



多串匹配算法 及其启示

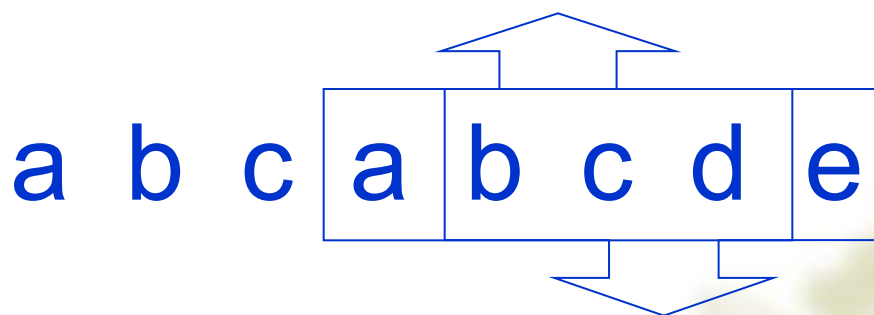
南京市外国语学校
朱泽园

问题提出

- ❖ 所谓多串匹配，就是给定一些模式串，在一段文章（只出现小写a到z这26个字母）中，找出第一个出现的任意一个模式串的位置，或者所有模式串出现的所有位置。

例子

- ❖ 模式串：“abcd” “bcde”
- ❖ 正文：



实际应用

- ❖ 含逻辑关键字的搜索引擎
- ❖ **DNA**序列搜索
- ❖

广!

因此用有效算法解决该问题能大大提高各行各业的工作效率!

数据规模

❖ 设共有 m 个模式串，长度分别为 L_1 、 $L_2 \dots L_m$

正文为一个长度为 n 的数组 $T[1..n]$ ，限定

$$\sum L \leq 100K, m \leq 1000, n \leq 900K$$

朴素想法

- ❖ 从小到大枚举每一个位置，并且对所有模式串进行检查。最坏情况下时间复杂度为

$$O(n \cdot \sum L)$$



- ❖ 对每一个模式串，使用**kmp**算法进行单串匹配，时间复杂度为

$$O(n \cdot m + \sum L)$$



我的算法

- ❖ 辅助算法1: **Knuth-Morris-Pratt**模式匹配
- ❖ 辅助算法2: 单词前缀树 (自创)
- ❖ 主算法1: 线性算法
- ❖ 辅助算法3: 后缀树
- ❖ 主算法2: 平均性能更好的算法



单词前缀树

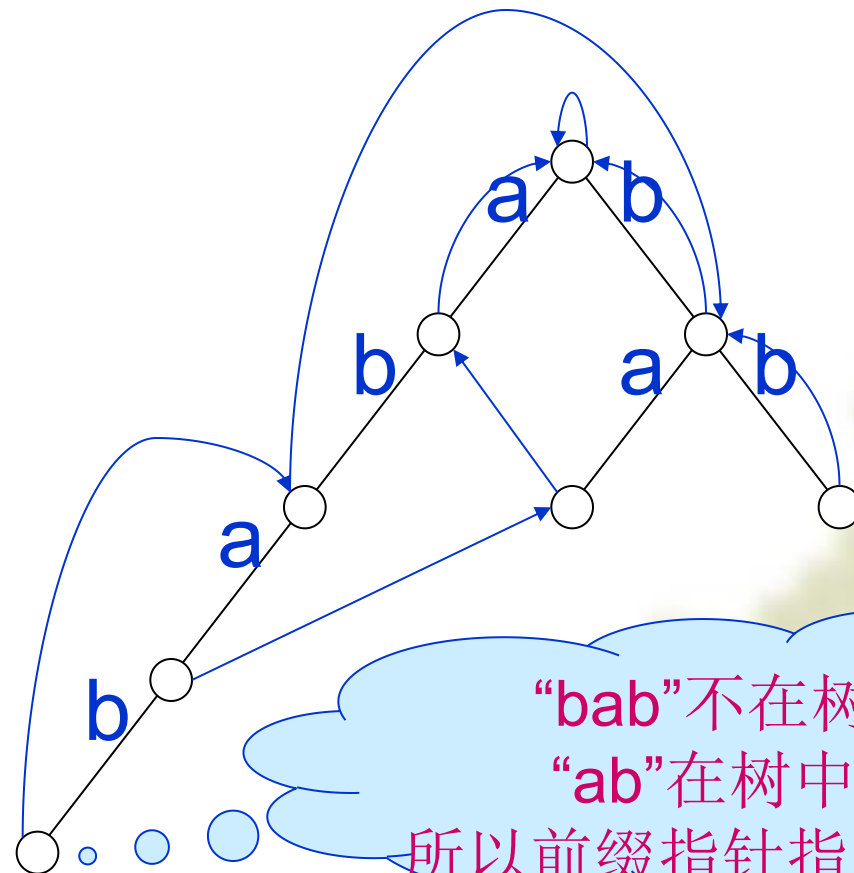
❖ 单词查找树

❖ 前缀指针的定义

- ❧ 单词前缀树之所以不同于单词树，是因为它的每一个非根结点上都有一个前缀指针（**Prefix Pointer**）。
- ❧ 设 s 为结点 p 在树中对应的字符串
- ❧ s 的所有后缀中，找到在单词树中出现的，最长的一个，设为 $s1$ 。
- ❧ p 结点的前缀指针指向 $s1$ 对应的结点。

