# 题目

给定两个由小写字母组成的字符串 s1 和 s2，请编写一个程序，确定其中一个字符串的字符重新排列后，能否变成另一个字符串。

示例 1：

输入: s1 = "abc", s2 = "bca"

输出: true

示例 2：

输入: s1 = "abc", s2 = "bad"

输出: false

说明：

0 <= len(s1) <= 100

0 <= len(s2) <= 100

# 分析

## 方法一：排序

思路：

代码：

class Solution {

public:

bool CheckPermutation(string s1, string s2) {

if (s1.length() != s2.length()) {

return false;

}

sort(s1.begin(), s1.end());

sort(s2.begin(), s2.end());

return s1 == s2;

}

};

## 方法二：哈希表

思路：

代码：

class Solution {

public:

bool CheckPermutation(std::string s1, std::string s2) {

// 如果字符串长度不同，则它们不可能是排列

if (s1.length() != s2.length()) {

return false;

}

// 使用哈希表记录字符的差值

std::unordered\_map<char, int> charDiff;

// 遍历两个字符串，同时更新字符计数的差值

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

charDiff[s1[i]]++;

charDiff[s2[i]]--;

}

// 检查哈希表中所有字符的计数是否都为零

for (const auto& pair : charDiff) {

if (pair.second != 0) {

return false;

}

}

return true;

}

};

或：

class Solution {

public:

bool CheckPermutation(string s1, string s2) {

if (s1.length() != s2.length()) {

return false;

}

vector<int> table(128, 0);

for (auto& ch: s1) {

table[ch]++;

}

for (auto& ch: s2) {

table[ch]--;

if (table[ch] < 0) {

return false;

}

}

return true;

}

};