# Problem 1794: Count Pairs of Equal Substrings With Minimum Difference

## Problem Information

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

## Problem Description

You are given two strings

firstString

and

secondString

that are

0-indexed

and consist only of lowercase English letters. Count the number of index quadruples

(i,j,a,b)

that satisfy the following conditions:

0 <= i <= j < firstString.length

0 <= a <= b < secondString.length

The substring of

firstString

that starts at the

i

th

character and ends at the

j

th

character (inclusive) is

equal

to the substring of

secondString

that starts at the

a

th

character and ends at the

b

th

character (inclusive).

j - a

is the

minimum

possible value among all quadruples that satisfy the previous conditions.

Return

the

number

of such quadruples

.

Example 1:

Input:

firstString = "abcd", secondString = "bccda"

Output:

1

Explanation:

The quadruple (0,0,4,4) is the only one that satisfies all the conditions and minimizes j - a.

Example 2:

Input:

firstString = "ab", secondString = "cd"

Output:

0

Explanation:

There are no quadruples satisfying all the conditions.

Constraints:

1 <= firstString.length, secondString.length <= 2 * 10

5

Both strings consist only of lowercase English letters.

## Code Snippets

**C++:**

```
class Solution {
public:
int countQuadruples(string firstString, string secondString) {

}
};
```

**Java:**

```
class Solution {
public int countQuadruples(String firstString, String secondString) {

}
}
```

**Python3:**

```
class Solution:
def countQuadruples(self, firstString: str, secondString: str) -> int:
```

**Python:**

```
class Solution(object):
def countQuadruples(self, firstString, secondString):
"""
:type firstString: str
```

```
:type secondString: str
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {string} firstString
* @param {string} secondString
* @return {number}
*/
var countQuadruples = function(firstString, secondString) {

};
```

**TypeScript:**

```
function countQuadruples(firstString: string, secondString: string): number {

};
```

**C#:**

```
public class Solution {
public int CountQuadruples(string firstString, string secondString) {

}
}
```

**C:**

```
int countQuadruples(char* firstString, char* secondString) {

}
```

**Go:**

```
func countQuadruples(firstString string, secondString string) int {

}
```

**Kotlin:**

```
class Solution {
fun countQuadruples(firstString: String, secondString: String): Int {

}
}
```

**Swift:**

```
class Solution {
func countQuadruples(_ firstString: String, _ secondString: String) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn count_quadruples(first_string: String, second_string: String) -> i32 {

}
}
```

**Ruby:**

```
# @param {String} first_string
# @param {String} second_string
# @return {Integer}
def count_quadruples(first_string, second_string)

end
```

**PHP:**

```
class Solution {

/**
* @param String $firstString
* @param String $secondString
* @return Integer
*/
function countQuadruples($firstString, $secondString) {

}
```

```
}
```

**Dart:**

```
class Solution {
int countQuadruples(String firstString, String secondString) {

}
}
```

**Scala:**

```
object Solution {
def countQuadruples(firstString: String, secondString: String): Int = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec count_quadruples(first_string :: String.t, second_string :: String.t)
:: integer
def count_quadruples(first_string, second_string) do

end
end
```

**Erlang:**

```
-spec count_quadruples(FirstString :: unicode:unicode_binary(), SecondString
:: unicode:unicode_binary()) -> integer().
count_quadruples(FirstString, SecondString) ->
.
```

**Racket:**

```
(define/contract (count-quadruples firstString secondString)
(-> string? string? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Count Pairs of Equal Substrings With Minimum Difference
* Difficulty: Medium
* Tags: string, tree, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
int countQuadruples(string firstString, string secondString) {

}
};
```

### Java Solution:

```
/**
* Problem: Count Pairs of Equal Substrings With Minimum Difference
* Difficulty: Medium
* Tags: string, tree, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int countQuadruples(String firstString, String secondString) {

}
}
```

### Python3 Solution:

```
"""
Problem: Count Pairs of Equal Substrings With Minimum Difference
```

```
Difficulty: Medium
Tags: string, tree, greedy, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
def countQuadruples(self, firstString: str, secondString: str) -> int:
# TODO: Implement optimized solution
pass
```

### Python Solution:

```
class Solution(object):
def countQuadruples(self, firstString, secondString):
"""
:type firstString: str
:type secondString: str
:rtype: int
"""
```

### JavaScript Solution:

```
/**
* Problem: Count Pairs of Equal Substrings With Minimum Difference
* Difficulty: Medium
* Tags: string, tree, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {string} firstString
* @param {string} secondString
* @return {number}
*/
var countQuadruples = function(firstString, secondString) {
```

```
};
```

### TypeScript Solution:

```
/**
* Problem: Count Pairs of Equal Substrings With Minimum Difference
* Difficulty: Medium
* Tags: string, tree, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function countQuadruples(firstString: string, secondString: string): number {

};
```

### C# Solution:

```
/*
* Problem: Count Pairs of Equal Substrings With Minimum Difference
* Difficulty: Medium
* Tags: string, tree, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* 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 CountQuadruples(string firstString, string secondString) {

}
}
```

### C Solution:

```
/*
* Problem: Count Pairs of Equal Substrings With Minimum Difference
```

```
* Difficulty: Medium
* Tags: string, tree, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int countQuadruples(char* firstString, char* secondString) {

}
```

### Go Solution:

```
// Problem: Count Pairs of Equal Substrings With Minimum Difference
// Difficulty: Medium
// Tags: string, tree, greedy, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func countQuadruples(firstString string, secondString string) int {

}
```

### Kotlin Solution:

```
class Solution {
fun countQuadruples(firstString: String, secondString: String): Int {

}
}
```

### Swift Solution:

```
class Solution {
func countQuadruples(_ firstString: String, _ secondString: String) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Count Pairs of Equal Substrings With Minimum Difference
// Difficulty: Medium
// Tags: string, tree, greedy, hash
//
// Approach: String manipulation with hash map or two pointers
// 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_quadruples(first_string: String, second_string: String) -> i32 {

}
}
```

### Ruby Solution:

```
# @param {String} first_string
# @param {String} second_string
# @return {Integer}
def count_quadruples(first_string, second_string)

end
```

### PHP Solution:

```
class Solution {

/**
* @param String $firstString
* @param String $secondString
* @return Integer
*/
function countQuadruples($firstString, $secondString) {

}
}
```

### Dart Solution:

```
class Solution {
int countQuadruples(String firstString, String secondString) {

}
}
```

### Scala Solution:

```
object Solution {
def countQuadruples(firstString: String, secondString: String): Int = {

}
}
```

### Elixir Solution:

```
defmodule Solution do
@spec count_quadruples(first_string :: String.t, second_string :: String.t)
:: integer
def count_quadruples(first_string, second_string) do

end
end
```

### Erlang Solution:

```
-spec count_quadruples(FirstString :: unicode:unicode_binary(), SecondString
:: unicode:unicode_binary()) -> integer().
count_quadruples(FirstString, SecondString) ->
.
```

### Racket Solution:

```
(define/contract (count-quadruples firstString secondString)
(-> string? string? exact-integer?)
)
```