【华为OD机考 统一考试机试C卷】字符串序列判定/最后一个有效字符（ C++ Java JavaScript python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

2023年11月份，华为官方已经将 华为OD机考：OD统一考试（A卷 / B卷）切换到 OD统一考试（C卷）和 OD统一考试（D卷） 。

真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明

专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）

华为OD面试真题精选：华为OD面试真题精选

在线OJ：点击立即刷题，模拟真实机考环境

题目描述：字符串序列判定/最后一个有效字符（本题分值100）

输入两个字符串S和L，都只包含英文小写字母。S长度<=100，L长度<=500,000。判定S是否是L的有效子串。

判定规则：

S中的每个字符在L中都能找到（可以不连续），

且S在 L 中字符的前后顺序与S中顺序要保持一致。

（例如，S=”ace”是L=”abcde”的一个子序列且有效字符是a、c、e，而”aec”不是有效子序列，且有效字符只有a、e）

输入描述

输入两个字符串S和L，都只包含英文小写字母。S长度<=100，L长度<=500,000。

先输入S，再输入L，每个字符串占一行。

输出描述

输出S串最后一个有效字符在L中的位置。（首位从0开始计算，无有效字符返回-1）

用例

用例1

输入

```
ace
abcde
```

输出

```
4
```

用例2

输入

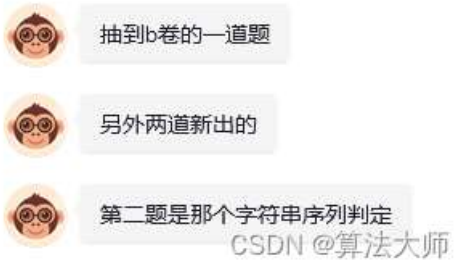
```
fgh
abcde
```

输出

```
-1
```

解题思路

注意： 本题在C卷中和B卷中都有。机考可能会存在变形，请注意审题。



我们初始化两个指针i和j，分别用于遍历S和L。

接下来，我们使用双指针法进行遍历，当i小于S的长度且j小于L的长度时，进行循环。

在循环中，我们判断S中的当前字符是否与L中的当前字符相等，如果相等，则将i指针向后移动一位。

无论字符是否相等，我们都将j指针向后移动一位。

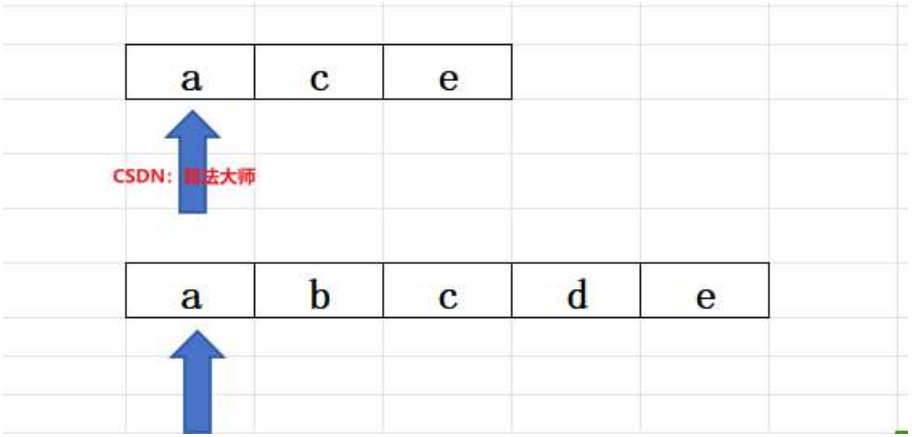
当循环结束后，我们判断i是否等于S的长度，如果等于，则说明S的所有字符都在L中找到了，打印L中最后一个有效字符的位置（即j的值减1）；否则，说明S还有字符没有在L中找到，打印-1。

