字符串匹配

胡船长
初航我带你，远航靠自己

二、多模匹配问题

1. 基于哈希：Rabin-Karp 算法
2. 初探 NFA：Shift-and/or 算法
3. 神兵利器：Trie 字典树
4. 飞升蜕变：AC 自动机

二、多模匹配问题

1. 基于哈希：Rabin-Karp 算法
2. 初探 NFA：Shift-and/or 算法
3. 神兵利器：Trie 字典树
4. 飞升蜕变：AC 自动机

字符串哈希的基本思想

`Hash(s) != Hash(t)` 一定不相等

`Hash(s) == Hash(t)` 不一定相等

两种常用的字符串哈希

$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{i}$$

$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{n-i-1}$$

两种常用的字符串哈希

$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{i} \qquad a * base^{0} + b * base^{1} + c * base^{2}$$

$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{n-i-1} \qquad a * base^{2} + b * base^{1} + c * base^{0}$$

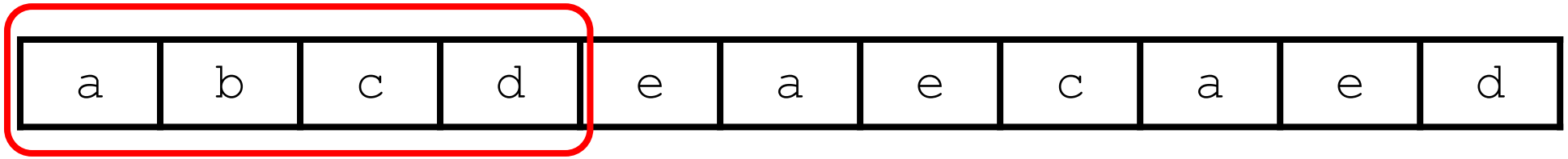
两种常用的字符串哈希

$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{n-i-1}$$

$$a * base^2 + b * base^1 + c * base^0$$

Rabin-Karp 算法思想

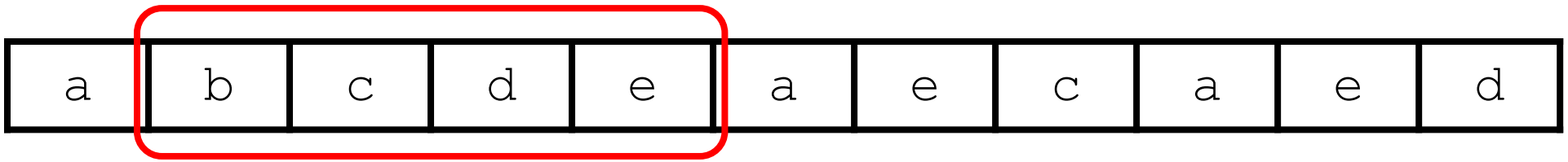
$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{n-i-1}$$



Hash1

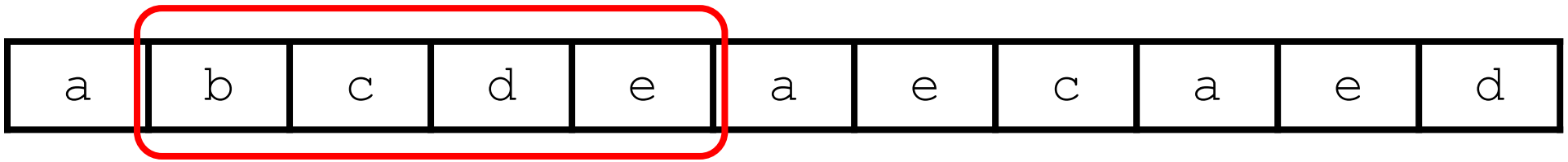
Rabin-Karp 算法思想

$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{n-i-1}$$



Rabin-Karp 算法思想

$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{n-i-1}$$

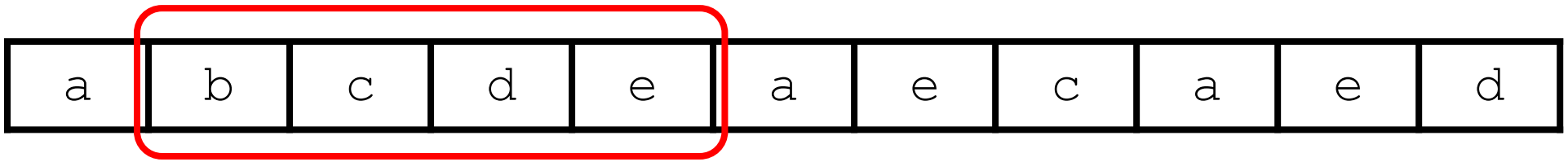


Hash2

Hash1 = a*base^3+b*base^2+c*base^1+d*base^0

Rabin-Karp 算法思想

$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{n-i-1}$$

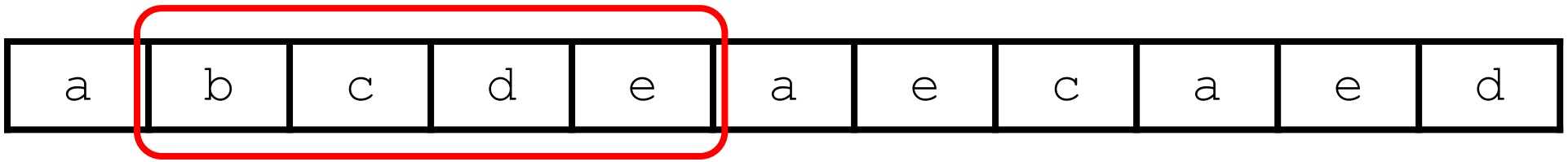


Hash1 = a*base^3+b*base^2+c*base^1+d*base^0

Hash2 = b*base^3+c*base^2+d*base^1+e*base^0

Rabin-Karp 算法思想

$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{n-i-1}$$

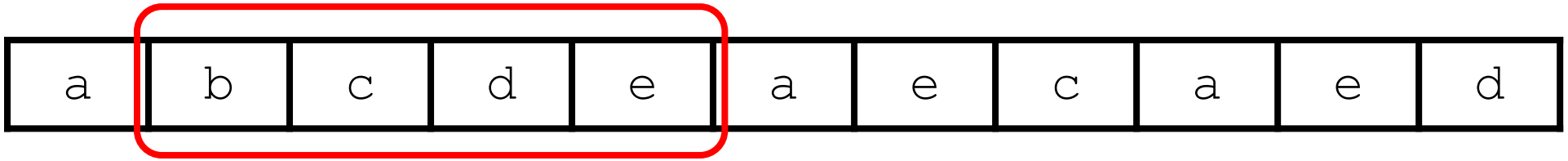


Hash1 = a*base^3+b*base^2+c*base^1+d*base^0

Hash2 = b*base^3+c*base^2+d*base^1+e*base^0

Rabin-Karp 算法思想

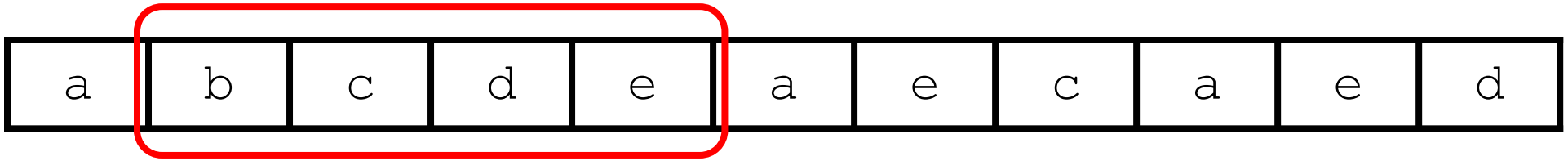
$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{n-i-1}$$



Hash2 = Hash1*base+e*base^0 -a*base^4

Rabin-Karp 算法思想

$$Hash(s) = \sum_{i=0}^{n} s[i] * base^{n-i-1}$$



Hash2 = Hash1*base+s[i]-s[i-n]*base^{n}

Rabin-Karp 算法思想

问题 1：两个不同的字符串拥有相同的哈希值，怎么办？

问题 2：算法的时间复杂度？

二、多模匹配问题

1. 基于哈希：Rabin-Karp 算法
2. 初探 NFA：Shift-and/or 算法
3. 神兵利器：Trie 字典树
4. 飞升蜕变：AC 自动机

二、多模匹配问题

1. 基于哈希：Rabin-Karp 算法
2. 初探 NFA：Shift-and/or 算法
3. 神兵利器：Trie 字典树
4. 飞升蜕变：AC 自动机

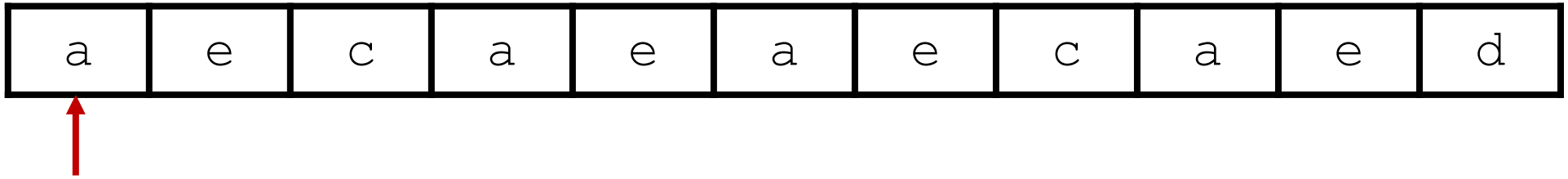
SHIFT-AND 算法

模式串 T

	a	e	c	a	e	d
	0	1	2	3	4	5
d[a]	1	0	0	1	0	0
d[c]	0	0	1	0	0	0
d[d]	0	0	0	0	0	1
d[e]	0	1	0	0	1	0

SHIFT-AND 算法

母串 S

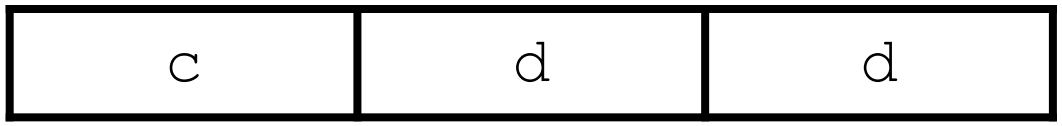


	0	1	2	3	4	5
d[a]	1	0	0	1	0	0
d[c]	0	0	1	0	0	0
d[d]	0	0	0	0	0	1
d[e]	0	1	0	0	1	0
P	0	0	0	0	0	0

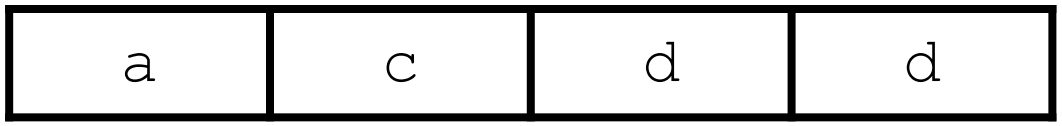
```
P = (P << 1 | 1) & d[s[i]]

P & (1 << (n - 1)) ?= 1
```

SHIFT-AND 算法--随堂练习1

模式串 T



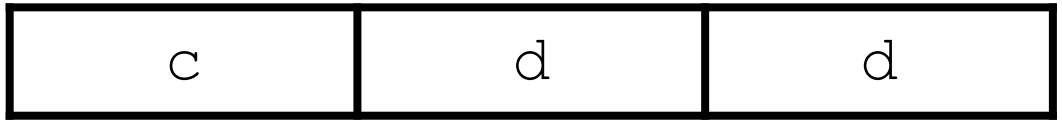
文本串s



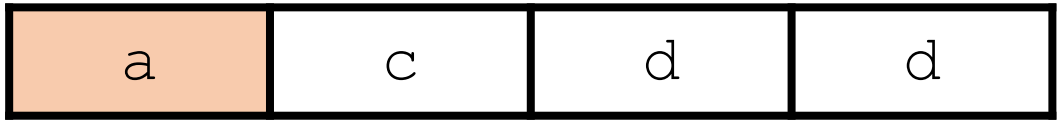

```
P = (P << 1 | 1) & d[s[i]]

P & (1 << (n - 1)) ?= 1
```

SHIFT-AND 算法

模式串 T



文本串 s



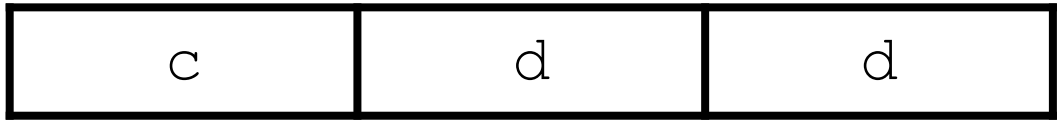
相应的位置为1，意味着当前文本串后缀能匹配到模式串的第几位前缀

c 编码：001
d 编码：110
P = (000 << 1 | 1) & code(a) = 001 & 000 = 000

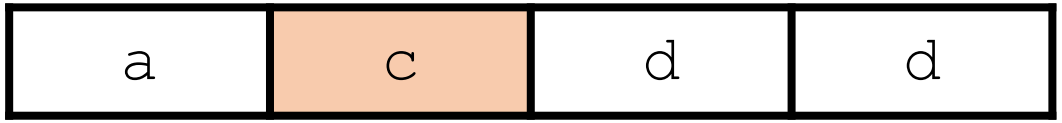
```
P = (P << 1 | 1) & d[s[i]]

P & (1 << (n - 1)) ?= 1
```

SHIFT-AND 算法

模式串 T



文本串s



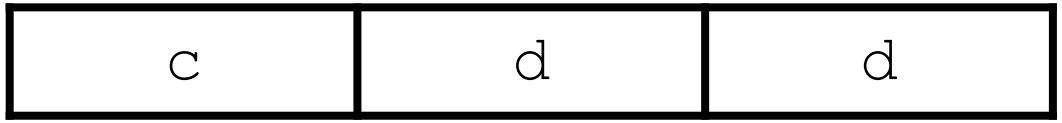
相应的位置为1，意味着当前文本串后缀能匹配到模式串的第几位前缀

c 编码：001
d 编码：110
P = (000 << 1 | 1) & code(a) = 001 & 000 = 000
P = (000 << 1 | 1) & code(c) = 001 & 001 = 001

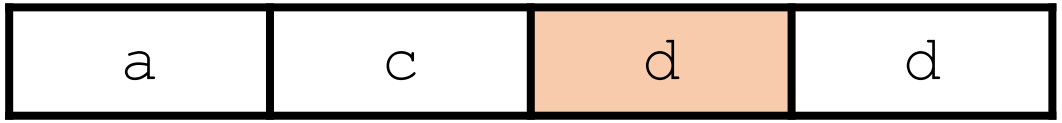
```
P = (P << 1 | 1) & d[s[i]]

P & (1 << (n - 1)) ?= 1
```

SHIFT-AND 算法

模式串 T



文本串s



相应的位置为1，意味着当前文本串后缀能匹配到模式串的第几位前缀

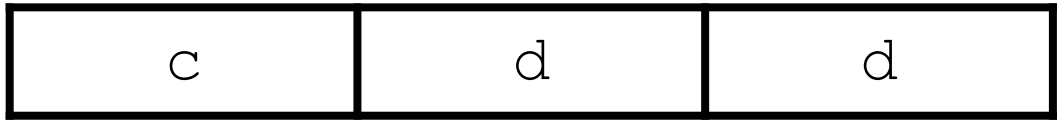
c 编码：001
d 编码：110

```
P = (000 << 1 | 1) & code(a) = 001 & 000 = 000
P = (000 << 1 | 1) & code(c) = 001 & 001 = 001
P = (001 << 1 | 1) & code(d) = 011 & 110 = 010
```

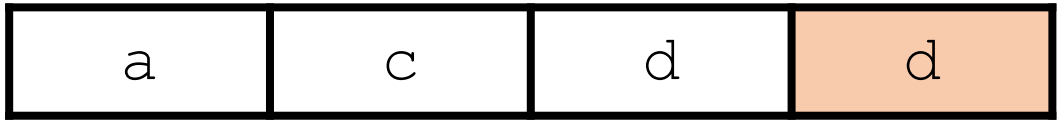
```
P = (P << 1 | 1) & d[s[i]]

P & (1 << (n - 1)) ?= 1
```

SHIFT-AND 算法

模式串 T



文本串s



相应的位置为1，意味着当前文本串后缀能匹配到模式串的第几位前缀

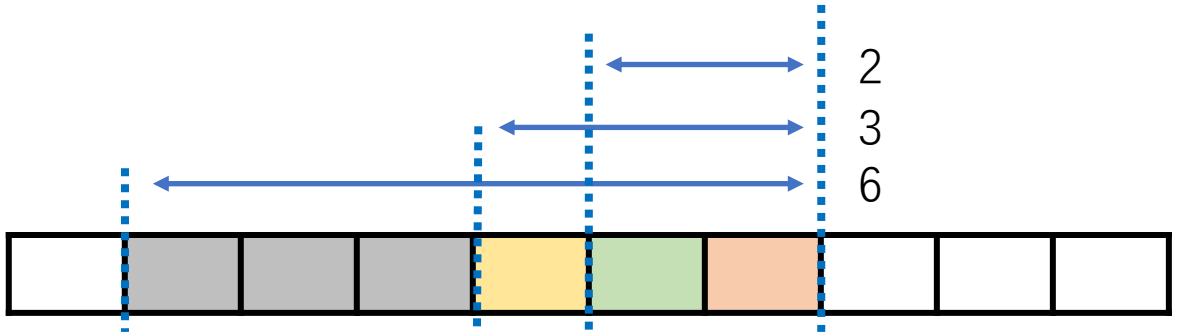
c 编码：001
d 编码：110

```
P = (000 << 1 | 1) & code(a) = 001 & 000 = 000
P = (000 << 1 | 1) & code(c) = 001 & 001 = 001
P = (001 << 1 | 1) & code(d) = 011 & 110 = 010
P = (010 << 1 | 1) & code(d) = 101 & 110 = 100
```

```
P = (P << 1 | 1) & d[s[i]]

P & (1 << (n - 1)) ?= 1
```

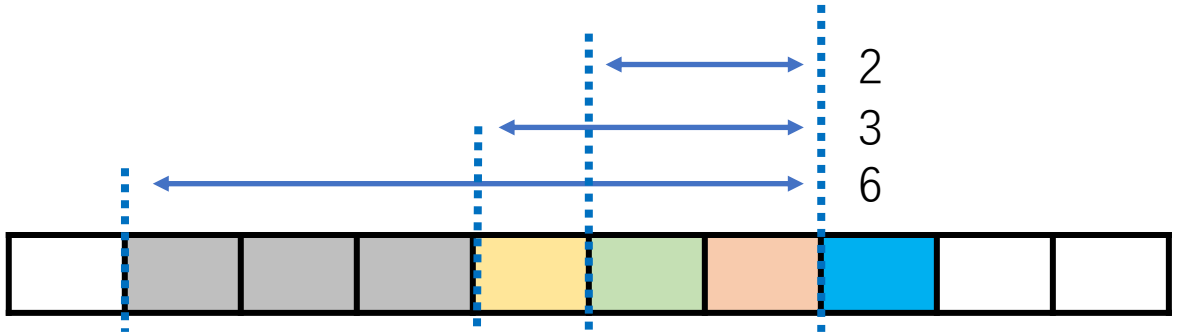
SHIFT-AND 算法

P = 100110



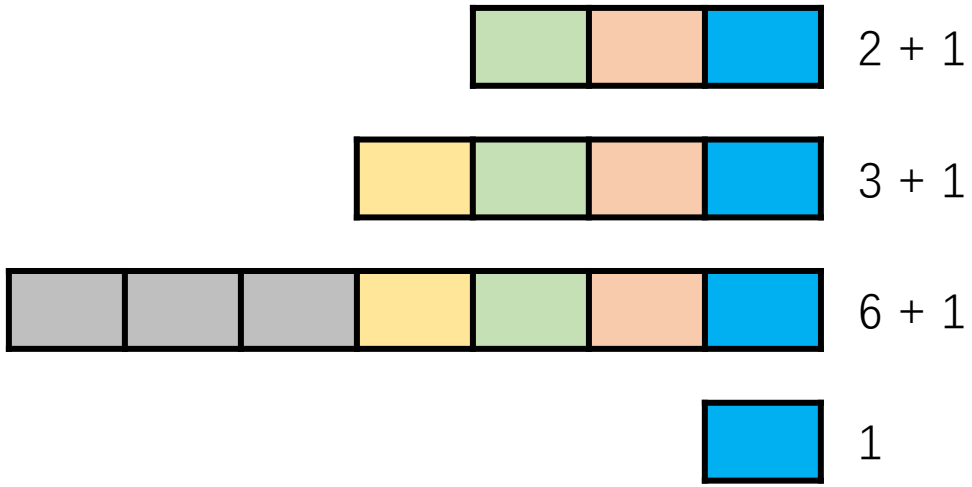
SHIFT-AND 算法

P = 100110



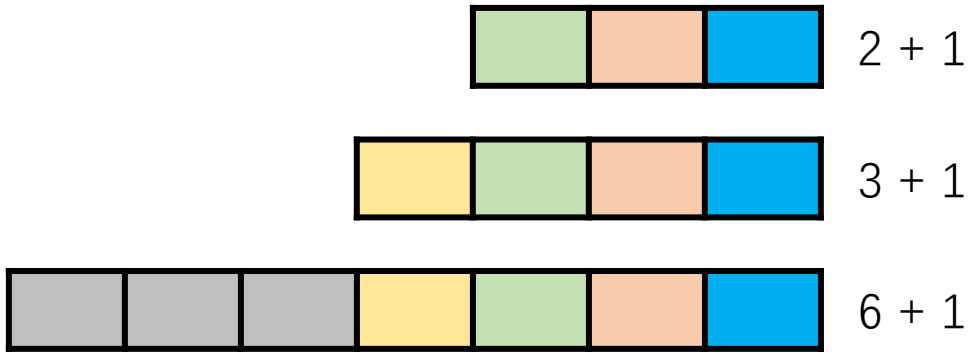
SHIFT-AND 算法

P = 100110



SHIFT-AND 算法

```
P = 100110
```

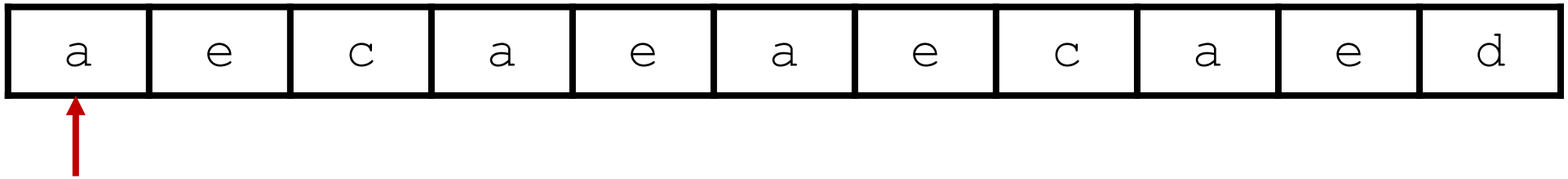



```
P = (P << 1 | 1) & code
```

```
P = (P << 1 | 1) & code
```

