Problem 2666: Allow One Function Call

Problem Information

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

Problem Description

Given a function

fn

, return a new function that is identical to the original function except that it ensures

fn

is called at most once.

The first time the returned function is called, it should return the same result as

fn

.

Every subsequent time it is called, it should return

undefined

.

Example 1:

Input:

fn = (a,b,c) => (a + b + c), calls = [[1,2,3],[2,3,6]]

Output:

[{"calls":1,"value":6}]

Explanation:

const onceFn = once(fn); onceFn(1, 2, 3); // 6 onceFn(2, 3, 6); // undefined, fn was not called

Example 2:

Input:

fn = (a,b,c) => (a * b * c), calls = [[5,7,4],[2,3,6],[4,6,8]]

Output:

[{"calls":1,"value":140}]

Explanation:

const onceFn = once(fn); onceFn(5, 7, 4); // 140 onceFn(2, 3, 6); // undefined, fn was not called onceFn(4, 6, 8); // undefined, fn was not called

Constraints:

calls

is a valid JSON array

1 <= calls.length <= 10

1 <= calls[i].length <= 100

2 <= JSON.stringify(calls).length <= 1000

Code Snippets

JavaScript:

```
/**
* @param {Function} fn
* @return {Function}
*/
var once = function(fn) {

return function(...args){

}
};

/**
* let fn = (a,b,c) => (a + b + c)
* let onceFn = once(fn)
*
* onceFn(1,2,3); // 6
* onceFn(2,3,6); // returns undefined without calling fn
*/
```

TypeScript:

```
type JSONValue = null | boolean | number | string | JSONValue[] | { [key:
string]: JSONValue };
type OnceFn = (...args: JSONValue[]) => JSONValue | undefined

function once(fn: Function): OnceFn {

return function (...args) {

};
}

/**
* let fn = (a,b,c) => (a + b + c)
* let onceFn = once(fn)
*
* onceFn(1,2,3); // 6
* onceFn(2,3,6); // returns undefined without calling fn
*/
```

Solutions

JavaScript Solution:

```
/**
* Problem: Allow One Function Call
* Difficulty: Easy
* Tags: array, string
*
* 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 {Function} fn
* @return {Function}
*/
var once = function(fn) {

return function(...args){

}
};

/**
* let fn = (a,b,c) => (a + b + c)
* let onceFn = once(fn)
*
* onceFn(1,2,3); // 6
* onceFn(2,3,6); // returns undefined without calling fn
*/
```

TypeScript Solution:

```
/**
* Problem: Allow One Function Call
* Difficulty: Easy
* Tags: array, string
*
* 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
*/

type JSONValue = null | boolean | number | string | JSONValue[] | { [key:
string]: JSONValue };
type OnceFn = (...args: JSONValue[]) => JSONValue | undefined

function once(fn: Function): OnceFn {

return function (...args) {

};
}

/**
* let fn = (a,b,c) => (a + b + c)
* let onceFn = once(fn)
*
* onceFn(1,2,3); // 6
* onceFn(2,3,6); // returns undefined without calling fn
*/
```