# 题目

给你两个字符串数组 words1 和 words2 ，请你返回在两个字符串数组中都恰好出现一次的字符串的数目。

示例 1：

输入：words1 = ["leetcode","is","amazing","as","is"], words2 = ["amazing","leetcode","is"]

输出：2

解释：

- "leetcode" 在两个数组中都恰好出现一次，计入答案。

- "amazing" 在两个数组中都恰好出现一次，计入答案。

- "is" 在两个数组中都出现过，但在 words1 中出现了 2 次，不计入答案。

- "as" 在 words1 中出现了一次，但是在 words2 中没有出现过，不计入答案。

所以，有 2 个字符串在两个数组中都恰好出现了一次。

示例 2：

输入：words1 = ["b","bb","bbb"], words2 = ["a","aa","aaa"]

输出：0

解释：没有字符串在两个数组中都恰好出现一次。

示例 3：

输入：words1 = ["a","ab"], words2 = ["a","a","a","ab"]

输出：1

解释：唯一在两个数组中都出现一次的字符串是 "ab" 。

提示：

1 <= words1.length, words2.length <= 1000

1 <= words1[i].length, words2[j].length <= 30

words1[i] 和 words2[j] 都只包含小写英文字母。

# 分析

## 方法一：哈希表

思路：

代码：

class Solution {

public:

int countWords(vector<string>& words1, vector<string>& words2) {

// 统计字符串出现频率

unordered\_map<string, int> freq1, freq2;

for (const string& w: words1) {

++freq1[w];

}

for (const string& w: words2) {

++freq2[w];

}

// 遍历 words1 出现的字符并判断是否满足要求

int res = 0;

for (const auto& [w, cnt1] : freq1) {

if (cnt1 == 1 && freq2[w] == 1) {

++res;

}

}

return res;

}

};

复杂度分析

时间复杂度：O(m+n)，其中m为word1中所有字符串的长度之和，n为word2中所有字符串的总长度。即为维护哈希表和遍历统计出现过一次的公共字符串数量的时间复杂度。

空间复杂度：O(m+n)O(m + n)O(m+n)，即为哈希表的空间开销。