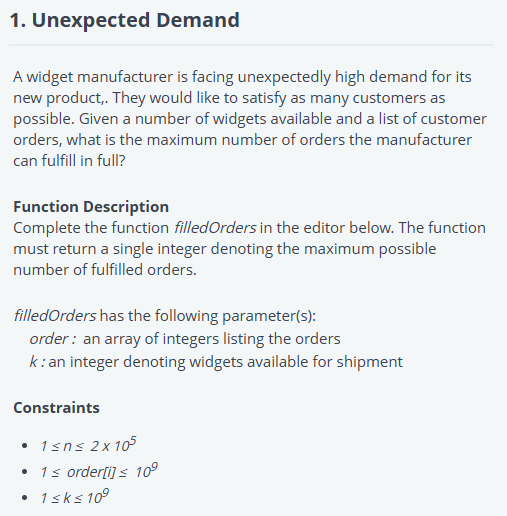
# Unexpected Demand



'use strict';

const fs = require('fs');

process.stdin.resume();

process.stdin.setEncoding('utf-8');

let inputString = '';

let currentLine = 0;

process.stdin.on('data', function (inputStdin) {

    inputString += inputStdin;

});

process.stdin.on('end', function () {

    inputString = inputString.split('\n');

    main();

});

function readLine() {

    return inputString[currentLine++];

}

/\*

 \* Complete the 'filledOrders' function below.

 \*

 \* The function is expected to return an INTEGER.

 \* The function accepts following parameters:

 \*  1. INTEGER\_ARRAY order

 \*  2. INTEGER k

 \*/

function filledOrders(order, k) {

    // Write your code here

    order.sort((a, b) => a - b);

    let fulfilledOrders = 0;

    // Try fulfilling the orders from the sorted list

    for (let i = 0; i < order.length; i++) {

        if (order[i] <= k) {

            fulfilledOrders++; // Fulfill this order

            k -= order[i];     // Decrease available widgets

        } else {

            break;  // Stop if there are not enough widgets for the next order

        }

    }

    return fulfilledOrders; // Return the number of orders that can be fulfilled

}

function main() {

    const ws = fs.createWriteStream(process.env.OUTPUT\_PATH);

    const orderCount = parseInt(readLine().trim(), 10);

    let order = [];

    for (let i = 0; i < orderCount; i++) {

        const orderItem = parseInt(readLine().trim(), 10);

        order.push(orderItem);

    }

    const k = parseInt(readLine().trim(), 10);

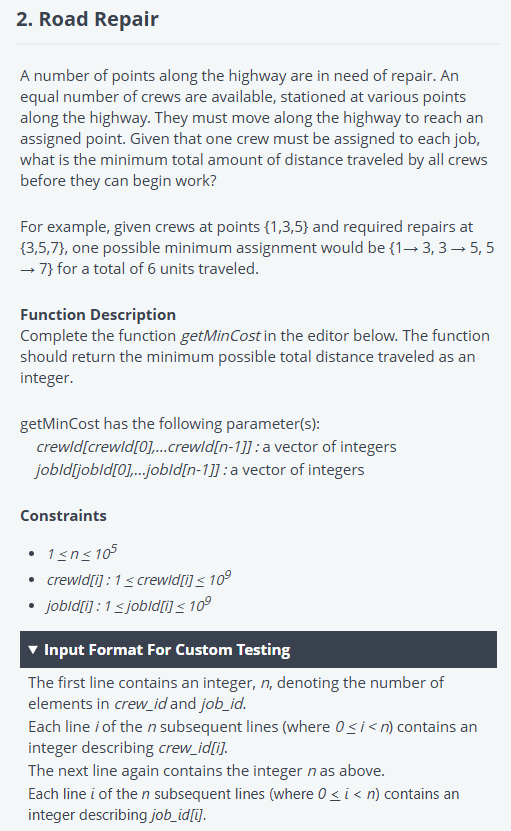
    const result = filledOrders(order, k);

    ws.write(result + '\n');

    ws.end();

}

# Road Repair



'use strict';

const fs = require('fs');

process.stdin.resume();

process.stdin.setEncoding('utf-8');

let inputString = '';

let currentLine = 0;

process.stdin.on('data', function(inputStdin) {

    inputString += inputStdin;

});

process.stdin.on('end', function() {

    inputString = inputString.split('\n');

    main();

});

function readLine() {

    return inputString[currentLine++];

}

/\*

 \* Complete the 'getMinCost' function below.

 \*

 \* The function is expected to return a LONG\_INTEGER.

 \* The function accepts following parameters:

 \*  1. INTEGER\_ARRAY crew\_id

 \*  2. INTEGER\_ARRAY job\_id

 \*/

function getMinCost(crew\_id, job\_id) {

    // Write your code here

    crew\_id.sort((a, b) => a - b);

    job\_id.sort((a, b) => a - b);

    let totalCost = 0;

    // Calculate total cost by pairing sorted crew\_id and job\_id

    for (let i = 0; i < crew\_id.length; i++) {

        totalCost += Math.abs(crew\_id[i] - job\_id[i]);

    }

    return totalCost; // Return the minimum total distance

}

function main() {

    const ws = fs.createWriteStream(process.env.OUTPUT\_PATH);

    const crew\_idCount = parseInt(readLine().trim(), 10);

    let crew\_id = [];

    for (let i = 0; i < crew\_idCount; i++) {

        const crew\_idItem = parseInt(readLine().trim(), 10);

        crew\_id.push(crew\_idItem);

    }

    const job\_idCount = parseInt(readLine().trim(), 10);

    let job\_id = [];

    for (let i = 0; i < job\_idCount; i++) {

        const job\_idItem = parseInt(readLine().trim(), 10);

        job\_id.push(job\_idItem);

    }

    const result = getMinCost(crew\_id, job\_id);

    ws.write(result + '\n');

    ws.end();

}