1790. Check if One String Swap Can Make Strings Equal

You are given two strings `s1` and `s2` of equal length. A **string swap** is an operation where you choose two indices in a string (not necessarily different) and swap the characters at these indices.

Return `true` *if it is possible to make both strings equal by performing* ***at most one string swap*** *on* ***exactly one*** *of the strings.* Otherwise, return `false`.

Example 1:

```
Input: s1 = "bank", s2 = "kanb"
Output: true
Explanation: For example, swap the first character with the last character of s2 to make "bank".
```

Example 2:

```
Input: s1 = "attack", s2 = "defend"
Output: false
Explanation: It is impossible to make them equal with one string swap.
```

Example 3:

```
Input: s1 = "kelb", s2 = "kelb"
Output: true
Explanation: The two strings are already equal, so no string swap operation is required.
```

Constraints:

- `1 <= s1.length, s2.length <= 100`
- `s1.length == s2.length`
- `s1` and `s2` consist of only lowercase English letters.

Overview

We are given two strings `s1` and `s2` and want to determine if it is possible to make the strings equal by performing at most one string swap on one of the strings. A string swap involves choosing any two indices in the string and swapping the characters at these positions.

1790. Check if One String Swap Can Make Strings Equal

```
/*
   Sorting+swap counting
   Time complexity: O(2nlogn+n)
   Space compelxity: O(2n)
*/
class Solution{
   public:
      bool areAlmostEqual(std::string& s1, std::string& s2) {
         int n=s1.size();

         // Save originals strings
         std::string s11=s1,s22=s2;

         // Sort both saved strings
         std::sort(s11.begin(),s11.end());
         std::sort(s22.begin(),s22.end());

         // If there are not eqaul return false
         if(s11!=s22) return false;

         // Otherwise, count number of swaps on original strings
        // If we found two different letters on same postion, means we need a swap
        // means for 1 swap, we count 2 (for 1 swap, we have 2 pairs of different letters at same index)
         int count_pairs=0;
         for(int i=0;i<n;++i){
            count_pairs+=(s1[i]!=s2[i]);

            // If #pairs of different letters at same position is more than 2, return false
            if(count_pairs>2) return false;
         }

         return true;
      }
};
```

1790. Check if One String Swap Can Make Strings Equal

```
/*
   swap counting
   Time complexity: O(n)
   Space compelxity: O(1)
*/
class Solution {
   public:
      bool areAlmostEqual(std::string& s1, std::string& s2) {
         int n=s1.size();

         // Count number of swaps on original strings
        // If we found two different letters on same postion, means we need a swap
        // means for 1 swap, we count 2 (for 1 swap, we have 2 pairs of different letters at same index)
         int count_pairs=0;

         // If there is one swap, save the positions
         int first_index=0,second_index=0;
         for(int i=0;i<n;++i){
            if(s1[i]!=s2[i]){
               count_pairs++;
               // If #pairs of different letters at same position is more than 2, return false
               if(count_pairs>2) return false;


               // Otherwise, If we found the first pair, save the first index
               if(count_pairs==1) first_index=i;

              // Otherwise, if we found the second pair(count_pairs==2), we found the second pair, save the second index
               else second_index=i;
            }
         }

        // To make a swap, the letter of string s1 at the first index, should be equal to the letter of string s2 at the second index
       // and the letter of string s1 at the second index, should be equal to the letter of string s2 at the first index
         return s1[first_index]==s2[second_index] && s1[second_index]==s2[first_index];
      }
};
```