SHIFT-AND 算法

母串 **S**



	0	1	2	3	4	5
d[a]	1	0	0	1	0	0
d[c]	0	0	1	0	0	0
d[d]	0	0	0	0	0	1
d[e]	0	1	0	0	1	0
P	0	0	0	0	0	0

```
P = (P << 1 | 1) & d[s[i]]

P & (1 << (n - 1)) ?= 1
```

SHIFT-AND 算法--随堂练习2

练习题：

给出一个正则表达式，形如：

```
(a|b|c)&(c|d)&e&(f|a|b)
```

再给出一段文本，问文本中有多少段不同的子文本可以被上述正则表达式匹配。

SHIFT-OR 算法

1. 有效位用 0 表示
2. 优化：少了左移以后的或1操作

SHIFT-OR 算法

1. 有效位用 0 表示
2. 优化：少了左移以后的或1操作

模式串 T

a	e	c	a	e	d

	0	1	2	3	4	5
d[a]	1	0	0	1	0	0
d[c]	0	0	1	0	0	0
d[d]	0	0	0	0	0	1
d[e]	0	1	0	0	1	0

SHIFT-OR 算法

1. 有效位用 0 表示
2. 优化：少了左移以后的或1操作

模式串 T

a	e	c	a	e	d

	0	1	2	3	4	5
d[a]	0	1	1	0	1	1
d[c]	1	1	0	1	1	1
d[d]	1	1	1	1	1	0
d[e]	1	0	1	1	0	1

SHIFT-OR 算法

1. 有效位用 0 表示
2. 优化：少了左移以后的或1操作

```
P = (P << 1) | d[s[i]]

P & (1 << (n - 1)) ?= 0
```

模式串 T

a	e	c	a	e	d

	0	1	2	3	4	5
d[a]	0	1	1	0	1	1
d[c]	1	1	0	1	1	1
d[d]	1	1	1	1	1	0
d[e]	1	0	1	1	0	1

SHIFT-OR 算法

1. 有效位用 0 表示
2. 优化：少了左移以后的或1操作

```
P = (P << 1) | d[s[i]]

P & (1 << (n - 1)) ?= 0
```

模式串 T

	a	e	c	a	e	d
	0	1	2	3	4	5
d[a]	0	1	1	0	1	1
d[c]	1	1	0	1	1	1
d[d]	1	1	1	1	1	0
d[e]	1	0	1	1	0	1

二、多模匹配问题

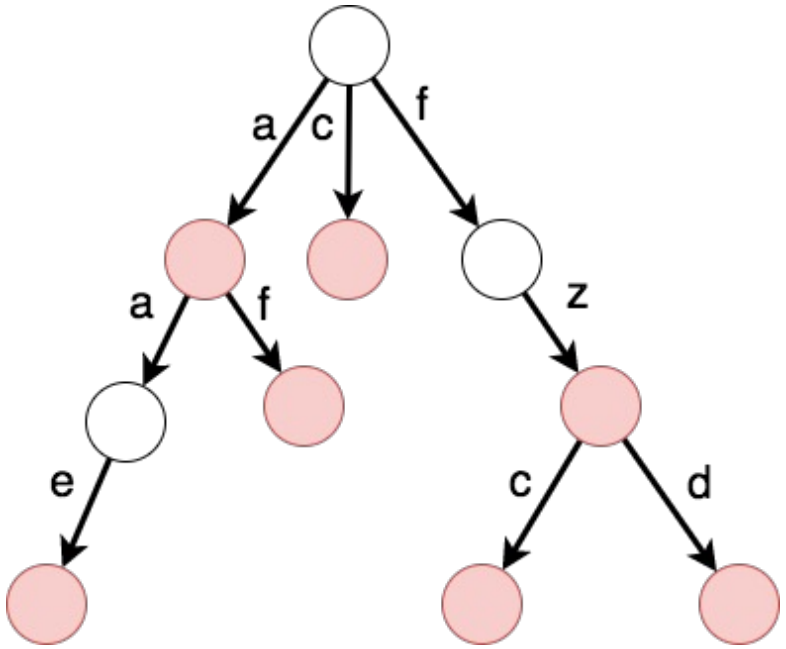
1. 基于哈希：Rabin-Karp 算法
2. 初探 NFA：Shift-and/or 算法
3. 神兵利器：Trie 字典树
4. 飞升蜕变：AC 自动机

二、多模匹配问题

1. 基于哈希：Rabin-Karp 算法
2. 初探 NFA：Shift-and/or 算法
3. **神兵利器：Trie 字典树**
4. 飞升蜕变：AC 自动机

字典树

字典树

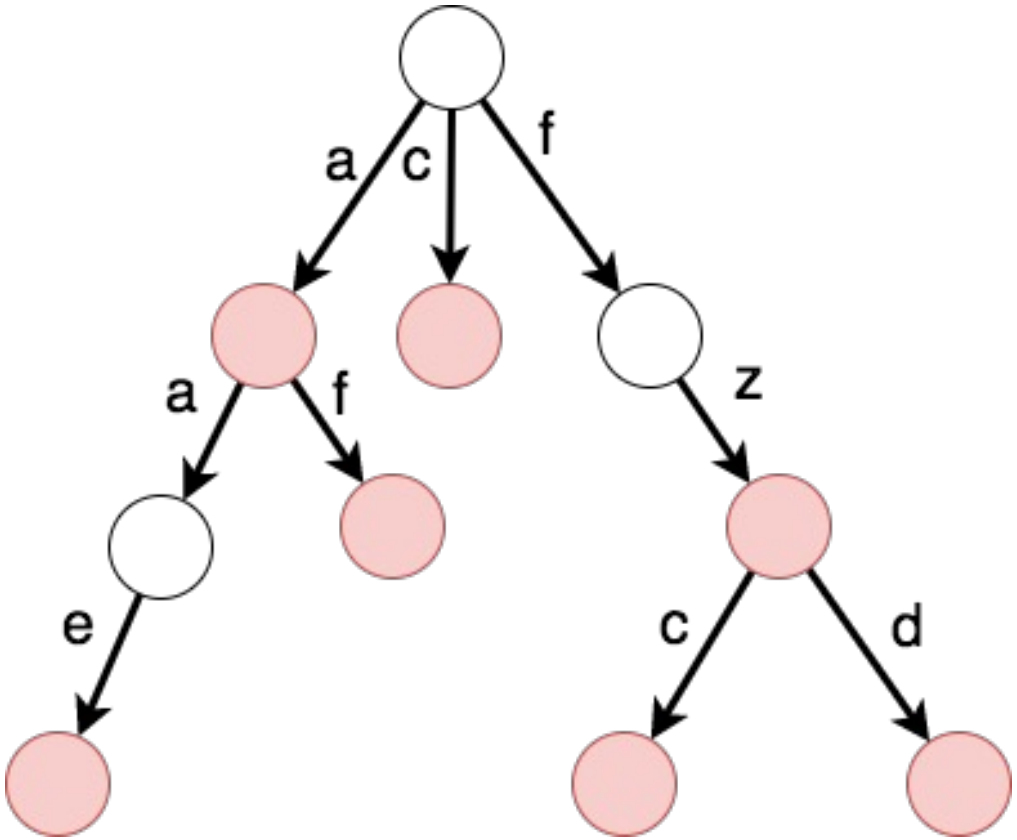


1、姓名：
Trie

2、曾用名：
字典树
单词查找树

3、作用：
单词查找
字符串排序

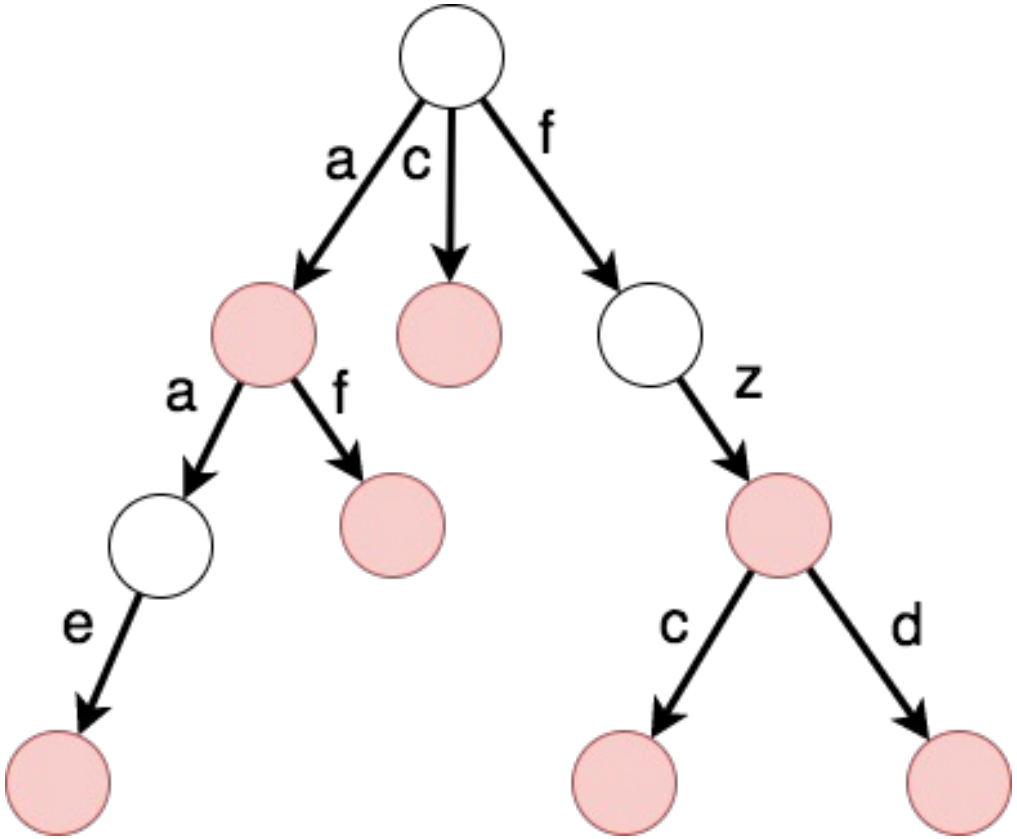
字典树



1、练习题：

按照字典序写出左侧字典树中所有的单词

字典树



1、练习题：

按照字典序写出左侧字典树中所有的单词

a
aae
af
c
fz
fzc
fzd

字典树升华

树的 **节点** 代表什么?

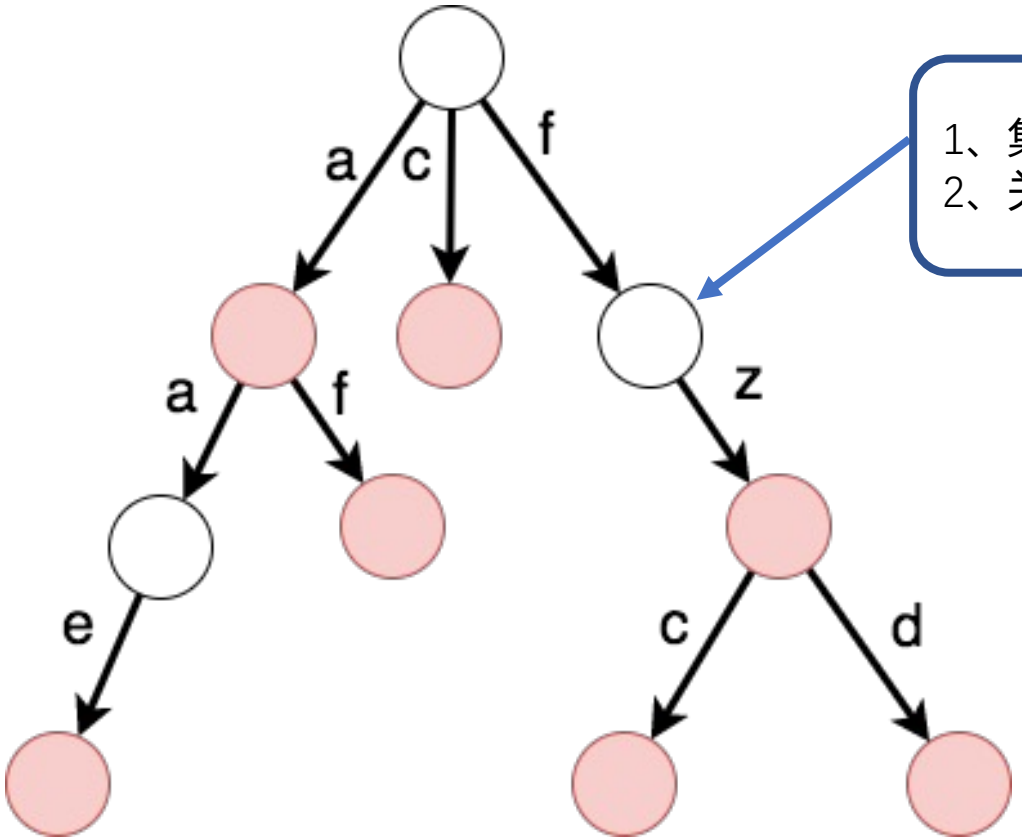
树的 **边** 代表什么?

字典树升华

树的 **节点** 代表 **集合**

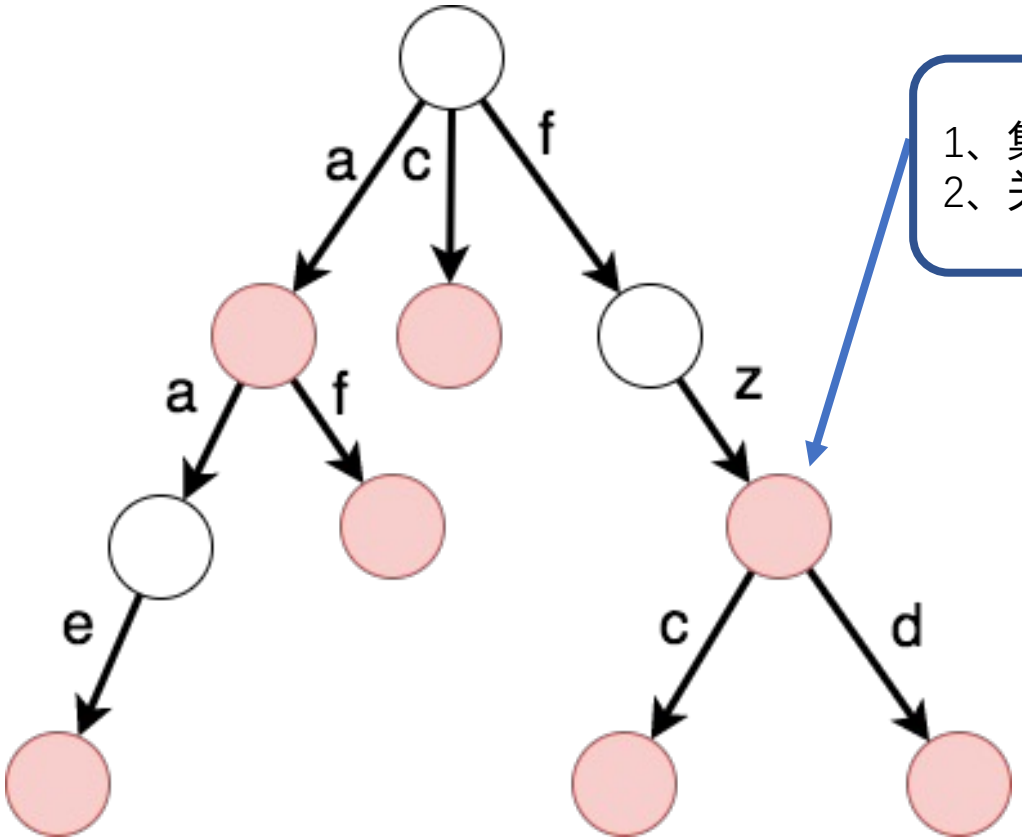
树的 **边** 代表 **关系**

再看：字典树



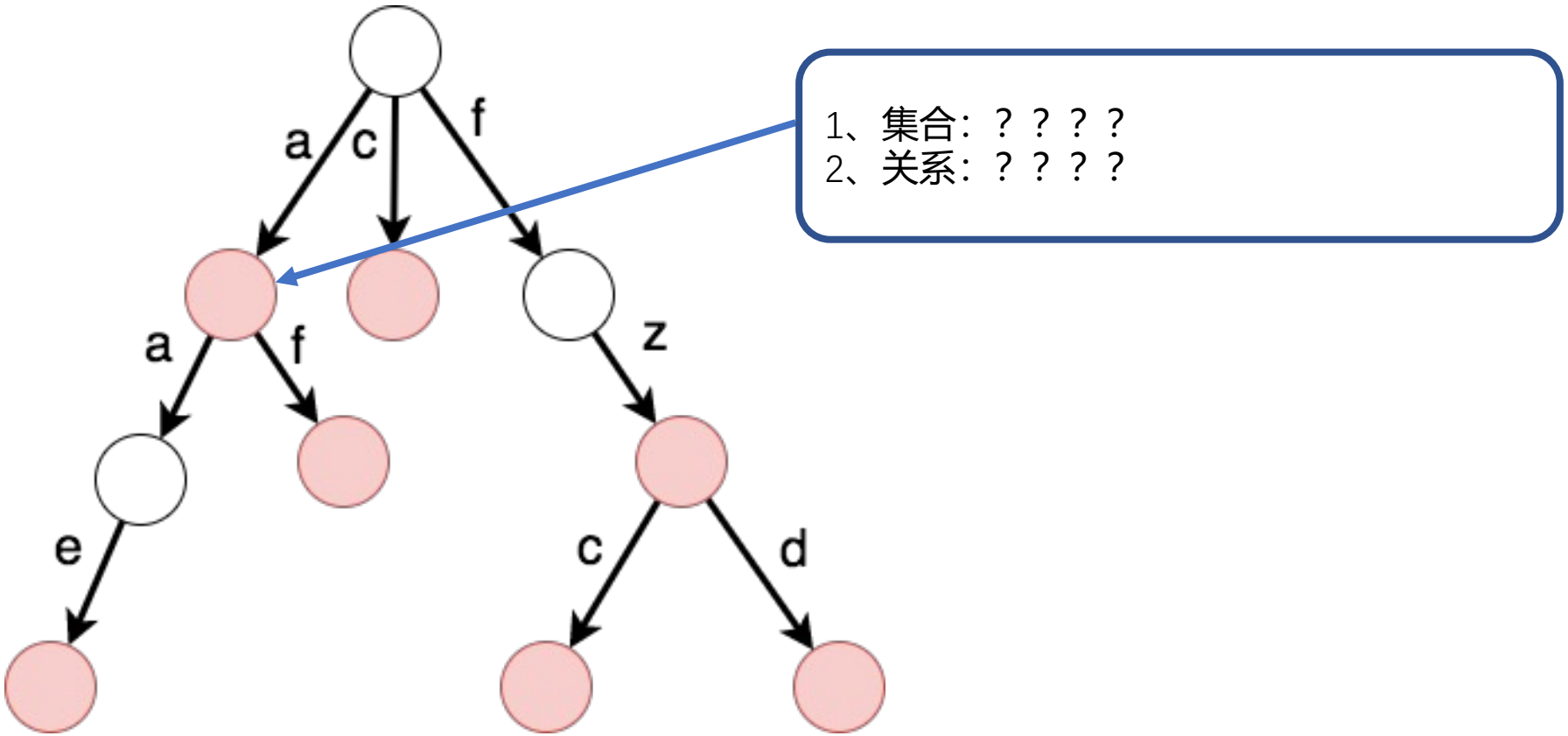
1、集合：包含了 `fz`、`fzc`、`fzd` 三个单词
2、关系：前缀是 `f` 的单词

