Problem 2121: Intervals Between Identical Elements

Problem Information

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

Problem Description

You are given a

0-indexed

array of

n

integers

arr

.

The

interval

between two elements in

arr

is defined as the

absolute difference

between their indices. More formally, the

interval

between

arr[i]

and

arr[j]

is

|i - j|

.

Return

an array

intervals

of length

n

where

intervals[i]

is

the sum of intervals

between

arr[i]

and each element in

arr

with the same value as

arr[i]

.

Note:

|x|

is the absolute value of

x

.

Example 1:

Input:

arr = [2,1,3,1,2,3,3]

Output:

[4,2,7,2,4,4,5]

Explanation:

- Index 0: Another 2 is found at index 4. |0 - 4| = 4 - Index 1: Another 1 is found at index 3. |1 - 3| = 2 - Index 2: Two more 3s are found at indices 5 and 6. |2 - 5| + |2 - 6| = 7 - Index 3: Another 1 is found at index 1. |3 - 1| = 2 - Index 4: Another 2 is found at index 0. |4 - 0| = 4 - Index 5: Two more 3s are found at indices 2 and 6. |5 - 2| + |5 - 6| = 4 - Index 6: Two more 3s are found at indices 2 and 5. |6 - 2| + |6 - 5| = 5

Example 2:

Input:

arr = [10,5,10,10]

Output:

[5,0,3,4]

Explanation:

- Index 0: Two more 10s are found at indices 2 and 3. |0 - 2| + |0 - 3| = 5 - Index 1: There is only one 5 in the array, so its sum of intervals to identical elements is 0. - Index 2: Two more 10s are found at indices 0 and 3. |2 - 0| + |2 - 3| = 3 - Index 3: Two more 10s are found at indices 0 and 2. |3 - 0| + |3 - 2| = 4

Constraints:

n == arr.length

1 <= n <= 10

5

1 <= arr[i] <= 10

5

Note:

This question is the same as

2615: Sum of Distances.

Code Snippets

C++:

```
class Solution {
public:
vector<long long> getDistances(vector<int>& arr) {

}
};
```

Java:

```
class Solution {
public long[] getDistances(int[] arr) {

}
}
```

Python3:

```
class Solution:
def getDistances(self, arr: List[int]) -> List[int]:
```

Python:

```
class Solution(object):
def getDistances(self, arr):
"""
:type arr: List[int]
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number[]} arr
* @return {number[]}
*/
var getDistances = function(arr) {

};
```

TypeScript:

```
function getDistances(arr: number[]): number[] {

};
```

C#:

```
public class Solution {
public long[] GetDistances(int[] arr) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
long long* getDistances(int* arr, int arrSize, int* returnSize) {

}
```

Go:

```
func getDistances(arr []int) []int64 {

}
```

Kotlin:

```
class Solution {
fun getDistances(arr: IntArray): LongArray {

}
}
```

Swift:

```
class Solution {
func getDistances(_ arr: [Int]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn get_distances(arr: Vec<i32>) -> Vec<i64> {

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @return {Integer[]}
def get_distances(arr)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer[]
*/
function getDistances($arr) {

}
}
```

Dart:

```
class Solution {
List<int> getDistances(List<int> arr) {

}
}
```

Scala:

```
object Solution {
def getDistances(arr: Array[Int]): Array[Long] = {

}
```

```
}
```

Elixir:

```
defmodule Solution do
@spec get_distances(arr :: [integer]) :: [integer]
def get_distances(arr) do

end
end
```

Erlang:

```
-spec get_distances(Arr :: [integer()]) -> [integer()].
get_distances(Arr) ->
.
```

Racket:

```
(define/contract (get-distances arr)
(-> (listof exact-integer?) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Intervals Between Identical Elements
* Difficulty: Medium
* Tags: array, 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:
vector<long long> getDistances(vector<int>& arr) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Intervals Between Identical Elements
* Difficulty: Medium
* Tags: array, 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 long[] getDistances(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Intervals Between Identical Elements
Difficulty: Medium
Tags: array, 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 getDistances(self, arr: List[int]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getDistances(self, arr):
"""
:type arr: List[int]
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Intervals Between Identical Elements
* Difficulty: Medium
* Tags: array, 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[]} arr
* @return {number[]}
*/
var getDistances = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Intervals Between Identical Elements
* Difficulty: Medium
* Tags: array, 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 getDistances(arr: number[]): number[] {

};
```

C# Solution:

```
/*
* Problem: Intervals Between Identical Elements
* Difficulty: Medium
* Tags: array, 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 long[] GetDistances(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Intervals Between Identical Elements
* Difficulty: Medium
* Tags: array, 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
long long* getDistances(int* arr, int arrSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Intervals Between Identical Elements
// Difficulty: Medium
// Tags: array, 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 getDistances(arr []int) []int64 {

}
```

Kotlin Solution:

```
class Solution {
fun getDistances(arr: IntArray): LongArray {

}
}
```

Swift Solution:

```
class Solution {
func getDistances(_ arr: [Int]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Intervals Between Identical Elements
// Difficulty: Medium
// Tags: array, 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 get_distances(arr: Vec<i32>) -> Vec<i64> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @return {Integer[]}
def get_distances(arr)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer[]
*/
function getDistances($arr) {

}
}
```

Dart Solution:

```
class Solution {
List<int> getDistances(List<int> arr) {

}
}
```

Scala Solution:

```
object Solution {
def getDistances(arr: Array[Int]): Array[Long] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_distances(arr :: [integer]) :: [integer]
def get_distances(arr) do
```

```
end
end
```

Erlang Solution:

```
-spec get_distances(Arr :: [integer()]) -> [integer()].
get_distances(Arr) ->
.
```

Racket Solution:

```
(define/contract (get-distances arr)
(-> (listof exact-integer?) (listof exact-integer?))
)
```