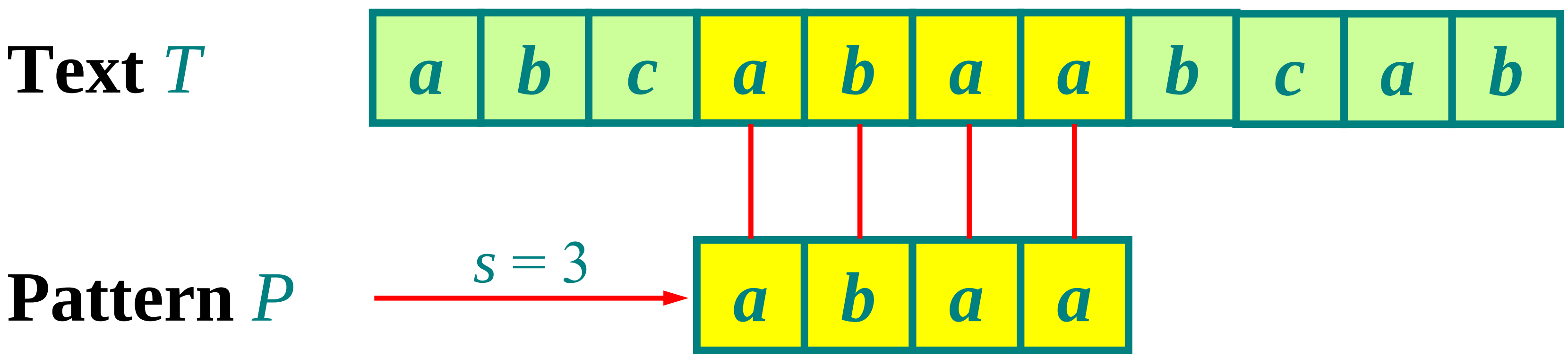
Data Structures and Algorithm

Xiaoqing Zheng
zhengxq@fudan.edu.cn

String matching problem

Pattern P ***occurs with shift*** s in text T (or , equivalently, that pattern P ***occurs beginning at position*** $s + 1$ in text T) if $T[s + 1 \ldots s + m] = P[1 \ldots m]$ and $0 \leq s \leq n - m$. If P occurs with shift s in T, the we call s a ***valid shift***.

The ***string-matching problem*** is the problem of finding ***all valid shifts*** with which a given pattern P occurs in a given text T.



Notation and terminology

Σ	finite alphabet. $\Sigma = \{a, b, ..., z\}$.
Σ^*	set of all finite-length strings formed using characters from the alphabet Σ.
ε	zero-length empty string belongs to Σ^*.
$\lvert x \rvert$	length of a string x.
xy	concatenation of two string x and y.
$w \triangleright x$	w is a prefix of a string x, if $x = wy$ for some string $y \in \Sigma^*$.
$w \triangleleft x$	w is a suffix of a string x, if $x = yw$ for some string $y \in \Sigma^*$.
P_k	k-character prefix $P[1 ... k]$ of the pattern P.

Naive string-matching algorithm

NAIVE-STRING-MATCHER(T, P)

1. $n \leftarrow length[T]$
2. $m \leftarrow length[P]$
3. **for** $s \leftarrow 0$ **to** $n - m$
4. **do if** $P[1 \ldots m] = T[s+1 \ldots s+m]$
5. **then** print "Pattern occurs with shift" s

Running time is $O((n - m + 1)m)$

Idea of Rabin-Karp algorithm

Input characters and string can be represented by ***graphical symbols*** or ***digits***.

Given a pattern $P[1 \ldots m]$, let p denote its corresponding decimal value.

$$p = P[m] + 10(P[m-1]) + 10(P[m-2] + \ldots + 10(P[2] + 10P[1])\ldots))$$

We also let t_s denote the decimal value of the length-m substring $T[s+1 \ldots s+m]$, for $s = 0, 1, \ldots, n-m$.

Then, $t_s = p$ if and only if $T[s+1 \ldots s+m] = P[1 \ldots m]$.

Idea of Rabin-Karp algorithm

t_{s+1} can be computed from t_s in constant time, since,

$t_{s+1} = 10(t_s - 10^{m-1} T[s + 1]) + T[s + m + 1]$.

Constant is precomputed which can be done in time $O(lgm)$

For example, if $m = 5$ and $t_s = 31415$, then we remove the high-order digit $T[s + 1] = 3$ and bring in the new low-order digit $T[s + 5 + 1] = 2$ to obtain

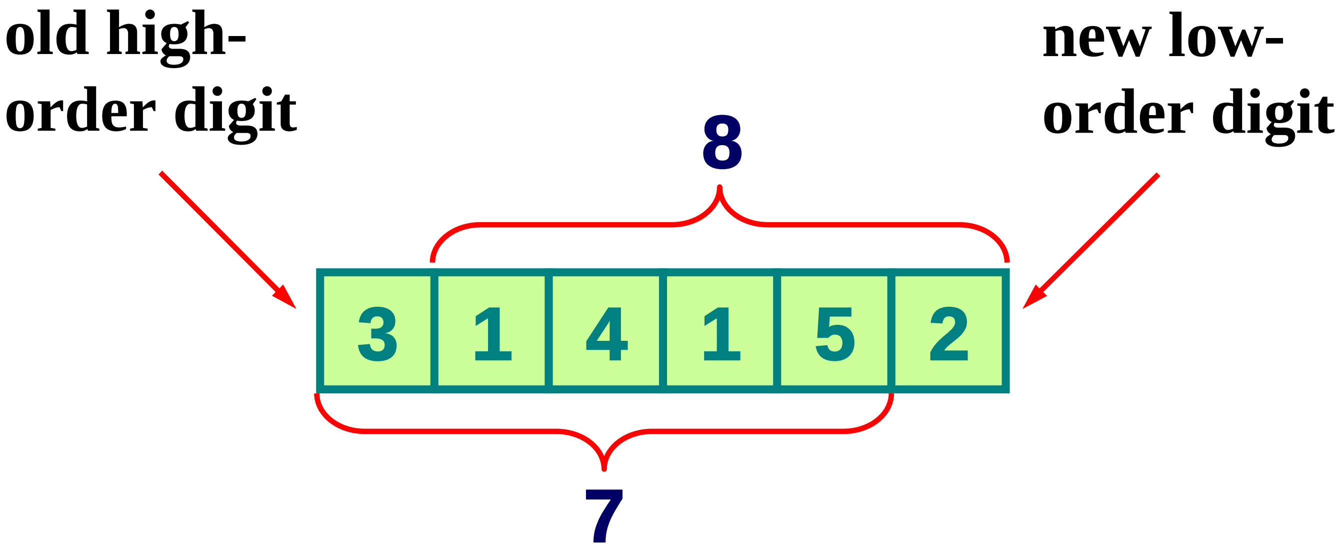
$$t_{s+1} = 10(31415 - 10000 \cdot 3) + 2$$
$$= 14152.$$

