Alice is taking a cryptography class and finding *anagrams* to be very useful. We consider two strings to be anagrams of each other if the first string's letters can be rearranged to form the second string. In other words, both strings must contain the same exact letters in the same exact frequency For example, bacdc and dcbac are anagrams, but bacdc and dcbad are not.

Alice decides on an encryption scheme involving two large strings where encryption is dependent on the minimum number of character deletions required to make the two strings anagrams. Can you help her find this number?

Given two strings,  and , that may or may not be of the same length, determine the minimum number of character deletions required to make  and  anagrams. Any characters can be deleted from either of the strings.

This challenge is also available in the following translations:

* [Chinese](https://hr-filepicker.s3.amazonaws.com/feb14/chinese/1364-chinese.md)
* [Russian](https://hr-filepicker.s3.amazonaws.com/feb-14/russian/1364-russian.md)

**Input Format**

The first line contains a single string, .   
The second line contains a single string, .

**Constraints**

* It is guaranteed that  and  consist of lowercase English alphabetic letters (i.e.,  through ).

**Output Format**

Print a single integer denoting the number of characters you must delete to make the two strings anagrams of each other.

**Sample Input**

cde

abc

**Sample Output**

4

**Explanation**

We delete the following characters from our two strings to turn them into anagrams of each other:

1. Remove d and e from cde to get c.
2. Remove a and b from abc to get c.

We must delete  characters to make both strings anagrams, so we print  on a new line.

Answer:

from collections import \*

a = Counter(raw\_input())

b = Counter(raw\_input())

c = a - b

d = b - a

e = c + d

print len(list(e.elements()))

<https://docs.python.org/2/library/collections.html>