Problem 1157: Online Majority Element In Subarray

Problem Information

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

Problem Description

Design a data structure that efficiently finds the

majority element

of a given subarray.

The

majority element

of a subarray is an element that occurs

threshold

times or more in the subarray.

Implementing the

MajorityChecker

class:

MajorityChecker(int[] arr)

Initializes the instance of the class with the given array

arr

.

int query(int left, int right, int threshold)

returns the element in the subarray

arr[left...right]

that occurs at least

threshold

times, or

-1

if no such element exists.

Example 1:

Input

["MajorityChecker", "query", "query", "query"] [[[1, 1, 2, 2, 1, 1]], [0, 5, 4], [0, 3, 3], [2, 3, 2]]

Output

[null, 1, -1, 2]

Explanation

MajorityChecker majorityChecker = new MajorityChecker([1, 1, 2, 2, 1, 1]);
majorityChecker.query(0, 5, 4); // return 1 majorityChecker.query(0, 3, 3); // return -1
majorityChecker.query(2, 3, 2); // return 2

Constraints:

1 <= arr.length <= 2 * 10

4

1 <= arr[i] <= 2 * 10

4

0 <= left <= right < arr.length

threshold <= right - left + 1

2 * threshold > right - left + 1

At most

10

4

calls will be made to

query

.

Code Snippets

C++:

```
class MajorityChecker {
public:
MajorityChecker(vector<int>& arr) {

}

int query(int left, int right, int threshold) {

}
};
```

```
/**
* Your MajorityChecker object will be instantiated and called as such:
* MajorityChecker* obj = new MajorityChecker(arr);
* int param_1 = obj->query(left,right,threshold);
*/
```

Java:

```
class MajorityChecker {

public MajorityChecker(int[] arr) {

}

public int query(int left, int right, int threshold) {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* MajorityChecker obj = new MajorityChecker(arr);
* int param_1 = obj.query(left,right,threshold);
*/
```

Python3:

```
class MajorityChecker:

def __init__(self, arr: List[int]):


def query(self, left: int, right: int, threshold: int) -> int:



# Your MajorityChecker object will be instantiated and called as such:
# obj = MajorityChecker(arr)
# param_1 = obj.query(left,right,threshold)
```

Python:

```
class MajorityChecker(object):

def __init__(self, arr):
"""
:type arr: List[int]
"""


def query(self, left, right, threshold):
"""
:type left: int
:type right: int
:type threshold: int
:rtype: int
"""




# Your MajorityChecker object will be instantiated and called as such:
# obj = MajorityChecker(arr)
# param_1 = obj.query(left,right,threshold)
```

JavaScript:

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

};

/**
* @param {number} left
* @param {number} right
* @param {number} threshold
* @return {number}
*/
MajorityChecker.prototype.query = function(left, right, threshold) {

};

/**
* Your MajorityChecker object will be instantiated and called as such:
```

```
* var obj = new MajorityChecker(arr)
* var param_1 = obj.query(left,right,threshold)
*/
```

TypeScript:

```
class MajorityChecker {
constructor(arr: number[]) {

}

query(left: number, right: number, threshold: number): number {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* var obj = new MajorityChecker(arr)
* var param_1 = obj.query(left,right,threshold)
*/
```

C#:

```
public class MajorityChecker {

public MajorityChecker(int[] arr) {

}

public int Query(int left, int right, int threshold) {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* MajorityChecker obj = new MajorityChecker(arr);
* int param_1 = obj.Query(left,right,threshold);
*/
```

C:

```
typedef struct {

} MajorityChecker;


MajorityChecker* majorityCheckerCreate(int* arr, int arrSize) {

}

int majorityCheckerQuery(MajorityChecker* obj, int left, int right, int
threshold) {

}

void majorityCheckerFree(MajorityChecker* obj) {

}

/**
* Your MajorityChecker struct will be instantiated and called as such:
* MajorityChecker* obj = majorityCheckerCreate(arr, arrSize);
* int param_1 = majorityCheckerQuery(obj, left, right, threshold);

