字符的最短距离

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) = 3 。

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]

```
class Solution {
public:
    vector<int> shortestToChar(string s, char c) {
        /*
        设置两个指针，遍历每一个字符，以每一个字符为中心，在这个中心向左右方向（用left，right 指针实现）寻找离这个中心最近的字符 c
        */
        vector<int> res;
        for(int i=0;i<s.size();i++)
        {
            int left=i;
            int right=i;
            while((left>0||right<s.size())&&s[left]!=c&&s[right]!=c)
            {
                if(left>0)
```

```
                    left--;
                if(right<s.size())
                    right++;
            }
            //此时 left  right 找到指向离 i 距离最近的字符 c  至少一个找到最近的字符 c
            if(s[left]==c)
            {
                res.push_back(abs(left-i));
            }
            else
            {
                res.push_back(abs(right-i));
            }
        }
        return res;
    }
};
```