Improved Rabin-Karp algorithm

$t_{s+1} = (10(t_s - T[s+1]h) + T[s+m+1]) \bmod q.$

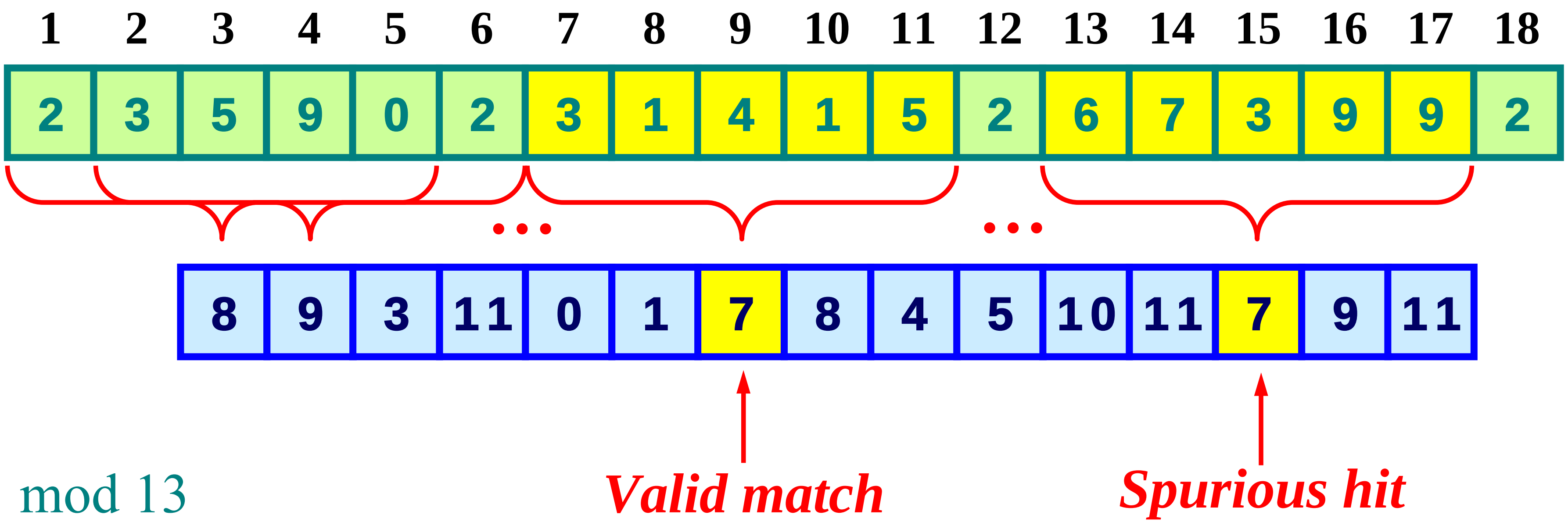
- q is typically chosen as a prime such that $10q$ just fits within one computer word.
- $h \equiv 10^{m-1} \pmod{q}$.

Improved Rabin-Karp algorithm



$$14152 \equiv 10 \cdot (31415 - 3 \cdot 10000) + 2 \pmod{13}$$
$$\equiv 10 \cdot (31415 - 3 \cdot 3) + 2 \pmod{13}$$
$$\equiv 8 \pmod{13}$$

Improved Rabin-Karp algorithm



mod 13

$t_s \equiv p \pmod{q}$ does not imply that $t_s = p$.

Rabin-Karp algorithm

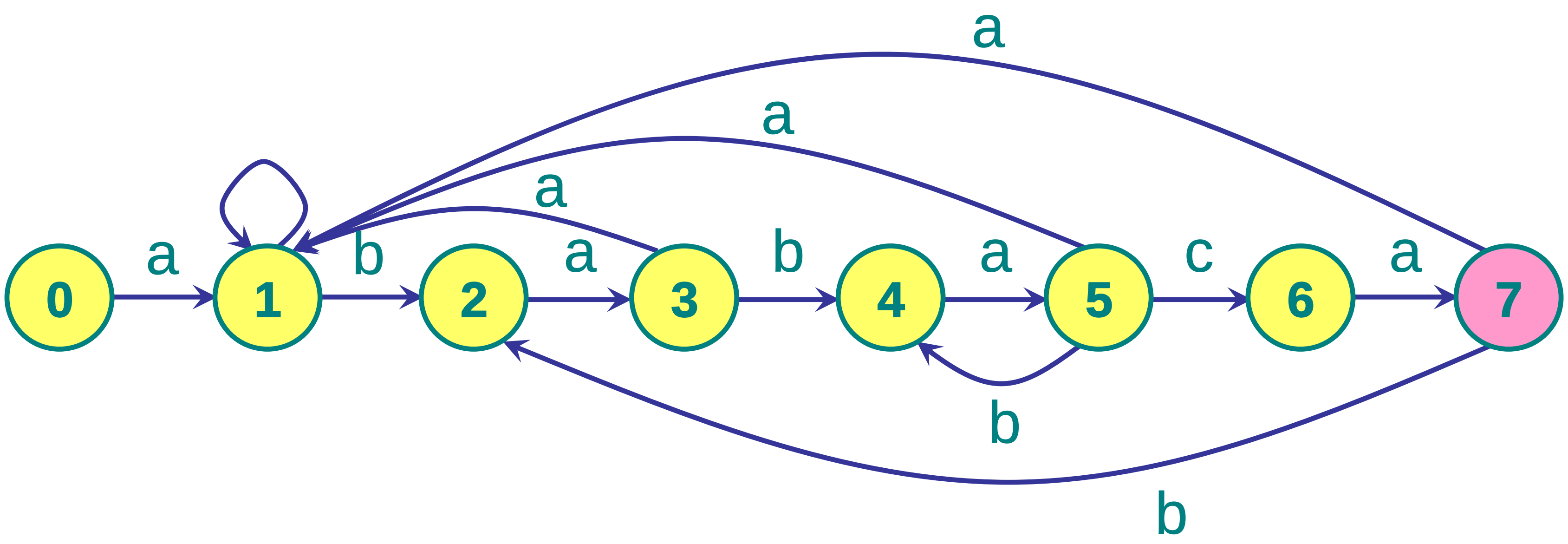
RABIN-KARP-MATCHER(T, P, d, q)

$n \leftarrow length[T]$
$m \leftarrow length[P]$
$h \leftarrow d^{m-1} \bmod q$
$p \leftarrow 0$
$t_0 \leftarrow 0$
for $i \leftarrow 1$ **to** m // Preprocessing.
    do $p \leftarrow (dp + P[i]) \bmod q$
    do $t_0 \leftarrow (dt_0 + T[i]) \bmod q$
for $s \leftarrow 0$ **to** $n - m$ // Matching.
    do if $p = t_s$
        then if $P[1 \ldots m] = T[s+1 \ldots s+m]$
            then print "Pattern occurs with shift" s
        if $s < n - m$
        then $t_{s+1} \leftarrow (d(t_s - T[s+1]h) + T[s+m+1]) \bmod q$