单词前缀树（续）

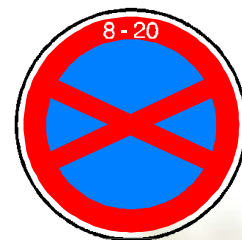
❖ 举例



单词前缀树（续）

❖ 前缀指针的生成

☞ 从定义出发，穷举+扫描

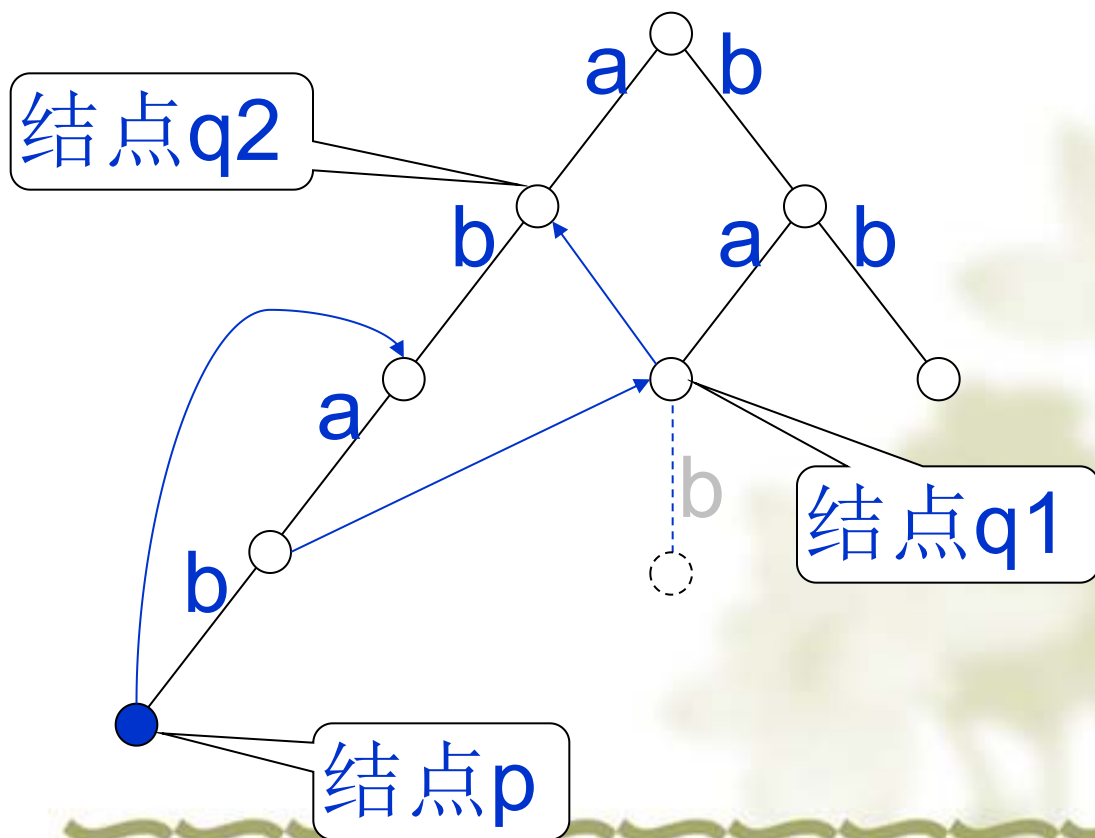


☞ 从kmp算法的前缀数组中吸取经验，通过父节点的前缀指针计算



单词前缀树（续）

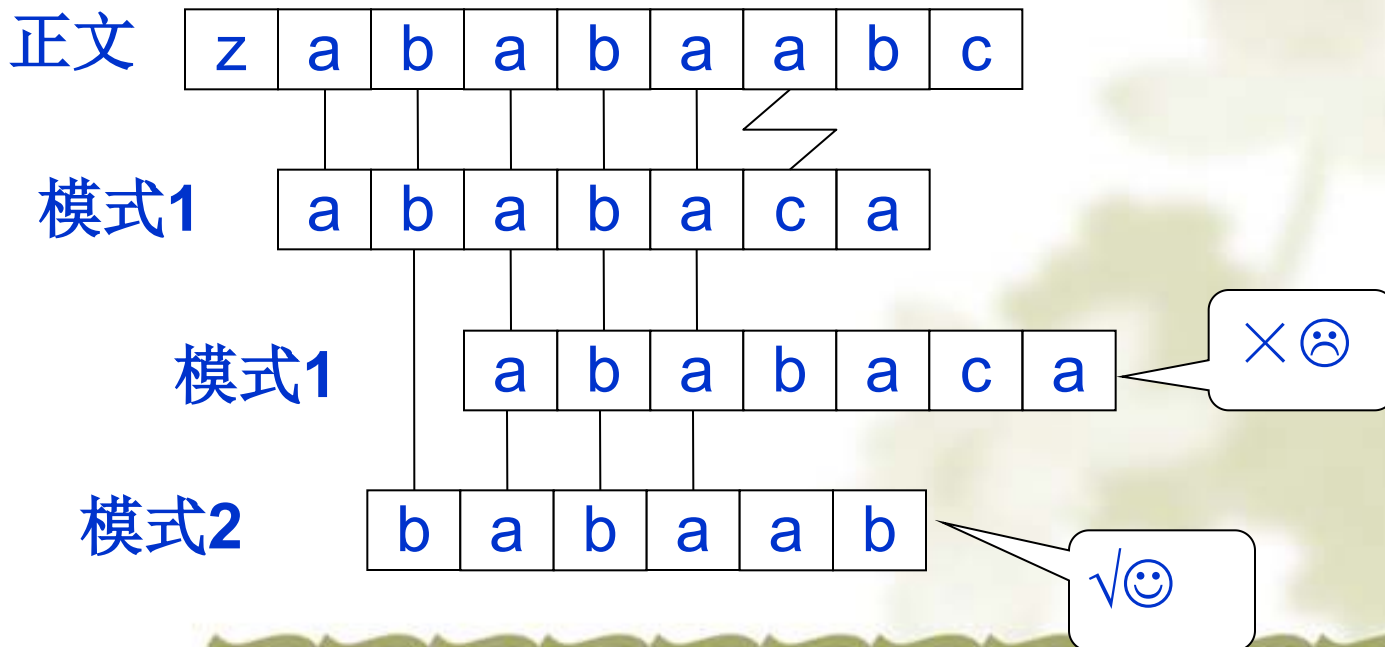
❖ 举例



主算法一

❖ kmp算法的启发

🌀 kmp算法的精髓是减少重复的计算，根据自身的位移匹配（特征），确定模式串的右移量。



