Problem 1385: Find the Distance Value Between Two Arrays

Problem Information

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

Problem Description

Given two integer arrays

arr1

and

arr2

, and the integer

d

,

return the distance value between the two arrays

.

The distance value is defined as the number of elements

arr1[i]

such that there is not any element

arr2[j]

where

|arr1[i]-arr2[j]| <= d

.

Example 1:

Input:

arr1 = [4,5,8], arr2 = [10,9,1,8], d = 2

Output:

2

Explanation:

For arr1[0]=4 we have: |4-10|=6 > d=2 |4-9|=5 > d=2 |4-1|=3 > d=2 |4-8|=4 > d=2 For arr1[1]=5 we have: |5-10|=5 > d=2 |5-9|=4 > d=2 |5-1|=4 > d=2 |5-8|=3 > d=2 For arr1[2]=8 we have:

|8-10|=2 <= d=2

|8-9|=1 <= d=2

|8-1|=7 > d=2

|8-8|=0 <= d=2

Example 2:

Input:

arr1 = [1,4,2,3], arr2 = [-4,-3,6,10,20,30], d = 3

Output:

2

Example 3:

Input:

arr1 = [2,1,100,3], arr2 = [-5,-2,10,-3,7], d = 6

Output:

1

Constraints:

1 <= arr1.length, arr2.length <= 500

-1000 <= arr1[i], arr2[j] <= 1000

0 <= d <= 100

Code Snippets

C++:

```
class Solution {
public:
int findTheDistanceValue(vector<int>& arr1, vector<int>& arr2, int d) {

}
};
```

Java:

```
class Solution {
public int findTheDistanceValue(int[] arr1, int[] arr2, int d) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} arr1
* @param {number[]} arr2
* @param {number} d
* @return {number}
*/
var findTheDistanceValue = function(arr1, arr2, d) {

};
```

TypeScript:

```
function findTheDistanceValue(arr1: number[], arr2: number[], d: number):
number {

};
```

C#:

```
public class Solution {
public int FindTheDistanceValue(int[] arr1, int[] arr2, int d) {

}
}
```

C:

```
int findTheDistanceValue(int* arr1, int arr1Size, int* arr2, int arr2Size,
int d) {

}
```

Go:

```
func findTheDistanceValue(arr1 []int, arr2 []int, d int) int {

}
```

Kotlin:

```
class Solution {
fun findTheDistanceValue(arr1: IntArray, arr2: IntArray, d: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

```
end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int findTheDistanceValue(List<int> arr1, List<int> arr2, int d) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
```

```
end
```

Erlang:

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

Racket:

```
(define/contract (find-the-distance-value arr1 arr2 d)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?
exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Distance Value Between Two Arrays
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int findTheDistanceValue(vector<int>& arr1, vector<int>& arr2, int d) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Distance Value Between Two Arrays
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int findTheDistanceValue(int[] arr1, int[] arr2, int d) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Distance Value Between Two Arrays
Difficulty: Easy
Tags: array, sort, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def findTheDistanceValue(self, arr1: List[int], arr2: List[int], d: int) ->
int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

```
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Distance Value Between Two Arrays
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} arr1
* @param {number[]} arr2
* @param {number} d
* @return {number}
*/
var findTheDistanceValue = function(arr1, arr2, d) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Distance Value Between Two Arrays
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function findTheDistanceValue(arr1: number[], arr2: number[], d: number):
number {

};
```

C# Solution:

```
/*
* Problem: Find the Distance Value Between Two Arrays
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int FindTheDistanceValue(int[] arr1, int[] arr2, int d) {

}
}
```

C Solution:

```
/*
* Problem: Find the Distance Value Between Two Arrays
* Difficulty: Easy
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int findTheDistanceValue(int* arr1, int arr1Size, int* arr2, int arr2Size,
int d) {

}
```

Go Solution:

```
// Problem: Find the Distance Value Between Two Arrays
// Difficulty: Easy
// Tags: array, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func findTheDistanceValue(arr1 []int, arr2 []int, d int) int {

}
```

Kotlin Solution:

```
class Solution {
fun findTheDistanceValue(arr1: IntArray, arr2: IntArray, d: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Distance Value Between Two Arrays
// Difficulty: Easy
// Tags: array, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn find_the_distance_value(arr1: Vec<i32>, arr2: Vec<i32>, d: i32) -> i32
{

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int findTheDistanceValue(List<int> arr1, List<int> arr2, int d) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (find-the-distance-value arr1 arr2 d)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?
exact-integer?)
)
```