# 题目

给你长度相等的两个字符串 s1 和 s2 。一次 字符串交换 操作的步骤如下：选出某个字符串中的两个下标（不必不同），并交换这两个下标所对应的字符。

如果对 其中一个字符串 执行 最多一次字符串交换 就可以使两个字符串相等，返回 true ；否则，返回 false 。

示例 1：

输入：s1 = "bank", s2 = "kanb"

输出：true

解释：例如，交换 s2 中的第一个和最后一个字符可以得到 "bank"

示例 2：

输入：s1 = "attack", s2 = "defend"

输出：false

解释：一次字符串交换无法使两个字符串相等

示例 3：

输入：s1 = "kelb", s2 = "kelb"

输出：true

解释：两个字符串已经相等，所以不需要进行字符串交换

示例 4：

输入：s1 = "abcd", s2 = "dcba"

输出：false

提示：

1 <= s1.length, s2.length <= 100

s1.length == s2.length

s1 和 s2 仅由小写英文字母组成

# 分析

## 方法一：计数统计

思路：

可以通过比较s1和s2中不同字符的数量以及它们的位置来确定是否可以通过一次交换使它们相等。

代码：

class Solution {

public:

bool areAlmostEqual(string s1, string s2) {

int diffCount = 0;

vector<int> diffIndices;

for (int i = 0; i < s1.length(); i++) {

if (s1[i] != s2[i]) {

diffCount++;

diffIndices.push\_back(i);

}

if (diffCount > 2) {

return false; // 如果不同字符数大于2，无法通过一次交换使其相等

}

}

return diffCount == 0 || (diffCount == 2 && s1[diffIndices[0]] == s2[diffIndices[1]] && s1[diffIndices[1]] == s2[diffIndices[0]]);

}

};

这段代码首先遍历s1和s2，找出它们不同的字符以及对应的下标。如果不同字符的数量大于2，那么无法通过一次交换使它们相等，直接返回false。如果不同字符的数量为0，说明两个字符串已经相等，返回true。如果不同字符的数量为2，那么检查这两个不同字符在两个字符串中的位置是否互相交换后可以使得两个字符串相等，如果是则返回true，否则返回false。