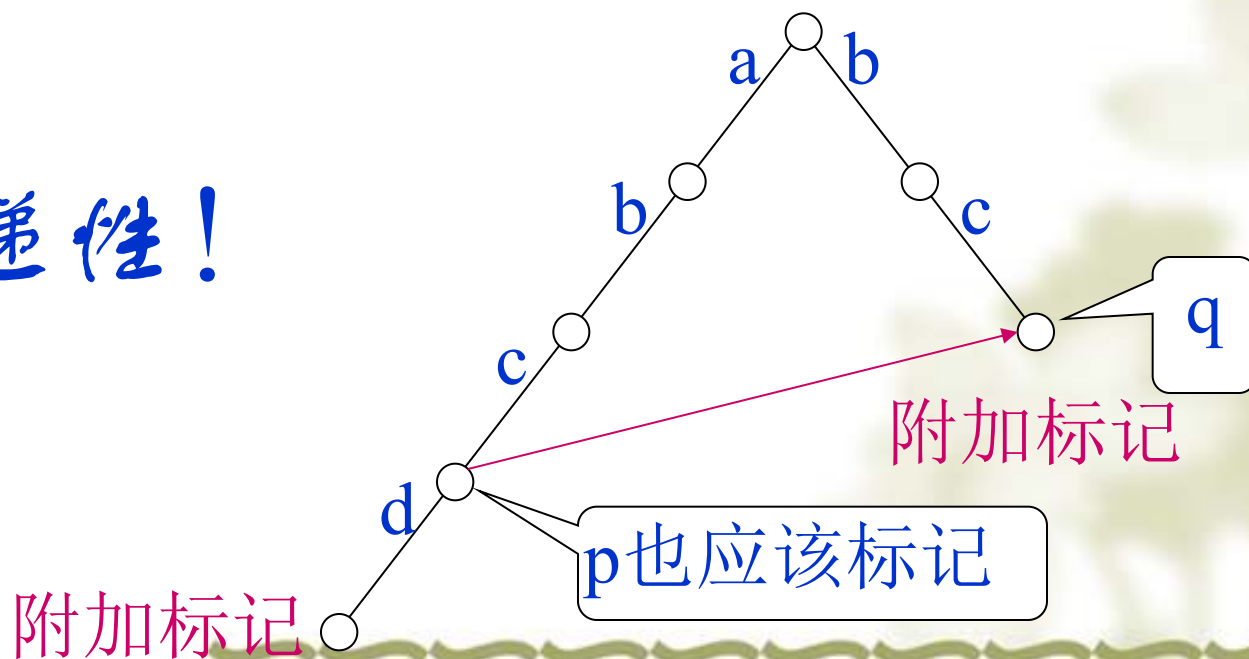
主算法一（续）

❖ 单词前缀树的使用和附加标记Okay

⌚ 模式串是构成单词前缀树的基本元素

⌚ 模式 “abcd” “bc”

传递性！

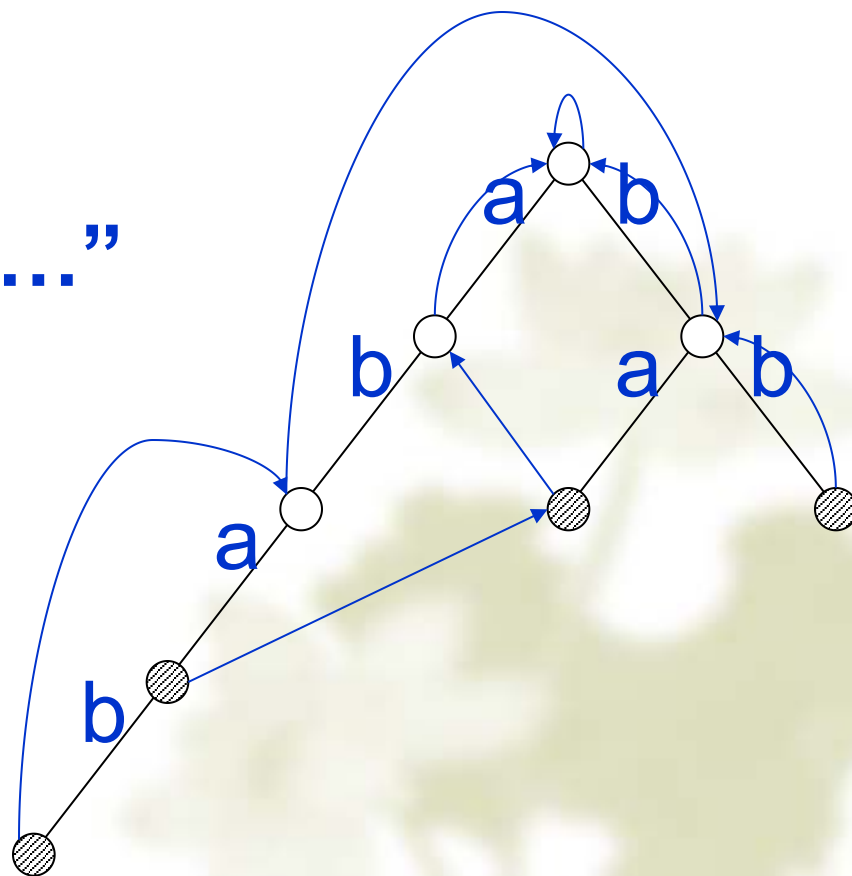


主算法一（续）

❖ 主过程

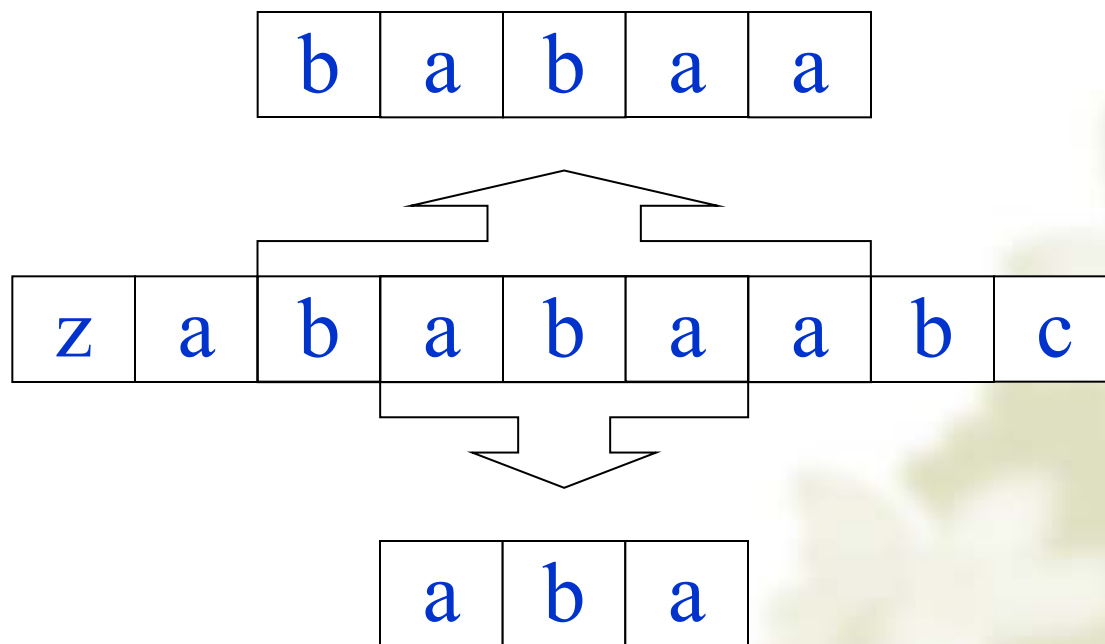
正文: “**a**~~ab~~**cc**~~ca~~**bb**.....”

找到匹配 “bb”!