再看：字典树



1、集合：？？？？
2、关系：？？？？

再看：字典树



双数组字典树

双数组字典树

逻辑结构还是那个结构，

只是换了一种信息的表示方法

双数组字典树

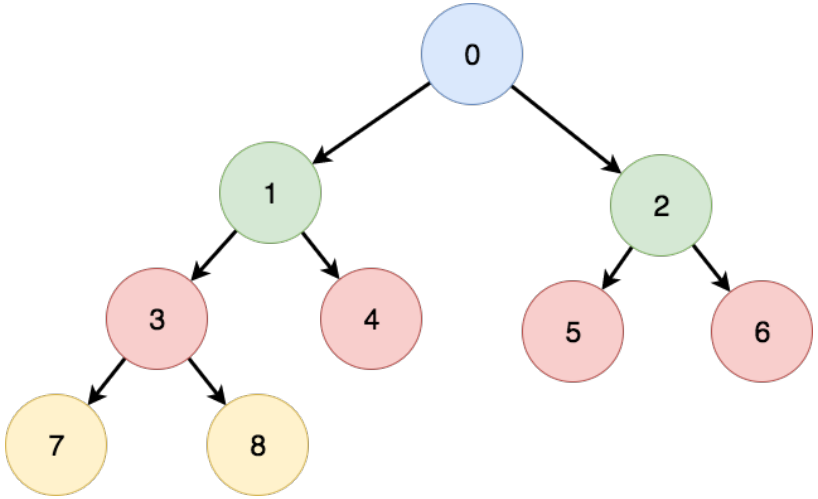
逻辑结构还是那个结构，

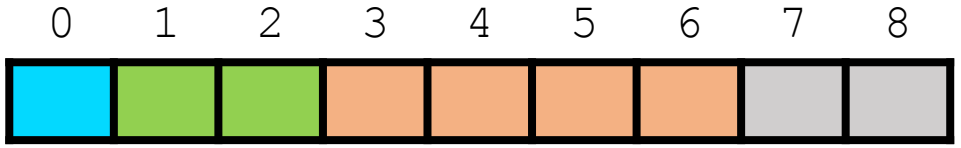
只是换了一种信息的表示方法

就像从阿拉伯数字

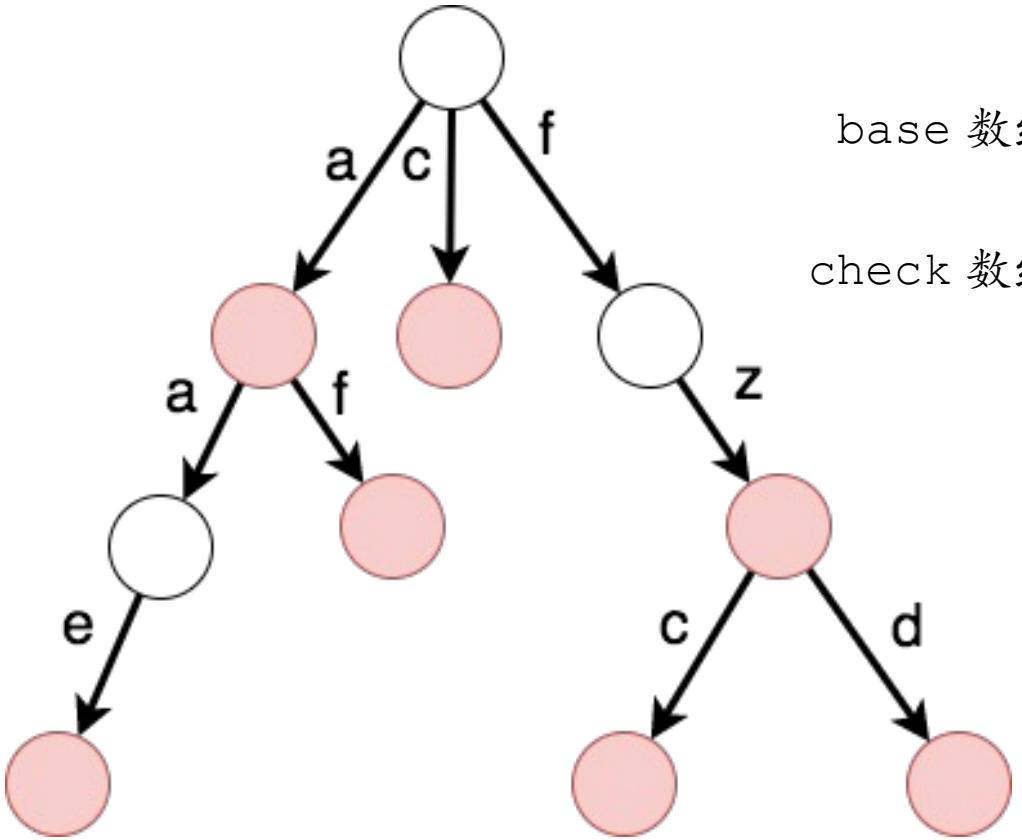
变成了罗马数字

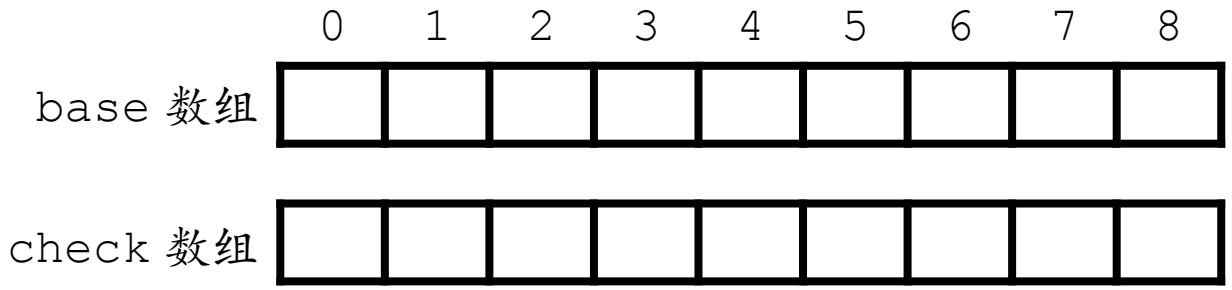
回想一下：完全二叉树





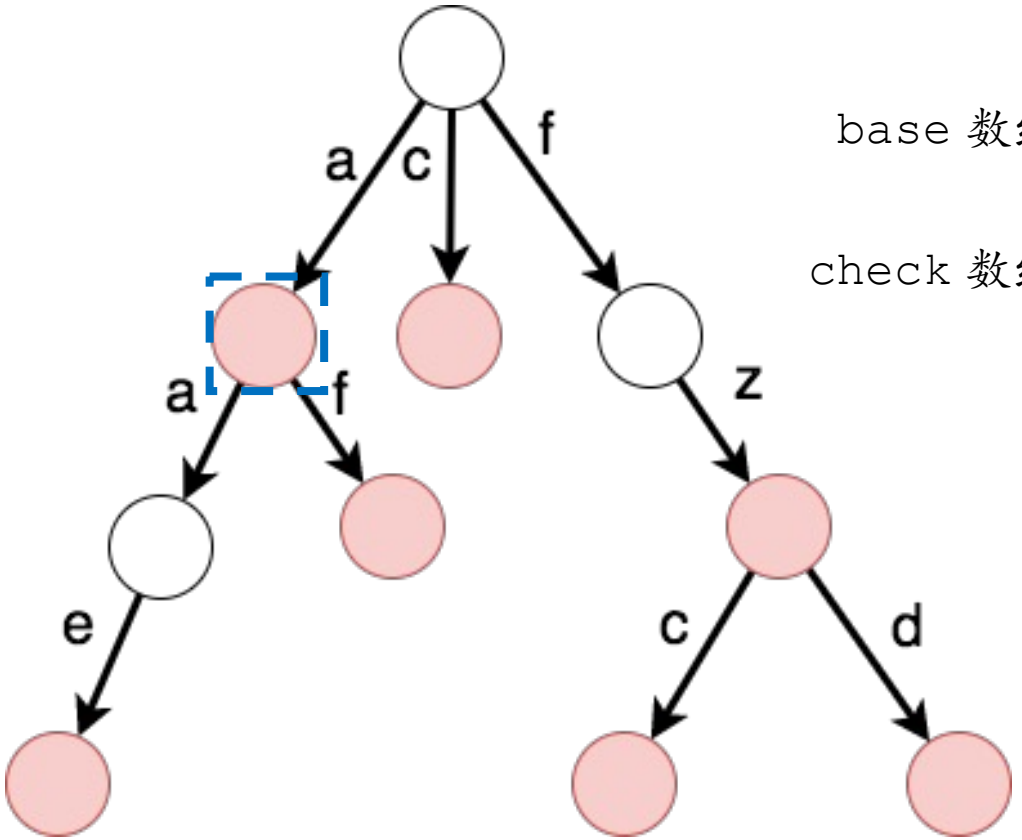
双数组字典树

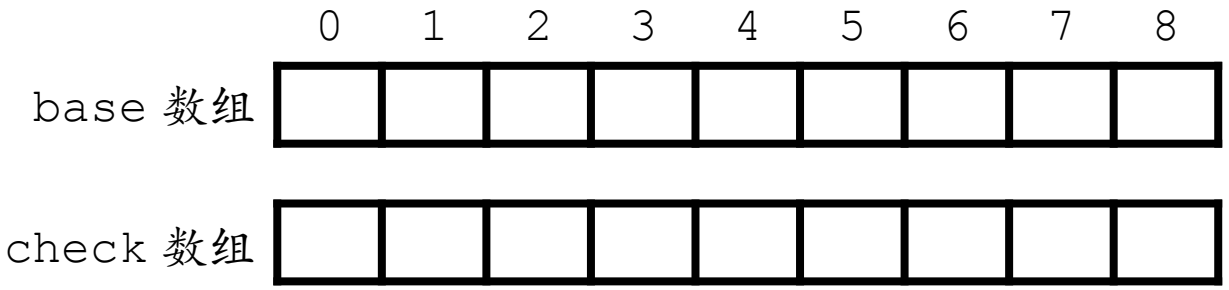





```
root_index = 1;
base[1] = 1;

child_i = base[father] + i;

check[child_i] = father;
check[child_i] = -father;
```

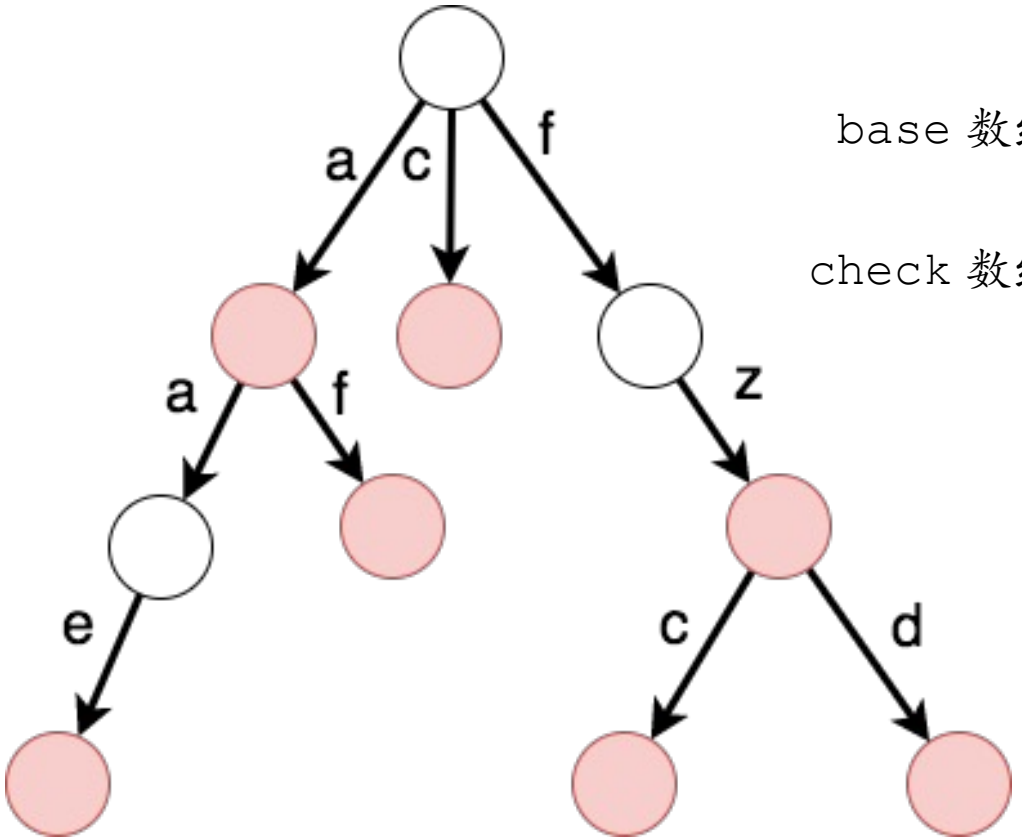
双数组字典树

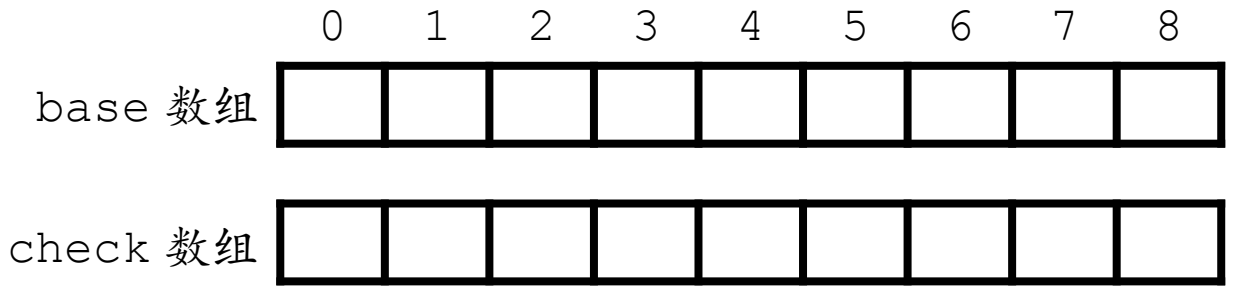





```
child_i = base[father] + i;

check[child_i] = father;
check[child_i] = -father;

base[child_i] = ?
```

双数组字典树

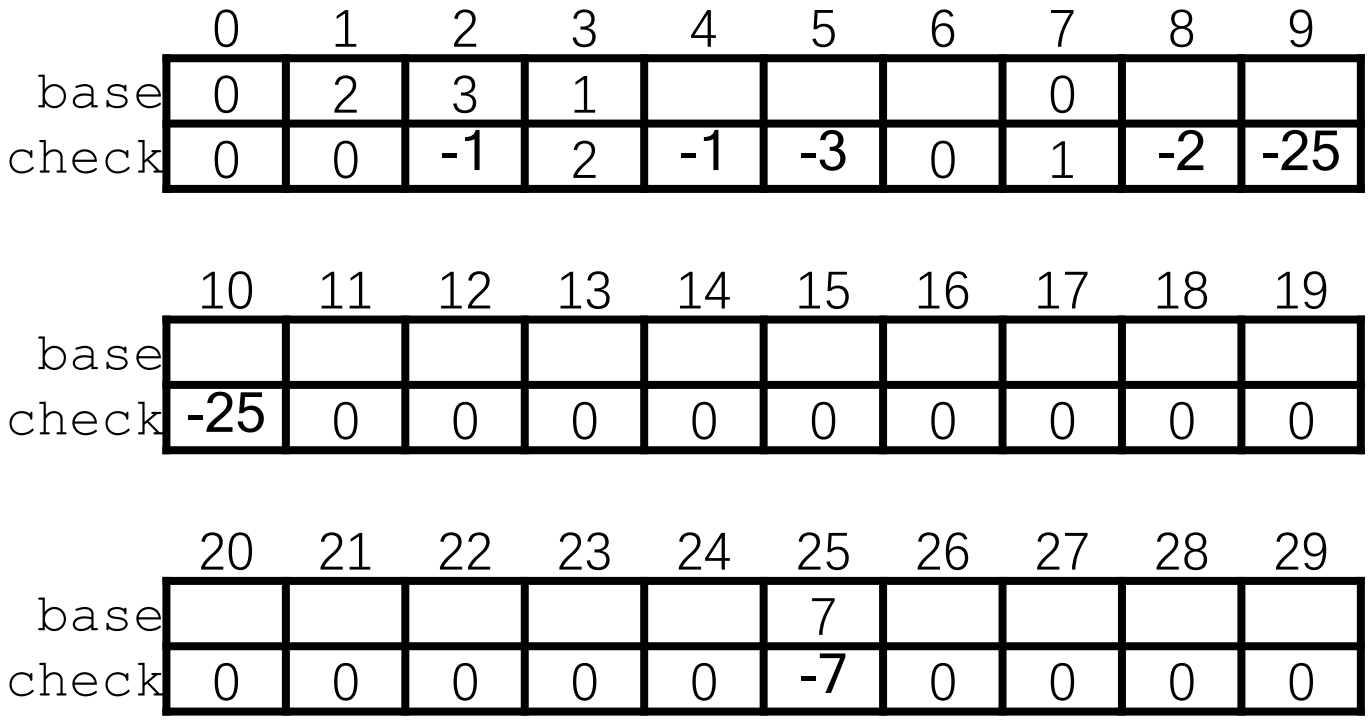




练习题：

请计算一组合理的base与check数组的值，使其对应左图 Trie 树。

双数组字典树


0 1 2 3 4 5 6 7 8 9
base 0 2 3 1 0
check 0 0 -1 2 -1 -3 0 1 -2 -25
10 11 12 13 14 15 16 17 18 19
base
check -25 0 0 0 0 0 0 0 0 0
20 21 22 23 24 25 26 27 28 29
base 7
check 0 0 0 0 0 -7 0 0 0 0