* majorityCheckerFree(obj);
*/
```

Go:

```
type MajorityChecker struct {

}


func Constructor(arr []int) MajorityChecker {

}


func (this *MajorityChecker) Query(left int, right int, threshold int) int {
```

```
}


/**
* Your MajorityChecker object will be instantiated and called as such:
* obj := Constructor(arr);
* param_1 := obj.Query(left,right,threshold);
*/
```

Kotlin:

```
class MajorityChecker(arr: IntArray) {

fun query(left: Int, right: Int, threshold: Int): Int {

}

}

/**
* Your MajorityChecker object will be instantiated and called as such:
* var obj = MajorityChecker(arr)
* var param_1 = obj.query(left,right,threshold)
*/
```

Swift:

```

class MajorityChecker {

init(_ arr: [Int]) {

}

func query(_ left: Int, _ right: Int, _ threshold: Int) -> Int {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* let obj = MajorityChecker(arr)
```

```
* let ret_1: Int = obj.query(left, right, threshold)
*/
```

Rust:

```
struct MajorityChecker {

}


/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl MajorityChecker {

fn new(arr: Vec<i32>) -> Self {

}

fn query(&self, left: i32, right: i32, threshold: i32) -> i32 {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* let obj = MajorityChecker::new(arr);
* let ret_1: i32 = obj.query(left, right, threshold);
*/
```

Ruby:

```
class MajorityChecker

=begin
:type arr: Integer[]
=end
def initialize(arr)

end
```

```
=begin
:type left: Integer
:type right: Integer
:type threshold: Integer
:rtype: Integer
=end
def query(left, right, threshold)

end



end

# Your MajorityChecker object will be instantiated and called as such:
# obj = MajorityChecker.new(arr)
# param_1 = obj.query(left, right, threshold)
```

PHP:

```
class MajorityChecker {
/**
* @param Integer[] $arr
*/
function __construct($arr) {

}

/**
* @param Integer $left
* @param Integer $right
* @param Integer $threshold
* @return Integer
*/
function query($left, $right, $threshold) {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* $obj = MajorityChecker($arr);
```

```
* $ret_1 = $obj->query($left, $right, $threshold);
*/
```

Dart:

```
class MajorityChecker {

MajorityChecker(List<int> arr) {

}

int query(int left, int right, int threshold) {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* MajorityChecker obj = MajorityChecker(arr);
* int param1 = obj.query(left,right,threshold);
*/
```

Scala:

```
class MajorityChecker(_arr: Array[Int]) {

def query(left: Int, right: Int, threshold: Int): Int = {

}

}

/**
* Your MajorityChecker object will be instantiated and called as such:
* val obj = new MajorityChecker(arr)
* val param_1 = obj.query(left,right,threshold)
*/
```

Elixir:

```
defmodule MajorityChecker do
@spec init_(arr :: [integer]) :: any
```

```
def init_(arr) do

end

@spec query(left :: integer, right :: integer, threshold :: integer) ::
integer
def query(left, right, threshold) do

end
end

# Your functions will be called as such:
# MajorityChecker.init_(arr)
# param_1 = MajorityChecker.query(left, right, threshold)

# MajorityChecker.init_ will be called before every test case, in which you
can do some necessary initializations.
```

Erlang:

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

-spec majority_checker_query(Left :: integer(), Right :: integer(), Threshold
:: integer()) -> integer().
majority_checker_query(Left, Right, Threshold) ->
.


%% Your functions will be called as such:
%% majority_checker_init_(Arr),
%% Param_1 = majority_checker_query(Left, Right, Threshold),

%% majority_checker_init_ will be called before every test case, in which you
can do some necessary initializations.
```

Racket:

```
(define majority-checker%
(class object%
(super-new)
```

```
; arr : (listof exact-integer?)
(init-field
arr)

; query : exact-integer? exact-integer? exact-integer? -> exact-integer?
(define/public (query left right threshold)
)))

;; Your majority-checker% object will be instantiated and called as such:
;; (define obj (new majority-checker% [arr arr]))
;; (define param_1 (send obj query left right threshold))
```

