substring(S) ->
.
```

Racket Solution:

```
(define/contract (last-substring s)
(-> string? string?)
)
```