Preprocessing
$\Phi(m)$

Matching
$O((n - m + 1)m)$

Finite automaton

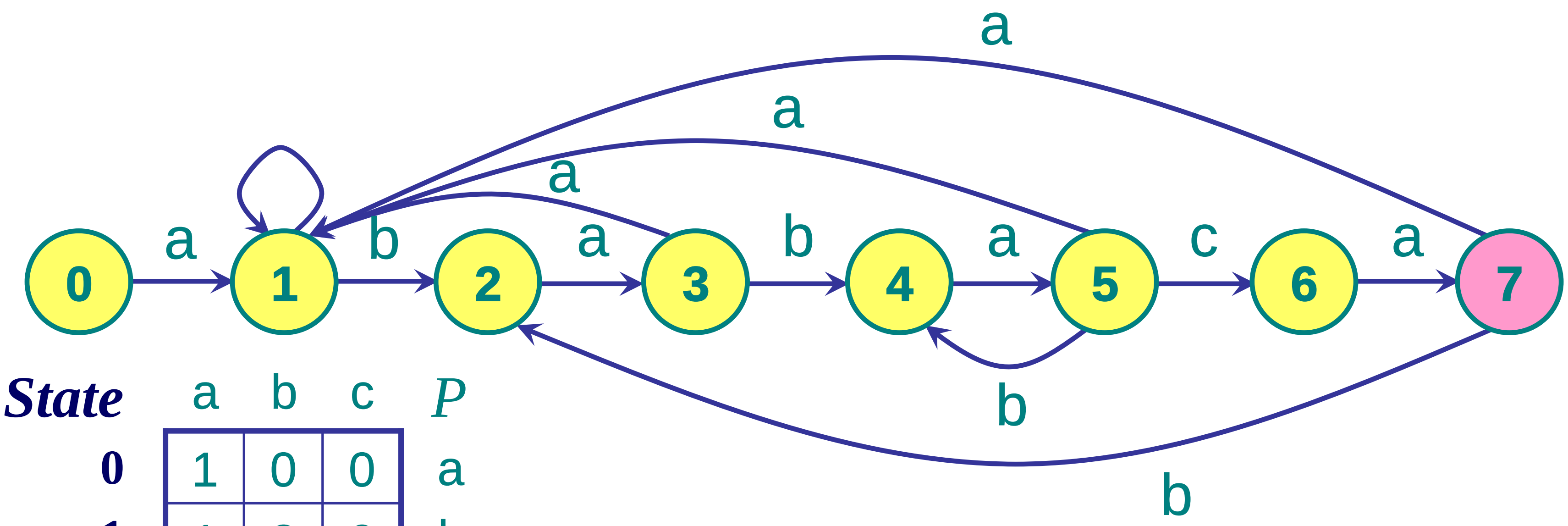


Finite automaton

A ***finite automaton*** M is a 5-tuple $(Q, q_0, A, \sum, \delta)$

- Q is a finite set of ***states***,
- $q_0 \in Q$ is the ***start states***,
- $A \subseteq Q$ is a distinguished set of ***accepting states***,
- $\sum$ is a finite ***input alphabet***,
- δ is a function from $Q \times \sum$ into Q, called the ***transition function*** of M.

