Problem 2836: Maximize Value of Function in a Ball Passing Game

Problem Information

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

Problem Description

You are given an integer array

receiver

of length

n

and an integer

k

.

n

players are playing a ball-passing game.

You choose the starting player,

i

. The game proceeds as follows: player

i

passes the ball to player

receiver[i]

, who then passes it to

receiver[receiver[i]]

, and so on, for

k

passes in total. The game's score is the sum of the indices of the players who touched the ball, including repetitions, i.e.

i + receiver[i] + receiver[receiver[i]] + ... + receiver

(k)

[i]

.

Return the

maximum

possible score.

Notes:

receiver

may contain duplicates.

receiver[i]

may be equal to

i

.

Example 1:

Input:

receiver = [2,0,1], k = 4

Output:

6

Explanation:

Starting with player

i = 2

the initial score is 2:

Pass

Sender Index

Receiver Index

Score

1

2

1

3

2

1

0

3

3

0

2

5

4

2

1

6

Example 2:

Input:

receiver = [1,1,1,2,3], k = 3

Output:

10

Explanation:

Starting with player

i = 4

the initial score is 4:

Pass

Sender Index

Receiver Index

Score

1

4

3

7

2

3

2

9

3

2

1

10

Constraints:

1 <= receiver.length == n <= 10

5

0 <= receiver[i] <= n - 1

1 <= k <= 10

10

Code Snippets

C++:

```
class Solution {
public:
long long getMaxFunctionValue(vector<int>& receiver, long long k) {

}
};
```

Java:

```
class Solution {
public long getMaxFunctionValue(List<Integer> receiver, long k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} receiver
* @param {number} k
* @return {number}
*/
var getMaxFunctionValue = function(receiver, k) {

};
```

TypeScript:

```
function getMaxFunctionValue(receiver: number[], k: number): number {

};
```

C#:

```
public class Solution {
public long GetMaxFunctionValue(IList<int> receiver, long k) {

}
}
```

C:

```
long long getMaxFunctionValue(int* receiver, int receiverSize, long long k) {

}
```

Go:

```
func getMaxFunctionValue(receiver []int, k int64) int64 {

}
```

Kotlin:

```
class Solution {
fun getMaxFunctionValue(receiver: List<Int>, k: Long): Long {

}
```

```
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn get_max_function_value(receiver: Vec<i32>, k: i64) -> i64 {

}
}
```

Ruby:

```
# @param {Integer[]} receiver
# @param {Integer} k
# @return {Integer}
def get_max_function_value(receiver, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $receiver
* @param Integer $k
* @return Integer
*/
function getMaxFunctionValue($receiver, $k) {

}
}
```

Dart:

```
class Solution {
int getMaxFunctionValue(List<int> receiver, int k) {

}
}
```

Scala:

```
object Solution {
def getMaxFunctionValue(receiver: List[Int], k: Long): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_max_function_value(receiver :: [integer], k :: integer) :: integer
def get_max_function_value(receiver, k) do

end
end
```

Erlang:

```
-spec get_max_function_value(Receiver :: [integer()], K :: integer()) ->
integer().
get_max_function_value(Receiver, K) ->
.
```

Racket:

```
(define/contract (get-max-function-value receiver k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximize Value of Function in a Ball Passing Game
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
long long getMaxFunctionValue(vector<int>& receiver, long long k) {

}
};
```

Java Solution:

```
/**
* Problem: Maximize Value of Function in a Ball Passing Game
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public long getMaxFunctionValue(List<Integer> receiver, long k) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximize Value of Function in a Ball Passing Game
Difficulty: Hard
Tags: array, dp
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def getMaxFunctionValue(self, receiver: List[int], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximize Value of Function in a Ball Passing Game
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number[]} receiver
* @param {number} k
* @return {number}
*/
var getMaxFunctionValue = function(receiver, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximize Value of Function in a Ball Passing Game
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function getMaxFunctionValue(receiver: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximize Value of Function in a Ball Passing Game
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public long GetMaxFunctionValue(IList<int> receiver, long k) {

}
}
```

C Solution:

```
/*
* Problem: Maximize Value of Function in a Ball Passing Game
* Difficulty: Hard
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
```

```
* Space Complexity: O(n) or O(n * m) for DP table
*/

long long getMaxFunctionValue(int* receiver, int receiverSize, long long k) {

}
```

Go Solution:

```
// Problem: Maximize Value of Function in a Ball Passing Game
// Difficulty: Hard
// Tags: array, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func getMaxFunctionValue(receiver []int, k int64) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun getMaxFunctionValue(receiver: List<Int>, k: Long): Long {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximize Value of Function in a Ball Passing Game
// Difficulty: Hard
```

```
// Tags: array, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn get_max_function_value(receiver: Vec<i32>, k: i64) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} receiver
# @param {Integer} k
# @return {Integer}
def get_max_function_value(receiver, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $receiver
* @param Integer $k
* @return Integer
*/
function getMaxFunctionValue($receiver, $k) {

}
}
```

Dart Solution:

```
class Solution {
int getMaxFunctionValue(List<int> receiver, int k) {

}
```

```
}
```

Scala Solution:

```
object Solution {
def getMaxFunctionValue(receiver: List[Int], k: Long): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_max_function_value(receiver :: [integer], k :: integer) :: integer
def get_max_function_value(receiver, k) do

end
end
```

Erlang Solution:

```
-spec get_max_function_value(Receiver :: [integer()], K :: integer()) ->
integer().
get_max_function_value(Receiver, K) ->
.
```

Racket Solution:

```
(define/contract (get-max-function-value receiver k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```