主算法一（续）

❖ 一点注意



主算法一（续）

❖ 时间复杂性分析

⌚ 单词前缀树的构建

$$O(26 \cdot \sum L)$$

⌚ 正文的检索

$$O(n)$$

❖ 空间复杂性分析

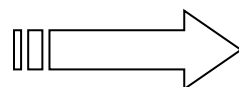
$$O(26 \cdot \sum L)$$

主算法一（续）

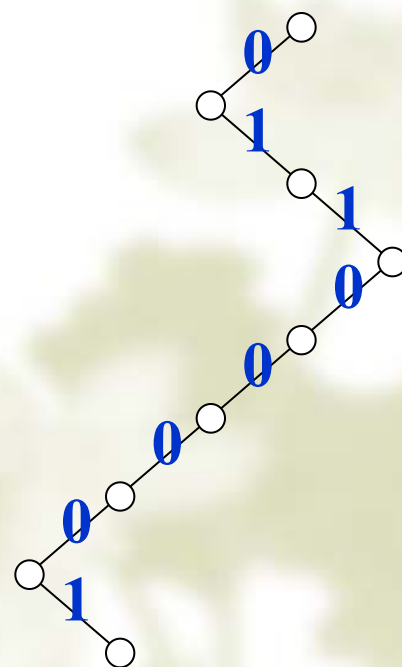
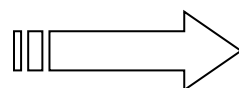
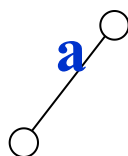
❖ 优化方案

🌀 二进制转化

a



0	1	1	0	0	0	0	1
---	---	---	---	---	---	---	---

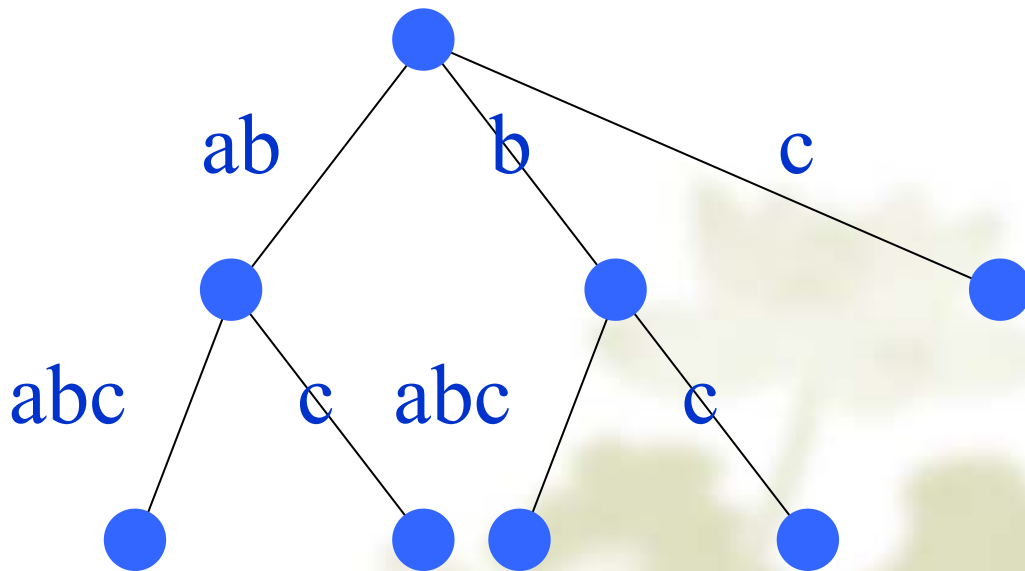


🌀 动态分配子结点+二分查找

后缀树概述

◆ 路经压缩

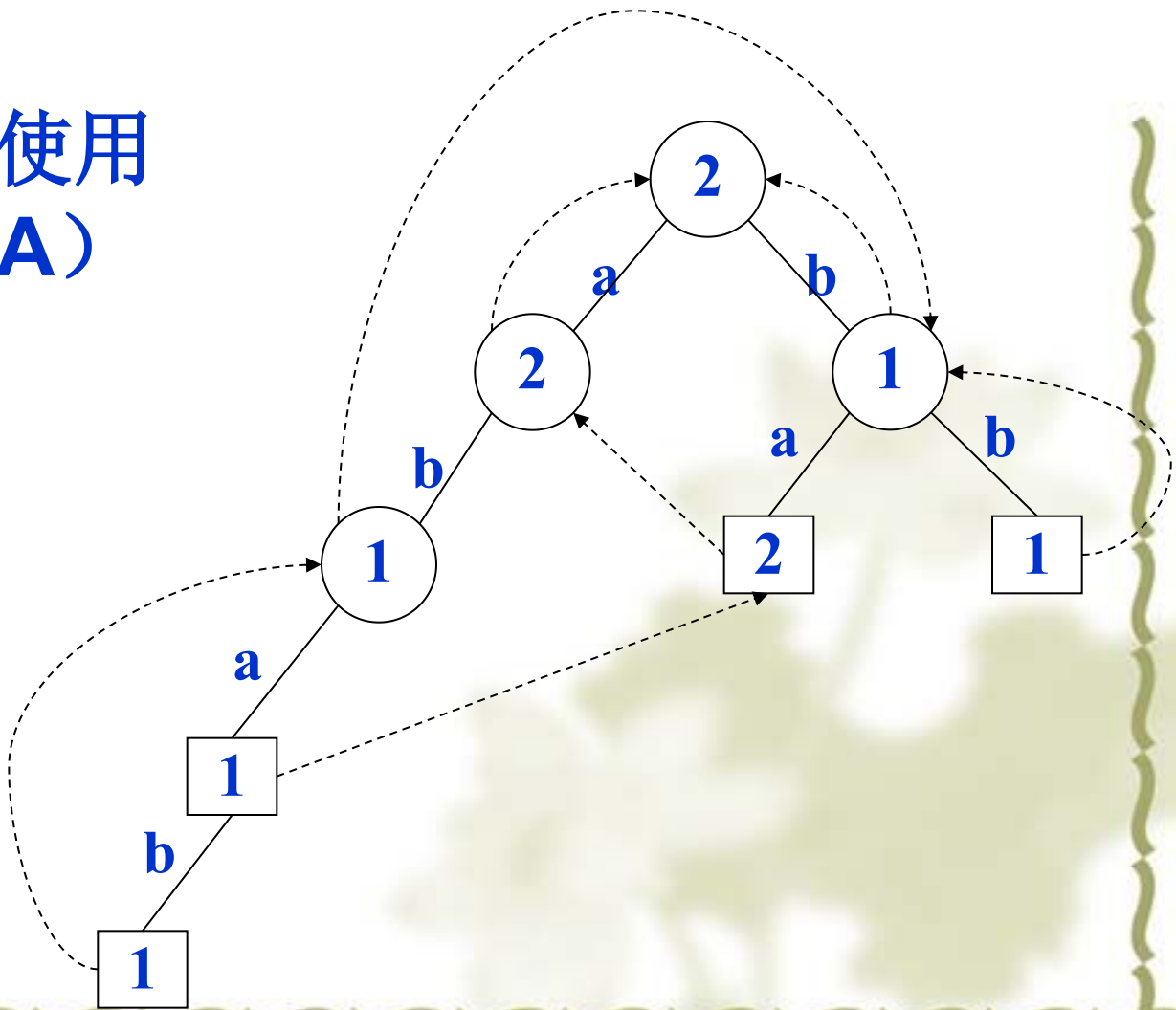
单词: “ababc”



❖ McCreight(1976), On-line Construction(1995)

