**Lexicographically Smallest Equivalent String**

You are given two strings of the same length s1 and s2 and a string baseStr.

We say s1[i] and s2[i] are equivalent characters.

* For example, if s1 = "abc" and s2 = "cde", then we have 'a' == 'c', 'b' == 'd', and 'c' == 'e'.

Equivalent characters follow the usual rules of any equivalence relation:

* **Reflexivity:** 'a' == 'a'.
* **Symmetry:** 'a' == 'b' implies 'b' == 'a'.
* **Transitivity:** 'a' == 'b' and 'b' == 'c' implies 'a' == 'c'.

For example, given the equivalency information from s1 = "abc" and s2 = "cde", "acd" and "aab" are equivalent strings of baseStr = "eed", and "aab" is the lexicographically smallest equivalent string of baseStr.

Return *the lexicographically smallest equivalent string of*baseStr*by using the equivalency information from*s1*and*s2.

**Example 1:**

**Input:** s1 = "parker", s2 = "morris", baseStr = "parser"

**Output:** "makkek"

**Explanation:** Based on the equivalency information in s1 and s2, we can group their characters as [m,p], [a,o], [k,r,s], [e,i].

The characters in each group are equivalent and sorted in lexicographical order.

So the answer is "makkek".

**Example 2:**

**Input:** s1 = "hello", s2 = "world", baseStr = "hold"

**Output:** "hdld"

**Explanation:** Based on the equivalency information in s1 and s2, we can group their characters as [h,w], [d,e,o], [l,r].

So only the second letter 'o' in baseStr is changed to 'd', the answer is "hdld".

**Example 3:**

**Input:** s1 = "leetcode", s2 = "programs", baseStr = "sourcecode"

**Output:** "aauaaaaada"

**Explanation:** We group the equivalent characters in s1 and s2 as [a,o,e,r,s,c], [l,p], [g,t] and [d,m], thus all letters in baseStr except 'u' and 'd' are transformed to 'a', the answer is "aauaaaaada".

**Constraints:**

* 1 <= s1.length, s2.length, baseStr <= 1000
* s1.length == s2.length
* s1, s2, and baseStr consist of lowercase English letters.