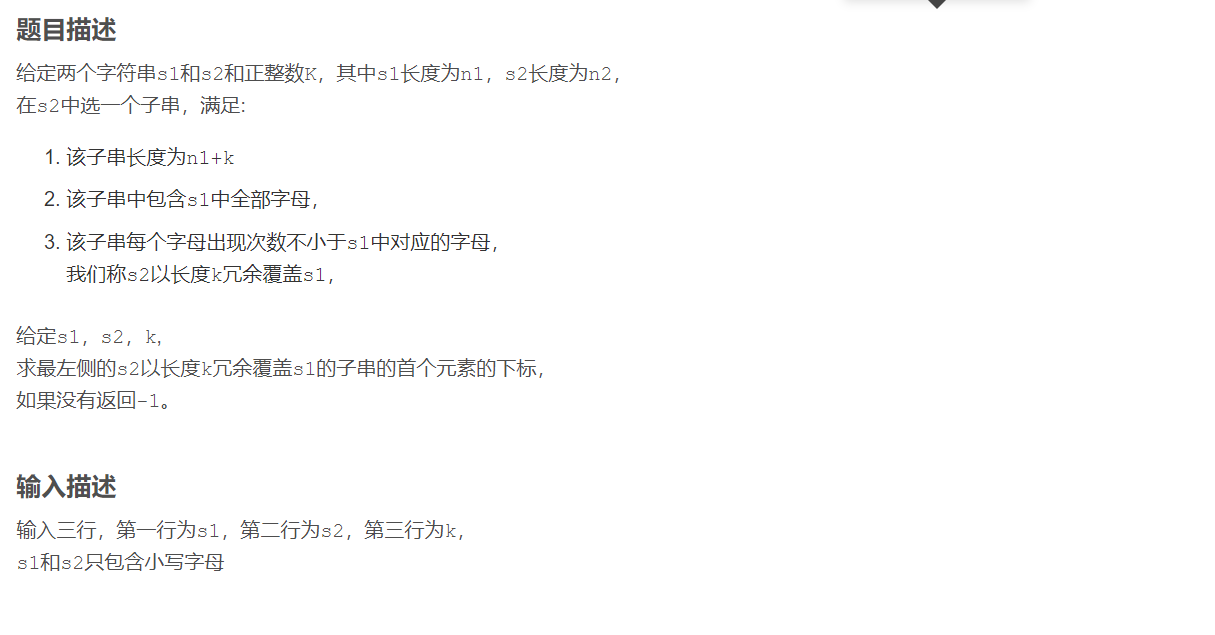
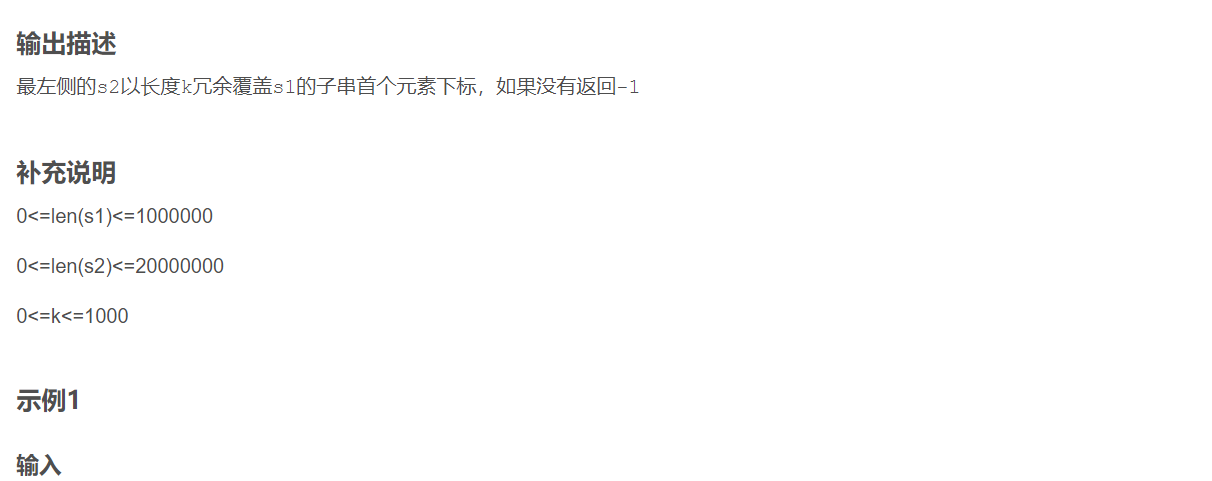
# **E卷-最左侧冗余覆盖子串[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**