主算法二

❖ 单词前缀树的使用和扩展 (**TreeA**)

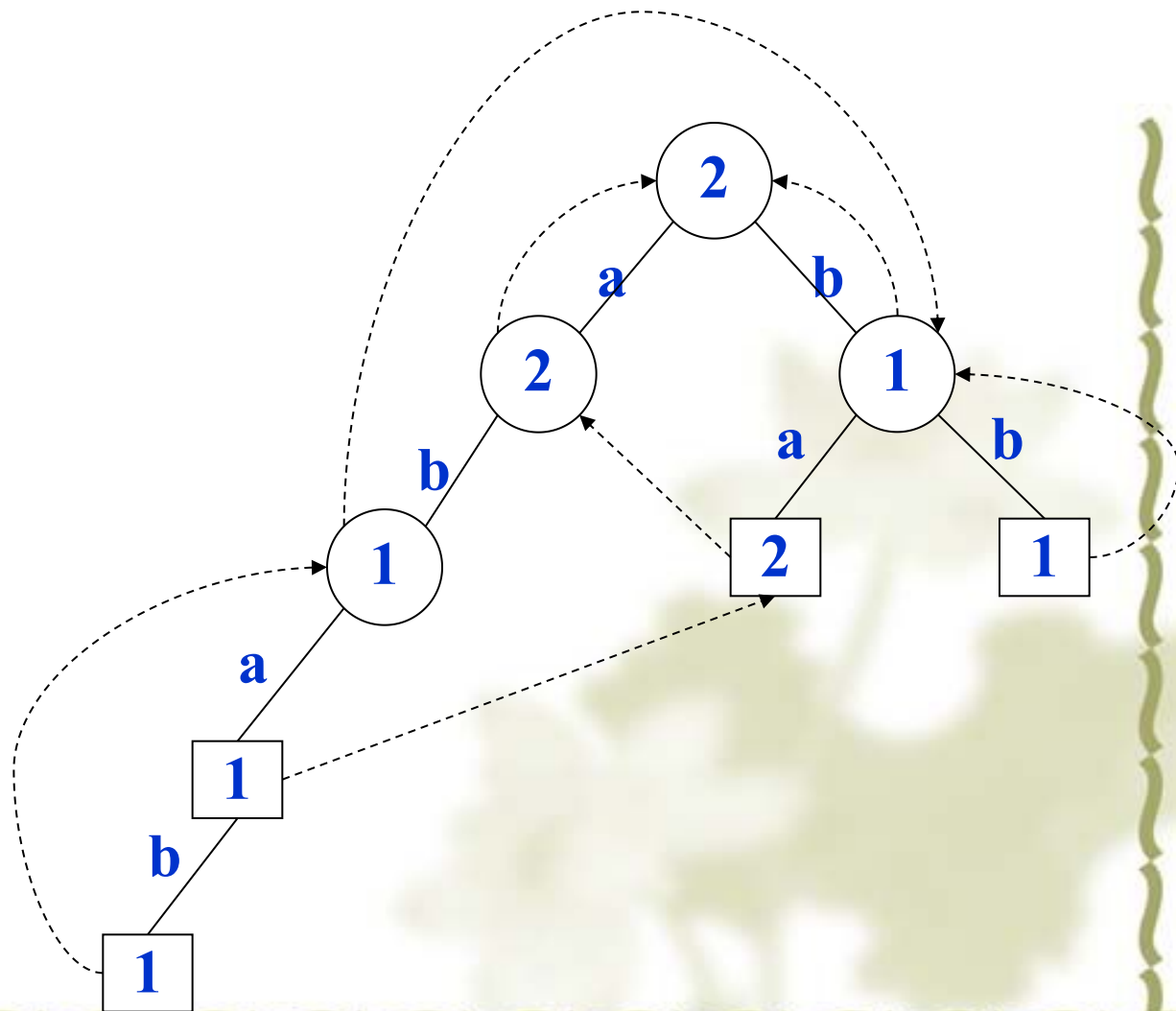


主算法二（续）

- ❖ 参数**Shift**，记录每一个结点到达任意一个**Okay**结点（自身除外）的最短路径（既可以通过树中的边，也可以通过前缀指针）

主算法二（续）

❖ 举例



主算法二（续）

❖ 后缀树的使用和扩展(TreeB)

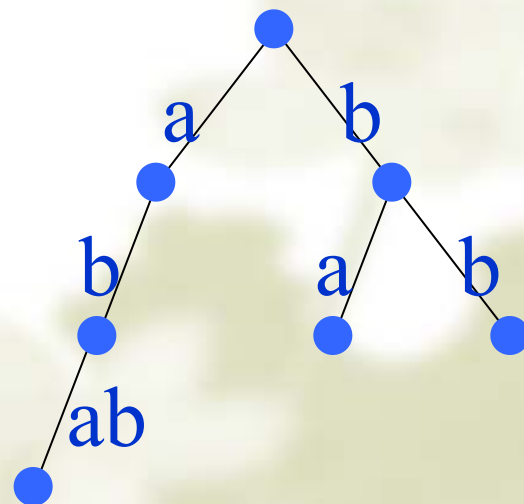
☞ 由所有模式串倒置后的所有后缀组成。

❖ 模式串为 “abab” “ba” “bb”

❖ 倒置: “baba” “ab” “bb”

❖ 作用:

☞ 在 $O(N)$ 的时间内，从后向前地查看一段长度为 N 的字符，检测它是否为任意一个模式串的子串

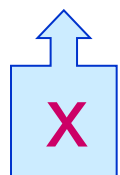


主算法二（续）

❖ TreeA上的函数ScanA

❧ Function ScanA(Left, Right, P);

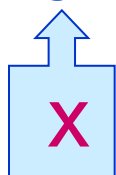
Left



X

X

Right

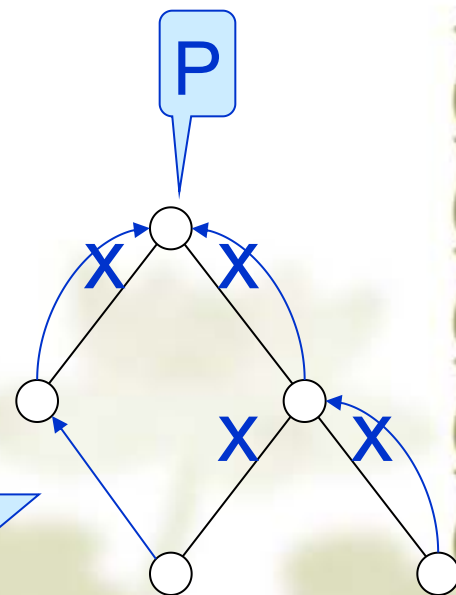
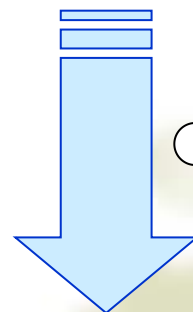