可持久化字典树

可持久化字典树

字典树：在 1~n 的单词中，是否存在单词 x

可持久化字典树：在 i~j 的单词中，是否存在单词 x

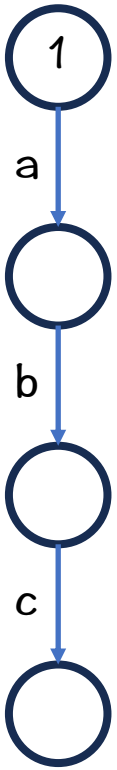
可持久化字典树

初试状态

可持久化字典树

初试状态

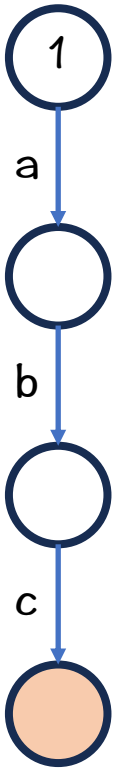
插入『abc』



可持久化字典树

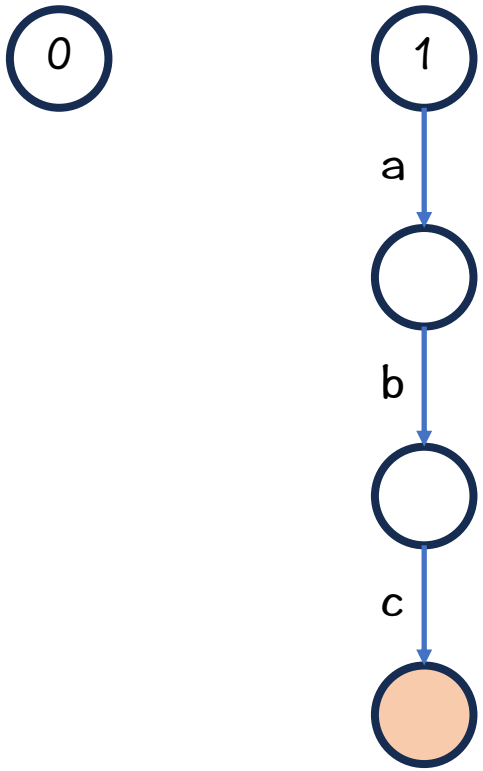
初试状态

插入『abc』

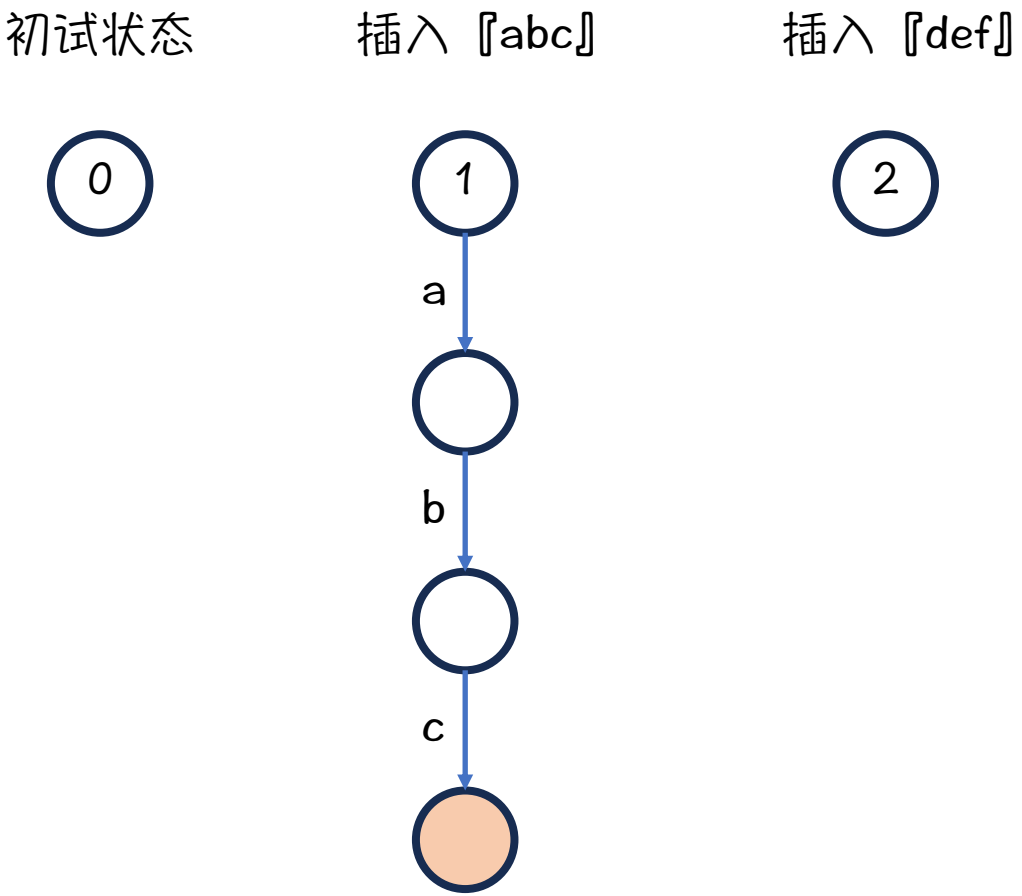


可持久化字典树

初试状态　　插入『abc』　　插入『def』

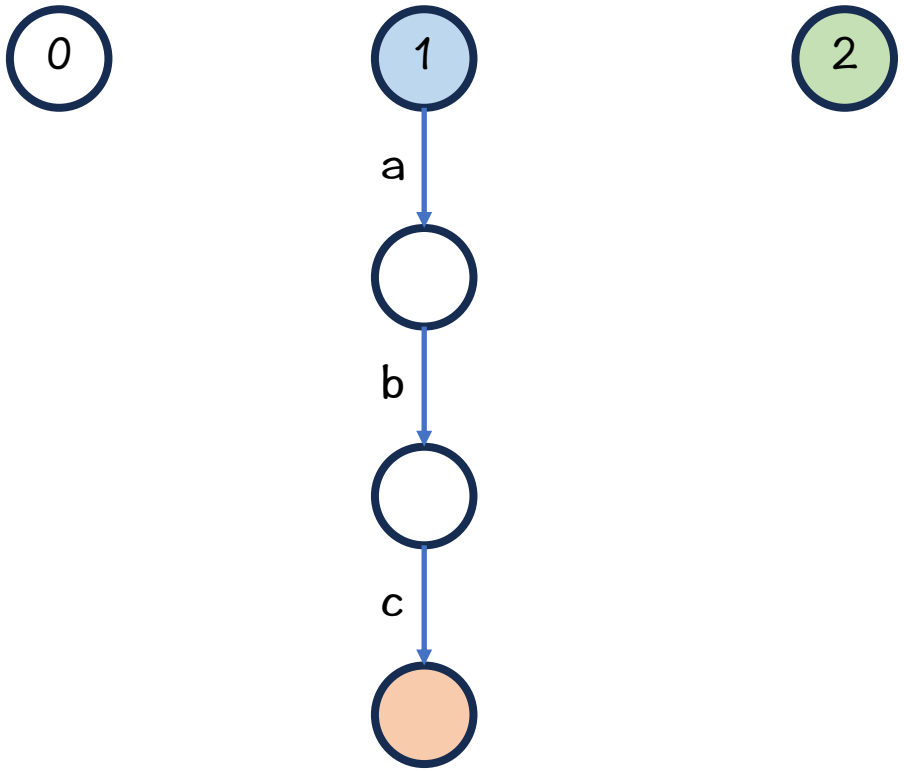


可持久化字典树



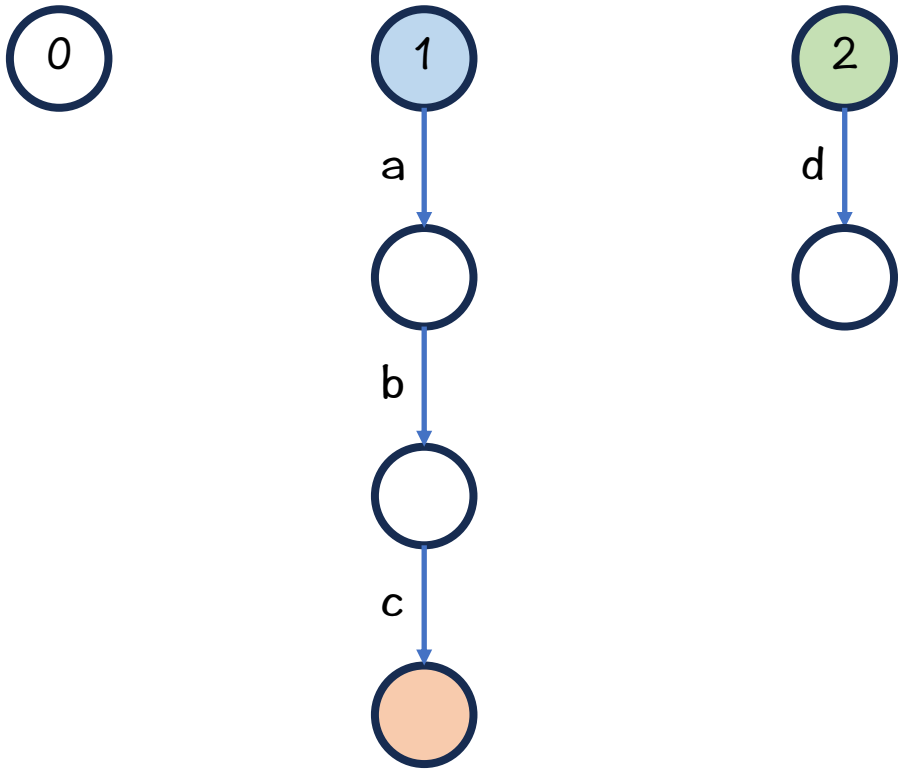
可持久化字典树

初试状态　　插入『abc』　　插入『def』

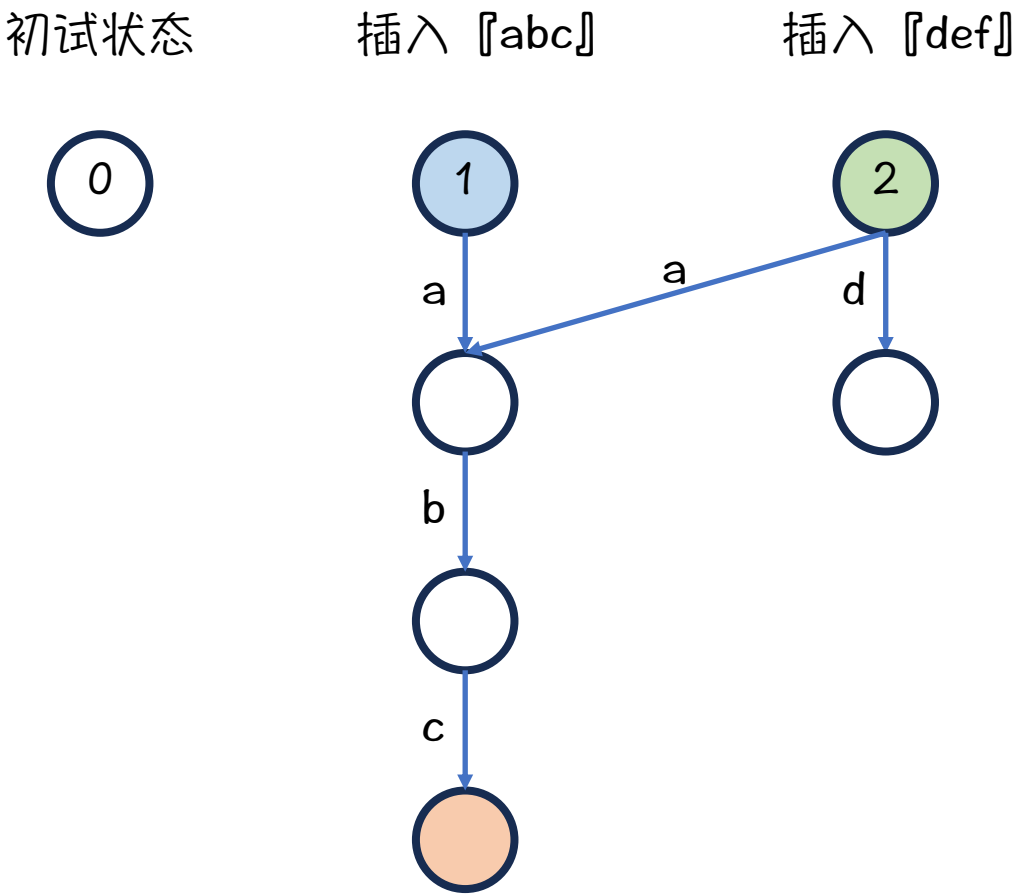


可持久化字典树

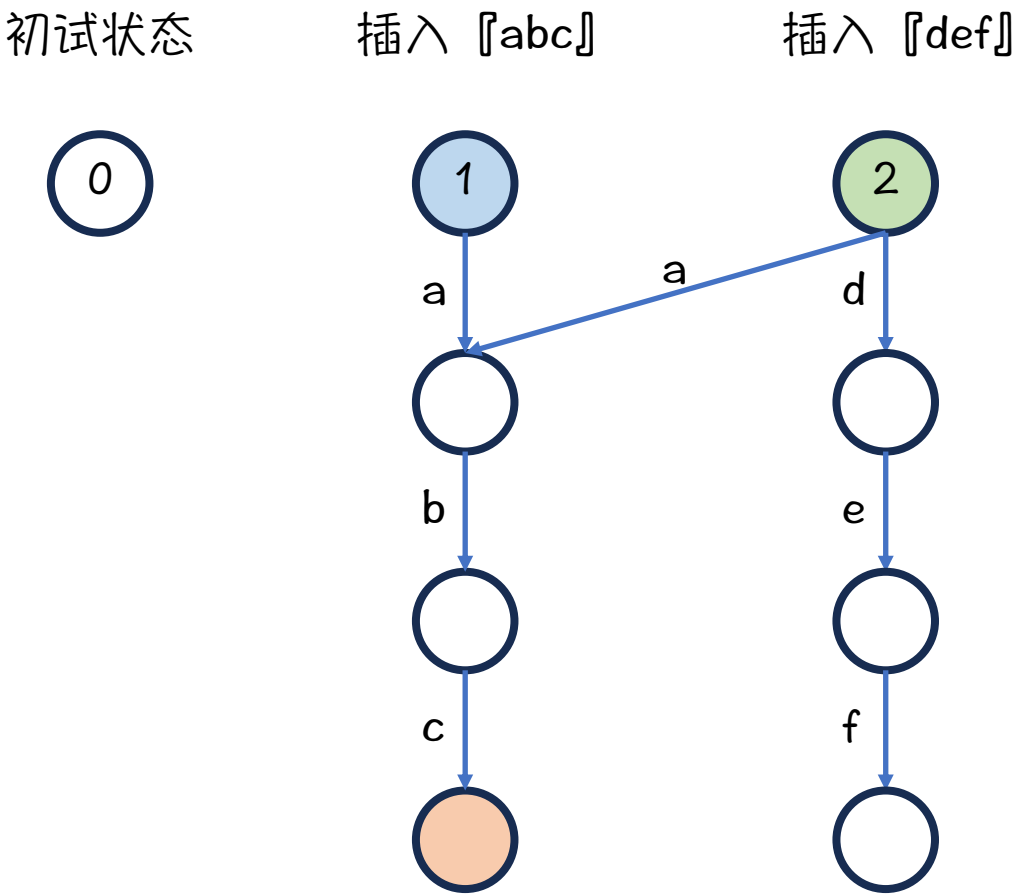
