Problem 3146: Permutation Difference between Two Strings

Problem Information

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

Problem Description

You are given two strings

s

and

t

such that every character occurs at most once in

s

and

t

is a permutation of

s

.

The

permutation difference

between

s

and

t

is defined as the

sum

of the absolute difference between the index of the occurrence of each character in

s

and the index of the occurrence of the same character in

t

.

Return the

permutation difference

between

s

and

t

.

Example 1:

Input:

s = "abc", t = "bac"

Output:

2

Explanation:

For

s = "abc"

and

t = "bac"

, the permutation difference of

s

and

t

is equal to the sum of:

The absolute difference between the index of the occurrence of

"a"

in

s

and the index of the occurrence of

"a"

in

t

.

The absolute difference between the index of the occurrence of

"b"

in

s

and the index of the occurrence of

"b"

in

t

.

The absolute difference between the index of the occurrence of

"c"

in

s

and the index of the occurrence of

"c"

in

t

.

That is, the permutation difference between

s

and

t

is equal to

|0 - 1| + |1 - 0| + |2 - 2| = 2

.

Example 2:

Input:

s = "abcde", t = "edbac"

Output:

12

Explanation:

The permutation difference between

s

and

t

is equal to

|0 - 3| + |1 - 2| + |2 - 4| + |3 - 1| + |4 - 0| = 12

.

Constraints:

1 <= s.length <= 26

Each character occurs at most once in

s

.

t

is a permutation of

s

.

s

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int findPermutationDifference(string s, string t) {

}
};
```

Java:

```
class Solution {
public int findPermutationDifference(String s, String t) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function findPermutationDifference(s: string, t: string): number {

};
```

C#:

```
public class Solution {
public int FindPermutationDifference(string s, string t) {
}
}
```

C:

```
int findPermutationDifference(char* s, char* t) {

}
```

Go:

```
func findPermutationDifference(s string, t string) int {

}
```

Kotlin:

```
class Solution {
fun findPermutationDifference(s: String, t: String): Int {

}
}
```

Swift:

```
class Solution {
func findPermutationDifference(_ s: String, _ t: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn find_permutation_difference(s: String, t: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {String} t
# @return {Integer}
def find_permutation_difference(s, t)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param String $t
* @return Integer
*/
function findPermutationDifference($s, $t) {

}
}
```

Dart:

```
class Solution {
int findPermutationDifference(String s, String t) {

}
}
```

Scala:

```
object Solution {
def findPermutationDifference(s: String, t: String): Int = {

}
}
```

Elixir:

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

```
end
end
```

Erlang:

```
-spec find_permutation_difference(S :: unicode:unicode_binary(), T ::
unicode:unicode_binary()) -> integer().
find_permutation_difference(S, T) ->
.
```

Racket:

```
(define/contract (find-permutation-difference s t)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Permutation Difference between Two Strings
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int findPermutationDifference(string s, string t) {

}
};
```

Java Solution:

```
/**
* Problem: Permutation Difference between Two Strings
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int findPermutationDifference(String s, String t) {

}
}
```

Python3 Solution:

```
"""
Problem: Permutation Difference between Two Strings
Difficulty: Easy
Tags: string, hash

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

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Permutation Difference between Two Strings
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} s
* @param {string} t
* @return {number}
*/
var findPermutationDifference = function(s, t) {

};
```

TypeScript Solution:

```
/**
* Problem: Permutation Difference between Two Strings
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function findPermutationDifference(s: string, t: string): number {

};
```

C# Solution:

```
/*
* Problem: Permutation Difference between Two Strings
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int FindPermutationDifference(string s, string t) {

}
}
```

C Solution:

```
/*
* Problem: Permutation Difference between Two Strings
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int findPermutationDifference(char* s, char* t) {

}
```

Go Solution:

```
// Problem: Permutation Difference between Two Strings
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func findPermutationDifference(s string, t string) int {
}
```

Kotlin Solution:

```
class Solution {
fun findPermutationDifference(s: String, t: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func findPermutationDifference(_ s: String, _ t: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Permutation Difference between Two Strings
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn find_permutation_difference(s: String, t: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {String} t
# @return {Integer}
def find_permutation_difference(s, t)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param String $t
* @return Integer
*/
function findPermutationDifference($s, $t) {

}
}
```

Dart Solution:

```
class Solution {
int findPermutationDifference(String s, String t) {

}
}
```

Scala Solution:

```
object Solution {
def findPermutationDifference(s: String, t: String): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec find_permutation_difference(S :: unicode:unicode_binary(), T ::
unicode:unicode_binary()) -> integer().
find_permutation_difference(S, T) ->
.
```

Racket Solution:

```
(define/contract (find-permutation-difference s t)
(-> string? string? exact-integer?)
)
```