Problem 940: Distinct Subsequences II

Problem Information

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

Problem Description

Given a string s, return

the number of

distinct non-empty subsequences

of

s

. Since the answer may be very large, return it

modulo

10

9

+ 7

.

A

subsequence

of a string is a new string that is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (i.e.,

"ace"

is a subsequence of

"

a

b

c

d

e

"

while

"aec"

is not.

Example 1:

Input:

s = "abc"

Output:

7

Explanation:

The 7 distinct subsequences are "a", "b", "c", "ab", "ac", "bc", and "abc".

Example 2:

Input:

s = "aba"

Output:

6

Explanation:

The 6 distinct subsequences are "a", "b", "ab", "aa", "ba", and "aba".

Example 3:

Input:

s = "aaa"

Output:

3

Explanation:

The 3 distinct subsequences are "a", "aa" and "aaa".

Constraints:

1 <= s.length <= 2000

s

consists of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int distinctSubseqII(char* s) {

}
```

Go:

```
func distinctSubseqII(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
```

```
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Distinct Subsequences II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

```
}
};
```

Java Solution:

```
/**
* Problem: Distinct Subsequences II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int distinctSubseqII(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Distinct Subsequences II
Difficulty: Hard
Tags: string, dp

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Distinct Subsequences II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Distinct Subsequences II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function distinctSubseqII(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Distinct Subsequences II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int DistinctSubseqII(string s) {

}
}
```

C Solution:

```
/*
* Problem: Distinct Subsequences II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int distinctSubseqII(char* s) {

}
```

Go Solution:

```
// Problem: Distinct Subsequences II
// Difficulty: Hard
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
```

```
// Space Complexity: O(n) or O(n * m) for DP table

func distinctSubseqII(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Distinct Subsequences II
// Difficulty: Hard
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
```

```
end
```

Erlang Solution:

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

Racket Solution:

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