# 题目

给你一个下标从 0 开始的 环形 字符串数组 words 和一个字符串 target 。环形数组 意味着数组首尾相连。

形式上， words[i] 的下一个元素是 words[(i + 1) % n] ，而 words[i] 的前一个元素是 words[(i - 1 + n) % n] ，其中 n 是 words 的长度。

从 startIndex 开始，你一次可以用 1 步移动到下一个或者前一个单词。

返回到达目标字符串 target 所需的最短距离。如果 words 中不存在字符串 target ，返回 -1 。

示例 1：

输入：words = ["hello","i","am","leetcode","hello"], target = "hello", startIndex = 1

输出：1

解释：从下标 1 开始，可以经由以下步骤到达 "hello" ：

- 向右移动 3 个单位，到达下标 4 。

- 向左移动 2 个单位，到达下标 4 。

- 向右移动 4 个单位，到达下标 0 。

- 向左移动 1 个单位，到达下标 0 。

到达 "hello" 的最短距离是 1 。

示例 2：

输入：words = ["a","b","leetcode"], target = "leetcode", startIndex = 0

输出：1

解释：从下标 0 开始，可以经由以下步骤到达 "leetcode" ：

- 向右移动 2 个单位，到达下标 3 。

- 向左移动 1 个单位，到达下标 3 。

到达 "leetcode" 的最短距离是 1 。

示例 3：

输入：words = ["i","eat","leetcode"], target = "ate", startIndex = 0

输出：-1

解释：因为 words 中不存在字符串 "ate" ，所以返回 -1 。

提示：

1 <= words.length <= 100

1 <= words[i].length <= 100

words[i] 和 target 仅由小写英文字母组成

0 <= startIndex < words.length

# 分析

## 方法一：一次遍历

思路：

代码：

class Solution {

public:

int closetTarget(vector<string>& words, string target, int startIndex) {

int n = words.size();

int minDist = INT\_MAX;

for (int i = 0; i < n; ++i) {

if (words[i] == target) {

int dist = min(abs(i - startIndex), n - abs(i - startIndex));

minDist = min(minDist, dist);

}

}

return minDist == INT\_MAX ? -1 : minDist;

}

};