# 题目

给你一个字符串 s 和一个字符 c ，且 c 是 s 中出现过的字符。

返回一个整数数组 answer ，其中 answer.length == s.length 且 answer[i] 是 s 中从下标 i 到离它 最近 的字符 c 的 距离 。

两个下标 i 和 j 之间的 距离 为 abs(i - j) ，其中 abs 是绝对值函数。

示例 1：

输入：s = "loveleetcode", c = "e"

输出：[3,2,1,0,1,0,0,1,2,2,1,0]

解释：字符 'e' 出现在下标 3、5、6 和 11 处（下标从 0 开始计数）。

距下标0最近的'e'出现在下标3，所以距离为abs(0 - 3) = 3 。

距下标1最近的'e'出现在下标3，所以距离为abs(1 - 3) = 2 。

对于下标4，出现在下标3和下标5处的'e'都离它最近，但距离是一样的abs(4 - 3) == abs(4 - 5) = 1。

距下标8最近的'e'出现在下标6，所以距离为abs(8 - 6) = 2。

示例 2：

输入：s = "aaab", c = "b"

输出：[3,2,1,0]

提示：

1 <= s.length <= 104

s[i] 和 c 均为小写英文字母

题目数据保证 c 在 s 中至少出现一次

# 分析

## 方法一：两次遍历

class Solution {

public:

vector<int> shortestToChar(string s, char c) {

int n = s.size();

vector<int> answer(n, INT\_MAX);

int pos = -1;

// 从左往右遍历，更新每个字符到字符 c 的最近距离

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

if (s[i] == c) {

pos = i;

}

if (pos != -1) {

answer[i] = i - pos;

}

}

pos = -1;

// 从右往左遍历，更新每个字符到字符 c 的最近距离

for (int i = n - 1; i >= 0; --i) {

if (s[i] == c) {

pos = i;

}

if (pos != -1) {

answer[i] = min(answer[i], pos - i);

}

}

return answer;

}

};

说明：

第二次遍历从右往左是为了确保对于每个字符，找到其左边和右边离它最近的字符 c 的距离中的最小值。

假设我们只进行一次遍历，从左往右，对于字符串 "loveleetcode" 和字符 'e'，如果我们只考虑从左往右的遍历，那么对于下标 4 的字符 'v' 来说，它离字符 'e' 最近的字符是下标 3 的字符 'e'，距离为 1。但是，实际上下标 5 的字符 'e' 离字符 'e' 更近，距离为 0。所以我们需要进行第二次遍历，从右往左，来更新距离，确保最终得到的是每个字符到字符 c 的最近距离。

通过这样的两次遍历，我们可以确保得到正确的结果。