

```

1  /*
2  给定一个字符串 S 和一个字符 C。返回一个代表字符串 S 中每个字符到字符串 S 中的字符 C 的最
   短距离的数组。
3
4  示例 1:
5
6  输入: S = "loveleetcode", C = 'e'
7  输出: [3, 2, 1, 0, 1, 0, 0, 1, 2, 2, 1, 0]
8  说明:
9
10 字符串 S 的长度范围为 [1, 10000]。
11  C 是一个单字符,且保证是字符串 S 里的字符。
12  S 和 C 中的所有字母均为小写字母。
13
14 来源: 力扣 (LeetCode)
15 链接: https://leetcode-cn.com/problems/shortest-distance-to-a-character
16 著作权归领扣网络所有。商业转载请联系官方授权,非商业转载请注明出处。
17 */

```

分析:

- 暴力法:
 - 首先遍历一次S,找到所有C的索引
 - 再遍历一次S,然后依次计算距离并保存最小值
 - 时间复杂度为 $O(M * N)$,其中M为S的长度, N是S中C的个数。
- 双索引法:
 - 遍历S的所有位置,若当前位置的值不为C,就从当前位置开始分别往S的左半部分和右半部分遍历直到遇到C,然后计算距离保存即可。

方法一:C++_暴力法

```

1  class Solution
2  {
3
4      public:
5          vector<int> shortestToChar(string S, char C)
6          {
7              vector<int> ret_val      ;
8              vector<int> temp_ind     ;
9              int i = 0 ;
10             int j = 0 ;
11             int min_ind = 0 ;
12
13             /*找到S中所有C的索引*/
14             for(i=0;i<S.size();i++)
15             {
16                 if(C==S.at(i))
17                 {
18                     temp_ind.push_back(i);
19                 }
20             }

```

```

21
22      /*遍历S中的每一个元素并计算*/
23      for(i=0;i<S.size();i++)
24      {
25          min_ind = 10000;
26          if(S.at(i)==C)
27          {
28              min_ind = 0;
29          }
30          else
31          {
32              for(j=0;j<temp_ind.size();j++)
33              {
34                  if(abs( temp_ind[j] - i ) < min_ind)
35                  {
36                      min_ind = abs( temp_ind[j] - i );
37                  }
38              }
39          }
40          ret_val.push_back(min_ind);
41      }
42      return ret_val;
43  }
44 };
45 /*
46 执行结果:
47 通过
48 显示详情
49 执行用时 :20 ms, 在所有 C++ 提交中击败了47.42%的用户
50 内存消耗 :8.9 MB, 在所有 C++ 提交中击败了74.80%的用户
51 */

```

方法二:C++_双索引法

```

1  class solution
2  {
3      public:
4          vector<int> shortestToChar(string S, char C)
5          {
6              vector<int> ret_val ;
7              int len = S.size();
8              int i = 0;
9              int left = 0;
10             int right = 0;
11             char ch = 0;
12             int min_ind = 0;
13
14             for(i = 0 ; i < len ; i++)
15             {
16                 ch = S.at(i);
17                 if(ch == C)
18                 {
19                     ret_val.push_back(0);
20                 }
21                 else
22                 {

```

```

23         left    = i - 1;
24         right   = i + 1;
25         while(1)
26         {
27             if( ( left >=0)
28                 &&( c == S.at(left))
29             )
30             {
31                 ret_val.push_back(i-left);
32                 break;
33             }
34             else
35             {
36                 left--;
37             }
38
39             if( (right < len)
40                 &&(c == S.at(right))
41             )
42             {
43                 ret_val.push_back(right-i);
44                 break;
45             }
46             else
47             {
48                 right++;
49             }
50         }
51     }
52 }
53 return ret_val;
54 }
55 };
56
57 /*
58 执行结果:
59 通过
60 显示详情
61 执行用时 :12 ms, 在所有 C++ 提交中击败了96.01%的用户
62 内存消耗 :8.9 MB, 在所有 C++ 提交中击败了77.17%的用户
63 */

```