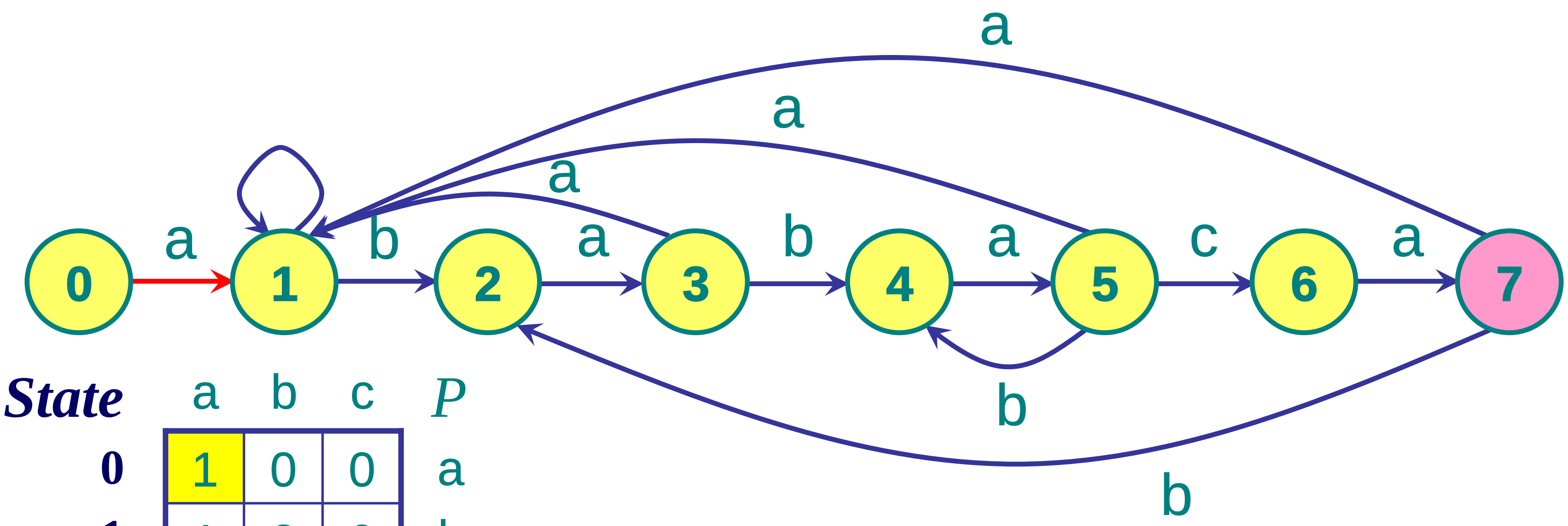
Operation of finite automaton



State	a	b	c	*P*
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**											

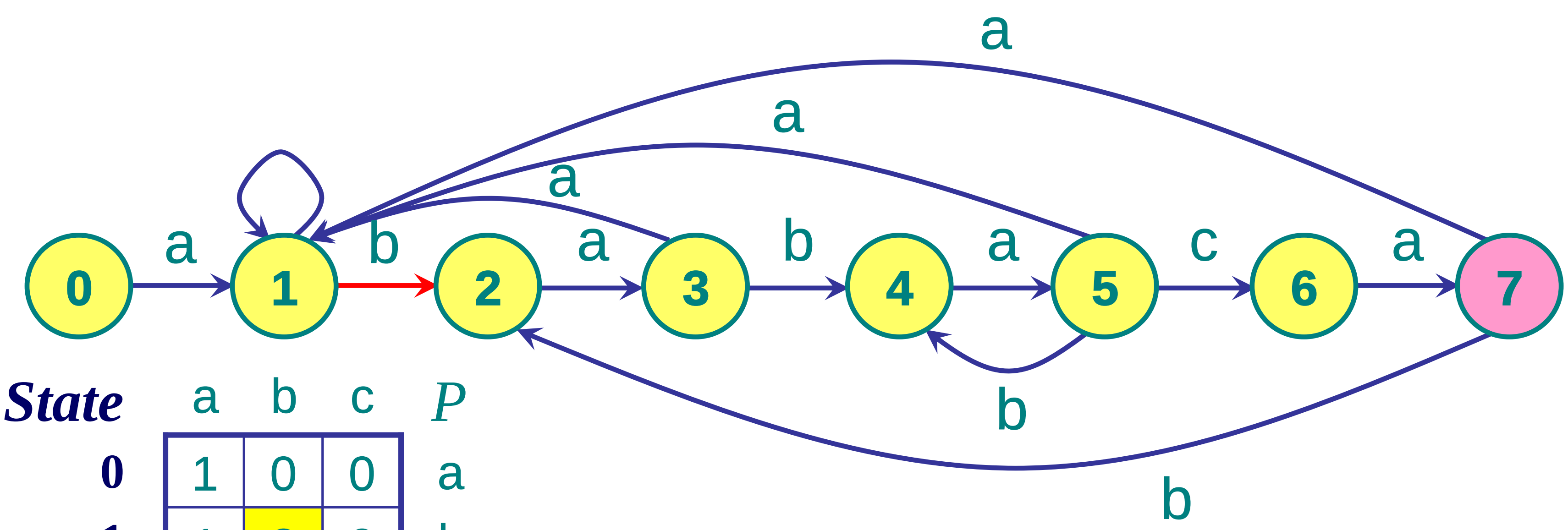
Operation of finite automaton



State	a	b	c	*P*
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**										

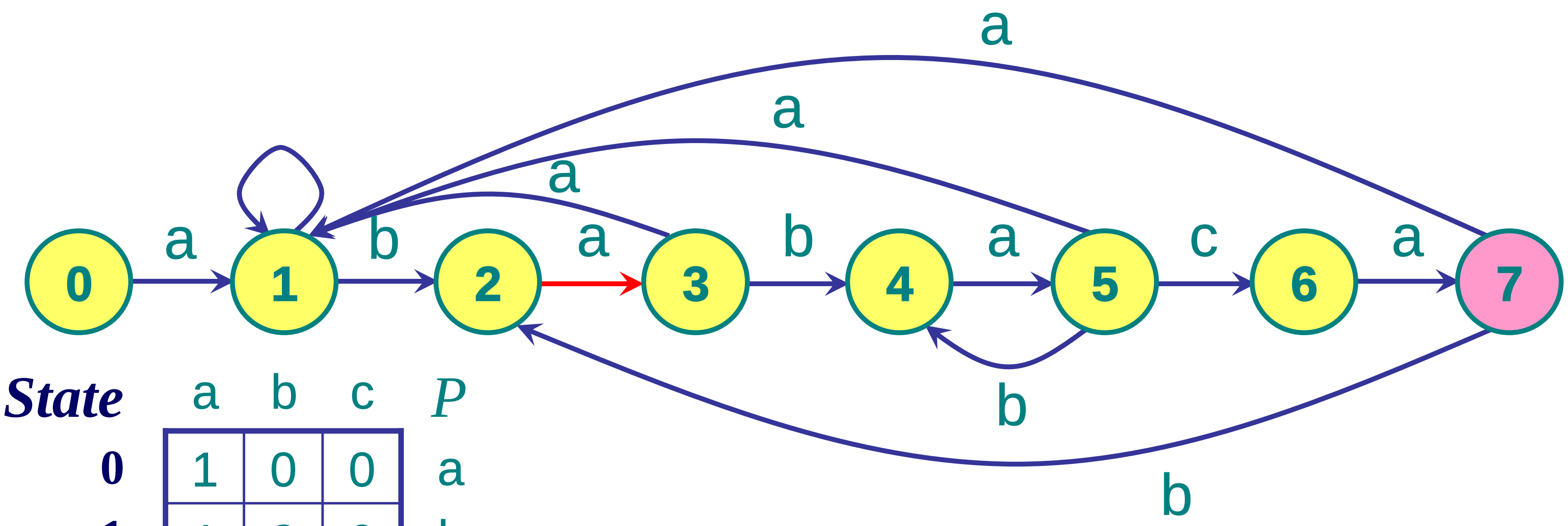
Operation of finite automaton



State	a	b	c	*P*
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**	**2**									

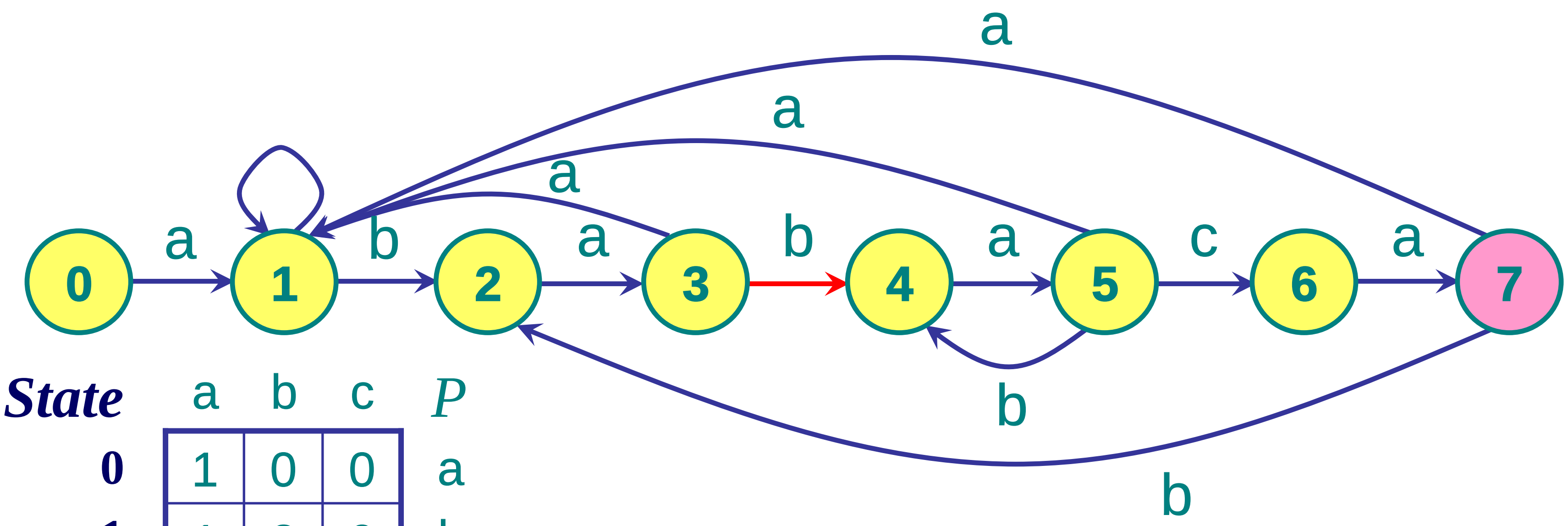
Operation of finite automaton



State	a	b	c	P
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**	**2**	**3**								

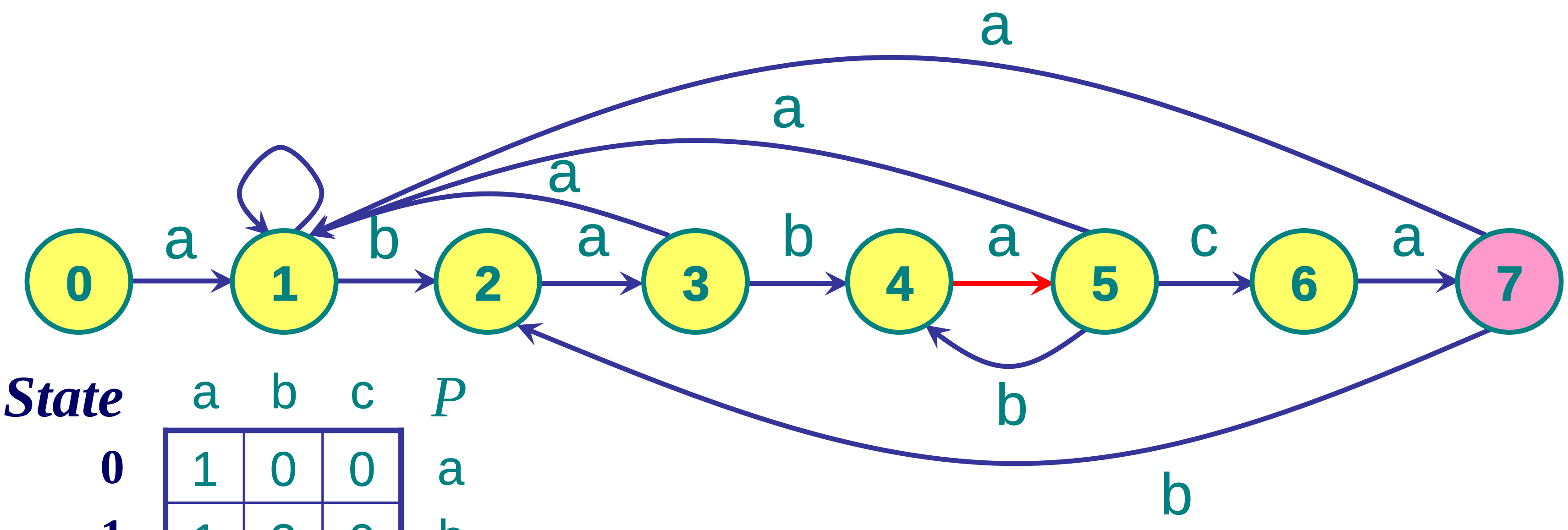
Operation of finite automaton



State	a	b	c	*P*
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**	**2**	**3**	**4**							

