Problem 911: Online Election

Problem Information

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

Problem Description

You are given two integer arrays

persons

and

times

. In an election, the

i

th

vote was cast for

persons[i]

at time

times[i]

.

For each query at a time

t

, find the person that was leading the election at time

t

. Votes cast at time

t

will count towards our query. In the case of a tie, the most recent vote (among tied candidates) wins.

Implement the

TopVotedCandidate

class:

TopVotedCandidate(int[] persons, int[] times)

Initializes the object with the

persons

and

times

arrays.

int q(int t)

Returns the number of the person that was leading the election at time

t

according to the mentioned rules.

Example 1:

Input

["TopVotedCandidate", "q", "q", "q", "q", "q", "q"] [[[0, 1, 1, 0, 0, 1, 0], [0, 5, 10, 15, 20, 25, 30]], [3], [12], [25], [15], [24], [8]]

Output

[null, 0, 1, 1, 0, 0, 1]

Explanation

TopVotedCandidate topVotedCandidate = new TopVotedCandidate([0, 1, 1, 0, 0, 1, 0], [0, 5, 10, 15, 20, 25, 30]); topVotedCandidate.q(3); // return 0, At time 3, the votes are [0], and 0 is leading. topVotedCandidate.q(12); // return 1, At time 12, the votes are [0,1,1], and 1 is leading. topVotedCandidate.q(25); // return 1, At time 25, the votes are [0,1,1,0,0,1], and 1 is leading (as ties go to the most recent vote.) topVotedCandidate.q(15); // return 0 topVotedCandidate.q(24); // return 0 topVotedCandidate.q(8); // return 1

Constraints:

1 <= persons.length <= 5000

times.length == persons.length

0 <= persons[i] < persons.length

0 <= times[i] <= 10

9

times

is sorted in a strictly increasing order.

times[0] <= t <= 10

9

At most

10

4

calls will be made to

q

.

Code Snippets

C++:

```
class TopVotedCandidate {
public:
TopVotedCandidate(vector<int>& persons, vector<int>& times) {

}

int q(int t) {

}
};

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* TopVotedCandidate* obj = new TopVotedCandidate(persons, times);
* int param_1 = obj->q(t);
*/
```

Java:

```
class TopVotedCandidate {

public TopVotedCandidate(int[] persons, int[] times) {

}
public int q(int t) {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* TopVotedCandidate obj = new TopVotedCandidate(persons, times);
* int param_1 = obj.q(t);
*/
```

Python3:

```
class TopVotedCandidate:

def __init__(self, persons: List[int], times: List[int]):



def q(self, t: int) -> int:




# Your TopVotedCandidate object will be instantiated and called as such:
# obj = TopVotedCandidate(persons, times)
# param_1 = obj.q(t)
```

Python:

```
class TopVotedCandidate(object):

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



def q(self, t):
"""
:type t: int
:rtype: int
```

```

"""



# Your TopVotedCandidate object will be instantiated and called as such:
# obj = TopVotedCandidate(persons, times)
# param_1 = obj.q(t)
```

JavaScript:

```
/**
* @param {number[]} persons
* @param {number[]} times
*/
var TopVotedCandidate = function(persons, times) {

};

/**
* @param {number} t
* @return {number}
*/
TopVotedCandidate.prototype.q = function(t) {

};

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* var obj = new TopVotedCandidate(persons, times)
* var param_1 = obj.q(t)
*/
```

TypeScript:

```
class TopVotedCandidate {
constructor(persons: number[], times: number[]) {

}

q(t: number): number {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* var obj = new TopVotedCandidate(persons, times)
* var param_1 = obj.q(t)
*/
```

C#:

```
public class TopVotedCandidate {

public TopVotedCandidate(int[] persons, int[] times) {

}

public int Q(int t) {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* TopVotedCandidate obj = new TopVotedCandidate(persons, times);
* int param_1 = obj.Q(t);
*/
```

