极客学院
jikexueyuan.com

名企数据结构面试题之字符串（下）

名企数据结构面试题之字符串（下） — 课程概要

- 同分异构体
- 反转单词顺序
- 数一数并读一读
- 模式匹配
- KMP算法简介

同分异构体

同分异构体

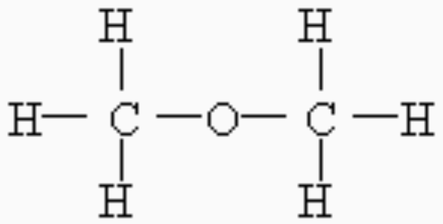
- 同分异构体的概念
- 问题描述
- 思路分析
- 代码实现
- 测试与提交
- 扩展问题

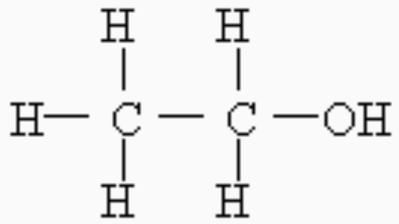
同分异构体 — **同分异构体的概念**

化学中的**同分异构体**：

- 化学式（分子式）相同
- 结构式不同

物质	化学式	结构式
甲醚	C_2H_6O	CH_3OCH_3
乙醇	C_2H_6O	CH_3CH_2OH





同分异构体 — **同分异构体的概念**

英语单词中的**同分异构体**，也称**Anagram**。

单词	中文大意	字母构成
lee	避风处	e_2l_1
eel	鳗鱼	e_2l_1

单词	中文大意	字母构成
read	读	$a_1e_1d_1r_1$
dear	亲	$a_1e_1d_1r_1$

单词	中文大意	字母构成
quiet	安静的	$e_1i_1u_1q_1t_1$
quite	非常地	$e_1i_1u_1q_1t_1$

单词	中文大意	字母构成
listen	听	$e_1i_1l_1n_1s_1t_1$
silent	沉默的	$e_1i_1l_1n_1s_1t_1$

同分异构体 — **问题描述**

leetCode 242： Valid Anagram

给定字符串s和字符串t，判断t是否为s的Anagram。

假设字符串只含有小写字母。

同分异构体 — **思路分析**

总体思想：

- 想办法把s与t写成形如$a_2b_0c_4\cdots z_1$的**分子式**形式
- 判断2个分子式是否相同

具体思路：

- 建立2个长度为26的数组，分别代表s与t的分子式
- 分别遍历s与t，填充相应的分子式
- 如果2个分子式，每个字母出现的次数都相同，就返回true，否则返回false

同分异构体 — **代码实现**

Key	Value
类名	_242ValidAnagram
方法名	isAnagram
时间复杂度	O(N)
空间复杂度	O(1)

- 哈希表
- 计数排序

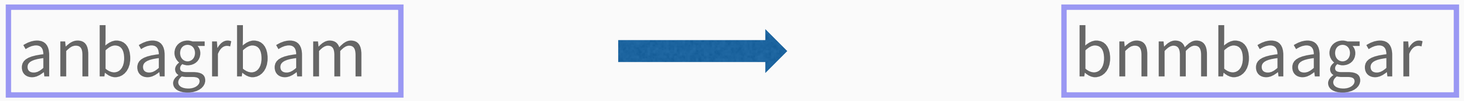
同分异构体 — 测试与提交

Key	Value
类名	_242ValidAnagram
方法名	test
测试输入1	s = "anagram", t = "nagaram"
测试输出1	true
测试输入2	s = "rat", t = "car"
测试输出2	false