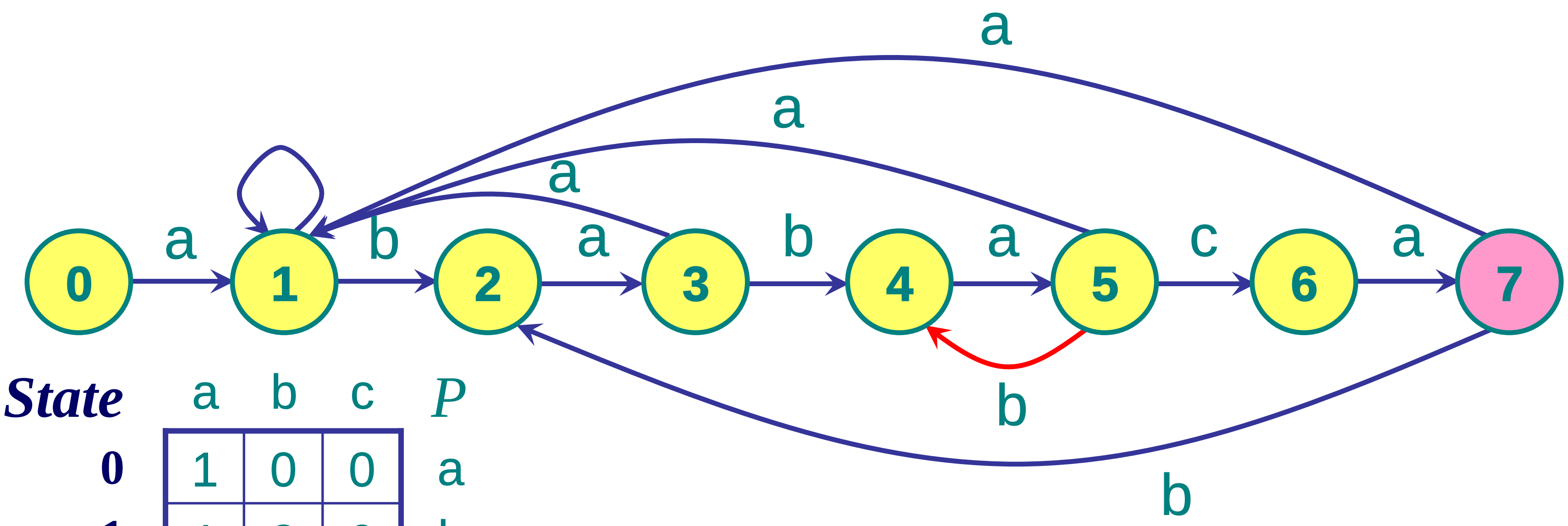
Operation of finite automaton



State	a	b	c	*P*
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**	**2**	**3**	**4**	**5**						

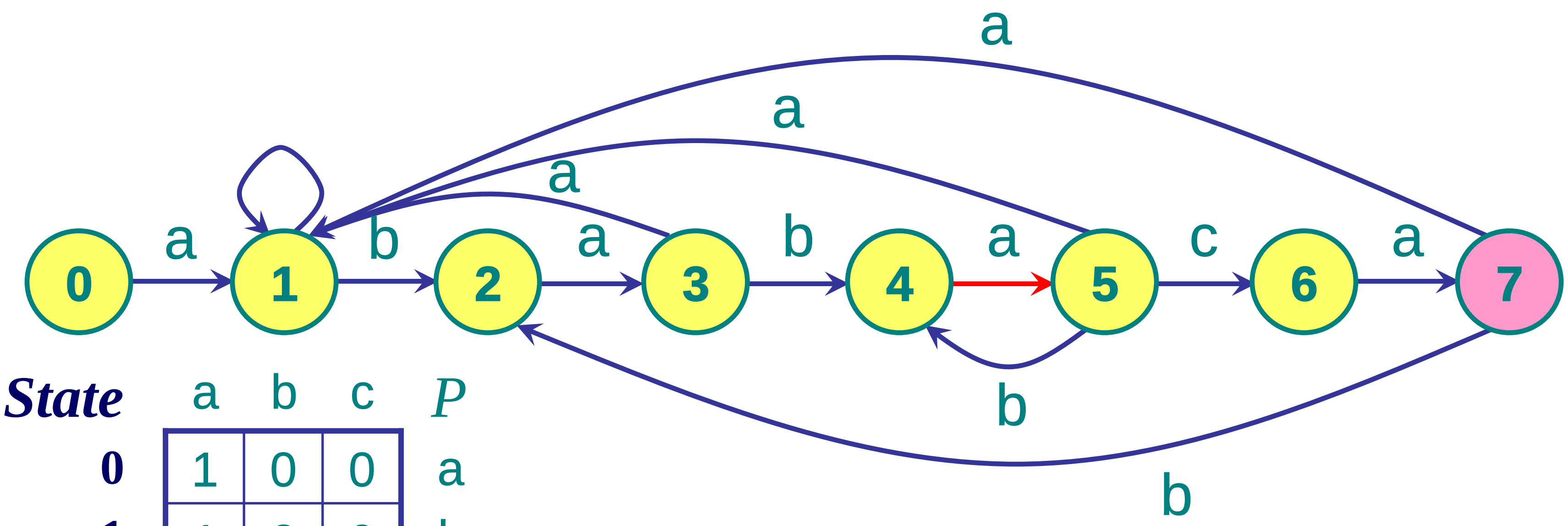
Operation of finite automaton



State	a	b	c	P
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**	**2**	**3**	**4**	**5**	**4**					

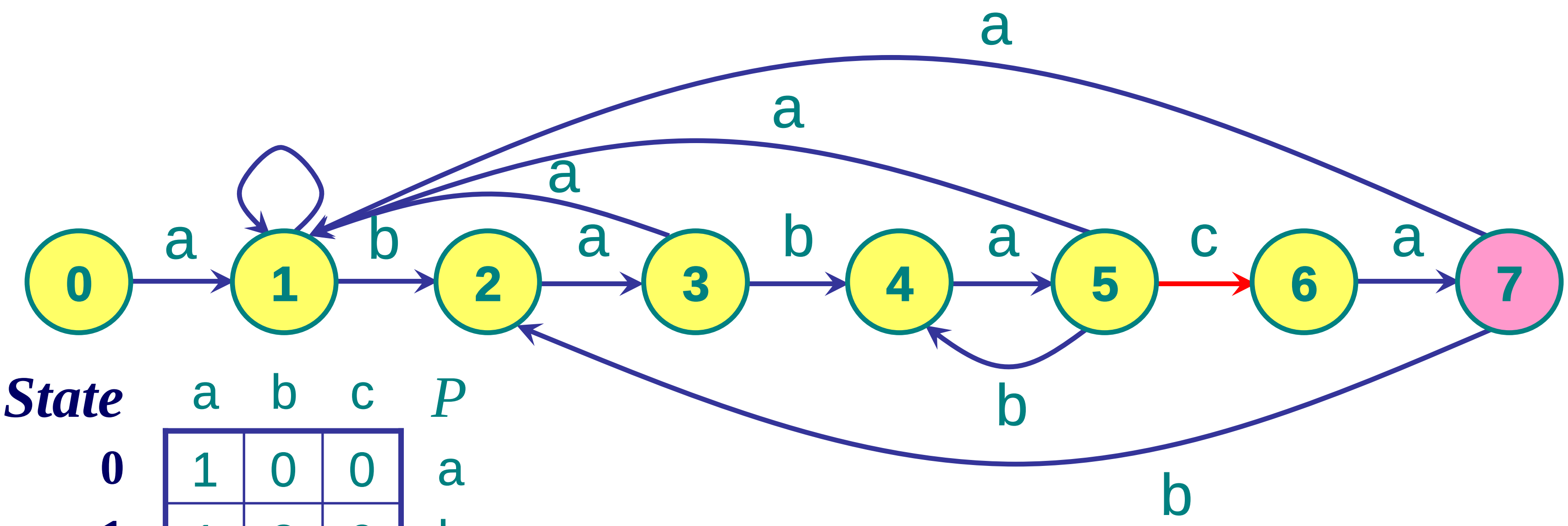
Operation of finite automaton



State	a	b	c	*P*
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**	**2**	**3**	**4**	**5**	**4**	**5**				

