

Problem 3043: Find the Length of the Longest Common Prefix

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given two arrays with

positive

integers

arr1

and

arr2

.

A

prefix

of a positive integer is an integer formed by one or more of its digits, starting from its

leftmost

digit. For example,

is a prefix of the integer

12345

, while

234

is

not

.

A

common prefix

of two integers

a

and

b

is an integer

c

, such that

c

is a prefix of both

a

and

b

. For example,

5655359

and

56554

have common prefixes

565

and

5655

while

1223

and

43456

do not

have a common prefix.

You need to find the length of the

longest common prefix

between all pairs of integers

(x, y)

such that

x

belongs to

arr1

and

y

belongs to

arr2

.

Return

the length of the

longest

common prefix among all pairs

.

If no common prefix exists among them

,

return

0

Example 1:

Input:

arr1 = [1,10,100], arr2 = [1000]

Output:

3

Explanation:

There are 3 pairs (arr1[i], arr2[j]): - The longest common prefix of (1, 1000) is 1. - The longest common prefix of (10, 1000) is 10. - The longest common prefix of (100, 1000) is 100. The longest common prefix is 100 with a length of 3.

Example 2:

Input:

arr1 = [1,2,3], arr2 = [4,4,4]

Output:

0

Explanation:

There exists no common prefix for any pair (arr1[i], arr2[j]), hence we return 0. Note that common prefixes between elements of the same array do not count.

Constraints:

$1 \leq \text{arr1.length}, \text{arr2.length} \leq 5 * 10$

4

$1 \leq \text{arr1}[i], \text{arr2}[i] \leq 10$

Code Snippets

C++:

```
class Solution {
public:
    int longestCommonPrefix(vector<int>& arr1, vector<int>& arr2) {
        }
    };
}
```

Java:

```
class Solution {
    public int longestCommonPrefix(int[] arr1, int[] arr2) {
        }
    }
}
```

Python3:

```
class Solution:
    def longestCommonPrefix(self, arr1: List[int], arr2: List[int]) -> int:
```

Python:

```
class Solution(object):
    def longestCommonPrefix(self, arr1, arr2):
        """
        :type arr1: List[int]
        :type arr2: List[int]
        :rtype: int
        """

```

JavaScript:

```
/**
 * @param {number[]} arr1
 * @param {number[]} arr2
```

```
* @return {number}
*/
var longestCommonPrefix = function(arr1, arr2) {
};
```

TypeScript:

```
function longestCommonPrefix(arr1: number[], arr2: number[]): number {
};
```

C#:

```
public class Solution {
public int LongestCommonPrefix(int[] arr1, int[] arr2) {
}
```

C:

```
int longestCommonPrefix(int* arr1, int arr1Size, int* arr2, int arr2Size) {
}
```

Go:

```
func longestCommonPrefix(arr1 []int, arr2 []int) int {
}
```

Kotlin:

```
class Solution {
fun longestCommonPrefix(arr1: IntArray, arr2: IntArray): Int {
}
```

Swift:

```
class Solution {  
    func longestCommonPrefix(_ arr1: [Int], _ arr2: [Int]) -> Int {  
        }  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn longest_common_prefix(arr1: Vec<i32>, arr2: Vec<i32>) -> i32 {  
        }  
    }  
}
```

Ruby:

```
# @param {Integer[]} arr1  
# @param {Integer[]} arr2  
# @return {Integer}  
def longest_common_prefix(arr1, arr2)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer[] $arr1  
     * @param Integer[] $arr2  
     * @return Integer  
     */  
    function longestCommonPrefix($arr1, $arr2) {  
  
    }  
}
```

Dart:

```
class Solution {  
    int longestCommonPrefix(List<int> arr1, List<int> arr2) {  
    }  
}
```

```
}
```

Scala:

```
object Solution {  
    def longestCommonPrefix(arr1: Array[Int], arr2: Array[Int]): Int = {  
        }  
        }  
}
```

Elixir:

```
defmodule Solution do  
    @spec longest_common_prefix(arr1 :: [integer], arr2 :: [integer]) :: integer  
    def longest_common_prefix(arr1, arr2) do  
  
    end  
    end
```

Erlang:

```
-spec longest_common_prefix(Arr1 :: [integer()], Arr2 :: [integer()]) ->  
integer().  
longest_common_prefix(Arr1, Arr2) ->  
.
```

