Problem 1698: Number of Distinct Substrings in a String

Problem Information

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

Problem Description

Given a string

s

, return

the number of

distinct

substrings of

s

.

A

substring

of a string is obtained by deleting any number of characters (possibly zero) from the front of the string and any number (possibly zero) from the back of the string.

Example 1:

Input:

s = "aabbaba"

Output:

21

Explanation:

The set of distinct strings is ["a","b","aa","bb","ab","ba","aab","abb","bab","bba","aba","aabb","abba","bbab","baba","aabba","abbab","bbaba","aabbab","abbaba","aabbaba"]

Example 2:

Input:

s = "abcdefg"

Output:

28

Constraints:

1 <= s.length <= 500

s

consists of lowercase English letters.

Follow up:

Can you solve this problem in

O(n)

time complexity?

Code Snippets

C++:

```
class Solution {
public:
int countDistinct(string s) {

}
};
```

Java:

```
class Solution {
public int countDistinct(String s) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function countDistinct(s: string): number {

};
```

C#:

```
public class Solution {
public int CountDistinct(string s) {

}
}
```

C:

```
int countDistinct(char* s) {

}
```

Go:

```
func countDistinct(s string) int {

}
```

Kotlin:

```
class Solution {
fun countDistinct(s: String): Int {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn count_distinct(s: String) -> i32 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int countDistinct(String s) {

}
}
```

Scala:

```
object Solution {
def countDistinct(s: String): Int = {

}
```

```
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec count_distinct(S :: unicode:unicode_binary()) -> integer().
count_distinct(S) ->
.
```

Racket:

```
(define/contract (count-distinct s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Distinct Substrings in a String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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:
int countDistinct(string s) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Number of Distinct Substrings in a String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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 int countDistinct(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Distinct Substrings in a String
Difficulty: Medium
Tags: array, string, tree, hash

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Distinct Substrings in a String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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 {number}
*/
var countDistinct = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Distinct Substrings in a String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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 countDistinct(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Number of Distinct Substrings in a String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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 int CountDistinct(string s) {

}
}
```

C Solution:

```
/*
* Problem: Number of Distinct Substrings in a String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* 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
*/

int countDistinct(char* s) {

}
```

Go Solution:

```
// Problem: Number of Distinct Substrings in a String
// Difficulty: Medium
// Tags: array, string, tree, hash
//
// 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 countDistinct(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun countDistinct(s: String): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number of Distinct Substrings in a String
// Difficulty: Medium
// Tags: array, string, tree, hash
//
// 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 count_distinct(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int countDistinct(String s) {

}
}
```

Scala Solution:

```
object Solution {
def countDistinct(s: String): Int = {

}
}
```

Elixir Solution:

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

end
```

```
end
```

Erlang Solution:

```
-spec count_distinct(S :: unicode:unicode_binary()) -> integer().
count_distinct(S) ->
.
```

Racket Solution:

```
(define/contract (count-distinct s)
(-> string? exact-integer?)
)
```