最后，我们得到了S串最后一个有效字符在L中的位置。

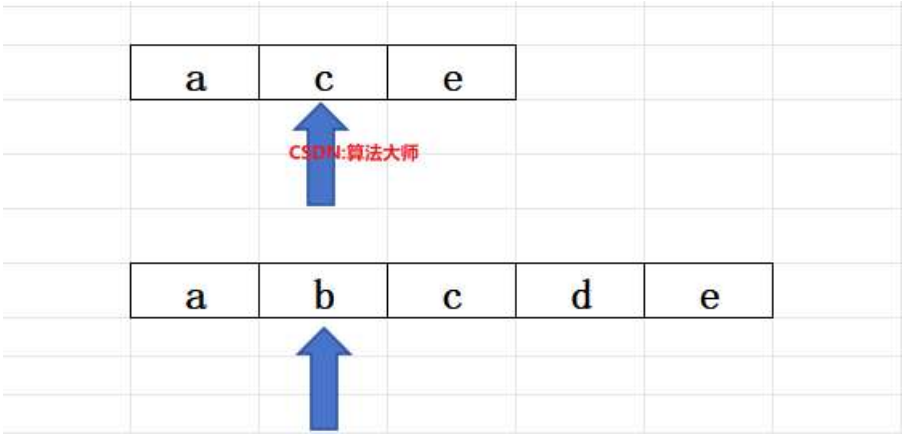
用例解析

在上面的用例中，indexS和indexL是通过循环逐步变化的。下面是它们的具体变化过程：

1. 初始化索引：
   indexS = 0, indexL = 0

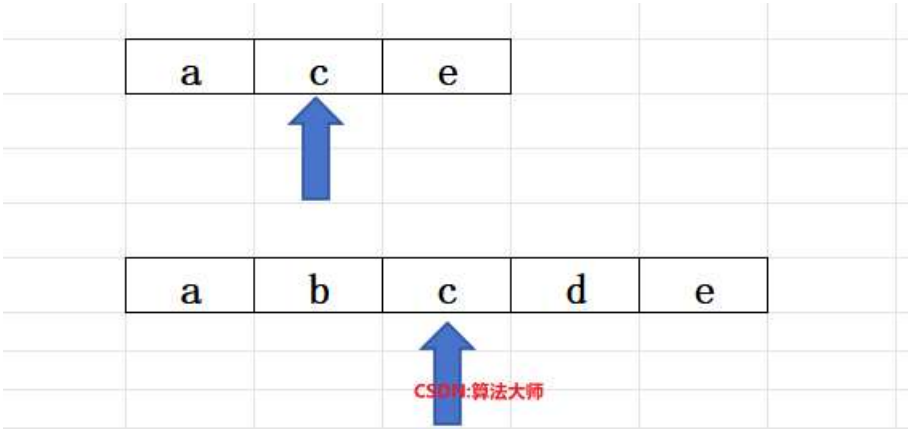


2. 第一次循环：
   - 检查S中的第一个字符'a'与L中的第一个字符'a'是否相同，相同则indexS加1，indexL加1。
   - indexS = 1, indexL = 1