X

X

X

X



❧ 如果Shift参数 $<$ 最短的模式串长度 $\text{div } 2$ ，继续读入字符并且P继续移动

❧ 输出所有遇到的匹配

主算法二（续）

❖ TreeB上的函数ScanB

❧ **Function ScanB(Left,Right);**

❧ 在TreeB中，将T[Left..Right]从右向左进行扫描，检查其是否为某个模式串的子串，返回最后扫描到的正文的位置。

❧ 定义：

当一个字符串是某个模式串的子串时，称其为“有效的”，反之为“无效的”。

主算法二（续）

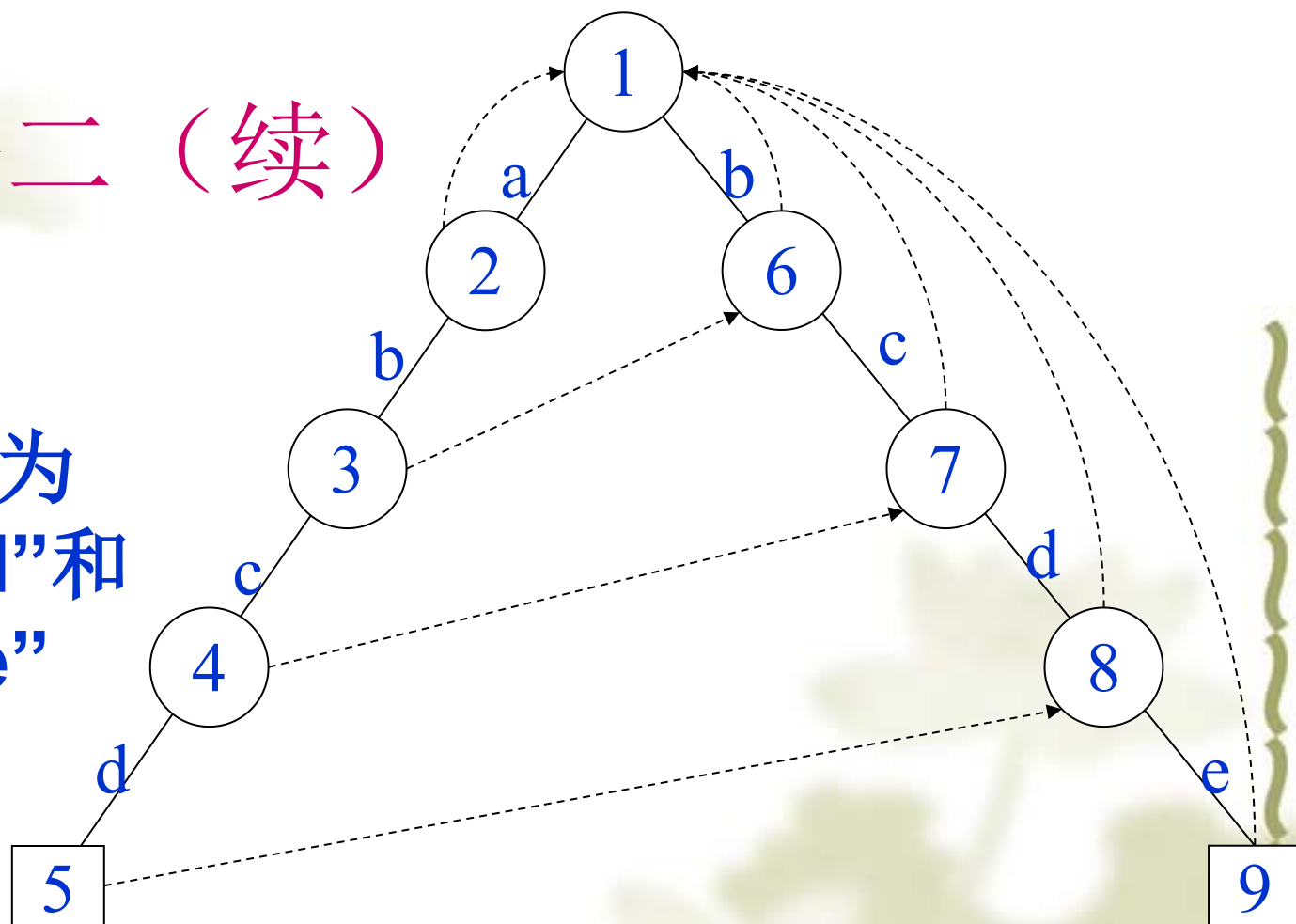
- ❖ 主过程的基本思想：
- ❖ 1、每次处理一个 $\text{Left}+1 \sim \text{Right}$ 的段落
- ❖ 2、从 Right 向左通过 ScanB 检索，最后到达位置 pos 。
- ❖ 3、从 pos 到 Right 进行 ScanA 检索。
- ❖ 4、下一个过程的 Left 为 ScanA 检索到的正文位置， Right 为 $\text{Left} + \text{当前TreeA上的结点的Shift参数}$

主算法二（续）

❖ 举例

模式串为
“abcd”和
“bcde”

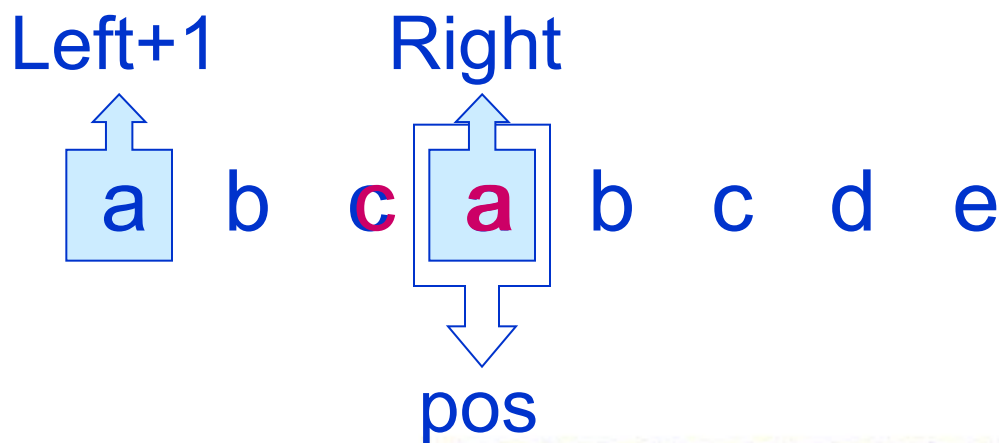
TreeA



编号	1	2	3	4	5	6	7	8	9
Shift	4	3	2	1	1	3	2	1	4

主算法二（续）

- ❖ $T = \text{"abcabcde"}$, $\text{Left} = 0$, $\text{Right} = 4$, $P = 1$
- ❖ 从 Right 到 $\text{Left} + 1$ 逆向进行ScanB
 - ⌘ “a”为“有效的”
 - ⌘ “ca”为“无效的”，所以 $\text{pos} = 4$ 。