同分异构体 — **扩展问题**

“交换字母”：

怎样通过“交换字母”操作，实现以下变换？



又如何求出最小的变换步数？

NP（完全）问题，以后再探讨！

反转单词顺序

反转单词顺序

- 问题描述
- 思路分析
- 代码实现
- 测试用例
- 思考题

反转单词顺序 — **问题描述**

输入一个英文句子，反转句子中单词的顺序，要求单词不变。

- 为了简便，假设：句子只含有字母、空格，且句子首尾没有空格
- 为了格式美观，假设：每一对相邻单词之间，空格数目都相等

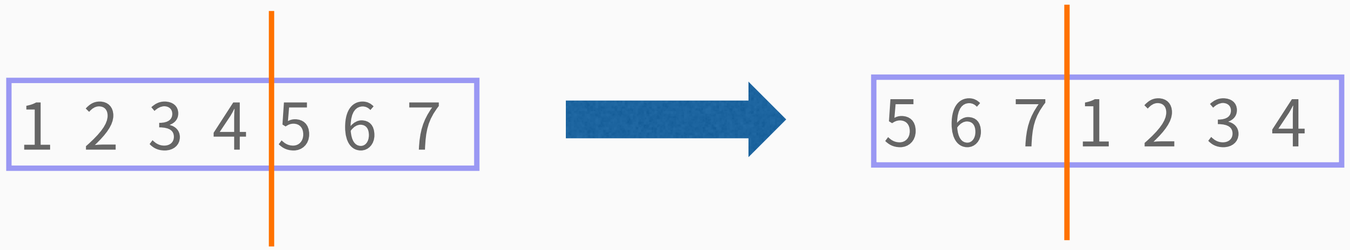
例如输入字符串“Thank you very much”，则输出“much very you Thank”。

