Given a string **sequence** consisting of the characters **'('**, **')'**, **'['**, **']'**, **'{'**, and **'}'**. Your task is to determine whether or not the **sequence** is a **valid** bracket sequence.

The **Valid** bracket sequence is defined in the following way:

* An empty bracket sequence is a valid bracket sequence.
* If **S** is a valid bracket sequence then **(S)**, **[S]** and **{S}** are also valid.
* If **A** and **B** are valid bracket sequences then **AB** is also valid.

Example

* For **sequence = "()"**, the output should be **solution(sequence) = true**;
* For **sequence = "()[]{}"**, the output should be **solution(sequence) = true**;
* For **sequence = "(]"**, the output should be **solution(sequence) = false**;
* For **sequence = "([)]"**, the output should be **solution(sequence) = false**;
* For **sequence = "{[]}"**, the output should be **solution(sequence) = true**.

Input/Output

* **[execution time limit] 4 seconds (js)**
* **[input] string sequence**

A bracket sequence, consisting of the characters **(**, **)**, **[**, **]**, **{**, and **}**.

*Guaranteed constraints:*  
**0 ≤ sequence.length ≤ 106**.

* **[output] boolean**

**true** if **sequence** is a valid bracket sequence and **false** otherwise.

**[JavaScript] Syntax Tips**

**// Prints help message to the console**

**// Returns a string**

**function helloWorld(name) {**

**console.log("This prints to the console when you Run Tests");**

**return "Hello, " + name;**

**}**

Given two strings **a** and **b**, both consisting only of lowercase English letters, your task is to calculate how many strings equal to **a** can be constructed using only letters from the string **b**? Each letter can be used only once and in one string only.

Example

* For **a = "abc"** and **b = "abccba"**, the output should be **solution(a, b) = 2**.

We can construct **2** strings **a = "abc"** using only letters from the string **b**.

* For **a = "ab"** and **b = "abcbcb"**, the output should be **solution(a, b) = 1**.

Input/Output

* **[execution time limit] 4 seconds (js)**
* **[input] string a**

String to construct, containing only lowercase English letters.

Guaranteed constraints:  
**1 ≤ a.length ≤ 105**.

* **[input] string b**

String containing needed letters, containing only lowercase English letters.

Guaranteed constraints:  
**1 ≤ b.length ≤ 105**.

* **[output] integer**

The number of strings **a** that can be constructed from the string **b**.

**[JavaScript] Syntax Tips**

**// Prints help message to the console**

**// Returns a string**

**function helloWorld(name) {**

**console.log("This prints to the console when you Run Tests");**

**return "Hello, " + name;**

**}**

You have an array **p** of points on a Cartesian plane. Find and return the minimum possible [Euclidian distance](http://www.cut-the-knot.org/pythagoras/DistanceFormula.shtml) between two points with different indices in **p**.

Example

For **p = [[0, 11], [-7, 1], [-5, -3]]**, the output should be  
**solution(p) = 4.472135955**.

Input/Output

* **[execution time limit] 4 seconds (js)**
* **[input] array.array.integer p**

Every inner array **p[i]** contains exactly 2 integers: the **x** and **y** coordinates of the **ith** point.

Guaranteed constraints:  
**2 ≤ p.length ≤ 2 · 104**,  
**p[i].length = 2**,  
**|p[i][j]| ≤ 107**.

* **[output] float**

The minimum possible distance between two points with different indices in **p**.

Your answer will be considered correct if its absolute error doesn't exceed **10-5**.

**[JavaScript] Syntax Tips**

**// Prints help message to the console**

**// Returns a string**

**function helloWorld(name) {**

**console.log("This prints to the console when you Run Tests");**

**return "Hello, " + name;**

**}**