

Problem 821: Shortest Distance to a Character

Problem Information

Difficulty: [Easy](#)

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given a string

s

and a character

c

that occurs in

s

, return

an array of integers

answer

where

answer.length == s.length

and

answer[i]

is the

distance

from index

i

to the

closest

occurrence of character

c

in

s

.

The

distance

between two indices

i

and

j

is

$\text{abs}(i - j)$

, where

`abs`

is the absolute value function.

Example 1:

Input:

`s = "loveleetcode", c = "e"`

Output:

`[3,2,1,0,1,0,0,1,2,2,1,0]`

Explanation:

The character 'e' appears at indices 3, 5, 6, and 11 (0-indexed). The closest occurrence of 'e' for index 0 is at index 3, so the distance is $\text{abs}(0 - 3) = 3$. The closest occurrence of 'e' for index 1 is at index 3, so the distance is $\text{abs}(1 - 3) = 2$. For index 4, there is a tie between the 'e' at index 3 and the 'e' at index 5, but the distance is still the same: $\text{abs}(4 - 3) == \text{abs}(4 - 5) = 1$. The closest occurrence of 'e' for index 8 is at index 6, so the distance is $\text{abs}(8 - 6) = 2$.

Example 2:

Input:

`s = "aaab", c = "b"`

Output:

`[3,2,1,0]`

Constraints:

$1 \leq s.\text{length} \leq 10$

4

`s[i]`

and

c

are lowercase English letters.

It is guaranteed that

c

occurs at least once in

s

.

Code Snippets

C++:

```
class Solution {  
public:  
    vector<int> shortestToChar(string s, char c) {  
        }  
    };
```

Java:

```
class Solution {  
public int[] shortestToChar(String s, char c) {  
    }  
}
```

Python3:

```
class Solution:  
    def shortestToChar(self, s: str, c: str) -> List[int]:
```

Python:

```
class Solution(object):
    def shortestToChar(self, s, c):
        """
        :type s: str
        :type c: str
        :rtype: List[int]
        """
```

JavaScript:

```
/**
 * @param {string} s
 * @param {character} c
 * @return {number[]}
 */
var shortestToChar = function(s, c) {
```

TypeScript:

```
function shortestToChar(s: string, c: string): number[] {
```

```
}
```

C#:

```
public class Solution {
    public int[] ShortestToChar(string s, char c) {
        }
}
```

C:

```
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* shortestToChar(char* s, char c, int* returnSize) {

}
```

Go:

```
func shortestToChar(s string, c byte) []int {  
    }  
}
```

Kotlin:

```
class Solution {  
    fun shortestToChar(s: String, c: Char): IntArray {  
        }  
        }  
}
```

Swift:

```
class Solution {  
    func shortestToChar(_ s: String, _ c: Character) -> [Int] {  
        }  
        }  
}
```

Rust:

```
impl Solution {  
    pub fn shortest_to_char(s: String, c: char) -> Vec<i32> {  
        }  
        }  
}
```

Ruby:

```
# @param {String} s  
# @param {Character} c  
# @return {Integer[]}  
def shortest_to_char(s, c)  
  
end
```

PHP:

```
class Solution {
```

```
/**  
 * @param String $s  
 * @param String $c  
 * @return Integer[]  
 */  
function shortestToChar($s, $c) {  
  
}  
}
```

Dart:

```
class Solution {  
List<int> shortestToChar(String s, String c) {  
  
}  
}
```

Scala:

```
object Solution {  
def shortestToChar(s: String, c: Char): Array[Int] = {  
  
}  
}
```

Elixir:

```
defmodule Solution do  
@spec shortest_to_char(s :: String.t, c :: char) :: [integer]  
def shortest_to_char(s, c) do  
  
end  
end
```

Erlang:

```
-spec shortest_to_char(S :: unicode:unicode_binary(), C :: char()) ->  
[integer()].  
shortest_to_char(S, C) ->  
.
```

Racket:

```
(define/contract (shortest-to-char s c)
  (-> string? char? (listof exact-integer?)))
)
```

Solutions

C++ Solution:

```
/*
 * Problem: Shortest Distance to a Character
 * Difficulty: Easy
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public:
vector<int> shortestToChar(string s, char c) {

}
};
```

Java Solution:

```
/**
 * Problem: Shortest Distance to a Character
 * Difficulty: Easy
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public int[] shortestToChar(String s, char c) {
```

```
}
```

```
}
```

Python3 Solution:

```
"""
Problem: Shortest Distance to a Character
Difficulty: Easy
Tags: array, string

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:

def shortestToChar(self, s: str, c: str) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def shortestToChar(self, s, c):
"""
:type s: str
:type c: str
:rtype: List[int]
"""


```

JavaScript Solution:

```
/**
 * Problem: Shortest Distance to a Character
 * Difficulty: Easy
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */
```

```

*/



/**
 * @param {string} s
 * @param {character} c
 * @return {number[]}
 */
var shortestToChar = function(s, c) {

};


```

TypeScript Solution:

```

/**



* Problem: Shortest Distance to a Character
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/



function shortestToChar(s: string, c: string): number[] {

};


```

C# Solution:

```

/*
* Problem: Shortest Distance to a Character
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/



public class Solution {
public int[] ShortestToChar(string s, char c) {


```

```
}
```

```
}
```

C Solution:

```
/*
 * Problem: Shortest Distance to a Character
 * Difficulty: Easy
 * Tags: array, string
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* shortestToChar(char* s, char c, int* returnSize) {

}
```

Go Solution:

```
// Problem: Shortest Distance to a Character
// Difficulty: Easy
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func shortestToChar(s string, c byte) []int {

}
```

Kotlin Solution:

```
class Solution {
    fun shortestToChar(s: String, c: Char): IntArray {
```

```
}
```

```
}
```

Swift Solution:

```
class Solution {  
    func shortestToChar(_ s: String, _ c: Character) -> [Int] {  
  
    }  
}
```

Rust Solution:

```
// Problem: Shortest Distance to a Character  
// Difficulty: Easy  
// Tags: array, string  
//  
// Approach: Use two pointers or sliding window technique  
// Time Complexity: O(n) or O(n log n)  
// Space Complexity: O(1) to O(n) depending on approach  
  
impl Solution {  
    pub fn shortest_to_char(s: String, c: char) -> Vec<i32> {  
  
    }  
}
```

Ruby Solution:

```
# @param {String} s  
# @param {Character} c  
# @return {Integer[]}  
def shortest_to_char(s, c)  
  
end
```

PHP Solution:

```
class Solution {
```

```

/**
 * @param String $s
 * @param String $c
 * @return Integer[]
 */
function shortestToChar($s, $c) {
}
}

```

Dart Solution:

```

class Solution {
List<int> shortestToChar(String s, String c) {
}
}

```

Scala Solution:

```

object Solution {
def shortestToChar(s: String, c: Char): Array[Int] = {
}
}

```

Elixir Solution:

```

defmodule Solution do
@spec shortest_to_char(s :: String.t, c :: char) :: [integer]
def shortest_to_char(s, c) do
end
end

```

Erlang Solution:

```

-spec shortest_to_char(S :: unicode:unicode_binary(), C :: char()) ->
[integer()].
shortest_to_char(S, C) ->
.

```

Racket Solution:

```
(define/contract (shortest-to-char s c)
  (-> string? char? (listof exact-integer?)))
)
```