Problem 1906: Minimum Absolute Difference Queries

Problem Information

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

Problem Description

The

minimum absolute difference

of an array

a

is defined as the

minimum value

of

|a[i] - a[j]|

, where

0 <= i < j < a.length

and

a[i] != a[j]

. If all elements of

a

are the

same

, the minimum absolute difference is

-1

.

For example, the minimum absolute difference of the array

[5,

2

,

3

,7,2]

is

|2 - 3| = 1

. Note that it is not

0

because

a[i]

and

a[j]

must be different.

You are given an integer array

nums

and the array

queries

where

queries[i] = [l

i

, r

i

]

. For each query

i

, compute the

minimum absolute difference

of the

subarray

nums[l

i

...r

i

]

containing the elements of

nums

between the

0-based

indices

l

i

and

r

i

(

inclusive

).

Return

an

array

ans

where

ans[i]

is the answer to the

i

th

query

.

A

subarray

is a contiguous sequence of elements in an array.

The value of

|x|

is defined as:

x

if

x >= 0

.

-x

if

x < 0

.

Example 1:

Input:

nums = [1,3,4,8], queries = [[0,1],[1,2],[2,3],[0,3]]

Output:

[2,1,4,1]

Explanation:

The queries are processed as follows: - queries[0] = [0,1]: The subarray is [

1

,

3

] and the minimum absolute difference is |1-3| = 2. - queries[1] = [1,2]: The subarray is [

3

,

4

] and the minimum absolute difference is |3-4| = 1. - queries[2] = [2,3]: The subarray is [

4

,

8

] and the minimum absolute difference is |4-8| = 4. - queries[3] = [0,3]: The subarray is [1,

3

,

4

,8] and the minimum absolute difference is |3-4| = 1.

Example 2:

Input:

nums = [4,5,2,2,7,10], queries = [[2,3],[0,2],[0,5],[3,5]]

Output:

[-1,1,1,3]

Explanation:

The queries are processed as follows: - queries[0] = [2,3]: The subarray is [2,2] and the minimum absolute difference is -1 because all the elements are the same. - queries[1] = [0,2]: The subarray is [

4

,

5

,2] and the minimum absolute difference is |4-5| = 1. - queries[2] = [0,5]: The subarray is [

4

,

5

,2,2,7,10] and the minimum absolute difference is |4-5| = 1. - queries[3] = [3,5]: The subarray is [2,

7

,

10

] and the minimum absolute difference is |7-10| = 3.

Constraints:

2 <= nums.length <= 10

5

1 <= nums[i] <= 100

1 <= queries.length <= 2 * 10

4

0 <= l

i

< r

i

< nums.length

Code Snippets

C++:

```
class Solution {
public:
vector<int> minDifference(vector<int>& nums, vector<vector<int>>& queries) {

}
};
```

Java:

```
class Solution {
public int[] minDifference(int[] nums, int[][] queries) {

}
}
```

Python3:

```
class Solution:
def minDifference(self, nums: List[int], queries: List[List[int]]) ->
List[int]:
```

Python:

```
class Solution(object):
def minDifference(self, nums, queries):
"""
:type nums: List[int]
:type queries: List[List[int]]
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @param {number[][]} queries
* @return {number[]}
*/
var minDifference = function(nums, queries) {

};
```

TypeScript:

```
function minDifference(nums: number[], queries: number[][]): number[] {

};
```

C#:

```
public class Solution {
public int[] MinDifference(int[] nums, int[][] queries) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* minDifference(int* nums, int numsSize, int** queries, int queriesSize,
int* queriesColSize, int* returnSize) {

}
```

Go:

```
func minDifference(nums []int, queries [][]int) []int {

}
```

Kotlin:

```
class Solution {
fun minDifference(nums: IntArray, queries: Array<IntArray>): IntArray {

}
}
```

Swift:

```
class Solution {
func minDifference(_ nums: [Int], _ queries: [[Int]]) -> [Int] {
}
}
```

Rust:

```
impl Solution {
pub fn min_difference(nums: Vec<i32>, queries: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer[][]} queries
# @return {Integer[]}
def min_difference(nums, queries)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer[][] $queries
* @return Integer[]
*/
function minDifference($nums, $queries) {

}
}
```

Dart:

```
class Solution {
List<int> minDifference(List<int> nums, List<List<int>> queries) {

}
}
```

Scala:

```
object Solution {
def minDifference(nums: Array[Int], queries: Array[Array[Int]]): Array[Int] =
{

}
}
```

Elixir:

```
defmodule Solution do
@spec min_difference(nums :: [integer], queries :: [[integer]]) :: [integer]
def min_difference(nums, queries) do

end
end
```

Erlang:

```
-spec min_difference(Nums :: [integer()], Queries :: [[integer()]]) ->
[integer()].
min_difference(Nums, Queries) ->
.
```

Racket:

```
(define/contract (min-difference nums queries)
(-> (listof exact-integer?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Absolute Difference Queries
* 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<int> minDifference(vector<int>& nums, vector<vector<int>>& queries) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Absolute Difference Queries
* 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 int[] minDifference(int[] nums, int[][] queries) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Absolute Difference Queries
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 minDifference(self, nums: List[int], queries: List[List[int]]) ->
List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minDifference(self, nums, queries):
"""
:type nums: List[int]
:type queries: List[List[int]]
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Absolute Difference Queries
* 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[]} nums
* @param {number[][]} queries
* @return {number[]}
*/
var minDifference = function(nums, queries) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Absolute Difference Queries
* 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 minDifference(nums: number[], queries: number[][]): number[] {

};
```

C# Solution:

```
/*
* Problem: Minimum Absolute Difference Queries
* 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 int[] MinDifference(int[] nums, int[][] queries) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Absolute Difference Queries
* 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().
*/
int* minDifference(int* nums, int numsSize, int** queries, int queriesSize,
int* queriesColSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Minimum Absolute Difference Queries
// 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 minDifference(nums []int, queries [][]int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun minDifference(nums: IntArray, queries: Array<IntArray>): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func minDifference(_ nums: [Int], _ queries: [[Int]]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Minimum Absolute Difference Queries
// 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 min_difference(nums: Vec<i32>, queries: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer[][]} queries
# @return {Integer[]}
def min_difference(nums, queries)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer[][] $queries
* @return Integer[]
*/
function minDifference($nums, $queries) {

}
}
```

Dart Solution:

```
class Solution {
List<int> minDifference(List<int> nums, List<List<int>> queries) {
}
}
```

Scala Solution:

```
object Solution {
def minDifference(nums: Array[Int], queries: Array[Array[Int]]): Array[Int] =
{

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_difference(nums :: [integer], queries :: [[integer]]) :: [integer]
def min_difference(nums, queries) do

end
end
```

Erlang Solution:

```
-spec min_difference(Nums :: [integer()], Queries :: [[integer()]]) ->
[integer()].
min_difference(Nums, Queries) ->
.
```

Racket Solution:

```
(define/contract (min-difference nums queries)
(-> (listof exact-integer?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```