**DSA Introduction: Assignment - Greatest Common Divisor**

**Instructions**

The Greatest Common Divisor (GCD) of two positive integers a and b is the largest of the common divisors of a and b. If a number c can be divided by another number d without leaving a remainder we say that d is a **divisor** of c.

For example, the GCD of 24 and 30 is 6 because, the divisors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24 and the divisors of 30 are 1, 2, 3, 5, 6, 10, 15, and 30.  
The common divisors of 24 and 30 are 1, 2, 3, and 6. And 6 is the largest of these common divisors.

**Existing Files**

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

**Tasks**

You are given two different algorithms that solve the GCD problem. For each of these algorithms implement them in JavaScript and observe their behaviour as they run.

**Algorithm 1**

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

This algorithm was developed by Euclid ( circa 300 B.C.). The steps are:

function gcd1

// accepts a - a non-negative integer

// b - a non-negative integer

if a is equal to 0 then the GCD of a and b is b, so return b

if b is equal to 0 then the GCD of a and b is a, so return a

while b is not 0 do

let r be the remainder of dividing a by b

set a to equal b

set b to equal r

return a

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

**Algorithm 2**

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

This is a brute force approach to solving the GCD problem. It iterates over all the divisors of both a and b and finds the common ones then finds the largest.

function gcd2

// accepts a - a non-negative integer

// b - a non-negative integer

if a is equal to 0 then the GCD of a and b is b, so return b

if b is equal to 0 then the GCD of a and b is a, so return a

initialize an array named divA with the value [1, a]

initialize an array named divB with the value [1, b]

initialize an empty array named common

for i = 2 to a - 1 do

if a can be divided by i without a remainder then

add i to the array divA

for i = 2 to b - 1 do

if b can be divided by i without a remainder then

add i to the array divB

for each number n in the array divA do:

if n is in the array divB then

add n to the array common

initialize a variable named largest to the smallest possible number

for each number n in commmon do:

if n is greater than largest then

assign largest the value n

return largest

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

**DSA Introduction: Assignment - Split Sum**

**Instructions**

You have been asked to help organize tours for groups of school children to view the Butterfly Pavilion at the museum. To your surprise the exhibit is quite popular and many groups have registered an interest in the tour.

You decide that you will have to hold the tours across two days to accommodate the groups. You need to select a number of groups to visit on the first day and a number of groups to visit on the second day such that each day has roughly the same number of visitors.

You have the sizes of each group recorded in an array. You wish to select the point at which to split the array so that the difference between the sums of each part of the array is minimum.

For example, suppose the following array of group sizes were given.

[4, 12, 17, 8, 13, 24, 9]

If you split at index 2, the first part of the array contains 4, 12, and 17 giving the sum 33. The second part of the array contains the numbers 8, 13, 24, and 9 giving the sum 54 giving the sum 54. The difference is 54 - 33 = 21.

However, if you split at index 3, the first part of the array contains 4, 12, 17, and 8 giving the sum 41 and the second part of the array contains 13, 24, and 9 giving the sum 46. The difference here is 46 - 41 = 5.

The objective is to find the smallest possible difference over all splits.

**Existing Files**

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

**Tasks**

You are given two different algorithms that solve the Split Sum problem. For each of these algorithms implement them in JavaScript and observe their behaviour as they run.

**Algorithm 1**

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

This algorithm works by first finding the sum of the whole array then iterating through the array. At each position i in the array the sum from 0 to i is calculated from the total sum and the sum from i + 1 to the end of the array is calculated from the total sum. These two sums are compared.

function splitSum1

// accepts tours - an array of group sizes

calculate the sum of all numbers in the tours array and assign it to a variable named total

initialize a variable named preSum to the value 0

initialize a variable named postSum to the value of total

initialize a variable named smallest to the largest posible number

for i = 0 to the length of tours - 1 do:

set preSum to the sum of preSum and tours[i]

set postSum to postSum - tours[i]

find the absolute difference between preSum and postSum

if the difference is less than smallest then

set smallest to the difference

return smallest

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

**Algorithm 2**

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

This algorithm once again iterates through the array of numbers. At each position i in the array the sum of the numbers from position 0 to i is calculated and the sum of the numbers from i + 1 to the end of the array is calculated. The difference is found and compared to the smallest number so far.

function splitSum2

// accepts tours - an array of group sizes

initialize a variable named smallest to the largest posible number

for k = 0 to the length of the tours array - 1 do:

calculate the sum of numbers from index 0 to index i of tours and assign to a variable named preSum

calculate the sum of the numbers from index i + 1 to the end of tours and assign to a variable named postSum

find the absolute difference between preSum and postSum

if the difference is less than smallest then

set smallest to the difference

return smallest

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