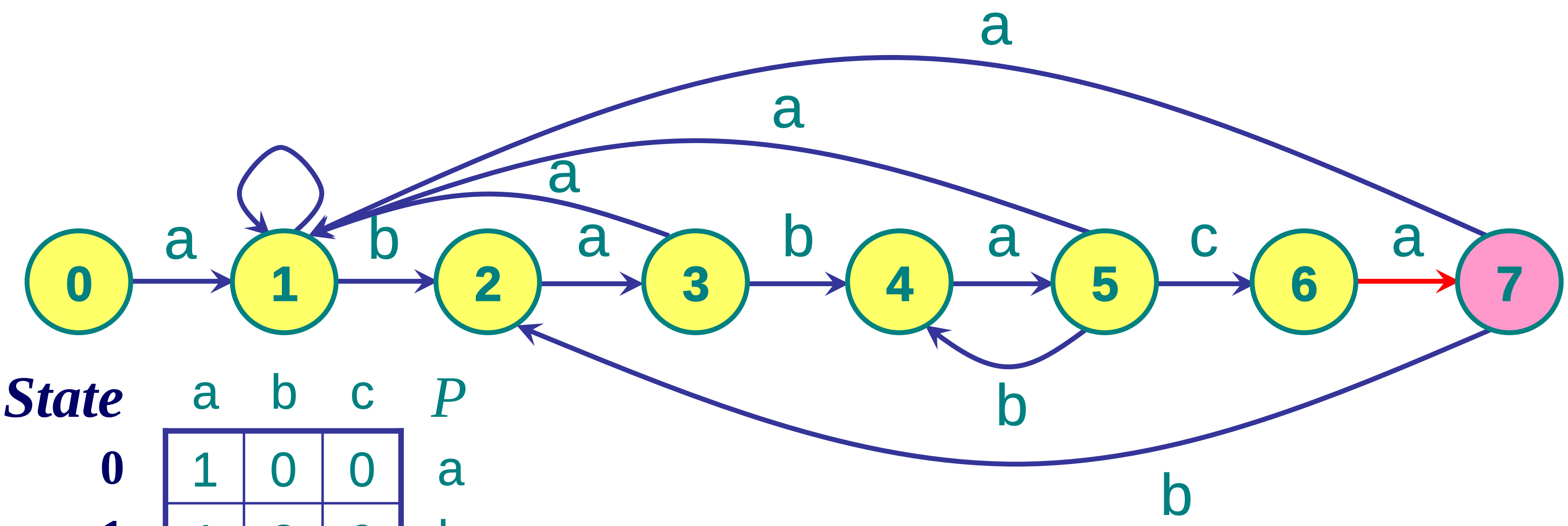
Operation of finite automaton



State	a	b	c	P
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**	**2**	**3**	**4**	**5**	**4**	**5**	**6**			

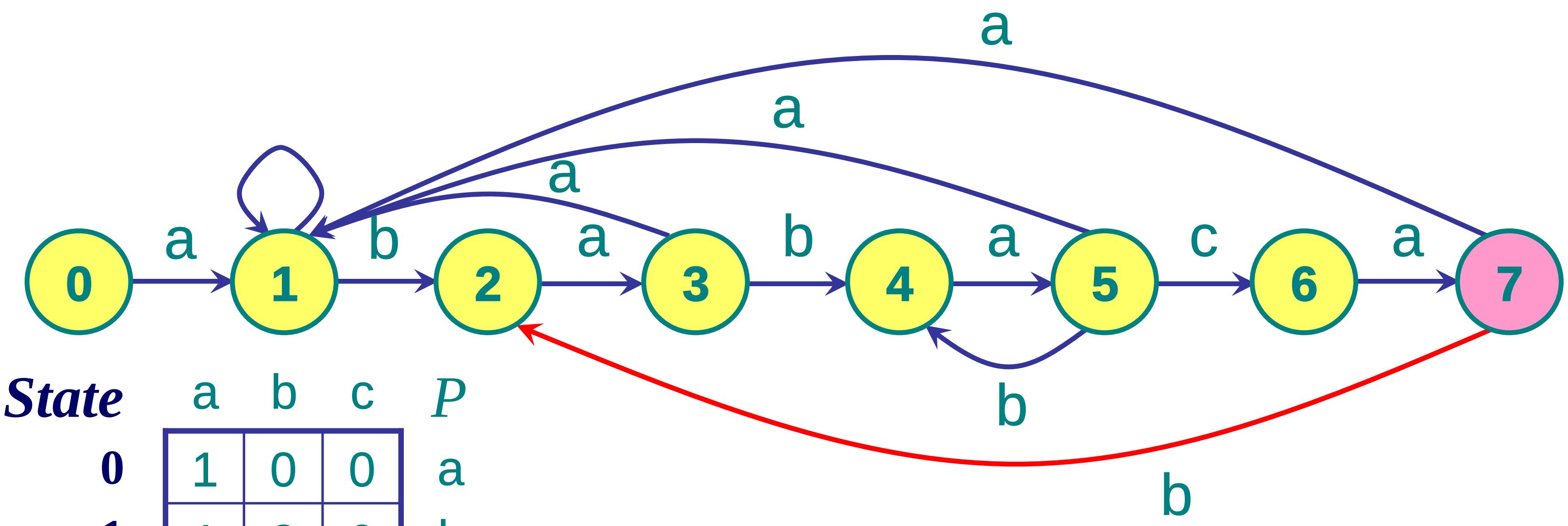
Operation of finite automaton



State	a	b	c	*P*
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**	**2**	**3**	**4**	**5**	**4**	**5**	**6**	**7**		

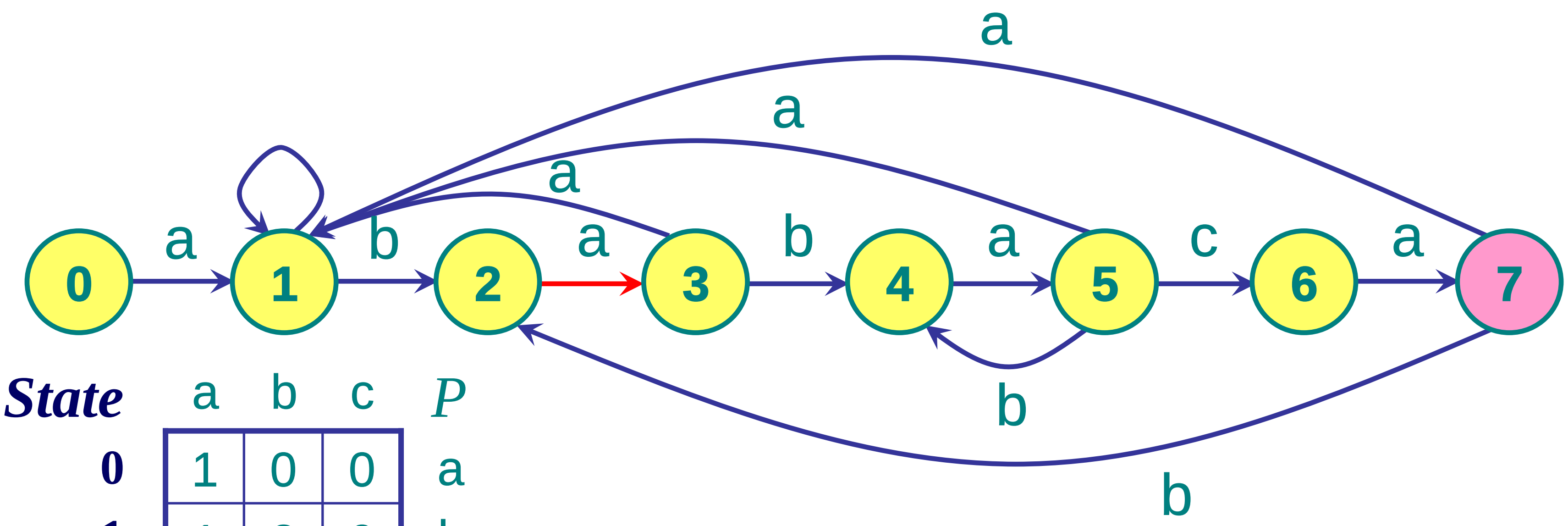
Operation of finite automaton



State	a	b	c	*P*
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**	**2**	**3**	**4**	**5**	**4**	**5**	**6**	**7**	**2**	