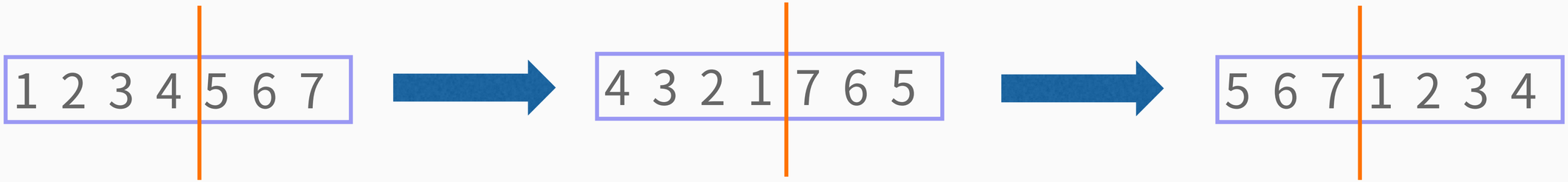
额外要求：

- 时间复杂度为O(N)，空间复杂度为O(1)
- **用C语言实现，不得使用Java字符串的split方法**

反转单词顺序 — **思路分析**

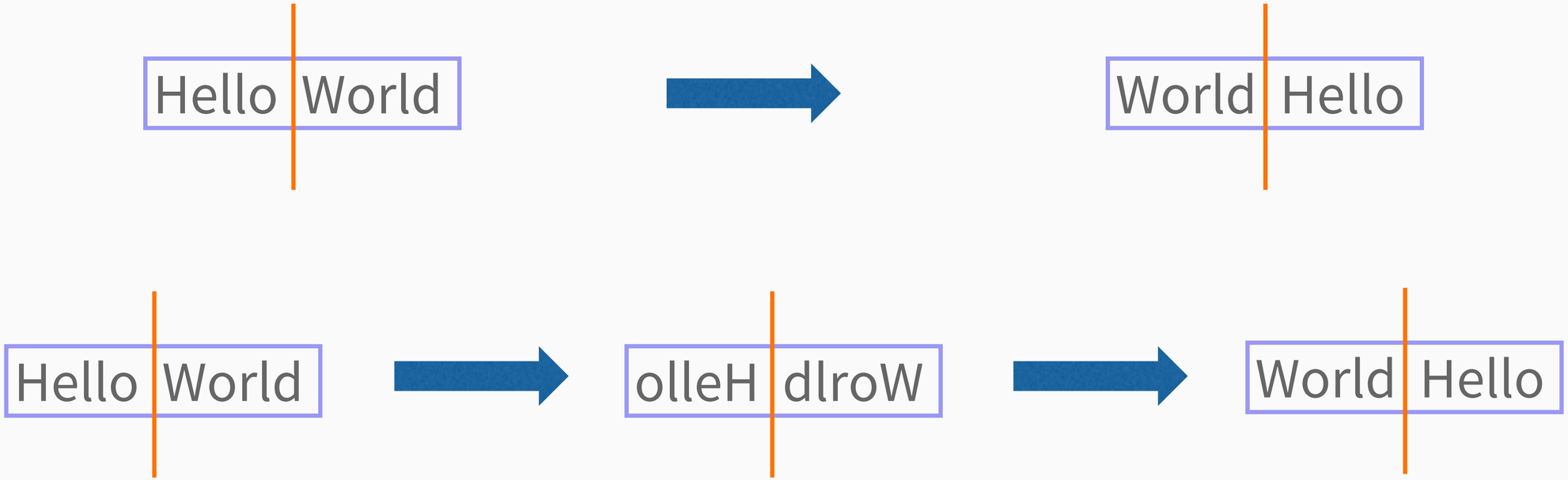
回忆**旋转数组**：





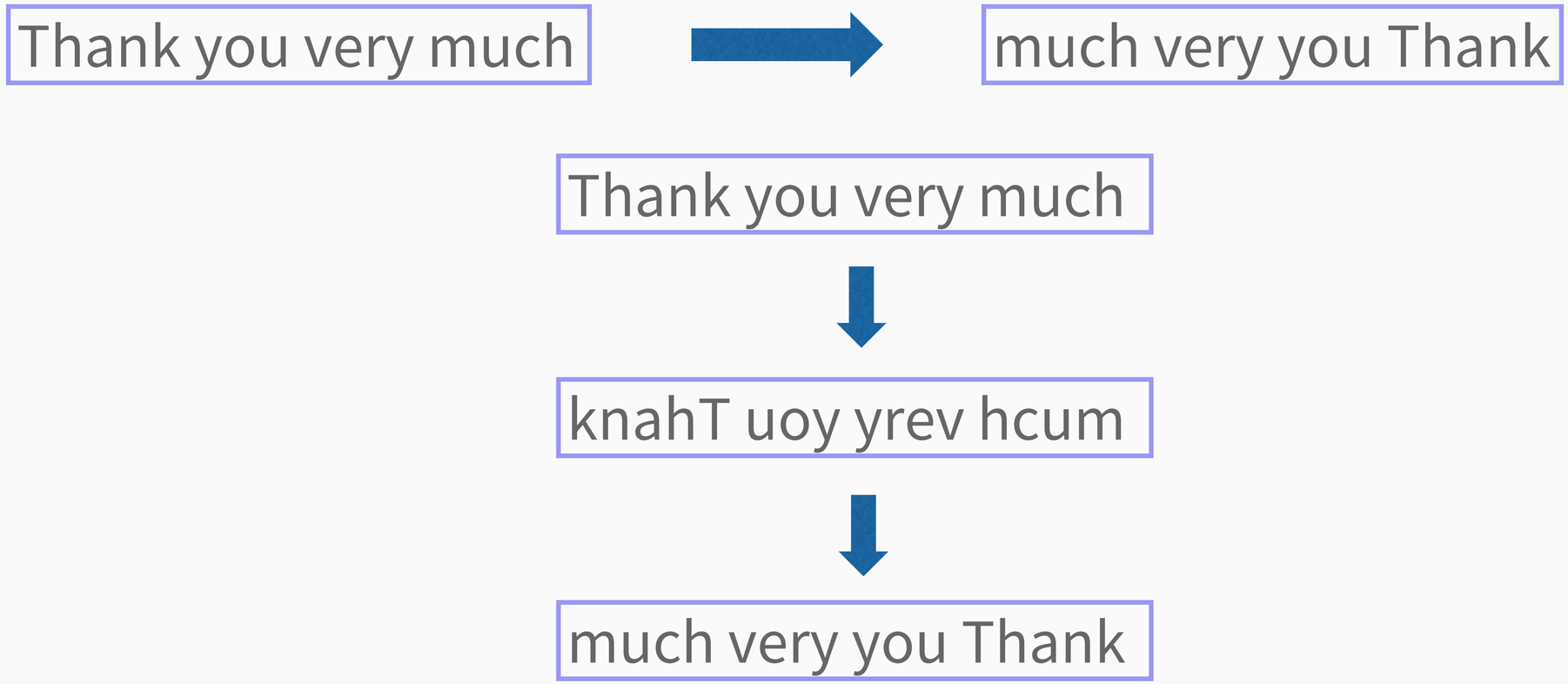
反转单词顺序 — **思路分析**

两个单词的情况：



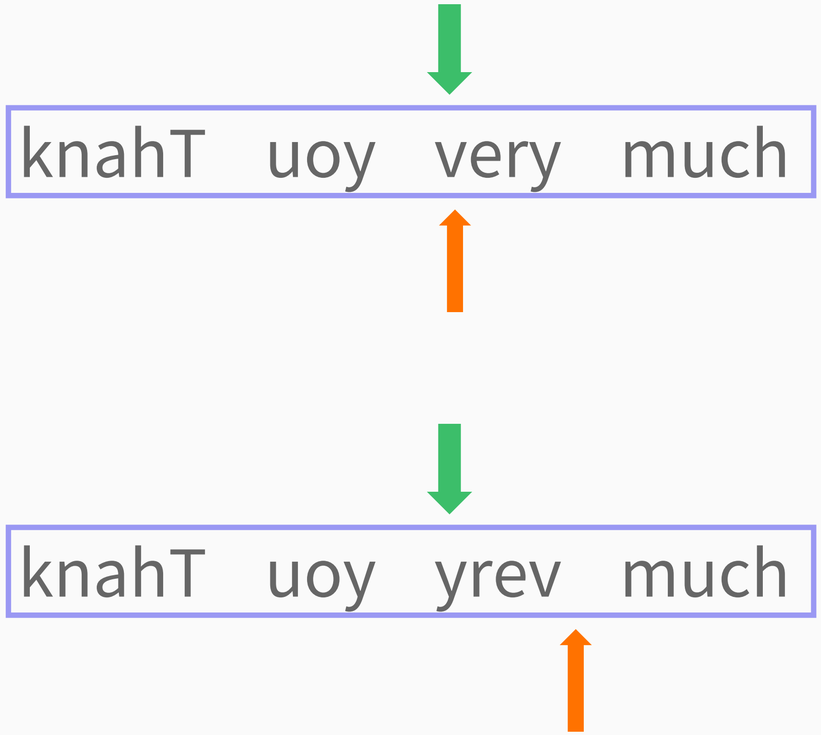
反转单词顺序 — **思路分析**

多个单词的情况：



反转单词顺序 — **思路分析**

双指针，i（绿）和j（橙）。



反转单词顺序 — 代码实现

Key	Value
文件名	ReverseWords.c
方法名	reverseWords
时间复杂度	O(N)
空间复杂度	O(1)

反转单词顺序 — 测试用例

Key	Value
文件名	ReverseWords.c
方法名	main
测试输入1	"Thank you very much"
测试输出1	"much very you Thank"
测试输入2	"What is your job"
测试输出2	"job your is What"

反转单词顺序 — **思考题**

leetCode 151：Reverse Words in a String

除了时间、空间、C语言的限制之外，对结果还有以下要求：

- 去除首尾空格
- 相邻单词之间只有一个空格

数一数并读一读

数一数并读一读

- 问题描述
- 思路分析
- 代码实现
- 测试与提交

数一数并读一读 — **问题描述**

leetCode 38：Count And Say

比如：

“1211”，有1个1、1个2、2个1，所以下一个字符串为“111221”；

“111221”，有3个1、2个2、1个1，所以下一个字符串为“312211”。

统计每个数字出现的次数，并把本次得到的**结果字符串**作为下一次的**当前字符串**。

