Problem 1044: Longest Duplicate Substring

Problem Information

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

Problem Description

Given a string

s

, consider all

duplicated substrings

: (contiguous) substrings of s that occur 2 or more times. The occurrences may overlap.

Return

any

duplicated substring that has the longest possible length. If

s

does not have a duplicated substring, the answer is

""

.

Example 1:

Input:

s = "banana"

Output:

"ana"

Example 2:

Input:

s = "abcd"

Output:

""

Constraints:

2 <= s.length <= 3 * 10

4

s

consists of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public string LongestDupSubstring(string s) {
}
}
```

C:

```
char* longestDupSubstring(char* s) {

}
```

Go:

```
func longestDupSubstring(s string) string {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Longest Duplicate Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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 longestDupSubstring(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Duplicate Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
```

```
* 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 longestDupSubstring(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Duplicate Substring
Difficulty: Hard
Tags: array, string, tree, hash, search

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 longestDupSubstring(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Duplicate Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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 longestDupSubstring = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Duplicate Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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 longestDupSubstring(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Longest Duplicate Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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 LongestDupSubstring(string s) {

}
}
```

C Solution:

```
/*
* Problem: Longest Duplicate Substring
* Difficulty: Hard
* Tags: array, string, tree, hash, search
*
* 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* longestDupSubstring(char* s) {

}
```

Go Solution:

```
// Problem: Longest Duplicate Substring
// Difficulty: Hard
// Tags: array, string, tree, hash, search
//
// 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 longestDupSubstring(s string) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Longest Duplicate Substring
// Difficulty: Hard
// Tags: array, string, tree, hash, search
//
// 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 longest_dup_substring(s: String) -> String {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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