Problem 436: Find Right Interval

Problem Information

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

Problem Description

You are given an array of

intervals

, where

intervals[i] = [start

i

, end

i

]

and each

start

i

is

unique

.

The

right interval

for an interval

i

is an interval

j

such that

start

j

>= end

i

and

start

j

is

minimized

. Note that

i

may equal

j

.

Return

an array of

right interval

indices for each interval

i

. If no

right interval

exists for interval

i

, then put

-1

at index

i

.

Example 1:

Input:

intervals = [[1,2]]

Output:

[-1]

Explanation:

There is only one interval in the collection, so it outputs -1.

Example 2:

Input:

intervals = [[3,4],[2,3],[1,2]]

Output:

[-1,0,1]

Explanation:

There is no right interval for [3,4]. The right interval for [2,3] is [3,4] since start

0

= 3 is the smallest start that is >= end

1

= 3. The right interval for [1,2] is [2,3] since start

1

= 2 is the smallest start that is >= end

2

= 2.

Example 3:

Input:

intervals = [[1,4],[2,3],[3,4]]

Output:

[-1,2,-1]

Explanation:

There is no right interval for [1,4] and [3,4]. The right interval for [2,3] is [3,4] since start

2

= 3 is the smallest start that is >= end

1

= 3.

Constraints:

1 <= intervals.length <= 2 * 10

4

intervals[i].length == 2

-10

6

<= start

i

<= end

i

<= 10

6

The start point of each interval is

unique

.

Code Snippets

C++:

```
class Solution {
public:
vector<int> findRightInterval(vector<vector<int>>& intervals) {

}
};
```

Java:

```
class Solution {
public int[] findRightInterval(int[][] intervals) {

}
}
```

Python3:

```
class Solution:
def findRightInterval(self, intervals: List[List[int]]) -> List[int]:
```

Python:

```
class Solution(object):
def findRightInterval(self, intervals):
```

```
"""
:type intervals: List[List[int]]
:rtype: List[int]
"""
```

JavaScript:

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

};
```

TypeScript:

```
function findRightInterval(intervals: number[][]): number[] {

};
```

C#:

```
public class Solution {
public int[] FindRightInterval(int[][] intervals) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* findRightInterval(int** intervals, int intervalsSize, int*
intervalsColSize, int* returnSize) {

}
```

Go:

```
func findRightInterval(intervals [][]int) []int {

}
```

Kotlin:

```
class Solution {
fun findRightInterval(intervals: Array<IntArray>): IntArray {

}
}
```

Swift:

```
class Solution {
func findRightInterval(_ intervals: [[Int]]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn find_right_interval(intervals: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[][]} intervals
# @return {Integer[]}
def find_right_interval(intervals)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $intervals
* @return Integer[]
```

```
*/
function findRightInterval($intervals) {

}
}
```

Dart:

```
class Solution {
List<int> findRightInterval(List<List<int>> intervals) {

}
}
```

Scala:

```
object Solution {
def findRightInterval(intervals: Array[Array[Int]]): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_right_interval(intervals :: [[integer]]) :: [integer]
def find_right_interval(intervals) do

end
end
```

Erlang:

```
-spec find_right_interval(Intervals :: [[integer()]]) -> [integer()].
find_right_interval(Intervals) ->
.
```

Racket:

```
(define/contract (find-right-interval intervals)
(-> (listof (listof exact-integer?)) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Right Interval
* Difficulty: Medium
* 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:
vector<int> findRightInterval(vector<vector<int>>& intervals) {

}
};
```

Java Solution:

```
/**
* Problem: Find Right Interval
* Difficulty: Medium
* 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[] findRightInterval(int[][] intervals) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Right Interval
Difficulty: Medium
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 findRightInterval(self, intervals: List[List[int]]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find Right Interval
* Difficulty: Medium
* 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[][]} intervals
* @return {number[]}
*/
var findRightInterval = function(intervals) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Find Right Interval
* Difficulty: Medium
* 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 findRightInterval(intervals: number[][]): number[] {

};
```

C# Solution:

```
/*
* Problem: Find Right Interval
* Difficulty: Medium
* 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[] FindRightInterval(int[][] intervals) {

}
}
```

C Solution:

```
/*
* Problem: Find Right Interval
* Difficulty: Medium
```

```
* 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* findRightInterval(int** intervals, int intervalsSize, int*
intervalsColSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Find Right Interval
// Difficulty: Medium
// 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 findRightInterval(intervals [][]int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun findRightInterval(intervals: Array<IntArray>): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func findRightInterval(_ intervals: [[Int]]) -> [Int] {
```

```
}
}
```

Rust Solution:

```
// Problem: Find Right Interval
// Difficulty: Medium
// 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_right_interval(intervals: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} intervals
# @return {Integer[]}
def find_right_interval(intervals)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $intervals
* @return Integer[]
*/
function findRightInterval($intervals) {

}
}
```

Dart Solution:

```
class Solution {
List<int> findRightInterval(List<List<int>> intervals) {

}
}
```

Scala Solution:

```
object Solution {
def findRightInterval(intervals: Array[Array[Int]]): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_right_interval(intervals :: [[integer]]) :: [integer]
def find_right_interval(intervals) do

end
end
```

Erlang Solution:

```
-spec find_right_interval(Intervals :: [[integer()]]) -> [integer()].
find_right_interval(Intervals) ->
.
```

Racket Solution:

```
(define/contract (find-right-interval intervals)
(-> (listof (listof exact-integer?)) (listof exact-integer?))
)
```