如此循环往复，那么执行n次之后是什么？

数一数并读一读 — 思路分析

统计每个数字出现的次数，并把本次得到的**结果字符串**作为下一次的**当前字符串**。

```
p=“332211” // 当前字符串
str=“” //结果字符串
count=1 //临时变量，记录数字出现的次数
temp=p[0] //临时变量，记录上一个数字
for(j=1;j<p.length();j++){
    if(p[j]==temp) count++;
    else{
    str+=count+temp;
    count=1;
    temp=p[j];
    }
}
str+=count+temp;
p=str
```

str=“13”

count=1

temp=2

str=“13”

count=2

temp=1

str=“1321”

count=2

temp=1

str=“1321”

count=1

temp=2

str=“132112”

数一数并读一读 — 代码实现

用StringBuilder的append方法代替字符串的“+”。

Key	Value
类名	_038CountAndSay
方法名	countAndSay
时间复杂度	约为$O(N^3)$
空间复杂度	约为$O(N^2)$

数一数并读一读 — 测试与提交

Key	Value
类名	_038CountAndSay
方法名	test
测试输入1	5
测试输出1	111221
测试输入2	6
测试输出2	312211
测试出入3	7
测试输出3	13112221

模式匹配

模式匹配

- 模式匹配的概念
- 问题描述
- BF算法的思路
- 代码实现
- 测试与提交

模式匹配 — 模式匹配的概念

给定文本串S（Source）、模式串P（Pattern），查找S中与P相同的所有子串：

S：ffabcdabceeabcdabaggabcdabahh

P：abcdaba

为了简便，只需找出子串出现的第一个位置，即为String类的indexOf方法。

模式匹配 — 问题描述

leetCode 28：Implement strStr()