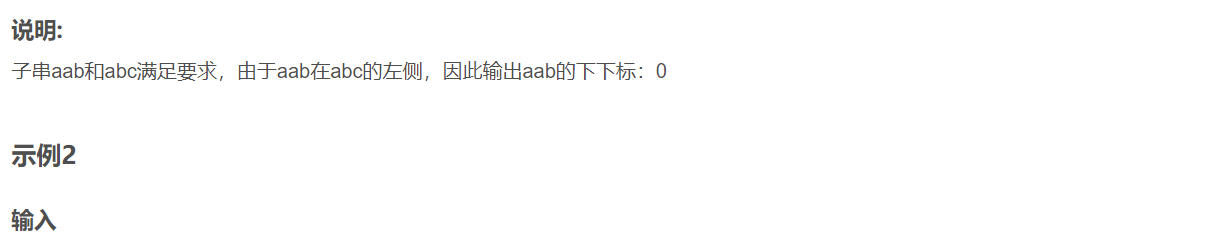
ab

aabcd

1



0



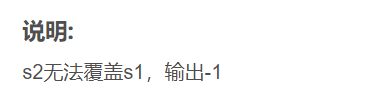
abc

dfs

10



-1





import java.util.\*;

public class Main {

// 声明全局数组，记录s1与当前窗口中每个字母的出现次数

static int[] cnts1 = new int[26];

static int[] cnts2 = new int[26];

// 比较函数：检查cnts2是否满足所有cnts1中的字母出现次数

static boolean cmp() {

for (int i = 0; i < 26; i++) {

if (cnts2[i] < cnts1[i]) return false;

}

return true;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// 输入字符串s1和s2

String s1 = sc.nextLine();

String s2 = sc.nextLine();

// 输入整数k

int k = sc.nextInt();

// 初始化cnts1，记录s1中每个字母的出现次数

for (int i = 0; i < s1.length(); i++) {

cnts1[s1.charAt(i) - 'a']++;

}

// 初始化cnts2，记录s2中前s1.length() + k长度子串的每个字母的出现次数

for (int i = 0; i < s1.length() + k; i++) {

cnts2[s2.charAt(i) - 'a']++;

}

// 子串长度为s1.length() + k

int len = s1.length() + k;

// 遍历s2中的每个可能的子串

for (int i = len; i <= s2.length(); i++) {

// 比较当前窗口是否满足要求

if (cmp()) {

System.out.println(i - len); // 如果满足要求，输出子串的起始下标

return;

}

// 移动窗口

if (i == s2.length()) break;

cnts2[s2.charAt(i - len) - 'a']--; // 减去窗口最左边一个字符的出现次数

cnts2[s2.charAt(i) - 'a']++; // 增加窗口最右边一个字符的出现次数

}

// 如果没有找到满足条件的子串，输出-1

System.out.println(-1);

}

}



def cmp(cnts1, cnts2):

# 比较cnts2是否满足所有cnts1中的字母出现次数

for i in range(26):

if cnts2[i] < cnts1[i]:

return False

return True

s1 = input() # 输入字符串s1

s2 = input() # 输入字符串s2

k = int(input()) # 输入整数k

cnts1 = [0] \* 26 # 初始化cnts1

cnts2 = [0] \* 26 # 初始化cnts2

# 初始化cnts1，记录s1中每个字母的出现次数

for c in s1:

cnts1[ord(c) - ord('a')] += 1

# 初始化cnts2，记录s2中前s1长度+k部分的字母出现次数

for i in range(len(s1) + k):

cnts2[ord(s2[i]) - ord('a')] += 1

# 子串长度

length = len(s1) + k

# 遍历s2中的每个可能子串

for i in range(length, len(s2) + 1):

# 检查当前窗口是否满足条件

if cmp(cnts1, cnts2):

print(i - length) # 如果满足条件，输出开始下标

exit()

# 移动窗口

if i == len(s2):

break

cnts2[ord(s2[i - length]) - ord('a')] -= 1 # 减去窗口最左端字符出现次数

cnts2[ord(s2[i]) - ord('a')] += 1 # 增加窗口最右端字符出现次数

# 没有找到满足条件的子串，输出-1

print(-1)



#include <bits/stdc++.h>

using namespace std;

int cnts1[26]; // 用于记录s1中每个字母的出现次数

int cnts2[26]; // 用于记录当前窗口中每个字母的出现次数

// 函数：比较当前窗口（cnts2）是否能满足包含s1所有字母并且每个字母出现次数不小于s1

bool cmp() {

for (int i = 0; i < 26; i++) {

if (cnts2[i] < cnts1[i]) return false; // 如果cnts2中的某个字母的数量少于cnts1，则返回false

}

return true; // 否则返回true

}

int main() {

string s1, s2;

cin >> s1 >> s2; // 输入字符串s1和s2

int k;

cin >> k; // 输入整数k

// 初始化cnts1，记录s1中每个字母的出现次数

for (int i = 0; i < s1.size(); i++) cnts1[s1[i] - 'a']++;

// 初始化cnts2，记录s2中前(s1的长度 + k)长度子串的每个字母的出现次数

for (int i = 0; i < s1.size() + k; i++) {

cnts2[s2[i] - 'a']++;

}

// 子串长度应为 s1.size() + k

int len = s1.size() + k;