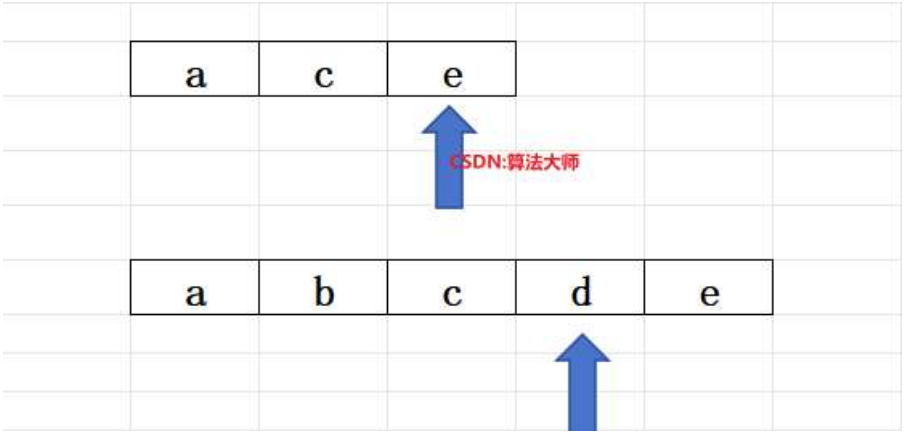
3. 第二次循环：

- 检查S中的第二个字符'c'与L中的第二个字符'b'是否相同，不同则indexS不变，indexL加1。
- indexS = 1, indexL = 2



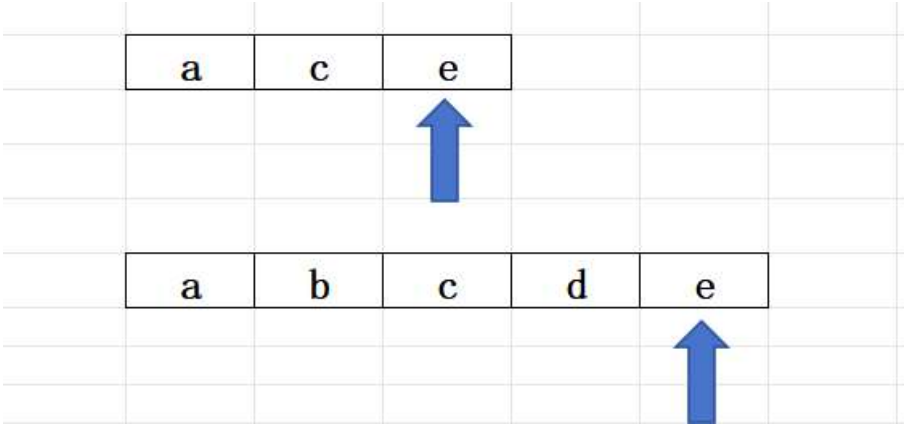
4. 第三次循环：

- 检查S中的第二个字符'c'与L中的第三个字符'c'是否相同，相同则则indexS加1，indexL加1。
- indexS = 2, indexL = 3



5. 第四次循环：

- 检查S中的第三个字符'e'与L中的第四个字符'd'是否相同，不同则indexS不变，indexL加1。
- indexS = 2, indexL = 4



6. 第五次循环：

- S已经遍历完，循环结束。

在循环结束后，判断indexS是否等于S的长度，即判断S的所有字符是否都在L中找到了。在这个用例中，indexS等于S的长度，表示S中的字符都在L中找到了。

最后，打印L中最后一个有效字符的位置（即L的当前索引减1），即输出为4。

C语言

```c
#include <stdio.h>
#include <string.h>

int main() {
  char stringS[1000];
  char stringL[1000];
  fgets(stringS, 1000, stdin);
  fgets(stringL, 1000, stdin);

  // 初始化两个索引，分别用于遍历S和L
  int indexS = 0;
  int indexL = 0;

  // 当S和L都没有遍历完时，继续遍历
  while (indexS < strlen(stringS) && indexL < strlen(stringL)) {
    // 如果S中的当前字符与L中的当前字符相同，则S的索引加1
    if (stringS[indexS] == stringL[indexL]) {
      indexS++;
    }
    // 无论字符是否相同，L的索引都加1
    indexL++;
  }

  // 如果S的所有字符都在L中找到了（即S已经遍历完了），则打印L中最后一个有效字符的位置（即L的当前索引减1）
  if (indexS == strlen(stringS) - 1) printf("%d\n", indexL - 1);
  // 如果S还有字符没有在L中找到，则打印-1
  else printf("-1\n");

  return 0;
}