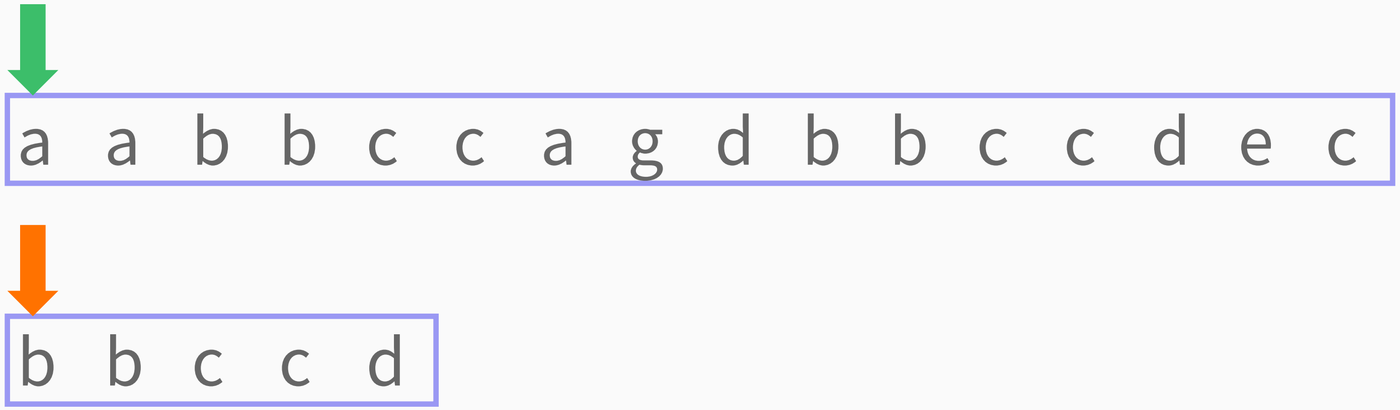
返回needle在haystack中的第一个出现位置，如果没找到，返回-1。

不允许直接使用indexOf方法！

模式匹配 — BF算法的思路

Brute Force（暴力匹配）算法思想：

两重循环，i指针（绿）与j指针（橙）



模式匹配 — 代码实现

Key	Value
类名	_028BruteForce
方法名	strStr
时间复杂度	O(M*N)
空间复杂度	O(1)

模式匹配 — 测试与提交

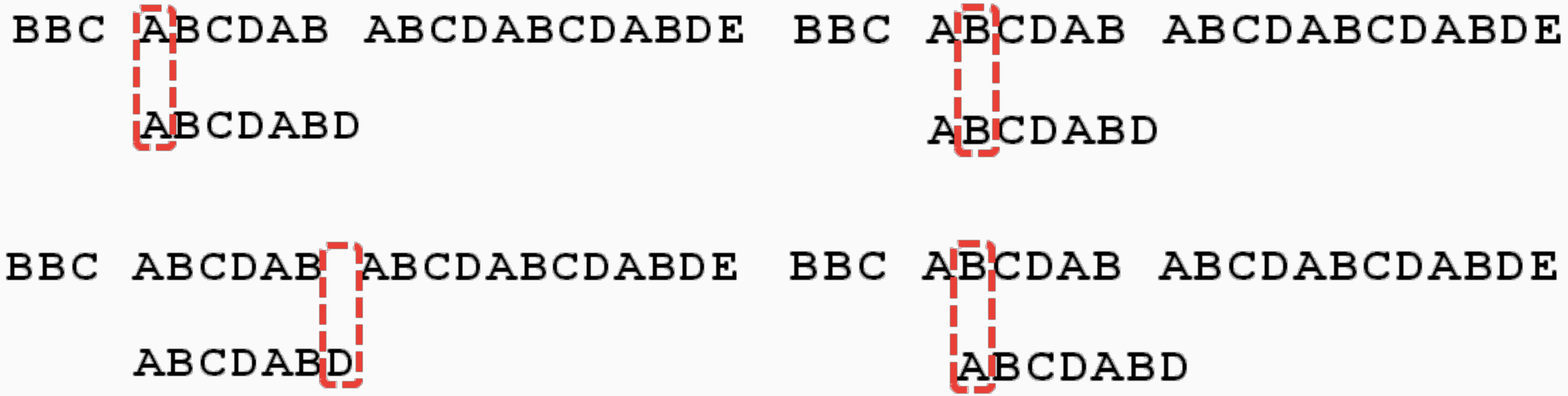
Key	Value
类名	_028BruteForce
方法名	test
测试输入1	s="aabbccagdbbccdec"; p="bbccd";
测试输出1	9
测试输入2	s="aabbccagdbbccdec"; p="bbccf";
测试输出2	-1