初试状态　　插入『abc』　　插入『def』



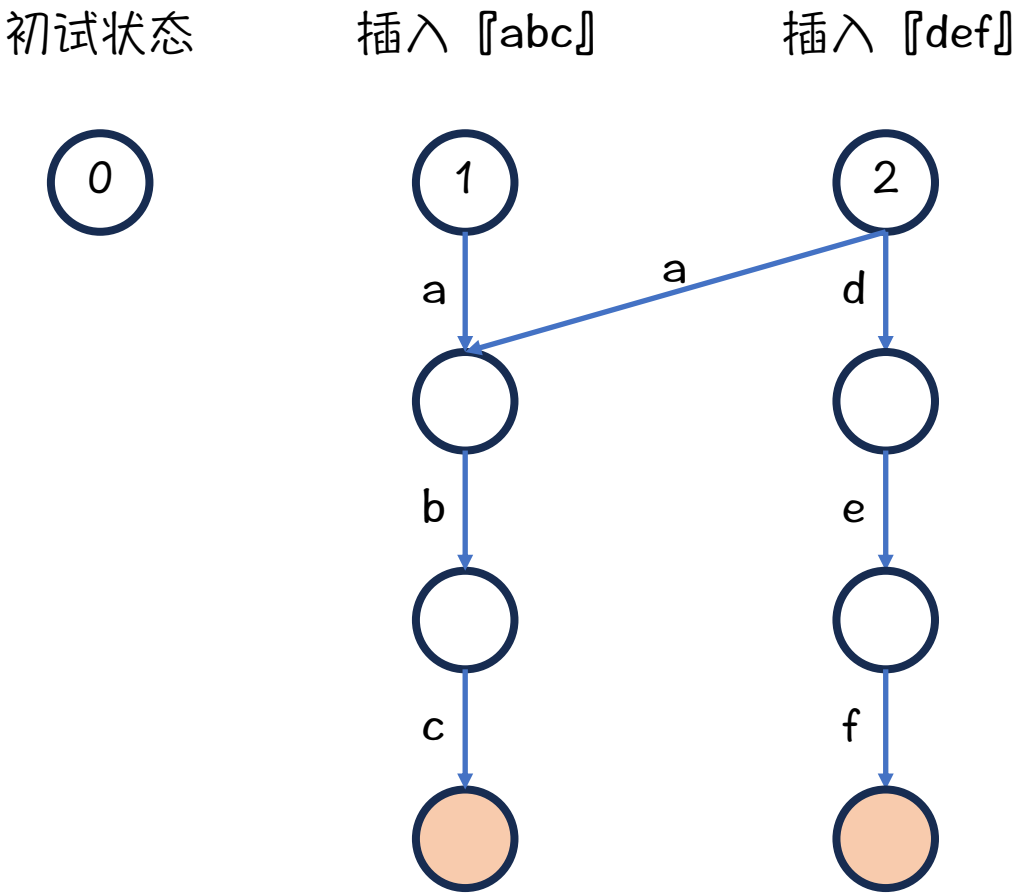
可持久化字典树



可持久化字典树

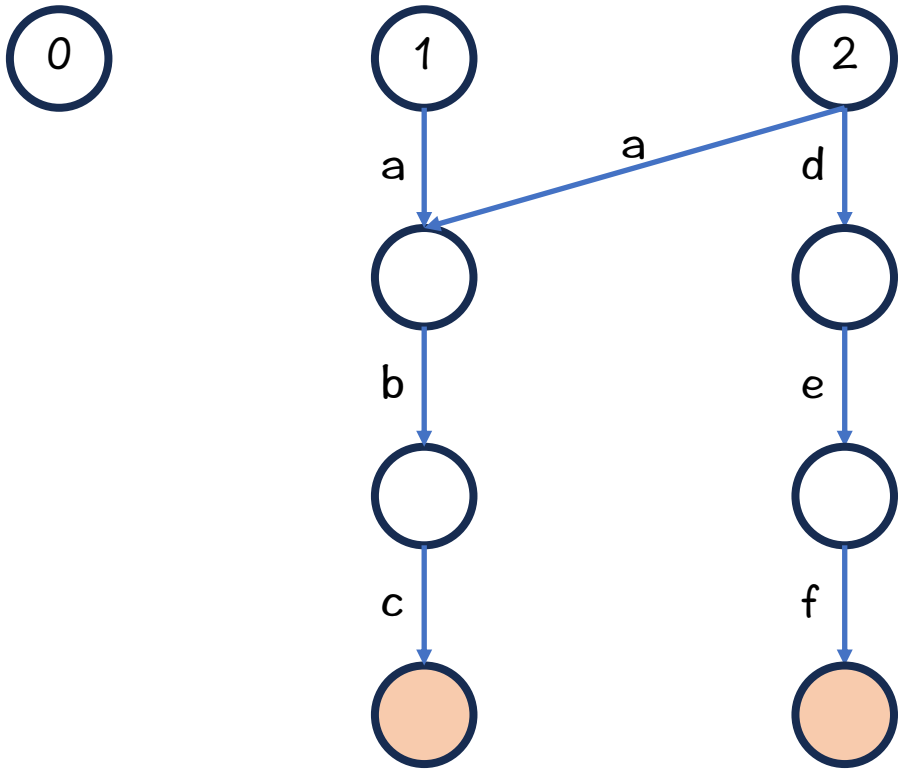


可持久化字典树



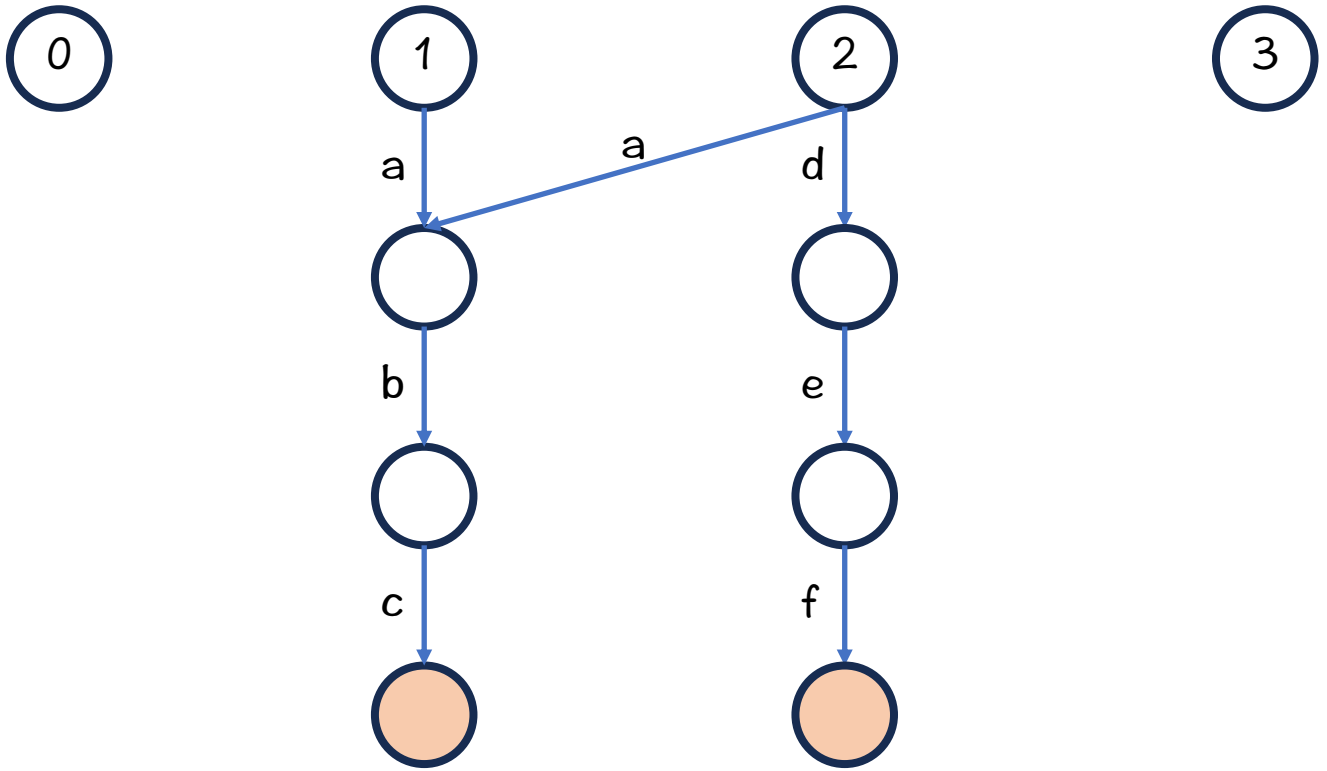
可持久化字典树

初试状态　　插入『abc』　　插入『def』　　插入『adb』



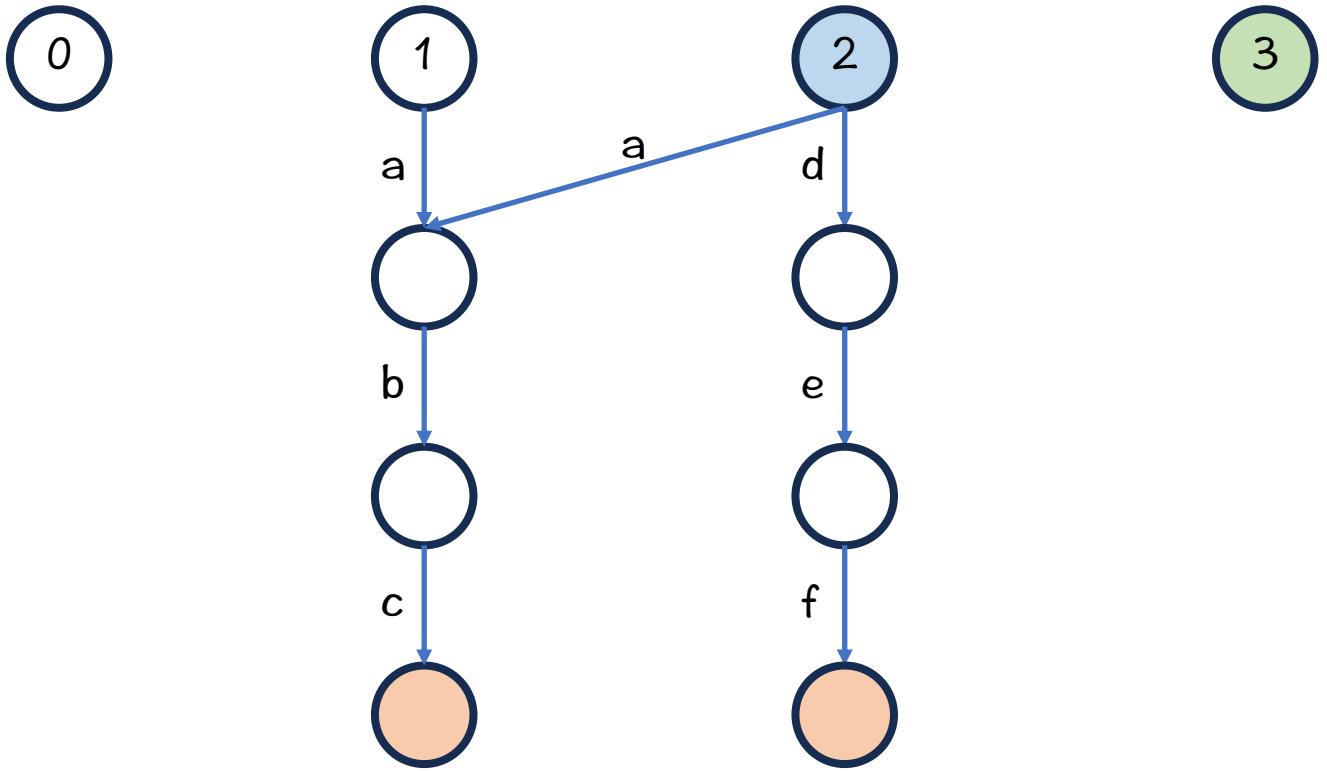
可持久化字典树

初试状态　　插入『abc』　　插入『def』　　插入『adb』

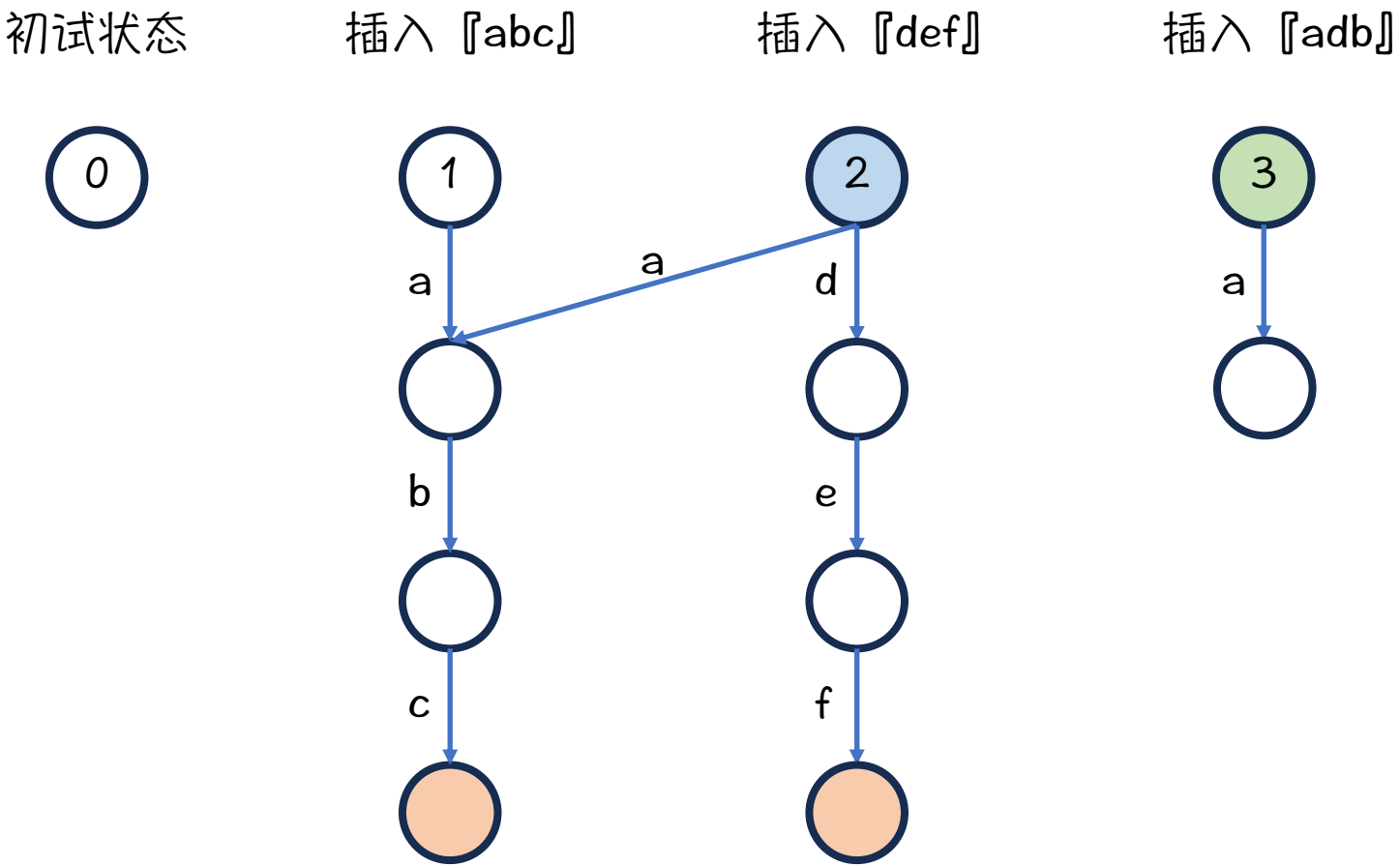


可持久化字典树

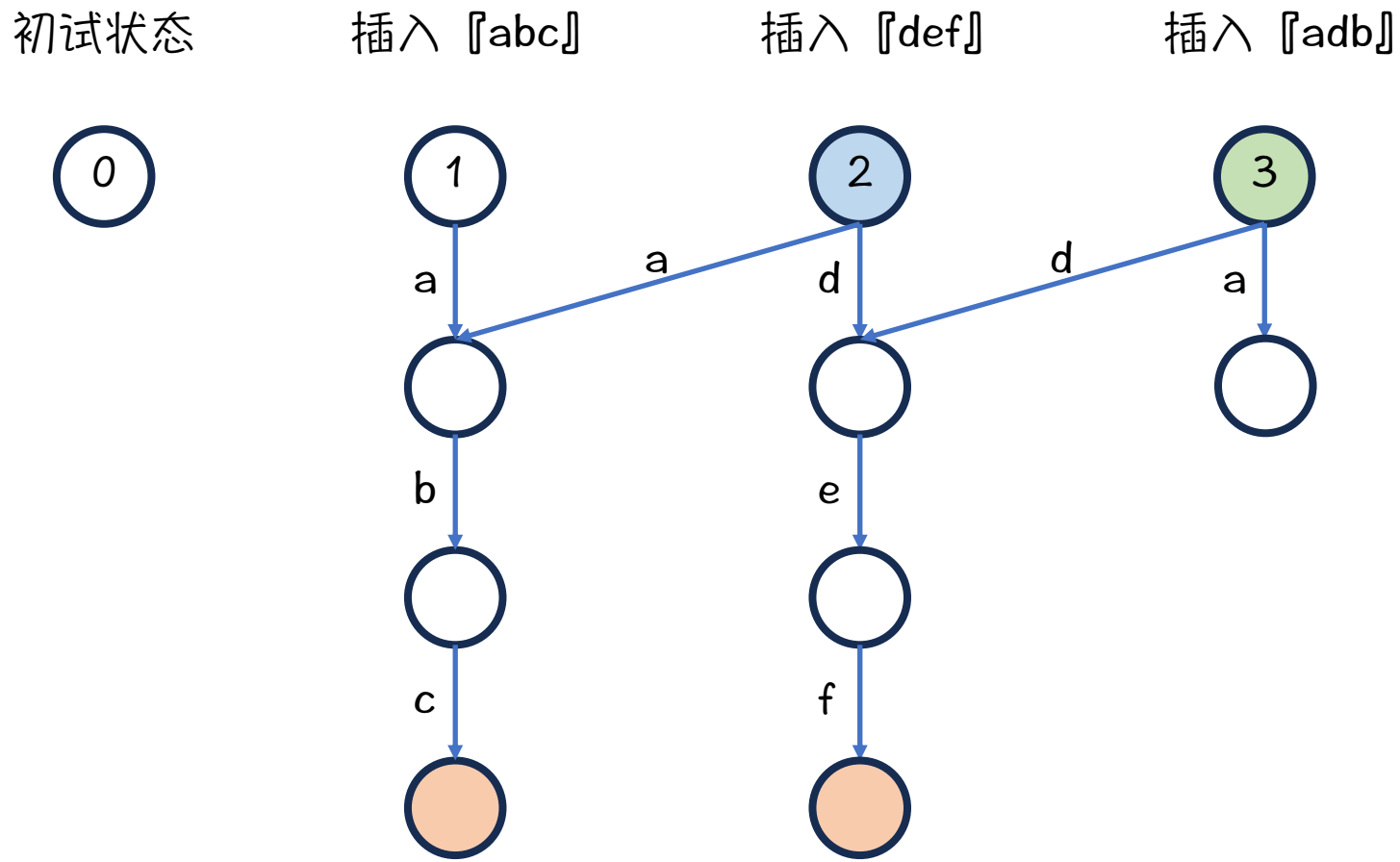
初试状态　　插入『abc』　　插入『def』　　插入『adb』



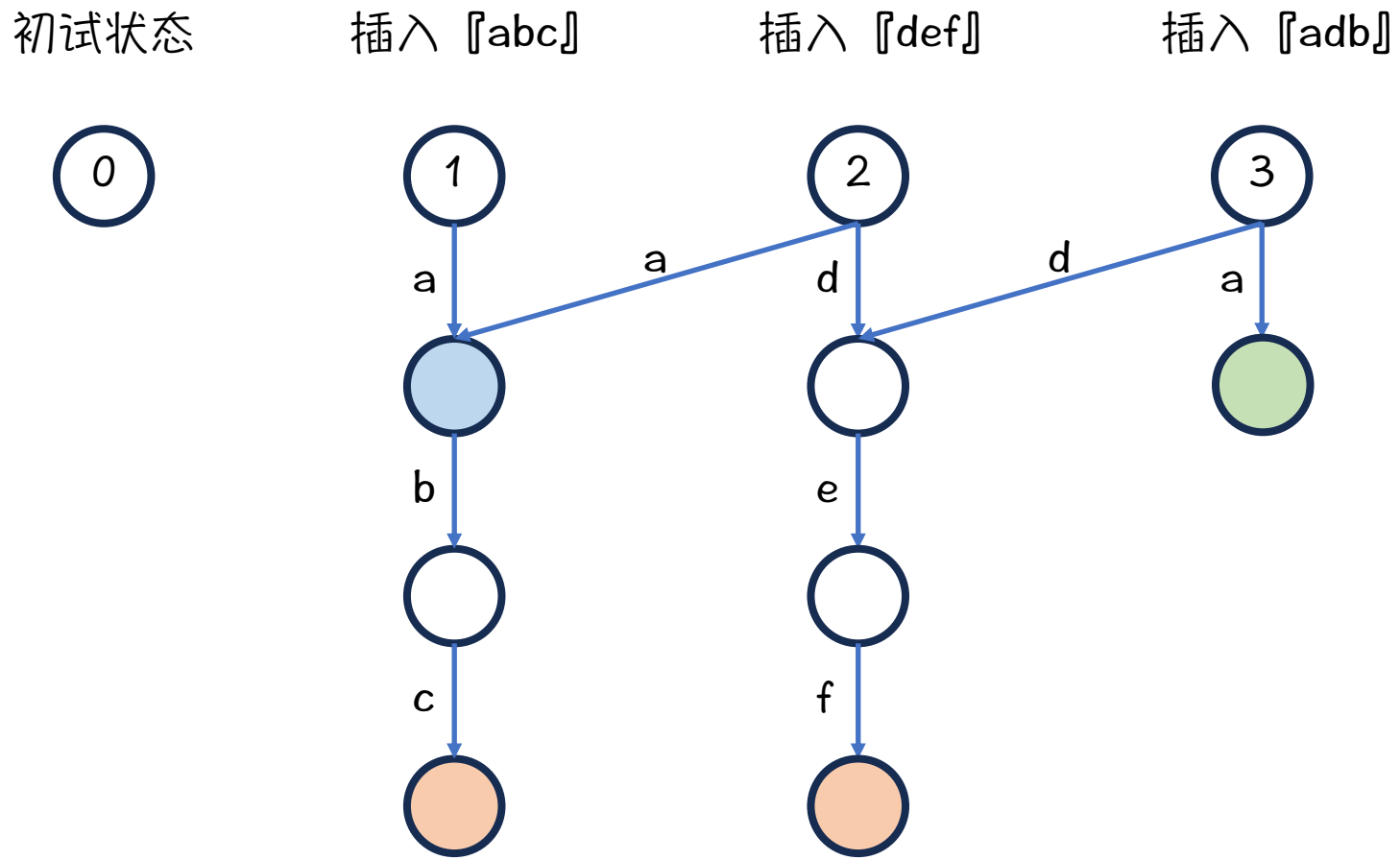
可持久化字典树



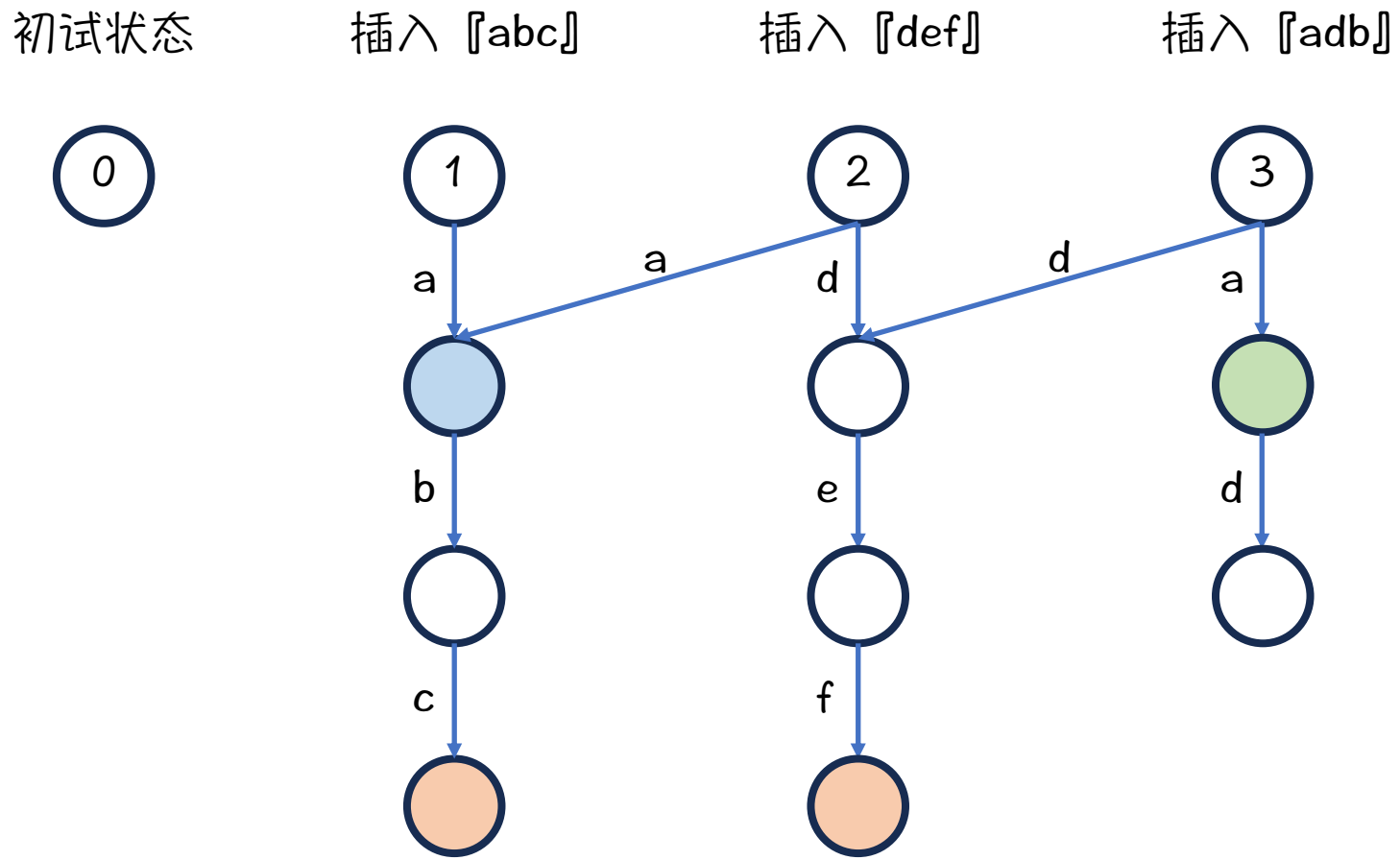
可持久化字典树



