Problem 3042: Count Prefix and Suffix Pairs I

Problem Information

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

Problem Description

You are given a

0-indexed

string array

words

.

Let's define a

boolean

function

isPrefixAndSuffix

that takes two strings,

str1

and

str2

:

isPrefixAndSuffix(str1, str2)

returns

true

if

str1

is

both

a

prefix

and a

suffix

of

str2

, and

false

otherwise.

For example,

isPrefixAndSuffix("aba", "ababa")

is

true

because

"aba"

is a prefix of

"ababa"

and also a suffix, but

isPrefixAndSuffix("abc", "abcd")

is

false

.

Return

an integer denoting the

number

of index pairs

(i, j)

such that

i < j

, and

isPrefixAndSuffix(words[i], words[j])

is

true

.

Example 1:

Input:

words = ["a","aba","ababa","aa"]

Output:

4

Explanation:

In this example, the counted index pairs are: i = 0 and j = 1 because isPrefixAndSuffix("a", "aba") is true. i = 0 and j = 2 because isPrefixAndSuffix("a", "ababa") is true. i = 0 and j = 3 because isPrefixAndSuffix("a", "aa") is true. i = 1 and j = 2 because isPrefixAndSuffix("aba", "ababa") is true. Therefore, the answer is 4.

Example 2:

Input:

words = ["pa","papa","ma","mama"]

Output:

2

Explanation:

In this example, the counted index pairs are: i = 0 and j = 1 because isPrefixAndSuffix("pa", "papa") is true. i = 2 and j = 3 because isPrefixAndSuffix("ma", "mama") is true. Therefore, the answer is 2.

Example 3:

Input:

words = ["abab","ab"]

Output:

0

Explanation:

In this example, the only valid index pair is i = 0 and j = 1, and isPrefixAndSuffix("abab", "ab") is false. Therefore, the answer is 0.

Constraints:

1 <= words.length <= 50

1 <= words[i].length <= 10

words[i]

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int countPrefixSuffixPairs(vector<string>& words) {

}
};
```

Java:

```
class Solution {
public int countPrefixSuffixPairs(String[] words) {

}
}
```

Python3:

```
class Solution:
def countPrefixSuffixPairs(self, words: List[str]) -> int:
```

Python:

```
class Solution(object):
def countPrefixSuffixPairs(self, words):
"""
:type words: List[str]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string[]} words
* @return {number}
*/
var countPrefixSuffixPairs = function(words) {

};
```

TypeScript:

```
function countPrefixSuffixPairs(words: string[]): number {

};
```

C#:

```
public class Solution {
public int CountPrefixSuffixPairs(string[] words) {

}
}
```

C:

```
int countPrefixSuffixPairs(char** words, int wordsSize) {

}
```

Go:

```
func countPrefixSuffixPairs(words []string) int {

}
```

Kotlin:

```
class Solution {
fun countPrefixSuffixPairs(words: Array<String>): Int {

}
}
```

Swift:

```
class Solution {
func countPrefixSuffixPairs(_ words: [String]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_prefix_suffix_pairs(words: Vec<String>) -> i32 {

}
}
```

Ruby:

```
# @param {String[]} words
# @return {Integer}
def count_prefix_suffix_pairs(words)

end
```

PHP:

```
class Solution {

/**
* @param String[] $words
* @return Integer
*/
function countPrefixSuffixPairs($words) {

}
}
```

Dart:

```
class Solution {
int countPrefixSuffixPairs(List<String> words) {

}
}
```

Scala:

```
object Solution {
def countPrefixSuffixPairs(words: Array[String]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_prefix_suffix_pairs(words :: [String.t]) :: integer
def count_prefix_suffix_pairs(words) do

end
end
```

Erlang:

```
-spec count_prefix_suffix_pairs(Words :: [unicode:unicode_binary()]) ->
integer().
count_prefix_suffix_pairs(Words) ->
```

```
.
```

Racket:

```
(define/contract (count-prefix-suffix-pairs words)
(-> (listof string?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Prefix and Suffix Pairs I
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int countPrefixSuffixPairs(vector<string>& words) {

}
};
```

Java Solution:

```
/**
* Problem: Count Prefix and Suffix Pairs I
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
class Solution {
public int countPrefixSuffixPairs(String[] words) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Prefix and Suffix Pairs I
Difficulty: Easy
Tags: array, string, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def countPrefixSuffixPairs(self, words: List[str]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countPrefixSuffixPairs(self, words):
"""
:type words: List[str]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count Prefix and Suffix Pairs I
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string[]} words
* @return {number}
*/
var countPrefixSuffixPairs = function(words) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Prefix and Suffix Pairs I
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function countPrefixSuffixPairs(words: string[]): number {

};
```

C# Solution:

```
/*
* Problem: Count Prefix and Suffix Pairs I
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
```

```
public int CountPrefixSuffixPairs(string[] words) {

}
}
```

C Solution:

```
/*
* Problem: Count Prefix and Suffix Pairs I
* Difficulty: Easy
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int countPrefixSuffixPairs(char** words, int wordsSize) {

}
```

Go Solution:

```
// Problem: Count Prefix and Suffix Pairs I
// Difficulty: Easy
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func countPrefixSuffixPairs(words []string) int {

}
```

Kotlin Solution:

```
class Solution {
fun countPrefixSuffixPairs(words: Array<String>): Int {

}
```

```
}
```

Swift Solution:

```
class Solution {
func countPrefixSuffixPairs(_ words: [String]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Prefix and Suffix Pairs I
// Difficulty: Easy
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn count_prefix_suffix_pairs(words: Vec<String>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String[]} words
# @return {Integer}
def count_prefix_suffix_pairs(words)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $words
* @return Integer
```

```
*/
function countPrefixSuffixPairs($words) {

}
}
```

Dart Solution:

```
class Solution {
int countPrefixSuffixPairs(List<String> words) {

}
}
```

Scala Solution:

```
object Solution {
def countPrefixSuffixPairs(words: Array[String]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_prefix_suffix_pairs(words :: [String.t]) :: integer
def count_prefix_suffix_pairs(words) do

end
end
```

Erlang Solution:

```
-spec count_prefix_suffix_pairs(Words :: [unicode:unicode_binary()]) ->
integer().
count_prefix_suffix_pairs(Words) ->
.
```

Racket Solution:

```
(define/contract (count-prefix-suffix-pairs words)
(-> (listof string?) exact-integer?)
)
```