// 遍历s2中的每个可能的子串

for (int i = len; i <= s2.size(); i++) {

// 比较当前窗口是否满足要求

if (cmp()) {

cout << i - len << endl; // 如果满足要求，输出子串的起始下标

return 0; // 结束程序

}

// 移动窗口

if (i == s2.size()) break; // 如果已经到了s2的末尾，跳出循环

cnts2[s2[i - len] - 'a']--; // 减去窗口最左边一个字符的出现次数

cnts2[s2[i] - 'a']++; // 增加窗口最右边一个字符的出现次数

}

// 如果没有找到满足条件的子串，输出-1

cout << -1 << endl;

}



#include <stdio.h>

#include <string.h>

int cnts1[26] = {0}; // 记录s1中每个字母的出现次数

int cnts2[26] = {0}; // 记录当前窗口中每个字母的出现次数

// 比较函数，检查cnts2是否满足cnts1的要求

int cmp() {

for(int i = 0; i < 26; i++) {

if(cnts2[i] < cnts1[i])

return 0;

}

return 1;

}

int main() {

char s1[1001], s2[1001];

// 输入字符串s1和s2

scanf("%s", s1);

scanf("%s", s2);

int k;

// 输入整数k

scanf("%d", &k);

int n1 = strlen(s1);

int n2 = strlen(s2);

// 记录s1中每个字母出现次数

for(int i = 0; i < n1; i++) {

cnts1[s1[i] - 'a']++;

}

// 记录s2前n1+k长度子串中的每个字母出现次数

for(int i = 0; i < n1 + k; i++) {

cnts2[s2[i] - 'a']++;

}

int len = n1 + k;

// 遍历s2中的每个可能子串

for(int i = len; i <= n2; i++) {

if(cmp()) {

printf("%d\n", i - len); // 如果满足要求，输出起始下标

return 0;

}

if(i == n2)

break;

// 移动窗口：减去左端字符，增加右端字符

cnts2[s2[i - len] - 'a']--;

cnts2[s2[i] - 'a']++;

}

// 没有找到满足条件的子串，输出-1

printf("-1\n");

return 0;

}



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

function cmp(cnts1, cnts2) {

// 比较函数，检查cnts2是否满足cnts1的要求

for (let i = 0; i < 26; i++) {

if (cnts2[i] < cnts1[i]) {

return false;

}

}

return true;

}

let inputLines = [];

let lineCount = 0;

rl.on("line", (line) => {

inputLines.push(line);

lineCount++;

if (lineCount === 3) {

// 读取输入

const s1 = inputLines[0];

const s2 = inputLines[1];

const k = parseInt(inputLines[2]);

let cnts1 = new Array(26).fill(0);

let cnts2 = new Array(26).fill(0);

// 记录s1中每个字母出现次数

for (let c of s1) {

cnts1[c.charCodeAt(0) - "a".charCodeAt(0)]++;

}

// 记录s2前n1+k长度子串中的每个字母出现次数

for (let i = 0; i < s1.length + k; i++) {

cnts2[s2[i].charCodeAt(0) - "a".charCodeAt(0)]++;

}

let len = s1.length + k;

// 遍历s2中的每个可能子串

for (let i = len; i <= s2.length; i++) {

if (cmp(cnts1, cnts2)) {

console.log(i - len); // 如果满足要求，输出起始下标

return;

}

if (i === s2.length) break;

// 移动窗口：减去左端字符，增加右端字符

cnts2[s2[i - len].charCodeAt(0) - "a".charCodeAt(0)]--;

cnts2[s2[i].charCodeAt(0) - "a".charCodeAt(0)]++;

}

// 没有找到满足条件的子串，输出-1

console.log(-1);

}

});



package main

import (

"fmt"

)

func cmp(cnts1, cnts2 []int) bool {

// 比较函数，检查cnts2是否满足cnts1的要求

for i := 0; i < 26; i++ {

if cnts2[i] < cnts1[i] {

return false

}

}

return true

}

func main() {

var s1,s2 string

fmt.Scanln(&s1)

fmt.Scanln(&s2)

var k int

fmt.Scan(&k)

cnts1 := make([]int, 26)

cnts2 := make([]int, 26)

// 记录s1中每个字母出现次数

for \_, c := range s1 {

cnts1[c-'a']++

}

// 记录s2前n1+k长度子串中的每个字母出现次数

for i := 0; i < len(s1)+k; i++ {

cnts2[s2[i]-'a']++

}

sz := len(s1) + k

// 遍历s2中的每个可能子串

for i := sz; i <= len(s2); i++ {

if cmp(cnts1, cnts2) {

fmt.Println(i - sz) // 如果满足要求，输出起始下标

return

}

if i == len(s2) {

break

}

// 移动窗口：减去左端字符，增加右端字符

cnts2[s2[i-sz]-'a']--

cnts2[s2[i]-'a']++

}

// 没有找到满足条件的子串，输出-1

fmt.Println(-1)

}