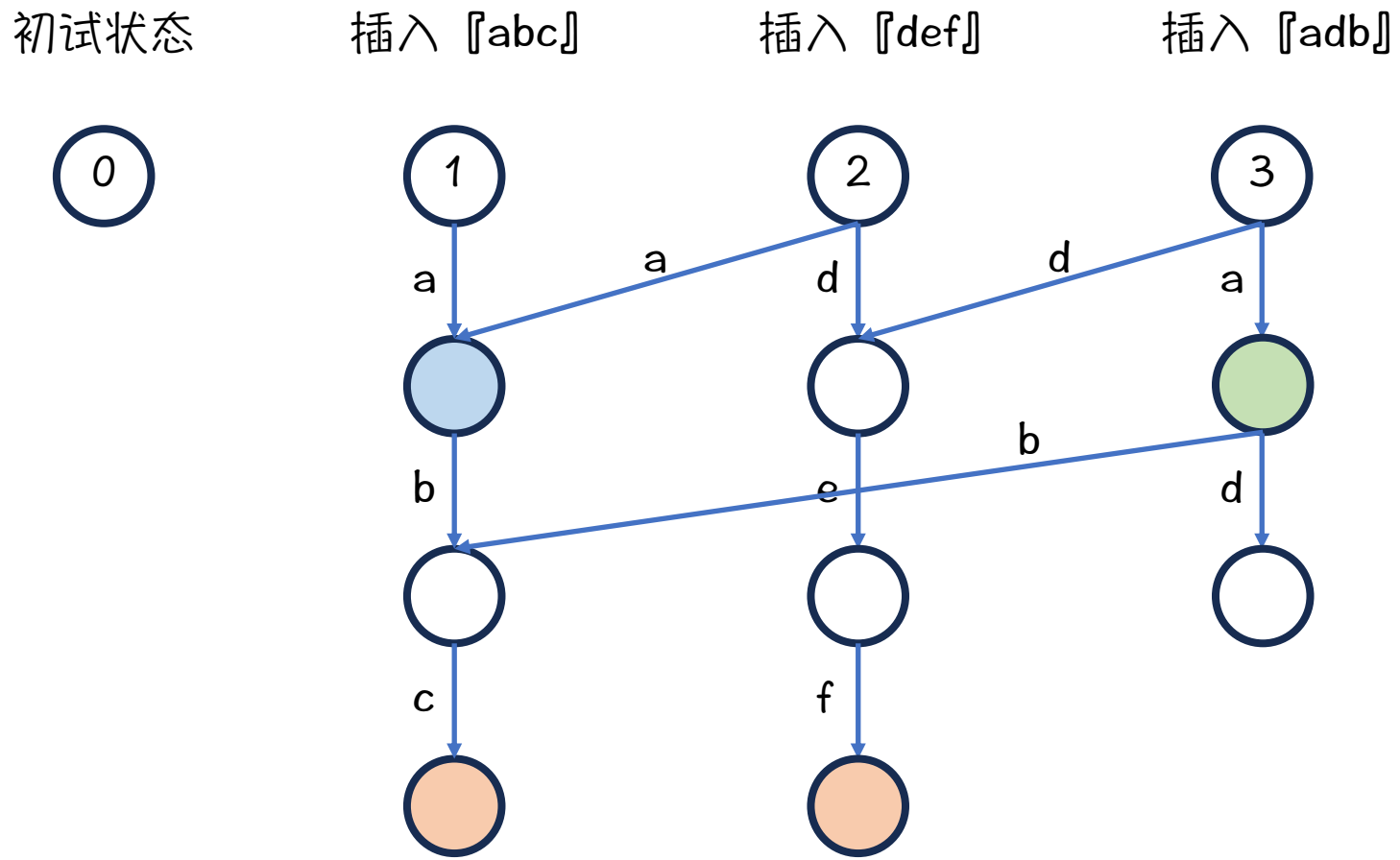
可持久化字典树



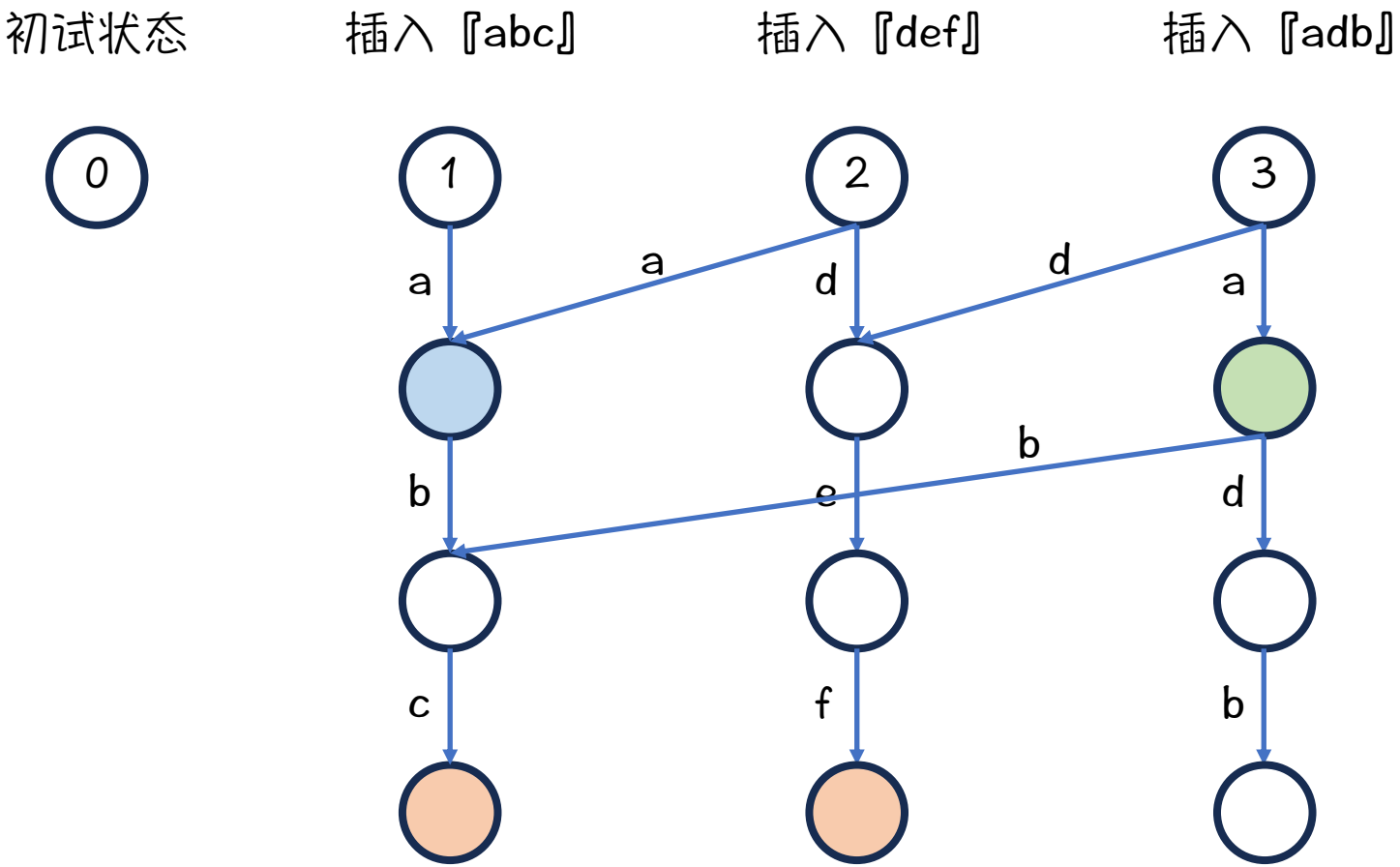
可持久化字典树



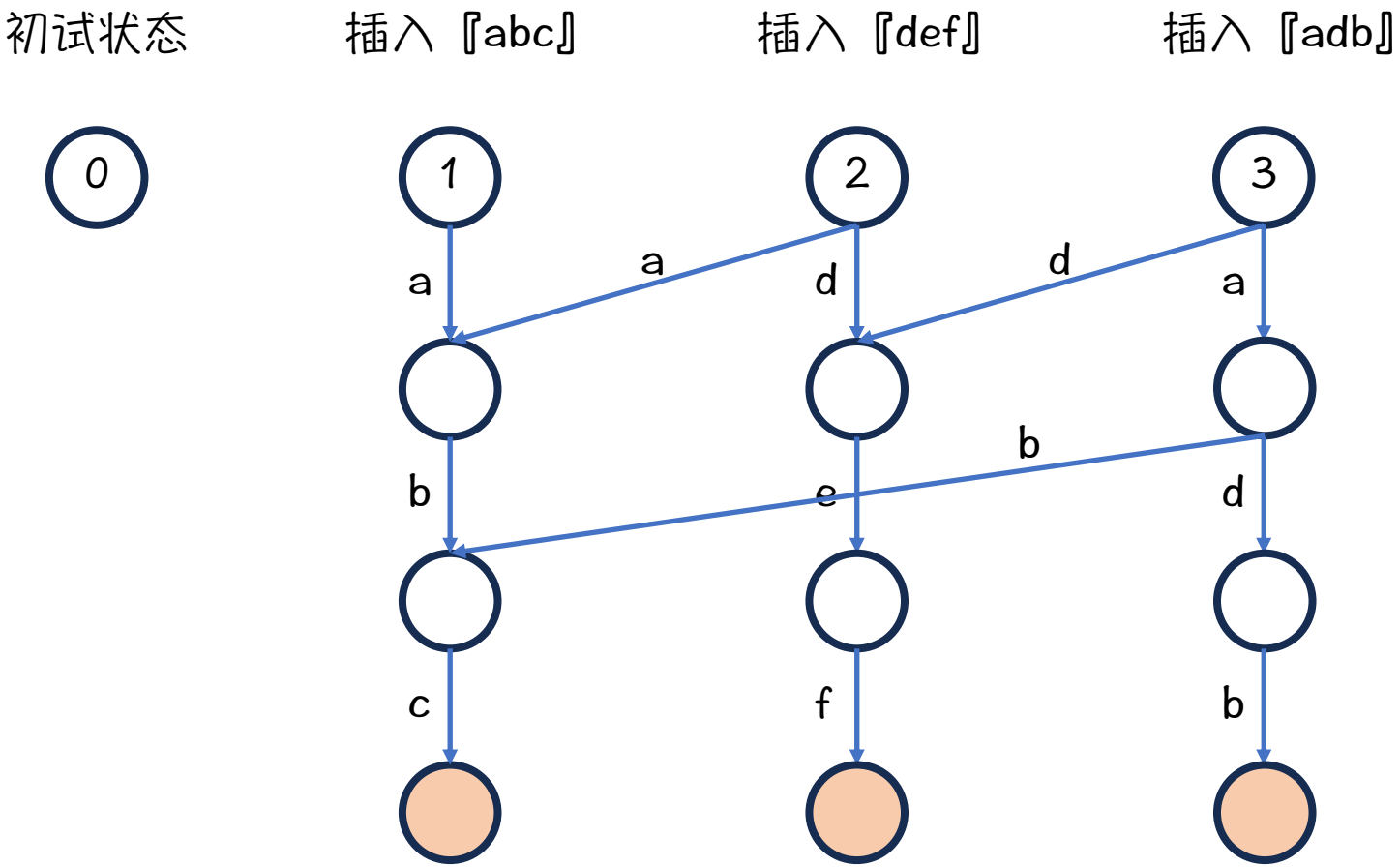
可持久化字典树



可持久化字典树

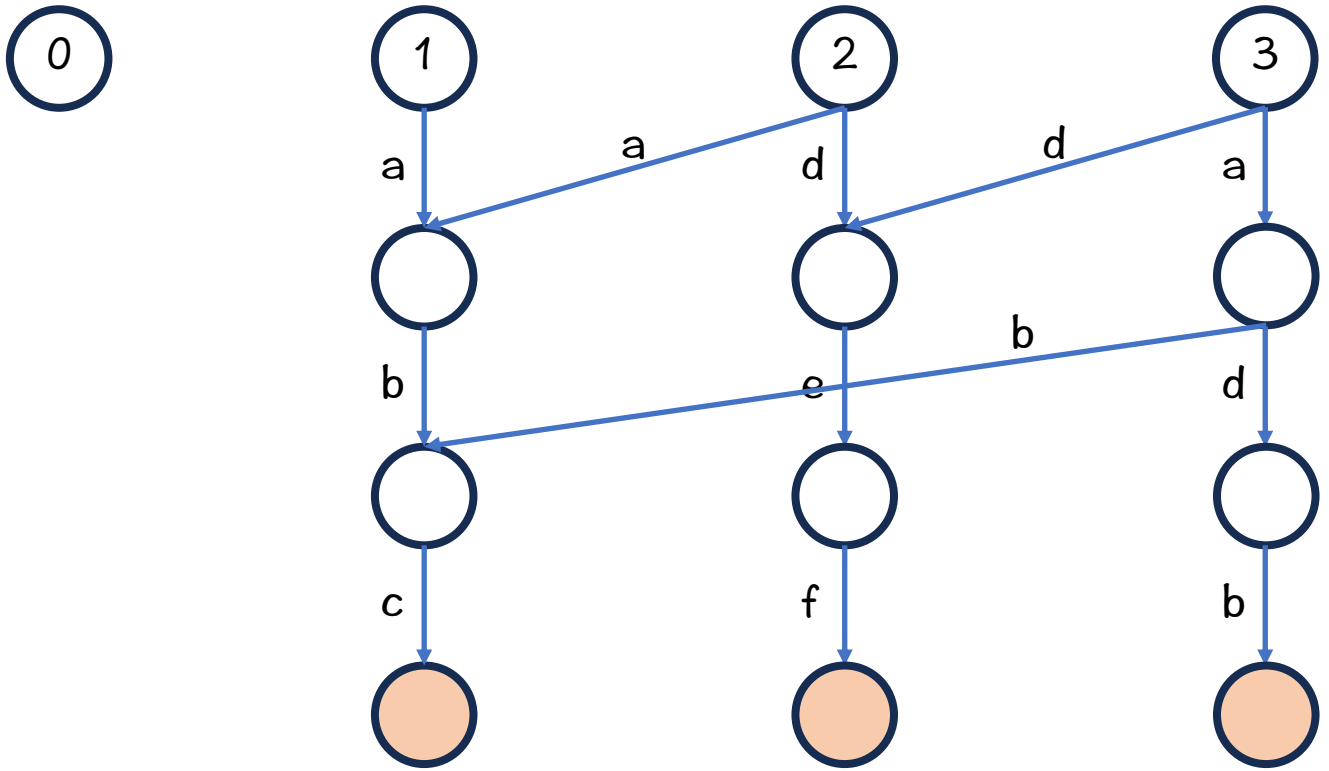


可持久化字典树

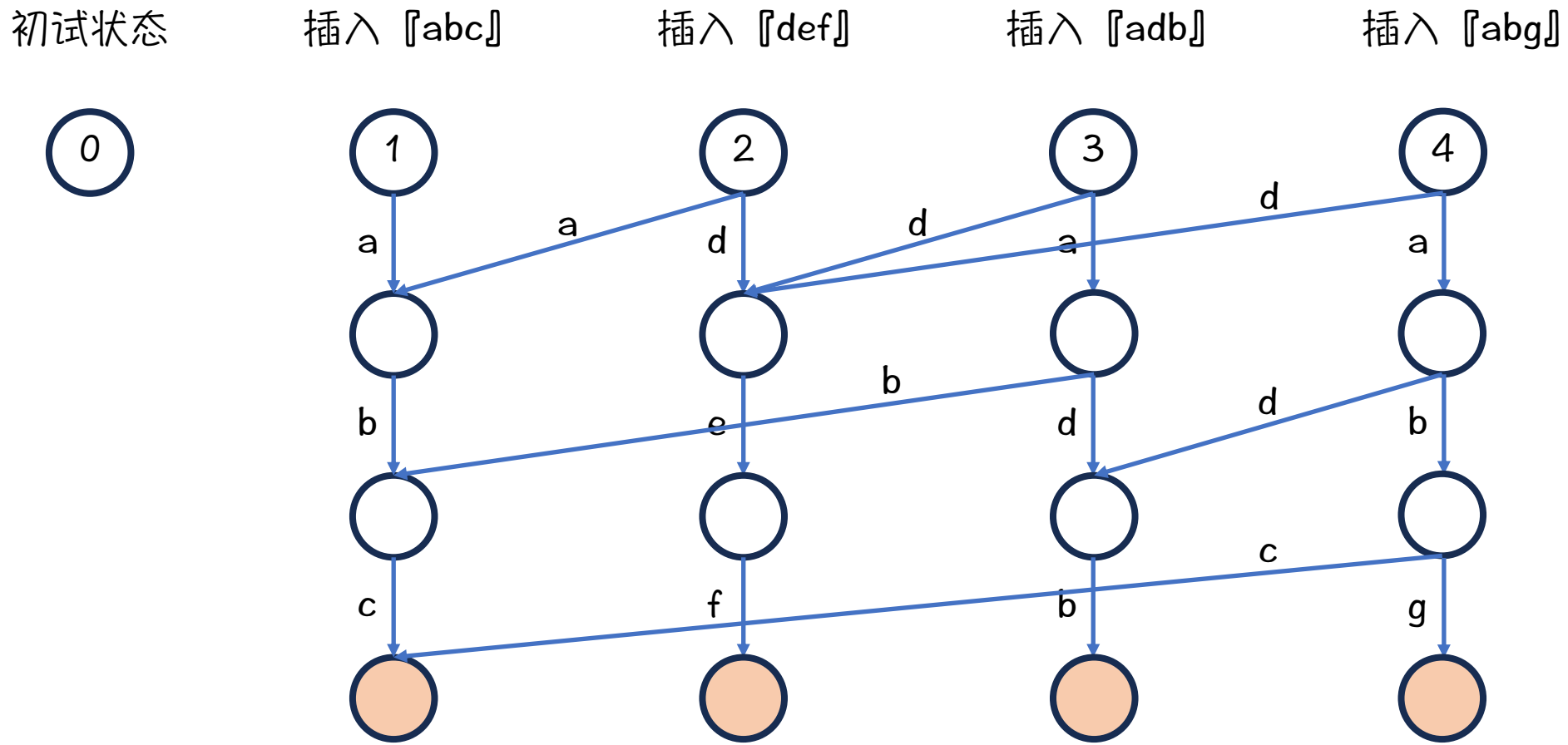


可持久化字典树

初试状态　　插入『abc』　　插入『def』　　插入『adb』　　插入『abg』



可持久化字典树



二、多模匹配问题

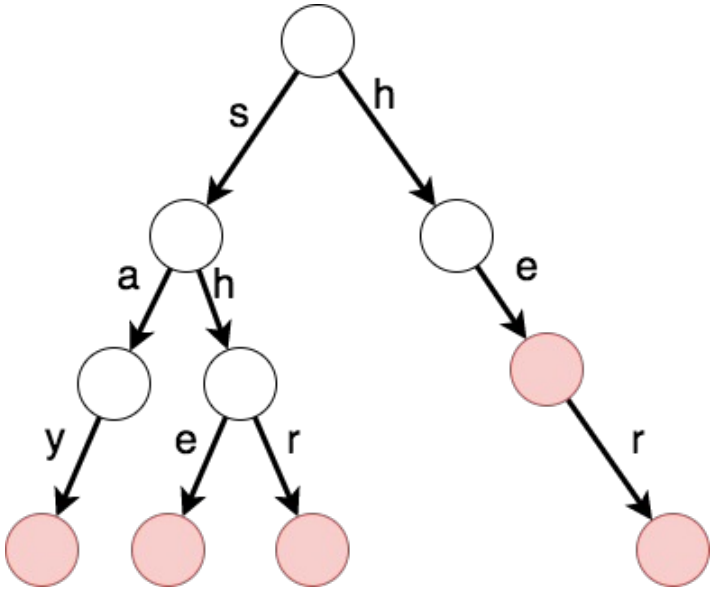
1. 基于哈希：Rabin-Karp 算法
2. 初探 NFA：Shift-and/or 算法
3. **神兵利器：Trie 字典树**
4. 飞升蜕变：AC 自动机

二、多模匹配问题

1. 基于哈希：Rabin-Karp 算法
2. 初探 NFA：Shift-and/or 算法
3. 神兵利器：Trie 字典树
4. **飞升蜕变：AC 自动机**