Solutions

C++ Solution:

```
/*
* Problem: Online Majority Element In Subarray
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class MajorityChecker {
public:
MajorityChecker(vector<int>& arr) {

}

int query(int left, int right, int threshold) {

}
};

/**
* Your MajorityChecker object will be instantiated and called as such:
```

```
* MajorityChecker* obj = new MajorityChecker(arr);
* int param_1 = obj->query(left,right,threshold);
*/
```

Java Solution:

```
/**
* Problem: Online Majority Element In Subarray
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class MajorityChecker {

public MajorityChecker(int[] arr) {

}

public int query(int left, int right, int threshold) {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* MajorityChecker obj = new MajorityChecker(arr);
* int param_1 = obj.query(left,right,threshold);
*/
```

Python3 Solution:

```
"""
Problem: Online Majority Element In Subarray
Difficulty: Hard
Tags: array, tree, search

Approach: Use two pointers or sliding window technique
```

```
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class MajorityChecker:

def __init__(self, arr: List[int]):


def query(self, left: int, right: int, threshold: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class MajorityChecker(object):

def __init__(self, arr):
"""
:type arr: List[int]
"""


def query(self, left, right, threshold):
"""
:type left: int
:type right: int
:type threshold: int
:rtype: int
"""




# Your MajorityChecker object will be instantiated and called as such:
# obj = MajorityChecker(arr)
# param_1 = obj.query(left,right,threshold)
```

JavaScript Solution:

```
/**
* Problem: Online Majority Element In Subarray
```

```
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {number[]} arr
*/
var MajorityChecker = function(arr) {

};

/**
* @param {number} left
* @param {number} right
* @param {number} threshold
* @return {number}
*/
MajorityChecker.prototype.query = function(left, right, threshold) {

};

/**
* Your MajorityChecker object will be instantiated and called as such:
* var obj = new MajorityChecker(arr)
* var param_1 = obj.query(left,right,threshold)
*/
```

TypeScript Solution:

```
/**
* Problem: Online Majority Element In Subarray
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
```

```
*/

class MajorityChecker {
constructor(arr: number[]) {

}

query(left: number, right: number, threshold: number): number {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* var obj = new MajorityChecker(arr)
* var param_1 = obj.query(left,right,threshold)
*/
```

C# Solution:

```
/*
* Problem: Online Majority Element In Subarray
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class MajorityChecker {

public MajorityChecker(int[] arr) {

}

public int Query(int left, int right, int threshold) {

}
}
```

```
/**
* Your MajorityChecker object will be instantiated and called as such:
* MajorityChecker obj = new MajorityChecker(arr);
* int param_1 = obj.Query(left,right,threshold);
*/
```

C Solution:

```
/*
* Problem: Online Majority Element In Subarray
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/




typedef struct {

} MajorityChecker;


MajorityChecker* majorityCheckerCreate(int* arr, int arrSize) {

}

int majorityCheckerQuery(MajorityChecker* obj, int left, int right, int
threshold) {

}

void majorityCheckerFree(MajorityChecker* obj) {

}

/**
* Your MajorityChecker struct will be instantiated and called as such:
```

```
* MajorityChecker* obj = majorityCheckerCreate(arr, arrSize);
* int param_1 = majorityCheckerQuery(obj, left, right, threshold);

* majorityCheckerFree(obj);
*/
```

Go Solution:

```
// Problem: Online Majority Element In Subarray
// Difficulty: Hard
// Tags: array, tree, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

type MajorityChecker struct {

}


func Constructor(arr []int) MajorityChecker {

}


func (this *MajorityChecker) Query(left int, right int, threshold int) int {

}


/**
* Your MajorityChecker object will be instantiated and called as such:
* obj := Constructor(arr);
* param_1 := obj.Query(left,right,threshold);
*/
```