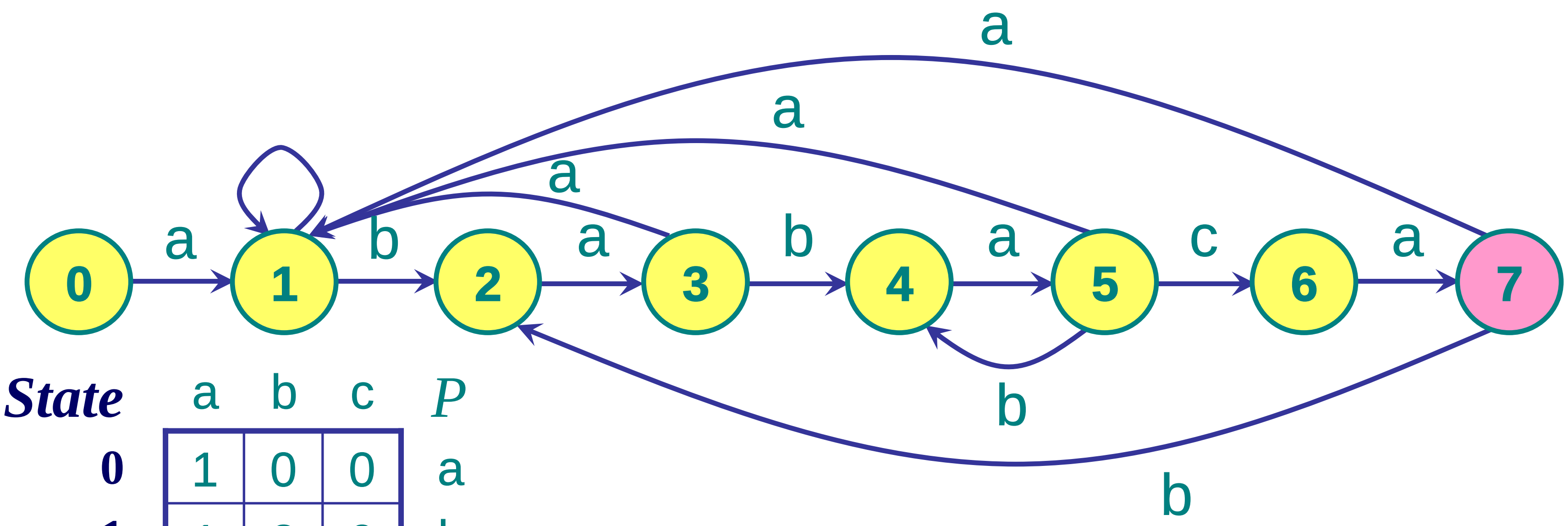
Operation of finite automaton



State	a	b	c	P
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	**0**	**1**	**2**	**3**	**4**	**5**	**4**	**5**	**6**	**7**	**2**	**3**

Operation of finite automaton



State	a	b	c	*P*
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

i	–	1	2	3	4	5	6	7	8	9	10	11
$T[i]$	–	a	b	a	b	a	b	a	c	a	b	a
δ	0	1	2	3	4	5	4	5	6	7	2	3

String matching with finite automata

FINITE-AUTOMATON-MATCHER(T, δ, m)

1. $n \leftarrow length[T]$
2. $q \leftarrow 0$
3. **for** $i \leftarrow 1$ **to** n
4. **do** $q \leftarrow \delta(q, T[i])$
5. **if** $q = m$
6. **then** print "Pattern occurs with shift" $i - m$

Running time is $\Theta(n)$

Computing the transition function

COMPUTE-TRANSITION-FUNCTION(P, Σ)

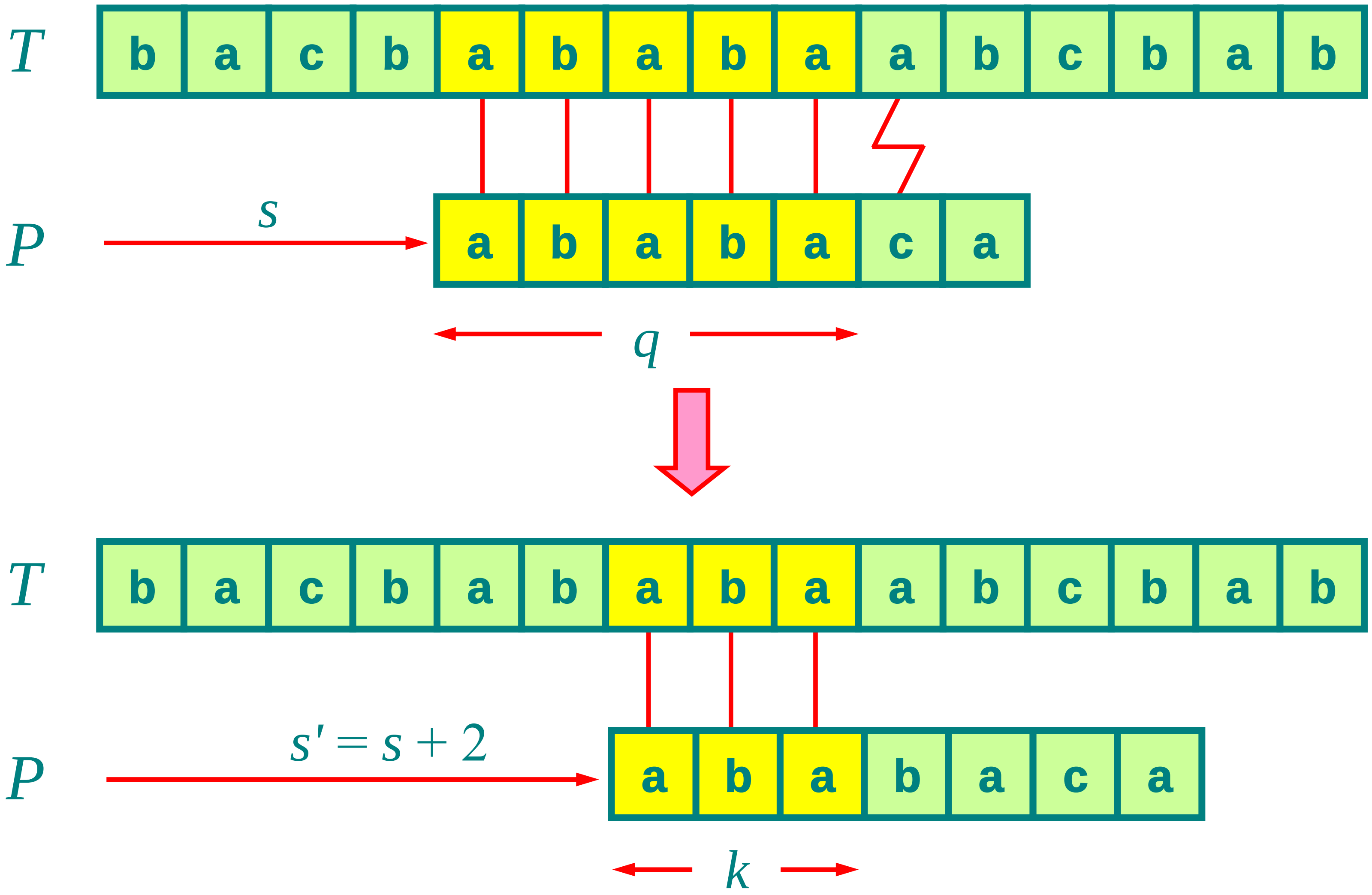
$m \leftarrow length[P]$
for $q \leftarrow 0$ **to** m
    do for each character $a \in \Sigma$
        do $k \leftarrow \min(m + 1, q + 2)$
            repeat $k \leftarrow k - 1$
            until $P_k \lhd P_q a$
            $\delta(q, a) \leftarrow k$
return δ