```

C++

```cpp
#include <iostream>
#include <string>
```

```cpp
using namespace std;

int main() {
  string stringS, stringL;
  getline(cin, stringS);
  getline(cin, stringL);

  // 初始化两个索引，分别用于遍历S和L
  int indexS = 0;
  int indexL = 0;

  // 当S和L都没有遍历完时，继续遍历
  while (indexS < stringS.length() && indexL < stringL.length()) {
    // 如果S中的当前字符与L中的当前字符相同，则S的索引加1
    if (stringS[indexS] == stringL[indexL]) {
      indexS++;
    }
    // 无论字符是否相同，L的索引都加1
    indexL++;
  }

  // 如果S的所有字符都在L中找到了（即S已经遍历完了），则打印L中最后一个有效字符的位置（即L的当前索引减1）
  if (indexS == stringS.length()) cout << indexL - 1 << endl;
  // 如果S还有字符没有在L中找到，则打印-1
  else cout << -1 << endl;

  return 0;
}

```

Java

```java
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    // 创建一个Scanner对象来读取用户的输入
    Scanner scanner = new Scanner(System.in);

    // 读取第一个字符串S
```

```
        String stringS = scanner.nextLine();
        // 读取第二个字符串L
        String stringL = scanner.nextLine();a

        // 初始化两个索引，分别用于遍历S和L
        int indexS = 0;
        int indexL = 0;

        // 当S和L都没有遍历完时，继续遍历
        while (indexS < stringS.length() && indexL < stringL.length()) {
            // 如果S中的当前字符与L中的当前字符相同，则S的索引加1
            if (stringS.charAt(indexS) == stringL.charAt(indexL)) {
                indexS++;
            }
            // 无论字符是否相同，L的索引都加1
            indexL++;
        }

        // 如果S的所有字符都在L中找到了（即S已经遍历完了），则打印L中最后一个有效字符的位置（即L的当前索引减1）
        if (indexS == stringS.length()) System.out.println(indexL - 1);
        // 如果S还有字符没有在L中找到，则打印-1
        else System.out.println(-1);
    }
}
```

JavaScript

```
const readline = require('readline');

// 创建一个readline接口对象来读取用户的输入
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// 读取第一个字符串S
rl.on('line', (stringS) => {
  // 读取第二个字符串L
  rl.on('line', (stringL) => {
```

```
    // 初始化两个索引，分别用于遍历S和L
    let indexS = 0;
    let indexL = 0;

    // 当S和L都没有遍历完时，继续遍历
    while (indexS < stringS.length && indexL < stringL.length) {
      // 如果S中的当前字符与L中的当前字符相同，则S的索引加1
      if (stringS.charAt(indexS) === stringL.charAt(indexL)) {
        indexS++;
      }
      // 无论字符是否相同，L的索引都加1
      indexL++;
    }

    // 如果S的所有字符都在L中找到了（即S已经遍历完了），则打印L中最后一个有效字符的位置（即L的当前索引减1）
    if (indexS === stringS.length) console.log(indexL - 1);
    // 如果S还有字符没有在L中找到，则打印-1
    else console.log(-1);

    rl.close();
  });
});
```

Python

```
# 创建一个Scanner对象来读取用户的输入
stringS = input("")
stringL = input("")

# 初始化两个索引，分别用于遍历S和L
indexS = 0
indexL = 0

# 当S和L都没有遍历完时，继续遍历
while indexS < len(stringS) and indexL < len(stringL):
    # 如果S中的当前字符与L中的当前字符相同，则S的索引加1
    if stringS[indexS] == stringL[indexL]:
        indexS += 1
    # 无论字符是否相同，L的索引都加1
```

```
    indexL += 1

# 如果S的所有字符都在L中找到了（即S已经遍历完了），则打印L中最后一个有效字符的位置（即L的当前索引减1）
if indexS == len(stringS):
    print(indexL - 1)
# 如果S还有字符没有在L中找到，则打印-1
else:
    print(-1)

```

完整用例

用例1

```
ace
abcde
```

用例2

```
fgh
abcde
```

用例3

```
abc
abcde
```

用例4

```
abcd
abcde
```

用例5

```
a
abcde
```

用例6

```
abcd
abcdef
```

用例7

```
aaa
aaabaaa
```

用例8

```
aaa
baaa
```

用例9

```
aaa
baabaaa
```

用例10

```
aaa
baabbaaa
```

文章目录