Problem 1535: Find the Winner of an Array Game

Problem Information

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

Problem Description

Given an integer array

arr

of

distinct

integers and an integer

k

.

A game will be played between the first two elements of the array (i.e.

arr[0]

and

arr[1]

). In each round of the game, we compare

arr[0]

with

arr[1]

, the larger integer wins and remains at position

0

, and the smaller integer moves to the end of the array. The game ends when an integer wins

k

consecutive rounds.

Return

the integer which will win the game

.

It is

guaranteed

that there will be a winner of the game.

Example 1:

Input:

arr = [2,1,3,5,4,6,7], k = 2

Output:

5

Explanation:

Let's see the rounds of the game: Round | arr | winner | win_count 1 | [2,1,3,5,4,6,7] | 2 | 1 2 | [2,3,5,4,6,7,1] | 3 | 1 3 | [3,5,4,6,7,1,2] | 5 | 1 4 | [5,4,6,7,1,2,3] | 5 | 2 So we can see that 4 rounds will be played and 5 is the winner because it wins 2 consecutive games.

Example 2:

Input:

arr = [3,2,1], k = 10

Output:

3

Explanation:

3 will win the first 10 rounds consecutively.

Constraints:

2 <= arr.length <= 10

5

1 <= arr[i] <= 10

6

arr

contains

distinct

integers.

1 <= k <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int getWinner(vector<int>& arr, int k) {

}
};
```

Java:

```
class Solution {
public int getWinner(int[] arr, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} arr
* @param {number} k
* @return {number}
*/
```

```
var getWinner = function(arr, k) {

};
```

TypeScript:

```
function getWinner(arr: number[], k: number): number {

};
```

C#:

```
public class Solution {
public int GetWinner(int[] arr, int k) {

}
}
```

C:

```
int getWinner(int* arr, int arrSize, int k) {

}
```

Go:

```
func getWinner(arr []int, k int) int {

}
```

Kotlin:

```
class Solution {
fun getWinner(arr: IntArray, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func getWinner(_ arr: [Int], _ k: Int) -> Int {
```

```
}
}
```

Rust:

```
impl Solution {
pub fn get_winner(arr: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @param {Integer} k
# @return {Integer}
def get_winner(arr, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @param Integer $k
* @return Integer
*/
function getWinner($arr, $k) {

}
}
```

Dart:

```
class Solution {
int getWinner(List<int> arr, int k) {

}
}
```

Scala:

```
object Solution {
def getWinner(arr: Array[Int], k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_winner(arr :: [integer], k :: integer) :: integer
def get_winner(arr, k) do

end
end
```

Erlang:

```
-spec get_winner(Arr :: [integer()], K :: integer()) -> integer().
get_winner(Arr, K) ->
.
```

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Find the Winner of an Array Game
* Difficulty: Medium
* Tags: array
*
* 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 getWinner(vector<int>& arr, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Winner of an Array Game
* Difficulty: Medium
* Tags: array
*
* 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 getWinner(int[] arr, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Winner of an Array Game
Difficulty: Medium
Tags: array

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 getWinner(self, arr: List[int], k: int) -> int:
```

```
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Winner of an Array Game
* Difficulty: Medium
* Tags: array
*
* 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[]} arr
* @param {number} k
* @return {number}
*/
var getWinner = function(arr, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Winner of an Array Game
* Difficulty: Medium
* Tags: array
*
```

```
* 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 getWinner(arr: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Winner of an Array Game
* Difficulty: Medium
* Tags: array
*
* 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 GetWinner(int[] arr, int k) {

}
}
```

C Solution:

```
/*
* Problem: Find the Winner of an Array Game
* Difficulty: Medium
* Tags: array
*
* 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
*/

int getWinner(int* arr, int arrSize, int k) {
```

```
}
```

Go Solution:

```
// Problem: Find the Winner of an Array Game
// Difficulty: Medium
// Tags: array
//
// 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 getWinner(arr []int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun getWinner(arr: IntArray, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func getWinner(_ arr: [Int], _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Winner of an Array Game
// Difficulty: Medium
// Tags: array
//
// 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 get_winner(arr: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @param {Integer} k
# @return {Integer}
def get_winner(arr, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @param Integer $k
* @return Integer
*/
function getWinner($arr, $k) {

}
}
```

Dart Solution:

```
class Solution {
int getWinner(List<int> arr, int k) {

}
}
```

Scala Solution:

```
object Solution {
def getWinner(arr: Array[Int], k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_winner(arr :: [integer], k :: integer) :: integer
def get_winner(arr, k) do

end
end
```

Erlang Solution:

```
-spec get_winner(Arr :: [integer()], K :: integer()) -> integer().
get_winner(Arr, K) ->
.
```

Racket Solution:

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