模式串
“abcd”
“bcde”
ca没出现

主算法二（续）

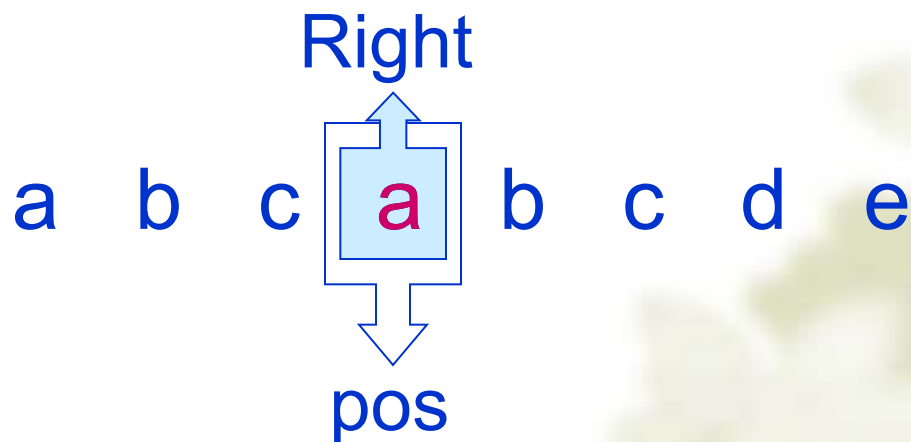
- ❖ 1..3的正文位置上，不可能出现模式的匹配

a	b	c	a	b	c	d	e
---	---	---	---	---	---	---	---

- ❖ **ScanA**的检索需要从**TreeA**根结点重新开始，**P**指针重置为**TreeA**的根结点。

主算法二（续）

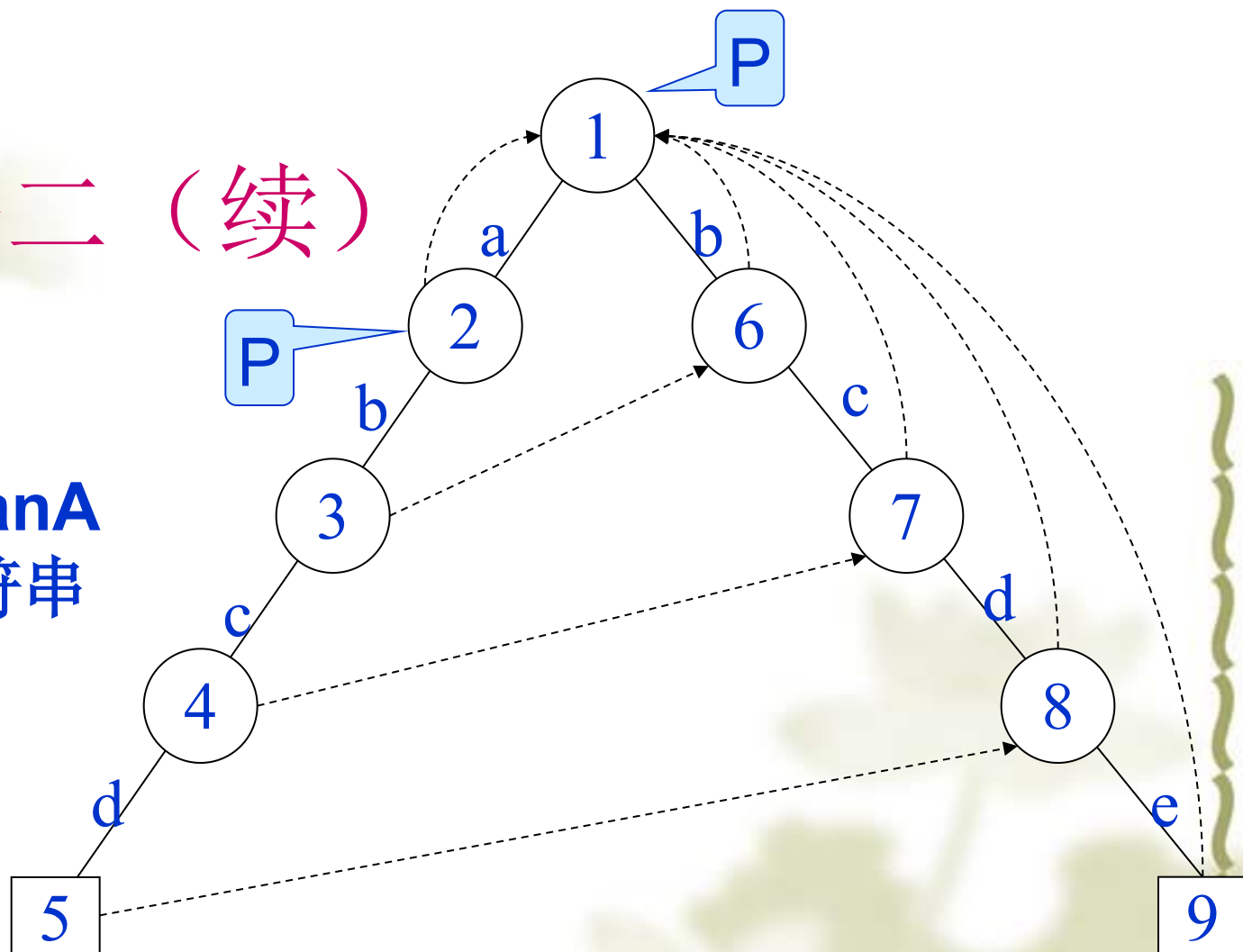
❖ 从pos到Right进行ScanA检索



主算法二（续）

❖ 阶段1:

☞ 正向ScanA
检索字符串
“a”



编号	1	2	3	4	5	6	7	8	9
Shift	4	3	2	1	1	3	2	1	4

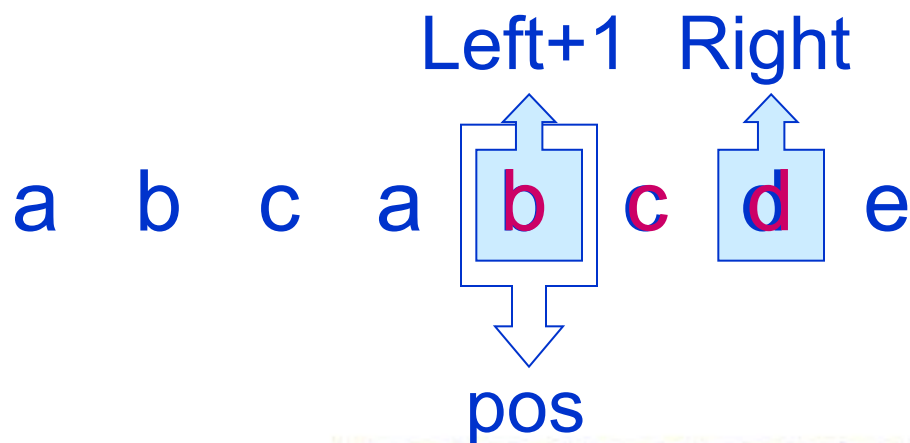
主算法二（续）

❖ $T = \text{"abcabcde"}$

⤿ $\text{Left} = 4, \text{Right} = \text{Left} + \text{Shift}[P] = 7, P = 2$

❖ 从 Right 到 $\text{Left}+1$ 逆向进行 ScanB

⤿ 有 “bcd” 为 “有效的”，所以 $\text{pos} = 5$ 。



模式串

“~~abc~~cd”