C:

```
typedef struct {

} TopVotedCandidate;


TopVotedCandidate* topVotedCandidateCreate(int* persons, int personsSize,
int* times, int timesSize) {

}
```

```
int topVotedCandidateQ(TopVotedCandidate* obj, int t) {

}

void topVotedCandidateFree(TopVotedCandidate* obj) {

}

/**
* Your TopVotedCandidate struct will be instantiated and called as such:
* TopVotedCandidate* obj = topVotedCandidateCreate(persons, personsSize,
times, timesSize);
* int param_1 = topVotedCandidateQ(obj, t);

* topVotedCandidateFree(obj);
*/
```

Go:

```
type TopVotedCandidate struct {

}


func Constructor(persons []int, times []int) TopVotedCandidate {

}


func (this *TopVotedCandidate) Q(t int) int {

}


/**
* Your TopVotedCandidate object will be instantiated and called as such:
* obj := Constructor(persons, times);
* param_1 := obj.Q(t);
*/
```

Kotlin:

```
class TopVotedCandidate(persons: IntArray, times: IntArray) {

fun q(t: Int): Int {

}

}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* var obj = TopVotedCandidate(persons, times)
* var param_1 = obj.q(t)
*/
```

Swift:

```

class TopVotedCandidate {

init(_ persons: [Int], _ times: [Int]) {

}

func q(_ t: Int) -> Int {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* let obj = TopVotedCandidate(persons, times)
* let ret_1: Int = obj.q(t)
*/
```

Rust:

```
struct TopVotedCandidate {

}



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

fn new(persons: Vec<i32>, times: Vec<i32>) -> Self {

}

fn q(&self, t: i32) -> i32 {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* let obj = TopVotedCandidate::new(persons, times);
* let ret_1: i32 = obj.q(t);
*/
```

Ruby:

```
class TopVotedCandidate

=begin
:type persons: Integer[]
:type times: Integer[]
=end
def initialize(persons, times)

end



=begin
:type t: Integer
:rtype: Integer
=end
def q(t)

end



end
```

```
# Your TopVotedCandidate object will be instantiated and called as such:
# obj = TopVotedCandidate.new(persons, times)
# param_1 = obj.q(t)
```

PHP:

```
class TopVotedCandidate {
/**
* @param Integer[] $persons
* @param Integer[] $times
*/
function __construct($persons, $times) {

}

/**
* @param Integer $t
* @return Integer
*/
function q($t) {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* $obj = TopVotedCandidate($persons, $times);
* $ret_1 = $obj->q($t);
*/
```

Dart:

```
class TopVotedCandidate {

TopVotedCandidate(List<int> persons, List<int> times) {

}

int q(int t) {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* TopVotedCandidate obj = TopVotedCandidate(persons, times);
* int param1 = obj.q(t);
*/
```

Scala:

```
class TopVotedCandidate(_persons: Array[Int], _times: Array[Int]) {

def q(t: Int): Int = {

}

}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* val obj = new TopVotedCandidate(persons, times)
* val param_1 = obj.q(t)
*/
```

Elixir:

```
defmodule TopVotedCandidate do
@spec init_(persons :: [integer], times :: [integer]) :: any
def init_(persons, times) do

end

@spec q(t :: integer) :: integer
def q(t) do

end
end

# Your functions will be called as such:
# TopVotedCandidate.init_(persons, times)
# param_1 = TopVotedCandidate.q(t)
# TopVotedCandidate.init_ will be called before every test case, in which you can do some necessary initializations.
```

Erlang:

```
-spec top_voted_candidate_init_(Persons :: [integer()], Times :: [integer()]) -> any().
top_voted_candidate_init_(Persons, Times) ->
.

-spec top_voted_candidate_q(T :: integer()) -> integer().
top_voted_candidate_q(T) ->
.



%% Your functions will be called as such:
%% top_voted_candidate_init_(Persons, Times),
%% Param_1 = top_voted_candidate_q(T),

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

Racket:

```
(define top-voted-candidate%
(class object%
(super-new)

; persons : (listof exact-integer?)
; times : (listof exact-integer?)
(init-field
persons
times)

; q : exact-integer? -> exact-integer?
(define/public (q t)
)))

;; Your top-voted-candidate% object will be instantiated and called as such:
;; (define obj (new top-voted-candidate% [persons persons] [times times]))
;; (define param_1 (send obj q t))
```

Solutions

C++ Solution:

```
/*
* Problem: Online Election
* Difficulty: Medium
* Tags: array, hash, sort, search
*
* 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 TopVotedCandidate {
public:
TopVotedCandidate(vector<int>& persons, vector<int>& times) {

}

int q(int t) {

}
};

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* TopVotedCandidate* obj = new TopVotedCandidate(persons, times);
* int param_1 = obj->q(t);
*/
```

Java Solution:

```
/**
* Problem: Online Election
* Difficulty: Medium
* Tags: array, hash, sort, search
*
* 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 TopVotedCandidate {

public TopVotedCandidate(int[] persons, int[] times) {

}

public int q(int t) {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* TopVotedCandidate obj = new TopVotedCandidate(persons, times);
* int param_1 = obj.q(t);
*/
```

Python3 Solution:

```
"""
Problem: Online Election
Difficulty: Medium
Tags: array, hash, sort, search

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 TopVotedCandidate:

def __init__(self, persons: List[int], times: List[int]):



def q(self, t: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class TopVotedCandidate(object):

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



def q(self, t):
"""
:type t: int
:rtype: int
"""




# Your TopVotedCandidate object will be instantiated and called as such:
# obj = TopVotedCandidate(persons, times)
# param_1 = obj.q(t)
```

JavaScript Solution:

```
/**
* Problem: Online Election
* Difficulty: Medium
* Tags: array, hash, sort, search
*
* 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[]} persons
* @param {number[]} times
*/
var TopVotedCandidate = function(persons, times) {

};

/**
```

```
* @param {number} t
* @return {number}
*/
TopVotedCandidate.prototype.q = function(t) {

};

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* var obj = new TopVotedCandidate(persons, times)
* var param_1 = obj.q(t)
*/
```

TypeScript Solution:

```
/**
* Problem: Online Election
* Difficulty: Medium
* Tags: array, hash, sort, search
*
* 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 TopVotedCandidate {
constructor(persons: number[], times: number[]) {

}

q(t: number): number {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* var obj = new TopVotedCandidate(persons, times)
* var param_1 = obj.q(t)
*/
```

C# Solution:

```
/*
* Problem: Online Election
* Difficulty: Medium
* Tags: array, hash, sort, search
*
* 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 TopVotedCandidate {

public TopVotedCandidate(int[] persons, int[] times) {

}

public int Q(int t) {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* TopVotedCandidate obj = new TopVotedCandidate(persons, times);
* int param_1 = obj.Q(t);
*/
```

C Solution:

```
/*
* Problem: Online Election
* Difficulty: Medium
* Tags: array, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
typedef struct {

} TopVotedCandidate;


TopVotedCandidate* topVotedCandidateCreate(int* persons, int personsSize,
int* times, int timesSize) {

}

int topVotedCandidateQ(TopVotedCandidate* obj, int t) {

}

void topVotedCandidateFree(TopVotedCandidate* obj) {

}

/**
* Your TopVotedCandidate struct will be instantiated and called as such:
* TopVotedCandidate* obj = topVotedCandidateCreate(persons, personsSize,
times, timesSize);
* int param_1 = topVotedCandidateQ(obj, t);

* topVotedCandidateFree(obj);
*/
```

Go Solution:

```
// Problem: Online Election
// Difficulty: Medium
// Tags: array, hash, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

type TopVotedCandidate struct {


}
```

```
func Constructor(persons []int, times []int) TopVotedCandidate {

}


func (this *TopVotedCandidate) Q(t int) int {

}


/**
* Your TopVotedCandidate object will be instantiated and called as such:
* obj := Constructor(persons, times);
* param_1 := obj.Q(t);
*/
```

Kotlin Solution:

```
class TopVotedCandidate(persons: IntArray, times: IntArray) {

fun q(t: Int): Int {

}

}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* var obj = TopVotedCandidate(persons, times)
* var param_1 = obj.q(t)
*/
```

Swift Solution:

```
class TopVotedCandidate {

init(_ persons: [Int], _ times: [Int]) {

}

func q(_ t: Int) -> Int {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* let obj = TopVotedCandidate(persons, times)
* let ret_1: Int = obj.q(t)
*/
```

Rust Solution:

```
// Problem: Online Election
// Difficulty: Medium
// Tags: array, hash, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

struct TopVotedCandidate {

}



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

fn new(persons: Vec<i32>, times: Vec<i32>) -> Self {

}

fn q(&self, t: i32) -> i32 {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* let obj = TopVotedCandidate::new(persons, times);
* let ret_1: i32 = obj.q(t);
*/
```

Ruby Solution:

```
class TopVotedCandidate

=begin
:type persons: Integer[]
:type times: Integer[]
=end
def initialize(persons, times)

end


=begin
:type t: Integer
:rtype: Integer
=end
def q(t)

end


end

# Your TopVotedCandidate object will be instantiated and called as such:
# obj = TopVotedCandidate.new(persons, times)
# param_1 = obj.q(t)
```

PHP Solution:

```
class TopVotedCandidate {
/**
* @param Integer[] $persons
* @param Integer[] $times
*/
function __construct($persons, $times) {

}

/**
* @param Integer $t
* @return Integer
*/
function q($t) {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* $obj = TopVotedCandidate($persons, $times);
* $ret_1 = $obj->q($t);
*/
```

Dart Solution:

```
class TopVotedCandidate {

TopVotedCandidate(List<int> persons, List<int> times) {

}

int q(int t) {

}
}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* TopVotedCandidate obj = TopVotedCandidate(persons, times);
* int param1 = obj.q(t);
*/
```

Scala Solution:

```
class TopVotedCandidate(_persons: Array[Int], _times: Array[Int]) {

def q(t: Int): Int = {

}

}

/**
* Your TopVotedCandidate object will be instantiated and called as such:
* val obj = new TopVotedCandidate(persons, times)
* val param_1 = obj.q(t)
*/
```

Elixir Solution:

```
defmodule TopVotedCandidate do
@spec init_(persons :: [integer], times :: [integer]) :: any
def init_(persons, times) do

end

@spec q(t :: integer) :: integer
def q(t) do

end
end

# Your functions will be called as such:
# TopVotedCandidate.init_(persons, times)
# param_1 = TopVotedCandidate.q(t)

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

Erlang Solution:

```
-spec top_voted_candidate_init_(Persons :: [integer()], Times :: [integer()]) -> any().
top_voted_candidate_init_(Persons, Times) ->
.
```

```
-spec top_voted_candidate_q(T :: integer()) -> integer().
top_voted_candidate_q(T) ->
.



%% Your functions will be called as such:
%% top_voted_candidate_init_(Persons, Times),
%% Param_1 = top_voted_candidate_q(T),

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

Racket Solution:

```
(define top-voted-candidate%
(class object%
(super-new)

; persons : (listof exact-integer?)
; times : (listof exact-integer?)
(init-field
persons
times)

; q : exact-integer? -> exact-integer?
(define/public (q t)
)))

;; Your top-voted-candidate% object will be instantiated and called as such:
;; (define obj (new top-voted-candidate% [persons persons] [times times]))
;; (define param_1 (send obj q t))
```