Problem 2725: Interval Cancellation

Problem Information

Difficulty: Easy
Acceptance Rate: 84.41%
Paid Only: No

Problem Description

Given a function `fn`, an array of arguments `args`, and an interval time `t`, return a cancel function `cancelFn`.

After a delay of `cancelTimeMs`, the returned cancel function `cancelFn` will be invoked.

setTimeout(cancelFn, cancelTimeMs)

The function `fn` should be called with `args` immediately and then called again every `t` milliseconds until `cancelFn` is called at `cancelTimeMs` ms.

Example 1:

Input: fn = (x) => x * 2, args = [4], t = 35 **Output:** [{"time": 0, "returned": 8}, {"time": 35, "returned": 8}, {"time": 70, "returned": 8}, {"time": 105, "returned": 8}, {"time": 140, "returned": 8}, {"time": 175, "returned": 8}] **Explanation:** const cancelTimeMs = 190; const cancelFn = cancellable((x) => x * 2, [4], 35); setTimeout(cancelFn, cancelTimeMs); Every 35ms, fn(4) is called. Until t=190ms, then it is cancelled. 1st fn call is at 0ms. fn(4) returns 8. 2nd fn call is at 35ms. fn(4) returns 8. 3rd fn call is at 70ms. fn(4) returns 8. 4th fn call is at 105ms. fn(4) returns 8. 5th fn call is at 140ms. fn(4) returns 8. 6th fn call is at 175ms. fn(4) returns 8. Cancelled at 190ms

Example 2:

Input: fn = (x1, x2) => (x1 * x2), args = [2, 5], t = 30 **Output:** [{"time": 0, "returned": 10}, {"time": 30, "returned": 10}, {"time": 60, "returned": 10}, {"time": 90, "returned": 10}, {"time": 120, "returned": 10}, {"time": 150, "returned": 10}] **Explanation:** const cancelTimeMs = 165; const cancelFn = cancellable((x1, x2) => (x1 * x2), [2, 5], 30) setTimeout(cancelFn,

cancelTimeMs) Every 30ms, fn(2, 5) is called. Until t=165ms, then it is cancelled. 1st fn call is at 0ms 2nd fn call is at 30ms 3rd fn call is at 60ms 4th fn call is at 90ms 5th fn call is at 120ms 6th fn call is at 150ms Cancelled at 165ms

Example 3:

Input: fn = (x1, x2, x3) => (x1 + x2 + x3), args = [5, 1, 3], t = 50 **Output:** [{"time": 0, "returned": 9}, {"time": 50, "returned": 9}, {"time": 100, "returned": 9}, {"time": 150, "returned": 9}] **Explanation:** const cancelTimeMs = 180; const cancelFn = cancellable((x1, x2, x3) => (x1 + x2 + x3), [5, 1, 3], 50) setTimeout(cancelFn, cancelTimeMs) Every 50ms, fn(5, 1, 3) is called. Until t=180ms, then it is cancelled. 1st fn call is at 0ms 2nd fn call is at 50ms 3rd fn call is at 100ms 4th fn call is at 150ms Cancelled at 180ms

Constraints:

* `fn` is a function * `args` is a valid JSON array * `1 <= args.length <= 10` * `30 <= t <= 100` * `10 <= cancelTimeMs <= 500`

Code Snippets

JavaScript:

```
/**
* @param {Function} fn
* @param {Array} args
* @param {number} t
* @return {Function}
*/
var cancellable = function(fn, args, t) {

};

/**
* const result = [];
*
* const fn = (x) => x * 2;
* const args = [4], t = 35, cancelTimeMs = 190;
*
* const start = performance.now();
*
```

```
* const log = (...argsArr) => {
* const diff = Math.floor(performance.now() - start);
* result.push({"time": diff, "returned": fn(...argsArr)});
* }
*
* const cancel = cancellable(log, args, t);
*
* setTimeout(cancel, cancelTimeMs);
*
* setTimeout(() => {
* console.log(result); // [
* // {"time":0,"returned":8},
* // {"time":35,"returned":8},
* // {"time":70,"returned":8},
* // {"time":105,"returned":8},
* // {"time":140,"returned":8},
* // {"time":175,"returned":8}
* // ]
* }, cancelTimeMs + t + 15)
*/
```

TypeScript:

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

function cancellable(fn: Fn, args: JSONValue[], t: number): Function {

};

/**
* const result = [];
*
* const fn = (x) => x * 2;
* const args = [4], t = 35, cancelTimeMs = 190;
*
* const start = performance.now();
*
* const log = (...argsArr) => {
* const diff = Math.floor(performance.now() - start);
* result.push({"time": diff, "returned": fn(...argsArr)});
```

```
* }
*
* const cancel = cancellable(log, args, t);
*
* setTimeout(cancel, cancelTimeMs);
*
* setTimeout(() => {
* console.log(result); // [
* // {"time":0,"returned":8},
* // {"time":35,"returned":8},
* // {"time":70,"returned":8},
* // {"time":105,"returned":8},
* // {"time":140,"returned":8},
* // {"time":175,"returned":8}
* // ]
* }, cancelTimeMs + t + 15)
*/
```