**DSA Introduction: Efficiency**

**Instructions**

Algorithms are solutions to well defined problems. In this challenge you are presented with two computational problems. For each of these computational problems you are presented with two algorithms that solves the given problem. You are required to implement both algorithms and observe their behaviour.

In particular, pay attention to the behaviour of the algorithms as the inputs increase in size.

**Note**: If the tests fail with a timeout error (Execution Timed Out (12000 ms)) simply wait a moment and try again.

**Existing Files**

| **File** | **Description** |
| --- | --- |
| \_\_tests\_\_/cable.test.js | Contains tests for the cable length problem. You are welcome to look at this file but do not modify it. |
| \_\_tests\_\_/missing.test.js | Contains tests for the missing number problem. You are welcome to look at this file but do not modify it. |
| src/missing.js | Write your solutions to the missing number problem in this file |
| src/cable.js | Write your solutions to the cable length problem in this file |

**Tasks**

**Important**: When you run the tests, the number of nanoseconds that the functions took to run will be logged in the output. Take a note of those numbers before moving on to the next challenge.

**Problem 1: The missing number**

A marathon organizer numbered the runners in the marathon from 1 to N, where N is the number of runners taking part in the race. As each runner passes the finish line their number is recorded in a list. It was found that only N - 1 numbers were recorded. That means that one of the runners did not pass the finish line. The organizer of the race wants to know which runner was missing from the list.

You are given an array of the numbers of the N-1 runners in the order in which they pass the finish line. Write an algorithm that will determine the missing number.

For example, if there were 10 runners and the array [8, 3, 4, 1, 2, 9, 10, 5, 7] was given, the algorithm will return 6 as the missing number.

The following assumptions hold true:

* N is an integer number in the range 1 to 100,000. That is, there could be as much as 100,000 runners in the race
* The array contain distinct numbers. That is, no duplicates. After all, a runner would not pass the finish line twice.

**The missing number algorithm 1**

The following algorithm is written in pseudocode. Your task is to implement it in JavaScript.

This algorithm works by iterating over all numbers from 1 to N. For each number, it searches the array for that number. If the number is found, the algorithm continues. If it is not found, the number that was not found is returned.

function missing1

// accepts: A an array of N - 1 numbers

declare a variable N and set it equal to the length of A + 1

for i = 1 to N do

declare variable found and set it to false

declare variable j and set it to 0

while found is false AND j is less than the length of A do

if i is equal to A[j] then

set found to true

increment j

if found is false then

return i

Write this algorithm in the file named src/missing.js.

**The missing number algorithm 2**

The following algorithm is written in pseudocode. Your task is to implement it in JavaScript.

This algorithm works by first calculating the sum of all the numbers in the array. Then, it finds the sum of all the numbers from 1 to N inclusive. Since each number in the array is distinct the missing number is the difference between the sum of all the numbers from 1 to N and the sum of the numbers in the array.

function missing2

// accepts: A an array of N - 1 numbers

declare a variable N and set it equal to the length of A + 1

declare a variable fullSum and set it to the sum of numbers from 1 to N inclusive

declare a variable sum. Find the sum of all numbers in A and store the sum in the variable sm

return the difference between fullSum and sum (i.e. fullSum - sum)

Write this algorithm in the file named src/missing.js

**Problem 2: The cable problem**

A company has been hired to install some cable between two locations in the country. After installing some of the cable they realized that they did not have enough to complete the job. Cable is sold in fixed length pieces. The company wishes to know the minimum number of lengths of cable that they need to buy to finish the job.

For example, if the company has to install cable over a distance of 10 metres and they have 2.5 metres of cable and cable is sold in 2 metre lengths then they need to buy a minimum of 4 lengths of cable. 4 lengths of cable at 2 metres per length gives a total of 8 metres plus the original 2.5 metres means they have 10.5 metres of cable to complete the job.

You are given three numbers:

1. currentLength - The length of cable that is already posessed by the company
2. requiredLength - The total length of cable needed for the task
3. saleLength - The length of a single piece of cable that can be bought

Calculate the minimum number of lengths of cable that need to be purchased

The following assumptions hold true:

* currentLength, requiredLegth and saleLength are integers in the range 1 to 1,000,000,000
* currentlength is less than or equal to requiredLength

**The cable problem algorithm 1**

The following algorithm is written in pseudocode. Your task is to implement it in JavaScript.

This algorithm works by iterating from currentlength to requiredLength in steps of saleLength and counting the number of steps.

function cable1

// accepts: currentLength - an integer between 1 and 1,000,000,000

// requireLength - an integer between 1 and 1,000,000,000

// saleLength - an integer between 1 and 1,000,000,000

declare a variable named count and set it equal to 0

declare a variable named i and set it equal to currentLength

while i is less than requiredLength do

Add saleLength to i

increment count

return count

Write this algorithm in the file named src/cable.js

**The cable problem algorithm 2**

The following algorithm is written in pseudocode. Your task is to implement it in JavaScript.

This algorithm calculates the remaining distance then calculates the number of lengths of cable needed as the smallest integer larger than the quotient of the remaining distance divided by the saleLength.

function cable2

// accepts: currentLength - an integer between 1 and 1,000,000,000

// requireLength - an integer between 1 and 1,000,000,000

// saleLength - an integer between 1 and 1,000,000,000

declare a variable named distance, assign it to the difference betwen requiredLength and currentlength

declare a vaiable named total,

Divide distance by saleLength and round up to the nearest integer, assign this value to total

return total

Write this algorithm in the file named src/cable.js