“bcde”

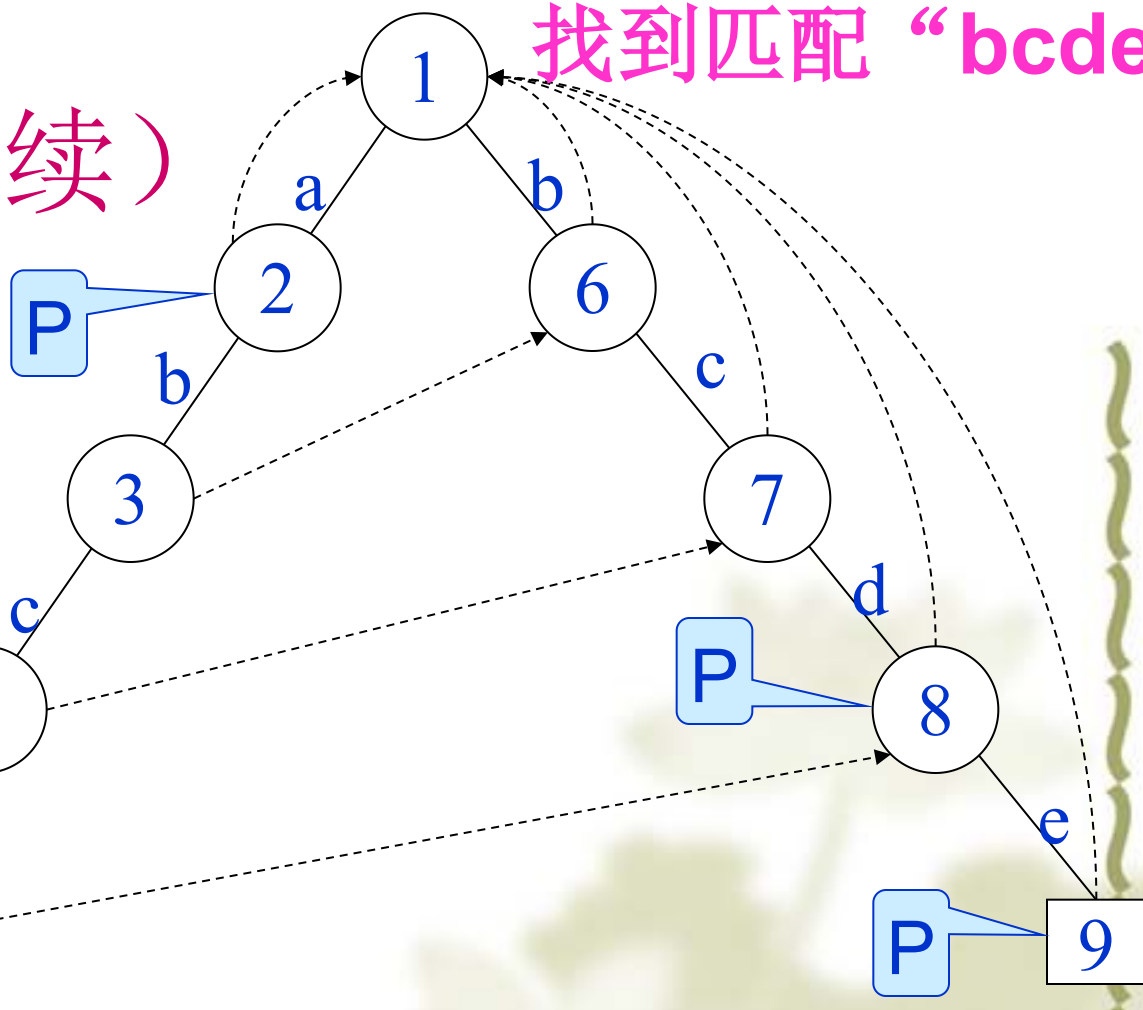
$\text{pos} = L + 1$

主算法二（续）

❖ 阶段1:

🌀 正向ScanA
检索字符串
“bcd”

🌀 再读入字符
“e”



编号	1	2	3	4	5	6	7	8	9
Shift	4	3	2	1	1	3	2	1	4

主算法二（续）

❖ 时间复杂度分析：

❧ 设最短的模式串长度为 θ

❧ 最坏情况 $O(N)$


❧ 设所有的模式串长度均为 θ ， θ 足够大时，若正文随机。

❧ **ScanB** 将所有的 $T[\text{Left}+1..\text{Right}]$ 的字符扫描完毕的概率并不大，可以证明平均复杂度：

$$O\left(\frac{n \cdot \log_{26} \theta}{\theta}\right)$$


算法总结——启示1

❖ $\frac{\theta}{2}$ 的使用

↪ 变大—— $\frac{2\theta}{3}$ 

中间值!

↪ **ScanA**将很难退出，平均复杂度变大!

↪ 变小—— $\frac{\theta}{3}$ 

↪ **Right-Left**的差变小，**ScanB**的pos回到Left+1的可能性变大，平均复杂度变大!

算法总结——启示2

- ❖ 优劣得所的思想
- ❖ 算术平均数 —— 本算法
- ❖ 几何平均数 —— Editor块状链表
- ❖ 不断更新的数组 $A[1..10000]$ ，求 $\max\{A[1..i]\}$
 - ↪ 更新： $O(10000)$ 。取值： $O(1)$
 - ↪ 二叉树（不易实现）
 - ↪ $\max1[i]$ 记录 $A[1*100 \sim (i-1)*100]$ 中的最大值
 - ↪ 更新： $O(100)$ 。取值： $O(100)$

启示

- ❖ 一条铁链的强度，决定于最弱的铁环的强度
一个水桶的水量，决定于最短的竹片的长度
- ❖ 在算法深度达到一定程度的前提下，我们应该将算法的广度拓宽，多种算法并用，从最弱的点找到解决问题的钥匙。
- ❖ 只要不断地从瓶颈处突破，解题将会“有山就有路，有河就能渡”！
- ❖ 最重要的是领悟“融会贯通”的思想



That's all!

**Thank you for
listening.**