Racket:

```
(define/contract (longest-common-prefix arr1 arr2)  
  (-> (listof exact-integer?) (listof exact-integer?) exact-integer?)  
)
```

Solutions

C++ Solution:

```
/*  
* Problem: Find the Length of the Longest Common Prefix  
* Difficulty: Medium  
* 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 longestCommonPrefix(vector<int>& arr1, vector<int>& arr2) {

}
};


```

Java Solution:

```

/**
* Problem: Find the Length of the Longest Common Prefix
* Difficulty: Medium
* 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 longestCommonPrefix(int[] arr1, int[] arr2) {

}
}


```

Python3 Solution:

```

"""
Problem: Find the Length of the Longest Common Prefix
Difficulty: Medium
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 longestCommonPrefix(self, arr1: List[int], arr2: List[int]) -> int:
        # TODO: Implement optimized solution
        pass
```

Python Solution:

```
class Solution(object):
    def longestCommonPrefix(self, arr1, arr2):
        """
        :type arr1: List[int]
        :type arr2: List[int]
        :rtype: int
        """
```

JavaScript Solution:

```
/**
 * Problem: Find the Length of the Longest Common Prefix
 * Difficulty: Medium
 * 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 {number[]} arr1
 * @param {number[]} arr2
 * @return {number}
 */
var longestCommonPrefix = function(arr1, arr2) {

};
```

TypeScript Solution:

```

/**
 * Problem: Find the Length of the Longest Common Prefix
 * Difficulty: Medium
 * 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 longestCommonPrefix(arr1: number[], arr2: number[]): number {
}

```

C# Solution:

```

/*
 * Problem: Find the Length of the Longest Common Prefix
 * Difficulty: Medium
 * 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 LongestCommonPrefix(int[] arr1, int[] arr2) {
}
}

```

C Solution:

```

/*
 * Problem: Find the Length of the Longest Common Prefix
 * Difficulty: Medium
 * 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

```

```
*/\n\nint longestCommonPrefix(int* arr1, int arr1Size, int* arr2, int arr2Size) {\n\n}
```

Go Solution:

```
// Problem: Find the Length of the Longest Common Prefix\n// Difficulty: Medium\n// Tags: array, string, hash\n//\n// Approach: Use two pointers or sliding window technique\n// Time Complexity: O(n) or O(n log n)\n// Space Complexity: O(n) for hash map\n\nfunc longestCommonPrefix(arr1 []int, arr2 []int) int {\n\n}
```

Kotlin Solution:

```
class Solution {\n    fun longestCommonPrefix(arr1: IntArray, arr2: IntArray): Int {\n        \n    }\n}
```

Swift Solution:

```
class Solution {\n    func longestCommonPrefix(_ arr1: [Int], _ arr2: [Int]) -> Int {\n        \n    }\n}
```

Rust Solution:

```
// Problem: Find the Length of the Longest Common Prefix\n// Difficulty: Medium\n// 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 longest_common_prefix(arr1: Vec<i32>, arr2: Vec<i32>) -> i32 {

}
}

```

Ruby Solution:

```

# @param {Integer[]} arr1
# @param {Integer[]} arr2
# @return {Integer}
def longest_common_prefix(arr1, arr2)

end

```

PHP Solution:

```

class Solution {

/**
 * @param Integer[] $arr1
 * @param Integer[] $arr2
 * @return Integer
 */
function longestCommonPrefix($arr1, $arr2) {

}
}

```

Dart Solution:

```

class Solution {
int longestCommonPrefix(List<int> arr1, List<int> arr2) {

}
}

```

Scala Solution:

```
object Solution {  
    def longestCommonPrefix(arr1: Array[Int], arr2: Array[Int]): Int = {  
  
    }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec longest_common_prefix(arr1 :: [integer], arr2 :: [integer]) :: integer  
  def longest_common_prefix(arr1, arr2) do  
  
  end  
end
```

Erlang Solution:

```
-spec longest_common_prefix(Arr1 :: [integer()], Arr2 :: [integer()]) ->  
integer().  
longest_common_prefix(Arr1, Arr2) ->  
.
```

Racket Solution:

```
(define/contract (longest-common-prefix arr1 arr2)  
  (-> (listof exact-integer?) (listof exact-integer?) exact-integer?)  
)
```