TRIE 树匹配

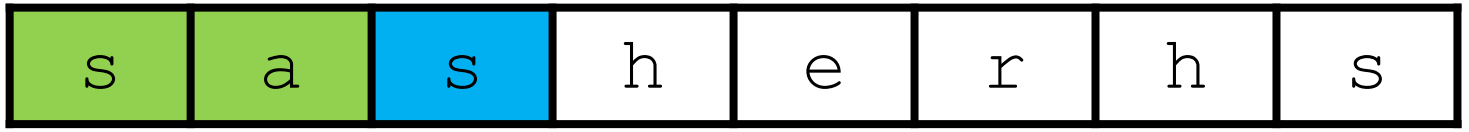
s	a	s	h	e	r	h	s



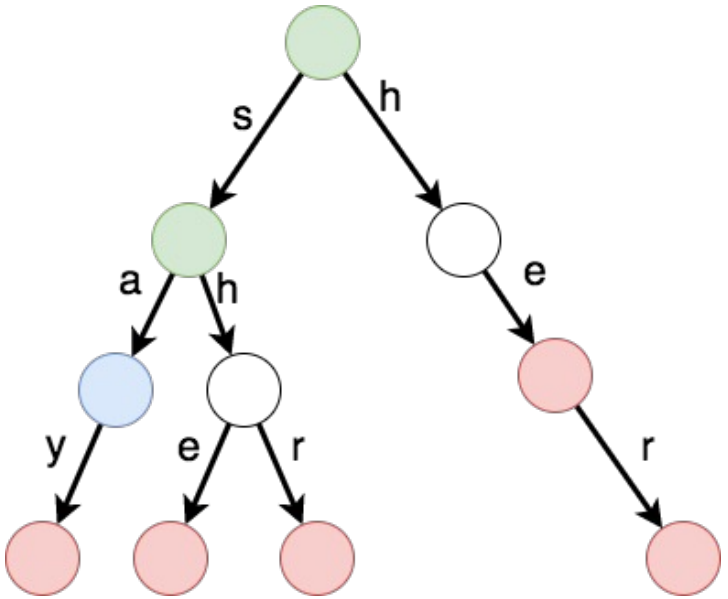
=

say
she
shr
he
her

TRIE 树匹配

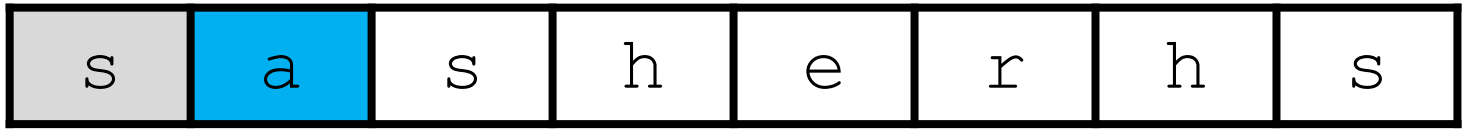


+3次

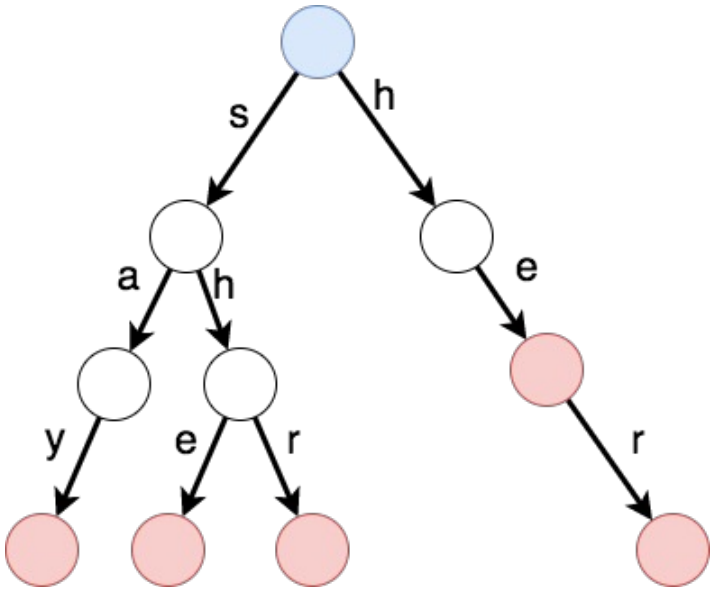


```
say
she
shr
he
her
```

TRIE 树匹配



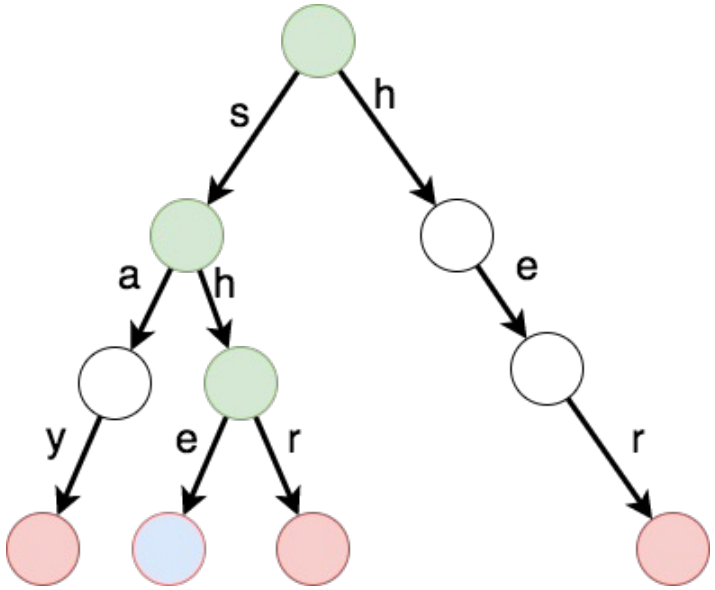
+1次



say
she
shr
he
her

TRIE 树匹配

+4次



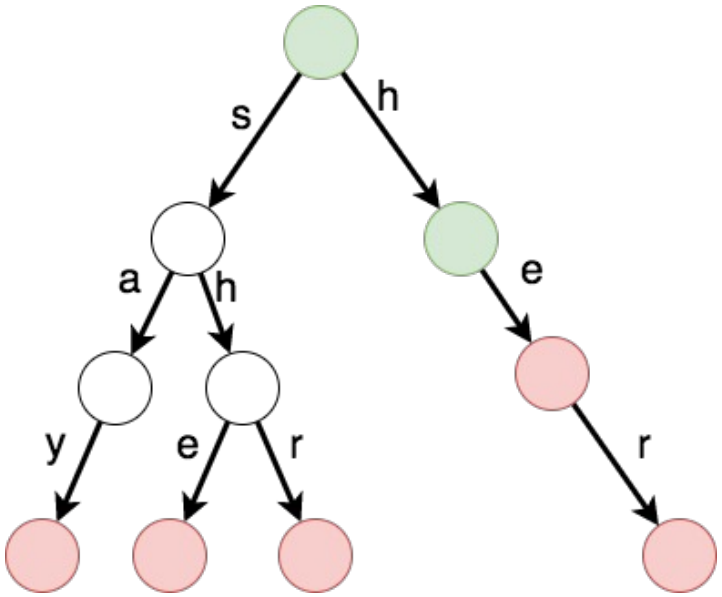
=

say
she
shr
he
her

TRIE 树匹配

s	a	s	h	e	r	h	s

+2次

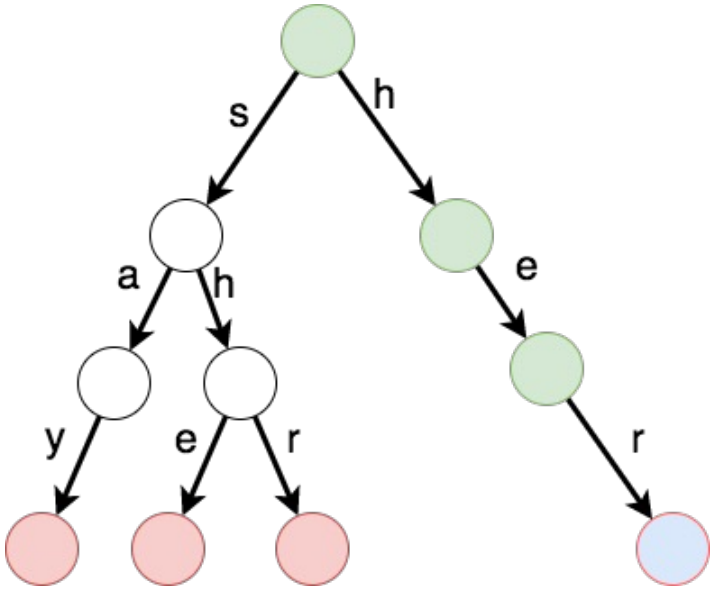


=

say
she
shr
he
her

TRIE 树匹配

+2次

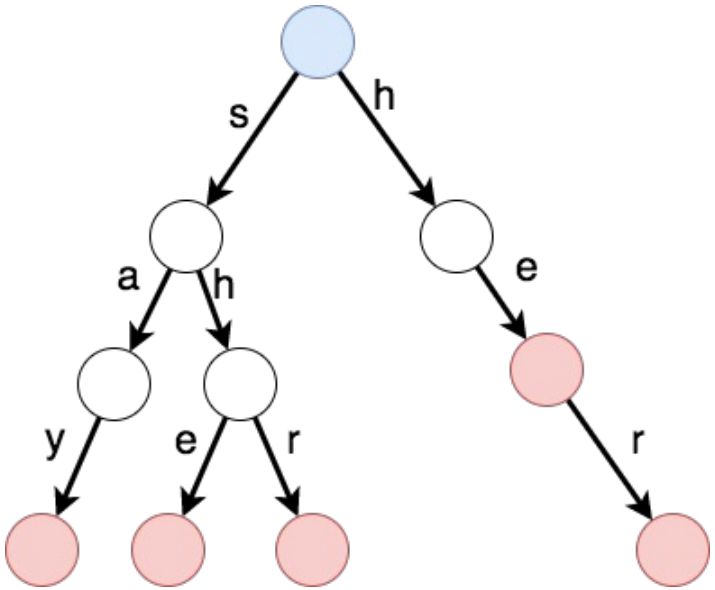


=

say
she
shr
he
her

TRIE 树匹配

+1次



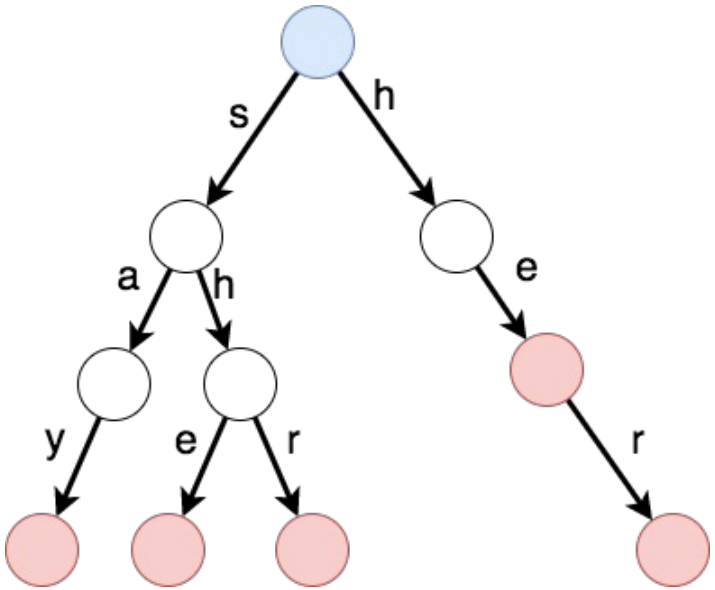
=

say
she
shr
he
her

TRIE 树匹配

s	a	s	h	e	r	h	s

+1次



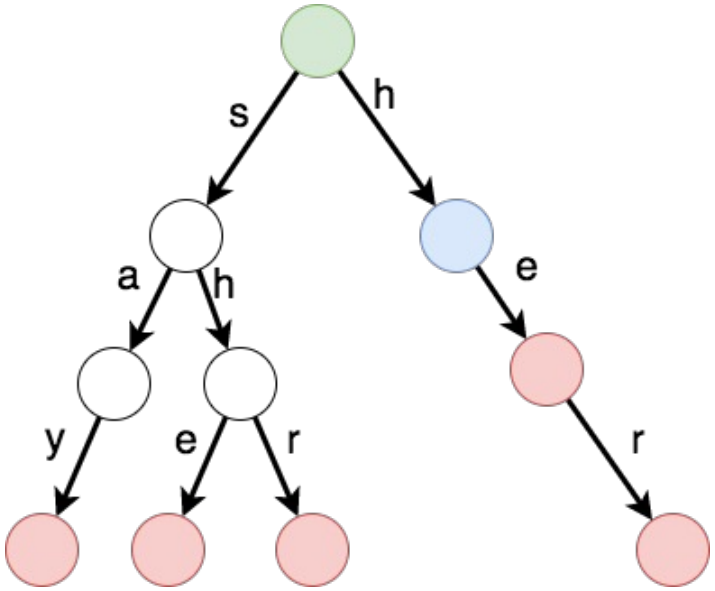
=

say
she
shr
he
her

TRIE 树匹配

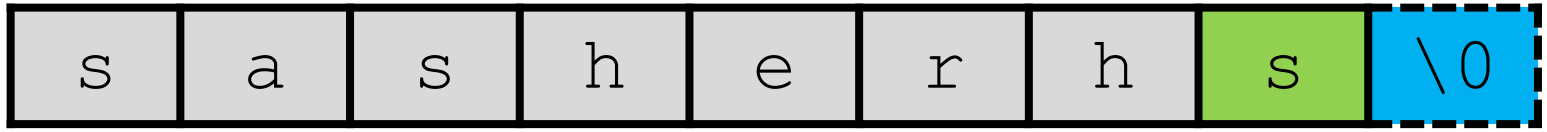
s	a	s	h	e	r	h	s

+2次

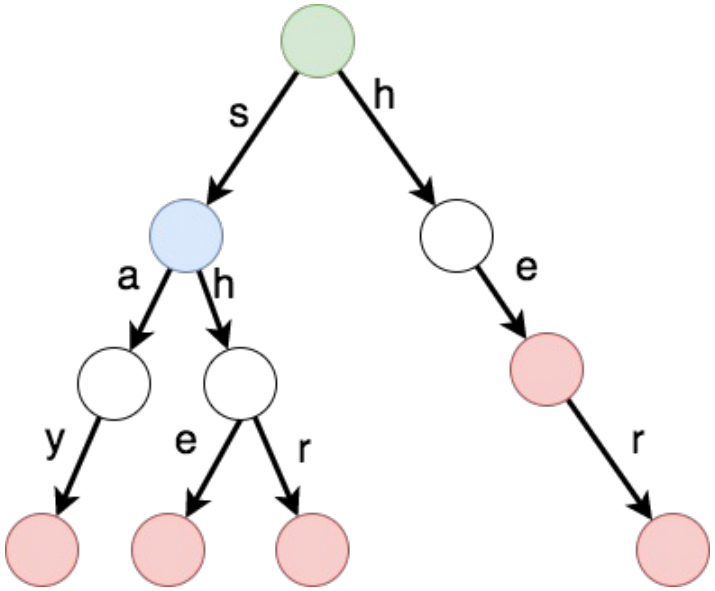


say
she
shr
he
her

TRIE 树匹配



+2次



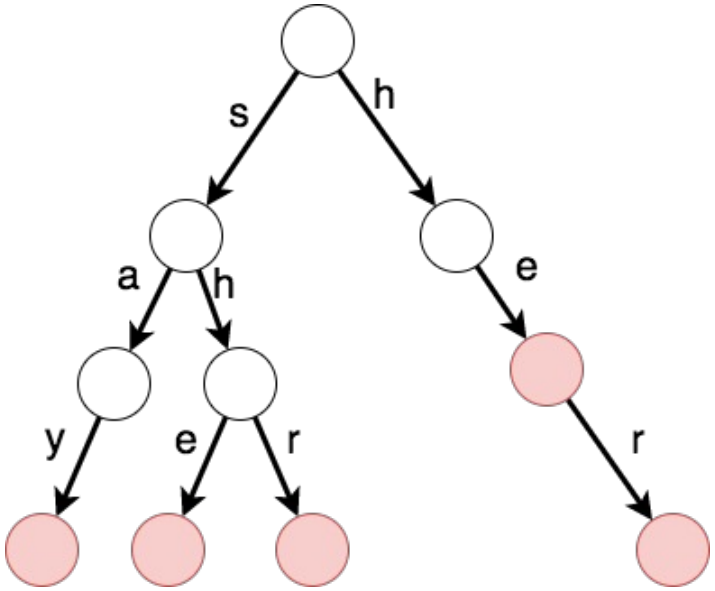
=

say
she
shr
he
her

TRIE 树匹配

s	a	s	h	e	r	h	s

18次

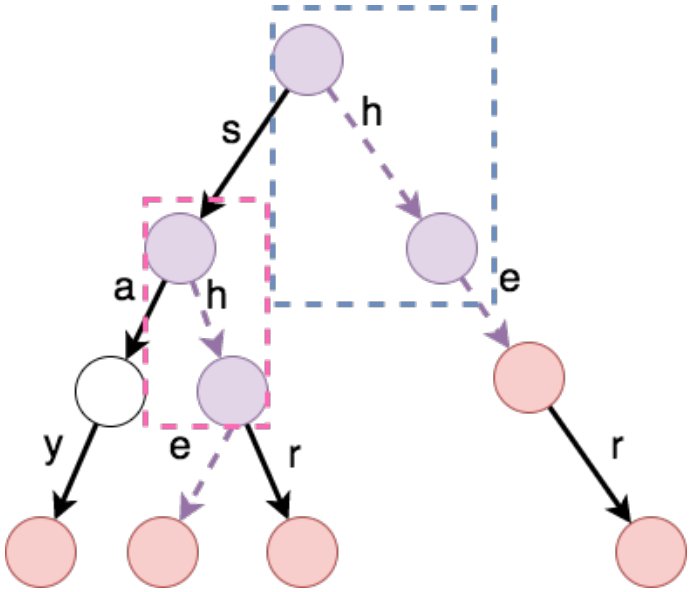


=

say
she
shr
he
her

TRIE 树匹配--问题

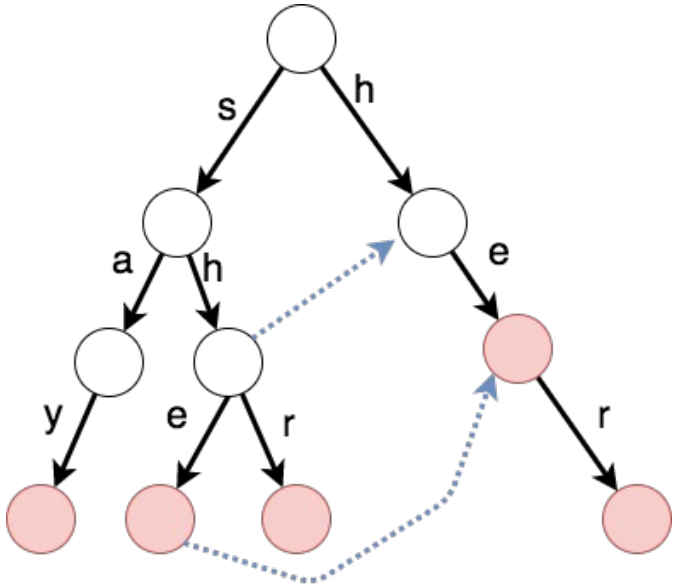
s	a	s	h	e	r	h	s



=

say
she
shr
he
her

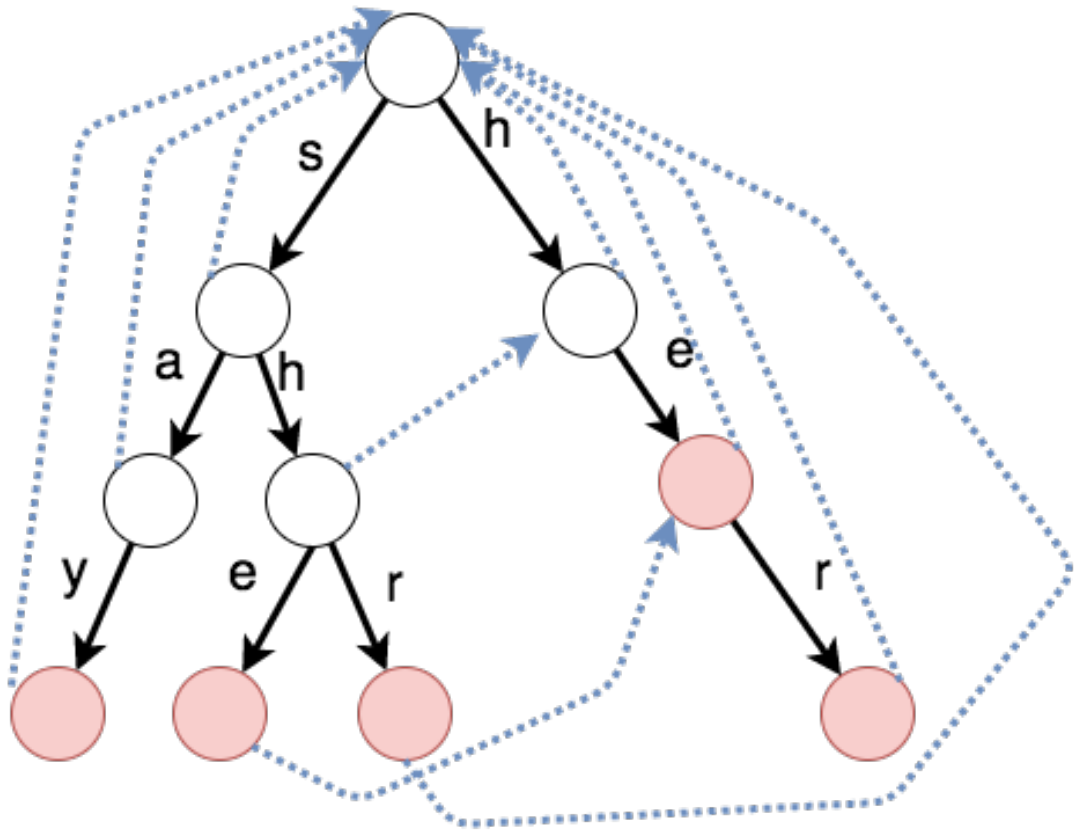
AC 自动机



=

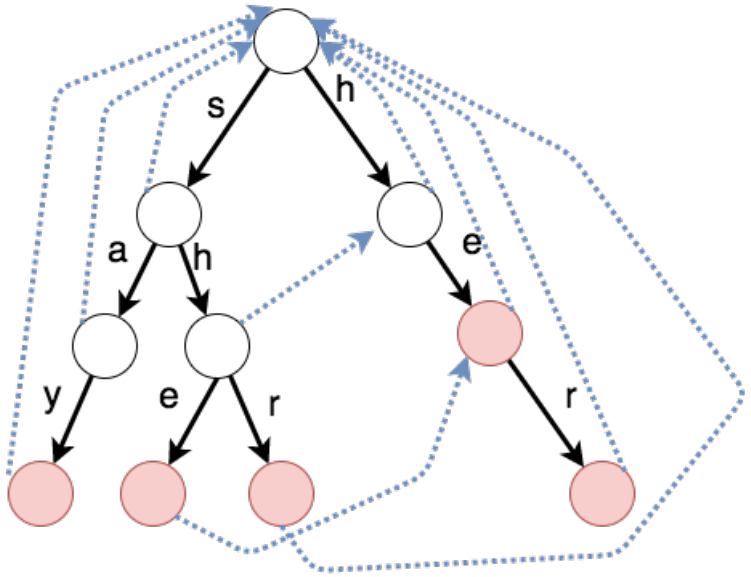
say
she
shr
he
her

AC 自动机



AC 自动机--匹配

s	a	s	h	e	r	h	s

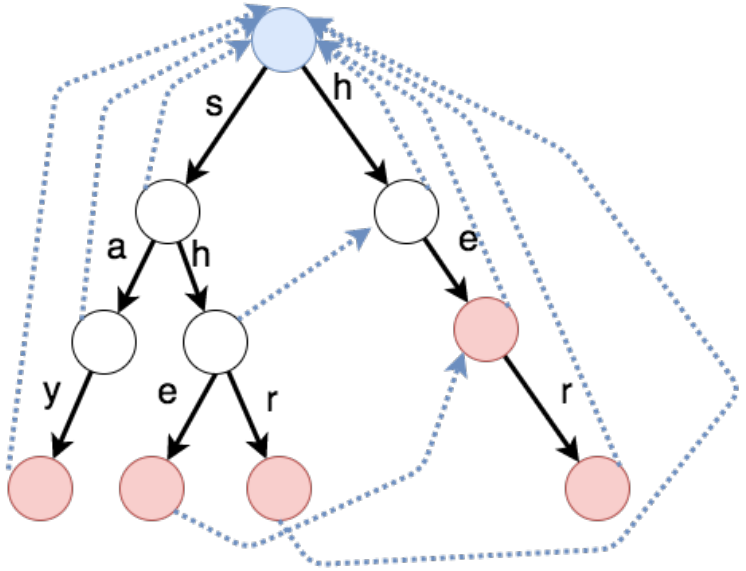