Kotlin Solution:

```
class MajorityChecker(arr: IntArray) {
```

```
fun query(left: Int, right: Int, threshold: Int): Int {

}

}

/**
* Your MajorityChecker object will be instantiated and called as such:
* var obj = MajorityChecker(arr)
* var param_1 = obj.query(left,right,threshold)
*/
```

Swift Solution:

```
class MajorityChecker {

init(_ arr: [Int]) {

}

func query(_ left: Int, _ right: Int, _ threshold: Int) -> Int {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* let obj = MajorityChecker(arr)
* let ret_1: Int = obj.query(left, right, threshold)
*/
```

Rust Solution:

```
// Problem: Online Majority Element In Subarray
// Difficulty: Hard
// Tags: array, tree, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height
```

```
struct MajorityChecker {

}


/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl MajorityChecker {

fn new(arr: Vec<i32>) -> Self {

}

fn query(&self, left: i32, right: i32, threshold: i32) -> i32 {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* let obj = MajorityChecker::new(arr);
* let ret_1: i32 = obj.query(left, right, threshold);
*/
```

Ruby Solution:

```
class MajorityChecker

=begin
:type arr: Integer[]
=end
def initialize(arr)

end


=begin
:type left: Integer
```

```
:type right: Integer
:type threshold: Integer
:rtype: Integer
=end
def query(left, right, threshold)

end



end

# Your MajorityChecker object will be instantiated and called as such:
# obj = MajorityChecker.new(arr)
# param_1 = obj.query(left, right, threshold)
```

PHP Solution:

```
class MajorityChecker {
/**
* @param Integer[] $arr
*/
function __construct($arr) {

}

/**
* @param Integer $left
* @param Integer $right
* @param Integer $threshold
* @return Integer
*/
function query($left, $right, $threshold) {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* $obj = MajorityChecker($arr);
* $ret_1 = $obj->query($left, $right, $threshold);
*/
```

Dart Solution:

```
class MajorityChecker {

MajorityChecker(List<int> arr) {

}

int query(int left, int right, int threshold) {

}
}

/**
* Your MajorityChecker object will be instantiated and called as such:
* MajorityChecker obj = MajorityChecker(arr);
* int param1 = obj.query(left,right,threshold);
*/
```

Scala Solution:

```
class MajorityChecker(_arr: Array[Int]) {

def query(left: Int, right: Int, threshold: Int): Int = {

}

}

/**
* Your MajorityChecker object will be instantiated and called as such:
* val obj = new MajorityChecker(arr)
* val param_1 = obj.query(left,right,threshold)
*/
```

Elixir Solution:

```
defmodule MajorityChecker do
@spec init_(arr :: [integer]) :: any
def init_(arr) do

end
```

```
@spec query(left :: integer, right :: integer, threshold :: integer) ::
integer
def query(left, right, threshold) do

end
end

# Your functions will be called as such:
# MajorityChecker.init_(arr)
# param_1 = MajorityChecker.query(left, right, threshold)

# MajorityChecker.init_ will be called before every test case, in which you
can do some necessary initializations.
```

Erlang Solution:

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

-spec majority_checker_query(Left :: integer(), Right :: integer(), Threshold
:: integer()) -> integer().
majority_checker_query(Left, Right, Threshold) ->
.


%% Your functions will be called as such:
%% majority_checker_init_(Arr),
%% Param_1 = majority_checker_query(Left, Right, Threshold),

%% majority_checker_init_ will be called before every test case, in which you
can do some necessary initializations.
```

Racket Solution:

```
(define majority-checker%
(class object%
(super-new)

; arr : (listof exact-integer?)
```

```
(init-field
arr)

; query : exact-integer? exact-integer? exact-integer? -> exact-integer?
(define/public (query left right threshold)
)))

;; Your majority-checker% object will be instantiated and called as such:
;; (define obj (new majority-checker% [arr arr]))
;; (define param_1 (send obj query left right threshold))
```