Running time is $O(m^3|\Sigma|)$

Computing the transition function

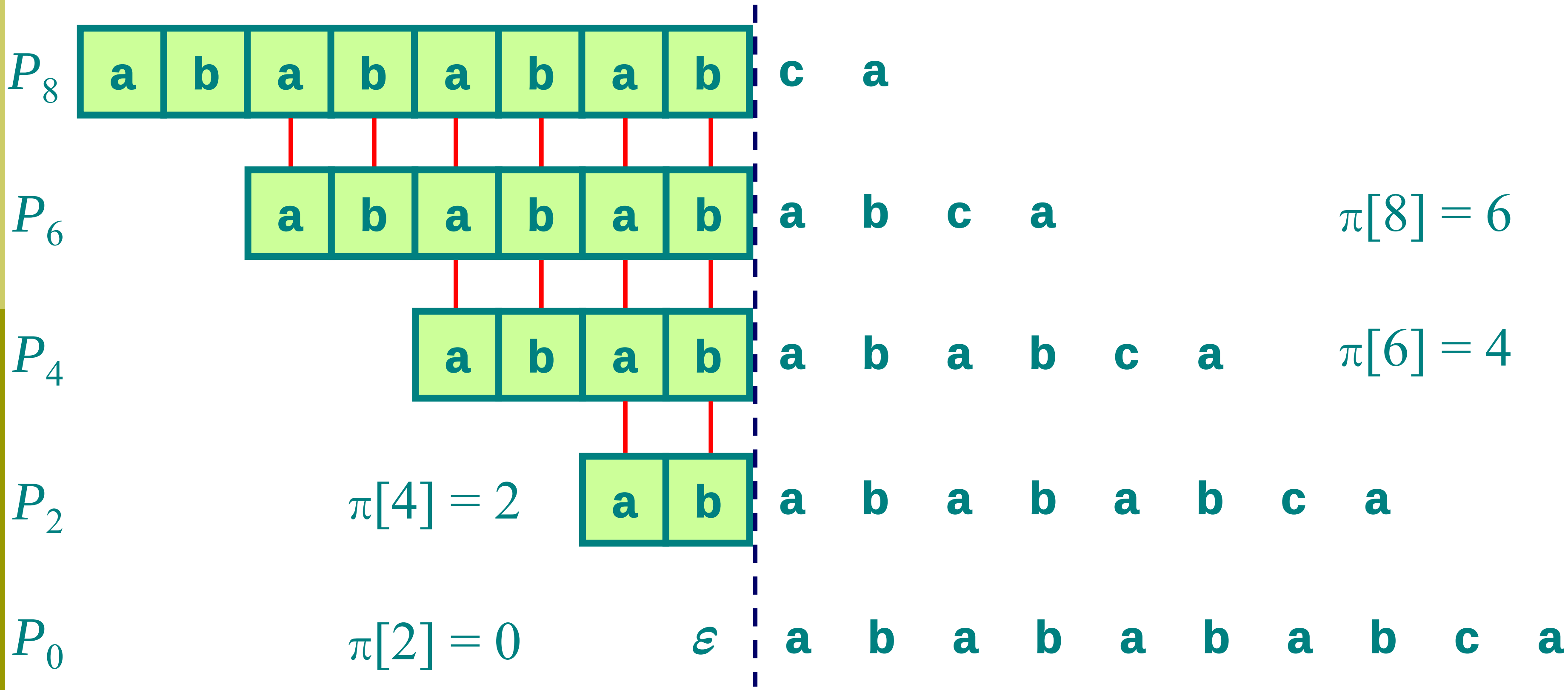
Step	m	q	a	k	$P_k \lhd P_q a$	δ
1	7	0	a	1	$a \lhd a$	$\delta(0, a) = 1$
2			b	1	$a \lhd b$	
3				0	$\varepsilon \lhd b$	$\delta(0, b) = 0$
4			c	1	$a \lhd c$	
5				0	$\varepsilon \lhd c$	$\delta(0, c) = 0$
6		1	a	2	$ab \lhd aa$	
7				1	$a \lhd aa$	$\delta(1, a) = 1$
8			b	2	$ab \lhd ab$	$\delta(1, b) = 2$
9			c	2	$ab \lhd ac$	
10				1	$a \lhd ac$	
11				0	$\varepsilon \lhd ac$	$\delta(1, c) = 0$
12	…	…	…	…	…	…

Idea of Knuth-Morris-Pratt algorithm



Idea of Knuth-Morris-Pratt algorithm

i	1	2	3	4	5	6	7	8	9	10
$P[i]$	a	b	a	b	a	b	a	b	c	a
$\pi[i]$	0	0	1	2	3	4	5	6	0	1



Knuth-Morris-Pratt algorithm

```
KMP-MATCHER(T, P)
n ← length[T]
m ← length[P]
π ← COMPUTE-PREFIX-FUNCTION(P)
q ← 0
for i ← 1 to n
        do while q > 0 and P[q + 1] ≠ T[i]
                do q ← π[q]
           if P[q + 1] = T[i]
             then q ← q + 1
           if q = m
             then print "Pattern occurs with shift" i − m
                  q ← π[q]
```

Running time is $\Theta(n)$

Computing prefix function

COMPUTE-PREFIX-FUNCTION(P)

1. $m \leftarrow length[P]$
2. $\pi[1] \leftarrow 0$
3. $k \leftarrow 0$
4. **for** $q \leftarrow 2$ **to** m
5. **do while** $k > 0$ and $P[k+1] \neq P[q]$
6. **do** $k \leftarrow \pi[k]$
7. **if** $P[k+1] = P[q]$
8. **then** $k \leftarrow k + 1$
9. $\pi[q] \leftarrow k$
10. **return** π

Running time is $\Theta(m)$

Computing prefix function

Step	m	q	k	$P[k+1] = P[q]$	π
1	10		0		$\pi(1) = 0$
2		2		$P[1] = a \neq b = P[2]$	$\pi(2) = 0$
3		3		$P[1] = a = a = P[3]$	
4			1		$\pi(3) = 1$
5		4		$P[2] = b = b = P[4]$	
6			2		$\pi(4) = 2$
7		5		$P[3] = a = a = P[5]$	
8			3		$\pi(5) = 3$
9		6		$P[4] = b = b = P[6]$	
10			4		$\pi(6) = 4$
11		7		$P[5] = a = a = P[7]$	
12			5		$\pi(7) = 5$

Computing prefix function (cont.)

Step	m	q	k	$P[k+1]=P[q]$	π
13	10	8		$P[6]=b=b=P[8]$	
14			6		$\pi(8)=6$
15		9		$P[7]=a \neq c=P[9]$	
16			4	$P[5]=a \neq c=P[9]$	
17			2	$P[3]=a \neq c=P[9]$	
18			0		$\pi(9)=0$

String matching algorithms

Algorithms	Preprocessing time	Matching time
Naive	0	$O((n-m+1)m)$
Rabin-Karp	$\Theta(m)$	$O((n-m+1)m)$
Finite automaton	$O(m\|\Sigma\|)$	$\Theta(n)$
Knuth-Morris-Pratt	$\Theta(m)$	$\Theta(n)$

Any question?

Xiaoqing Zheng
Fundan University