=

say
she
shr
he
her

AC 自动机--匹配

s	a	s	h	e	r	h	s



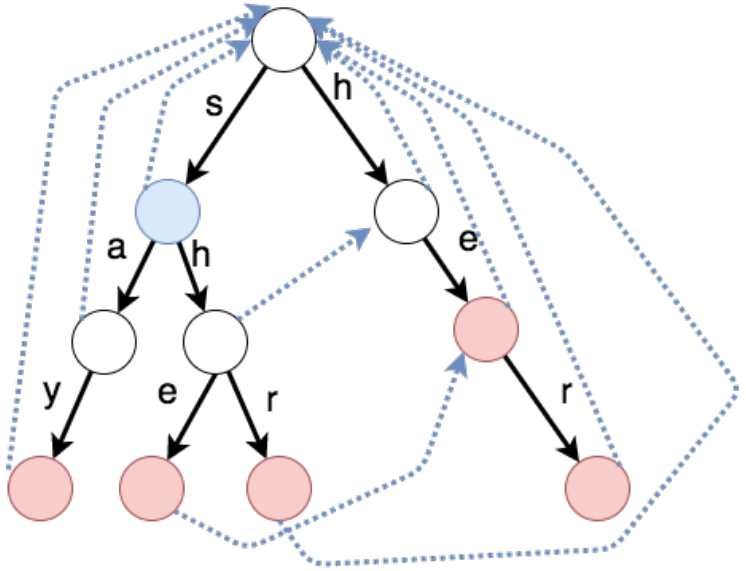
=

say
she
shr
he
her

AC 自动机--匹配

s	a	s	h	e	r	h	s

+1次



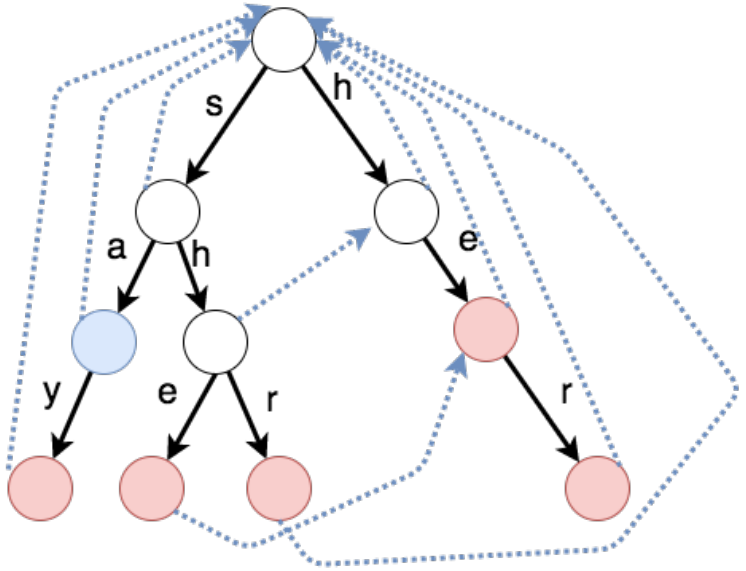
=

say
she
shr
he
her

AC 自动机--匹配

s	a	s	h	e	r	h	s

+1次



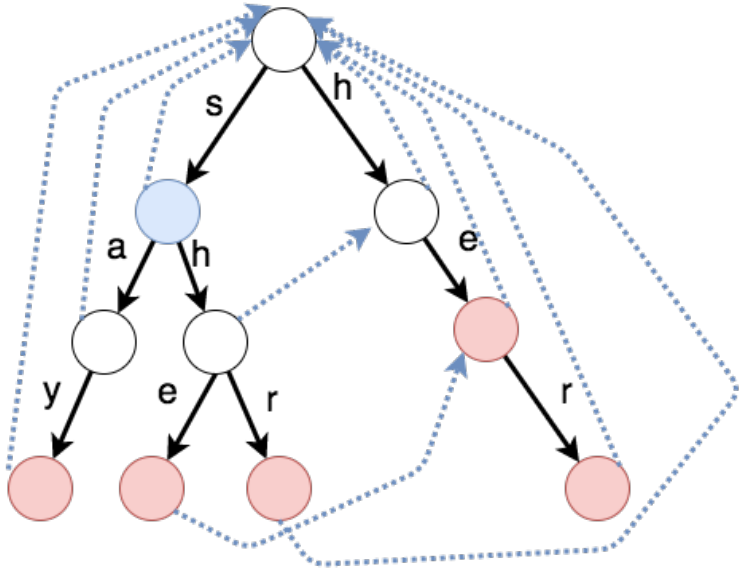
=

say
she
shr
he
her

AC 自动机--匹配

s	a	s	h	e	r	h	s

+1次



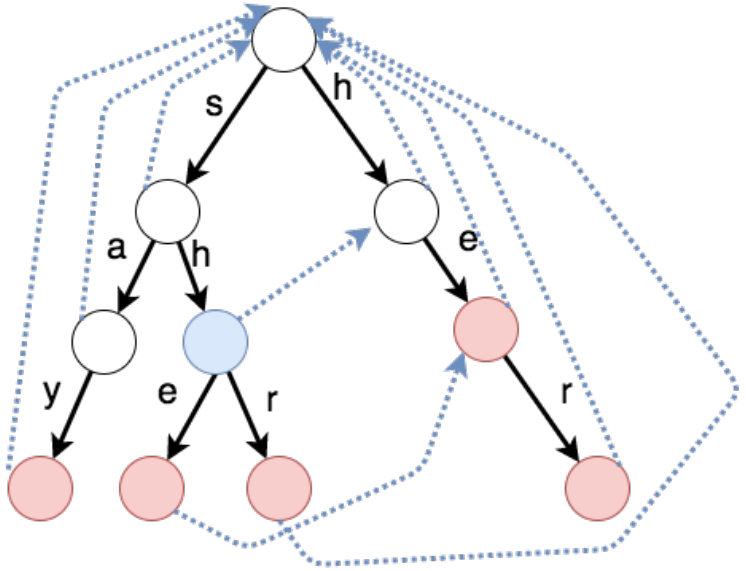
=

say
she
shr
he
her

AC 自动机--匹配

s	a	s	h	e	r	h	s

+1次



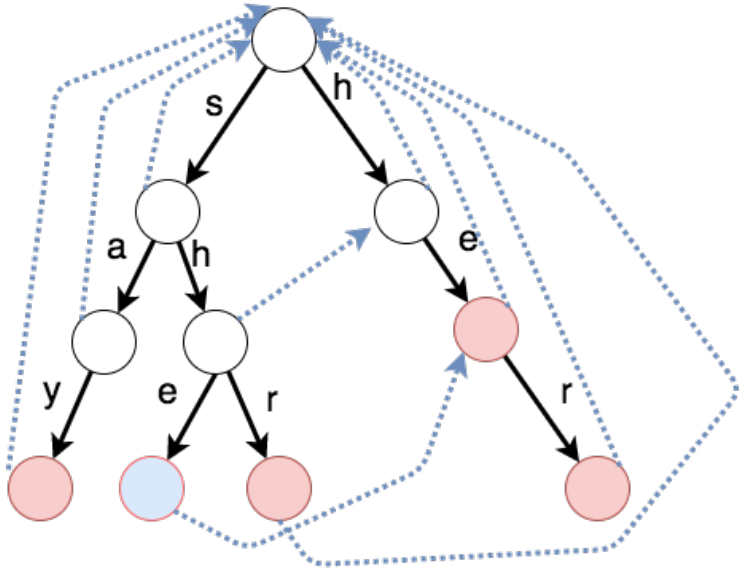
=

say
she
shr
he
her

AC 自动机--匹配

s	a	s	h	e	r	h	s

+1次

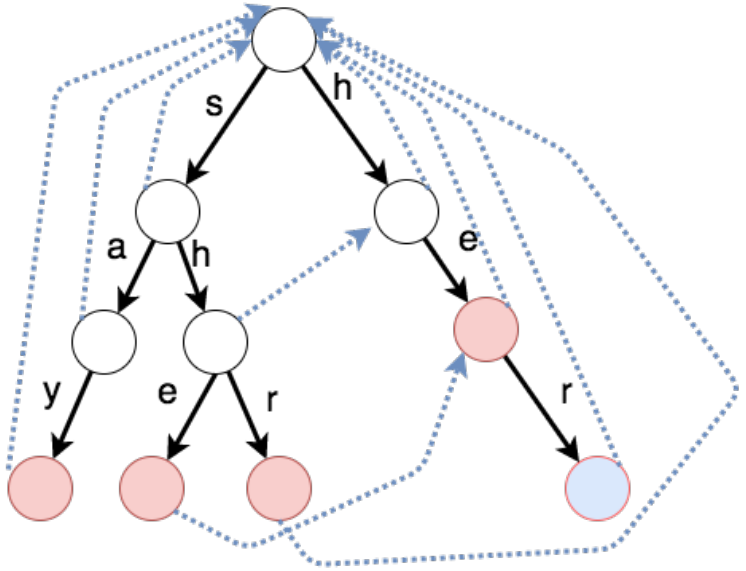


say
she
shr
he
her

AC 自动机--匹配

s	a	s	h	e	r	h	s

+1次



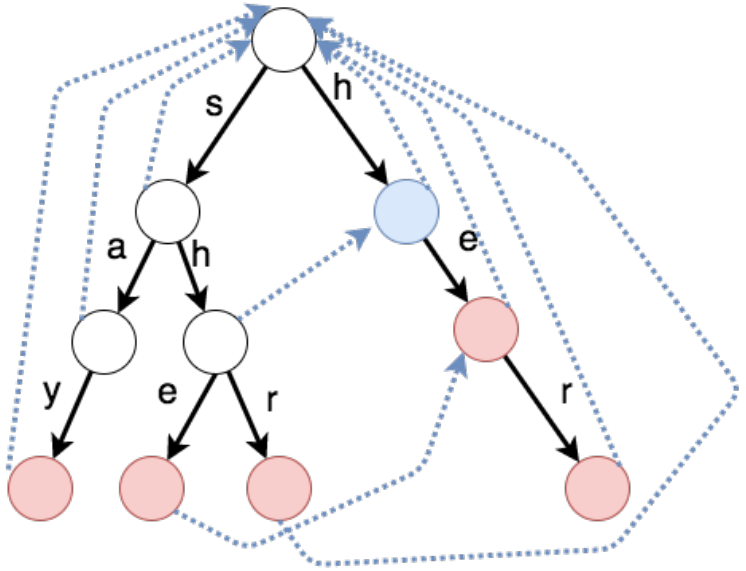
=

say
she
shr
he
her

AC 自动机--匹配

s	a	s	h	e	r	h	s

+1次



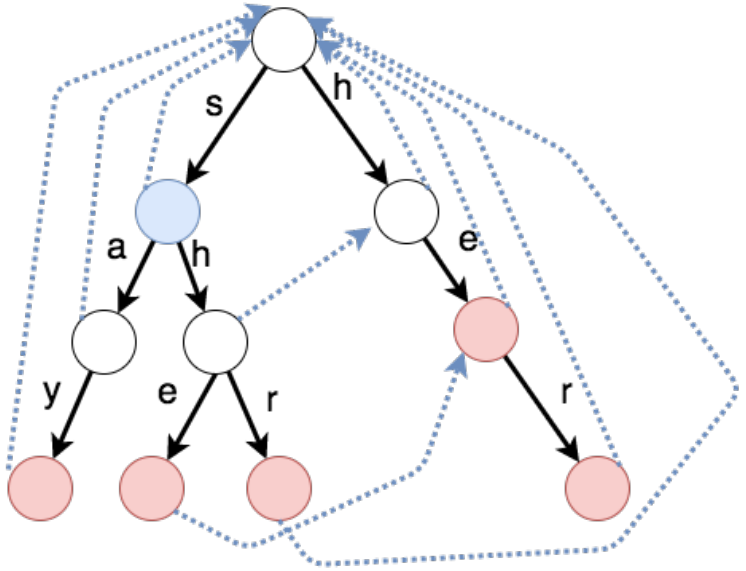
=

say
she
shr
he
her

AC 自动机--匹配

s	a	s	h	e	r	h	s

8次

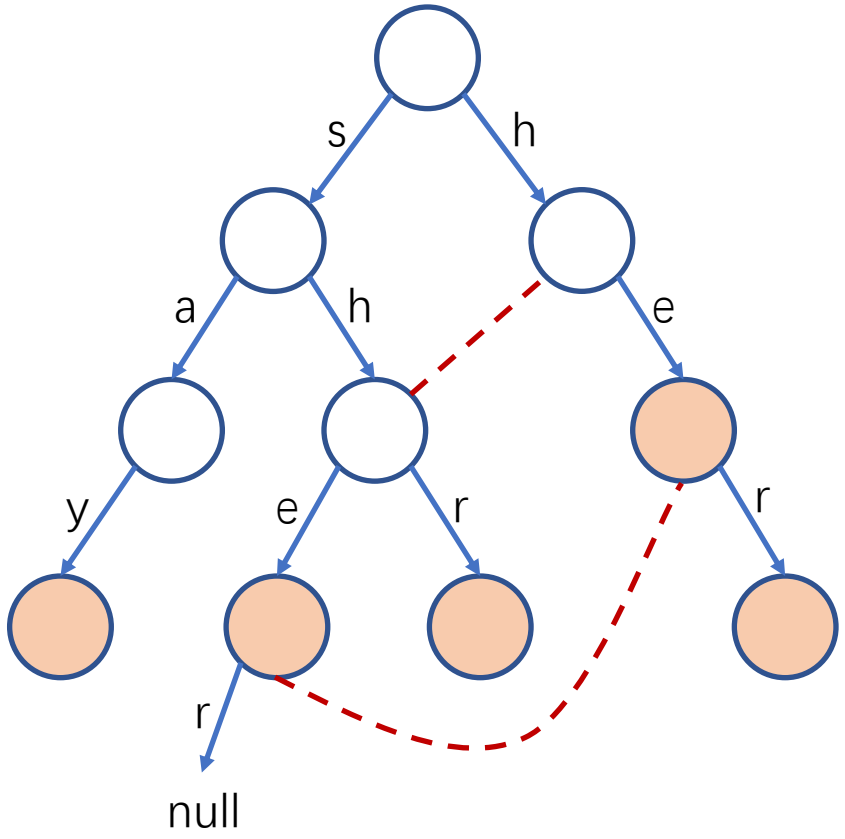


=

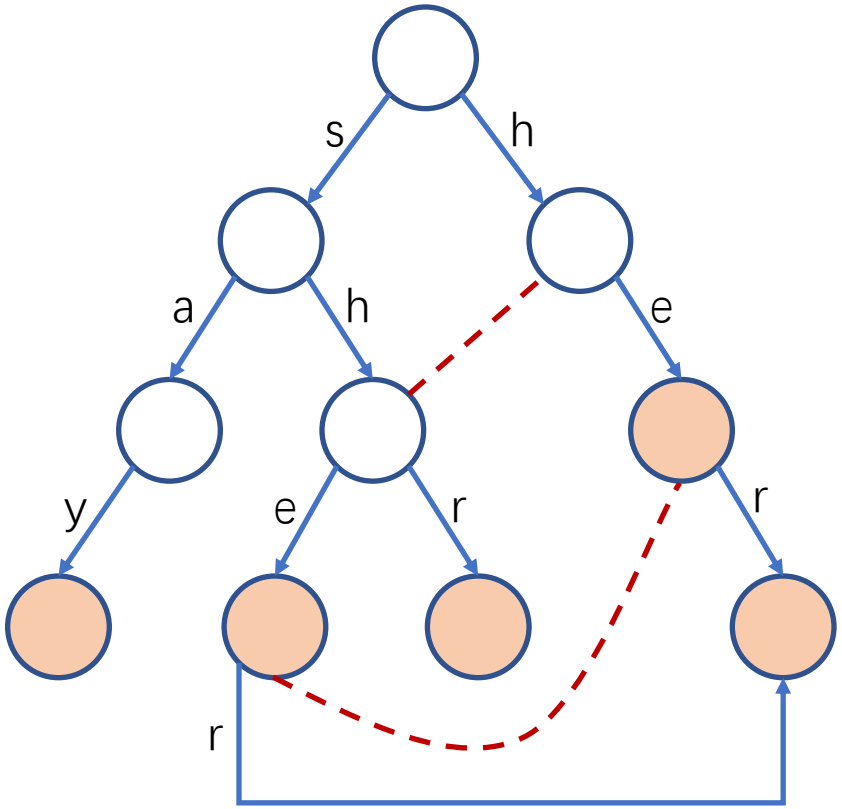
say
she
shr
he
her

AC自动机的优化

AC自动机的优化



AC自动机的优化



实战演练

实战演练

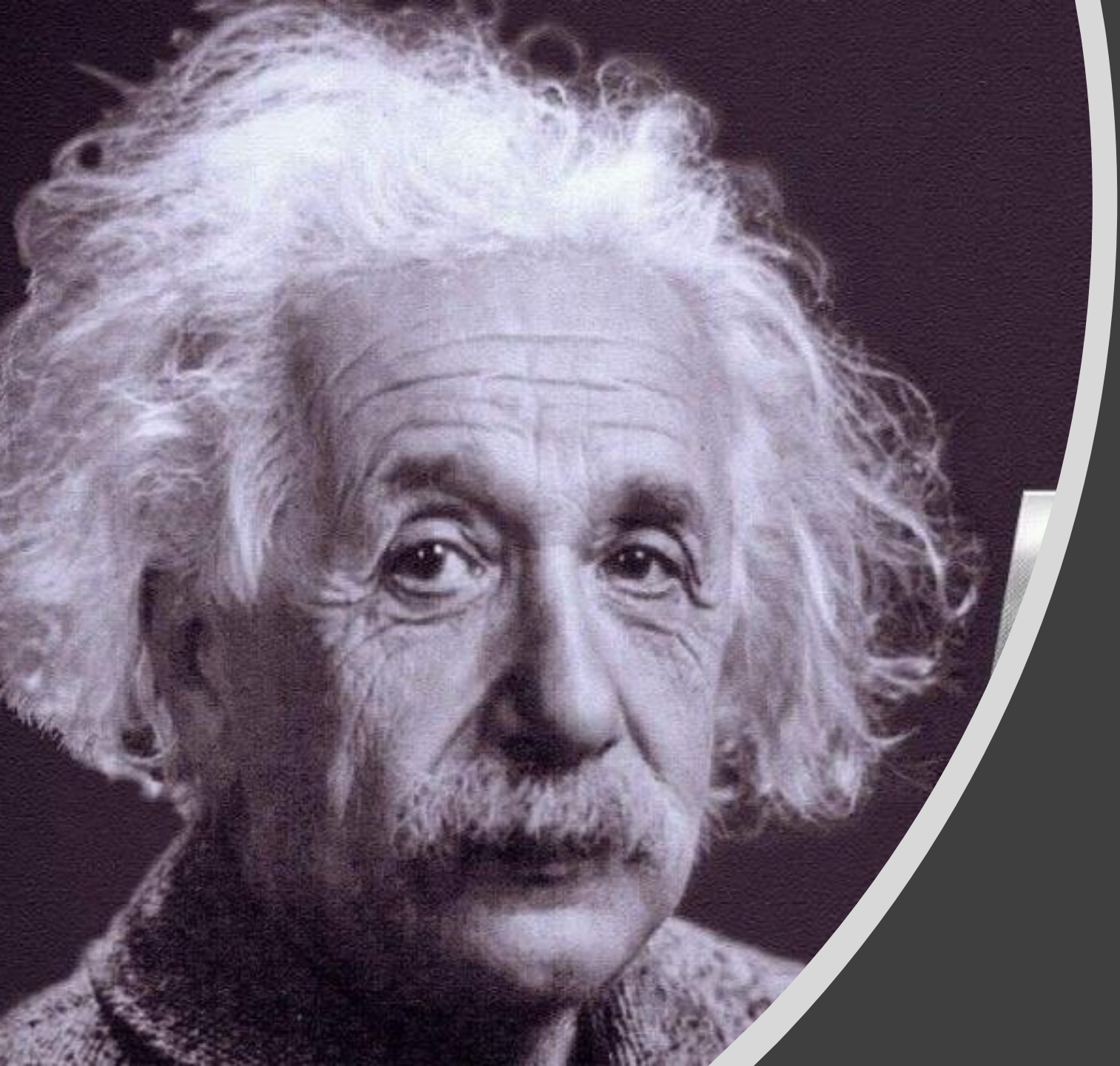
P3808 【模板】AC 自动机（简单版）

P3796 【模板】AC 自动机（加强版）

https://www.luogu.com.cn/problem/P3808
https://www.luogu.com.cn/problem/P3796

三、字符串匹配-课后实战题

1. HZOJ-278：循环的字符串
2. HZOJ-279：项链的主人
3. HZOJ-281：前缀统计
4. HZOJ-282：最大异或对
5. HZOJ-283：拨号
6. P3370：【模板】字符串哈希
7. P5410：【模板】扩展 KMP
8. P1470：最长前缀
9. P8306：【模板】字典树
10. P2292：L 语言



为什么
会出一样的题目？

《船说：算法与数据结构》
第15章-字符串匹配