KMP算法简介

KMP算法简介

- BF算法的不足
- next数组
- KMP的代码实现
- 测试与提交

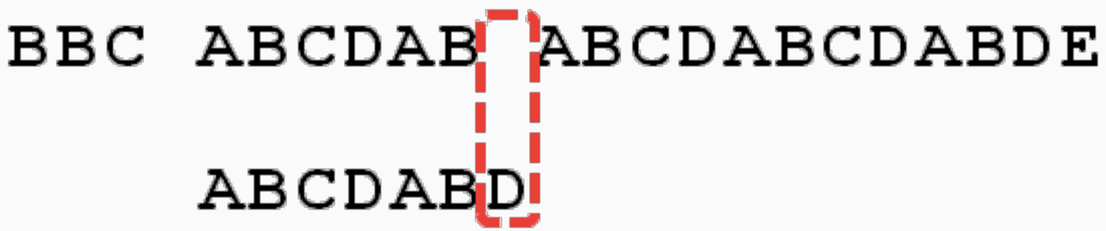
KMP算法简介 — **BF算法的不足**

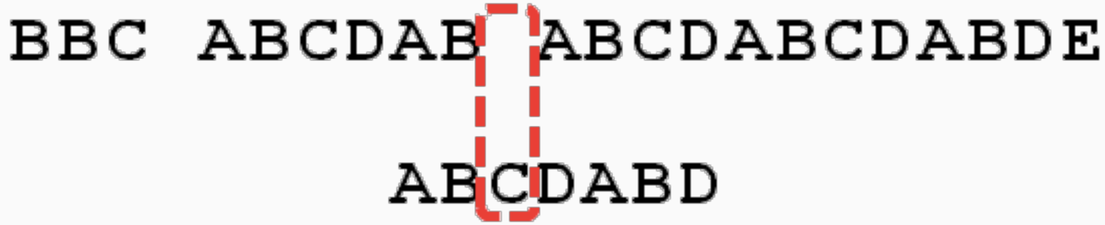


能否让i指针不回退，并让j指针移动到有效位置？

KMP算法简介 — **BF算法的不足**

KMP算法，充分利用**模式串自身特性**、**已经部分匹配**这两个重要信息，保持i指针不回退，并让j指针移动到尽量有效的位置。





KMP算法简介 — next数组

前缀子串、前缀、后缀

给定模式串“ABCDABD”，列出各个前缀子串的前缀、后缀：

前缀子串	前缀	后缀	公共元素的最大长度
A	null	null	0
AB	A	B	0
ABC	A,AB	C,BC	0
ABCD	A,AB,ABC	D,CD,BCD	0
ABCDA	**A**,AB,ABC,ABCD	**A**,DA,CDA,BCDA	1
ABCDAB	A,**AB**,ABC,ABCD,ABCDA	B,**AB**,DAB,CDAB,BCDAB	2
ABCDABD	A,AB,ABC,ABCD,ABCDA,ABCDAB	D,BD,ABD,DABD,CDABD,BCDABD	0

KMP算法简介 — **next数组**

列出**最大长度表**：

模式串	A	B	C	D	A	B	D
公共元素的最大长度	0	0	0	0	1	2	0

所有数字往后移动一位，0号位置元素置为-1，即得**next数组**：

模式串	A	B	C	D	A	B	D
next值	-1	0	0	0	0	1	2

求next数组的算法，时间复杂度并不是$O(N^2)$，而是O(N)。

参考：http://wiki.jikexueyuan.com/project/kmp-algorithm/define.html

KMP算法简介 — KMP的代码实现

Key	Value
类名	_028KMP
方法名	strStr
时间复杂度	O(M+N)，M表示文本串长度，N表示模式串长度
空间复杂度	O(N)

方法名	作用
kmpSearch	KMP算法
getNext	求next数组

KMP算法简介 — **测试与提交**

Key	Value
类名	_028KMP
方法名	test
测试输入1	s="AABAABAC AAAAABCD CD"; p="AAAAABCD";
测试输出1	9
测试输入2	s="BBC ABCDAB ABCDABCDABDE"; p="ABCDABD";
测试输出2	15

参考极客学院wiki，研读+举例+调试+**记忆**！

名企数据结构面试题之字符串（下）

本套课程中我们学习了名企数据结构面试之字符串（下）。你应当掌握了以下知识：

- 同分异构体
- 反转单词顺序
- 数一数并读一读
- 模式匹配
- KMP算法简介

你可以使用leetCode验证程序是否正确，还可以在白纸上书写代码；如果想进一步提高，你可以继续在极客学院学习**名企数据结构面试题之链表（上）**课程。

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台