String Matching

Using Finite Automata

Example (I)

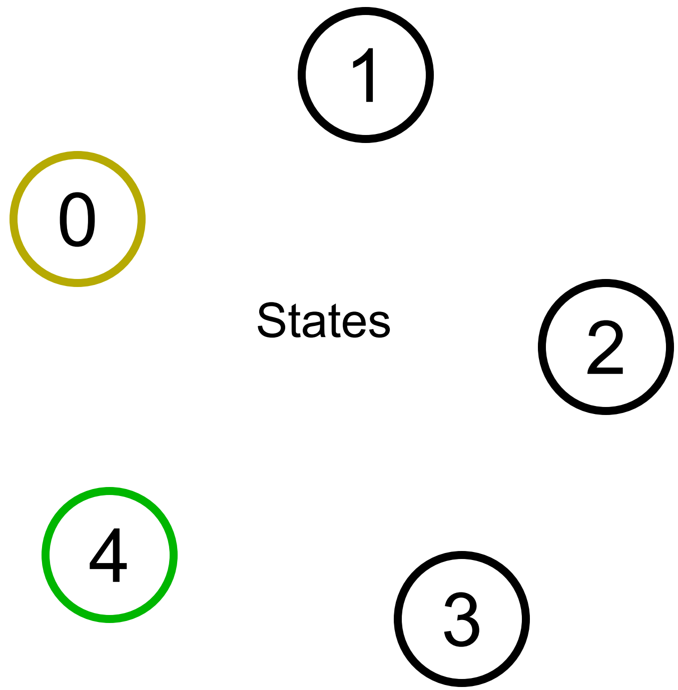
Q is a finite set of states

$q_0 \in \mathbf{Q}$ is the start state

Q is a set of accepting sates

Σ: input alphabet

$\delta: Q \times \Sigma \rightarrow Q$: transition function



input:

a	b	a	b	b	a	b	b	a	a

Example (II)

Q is a finite set of states

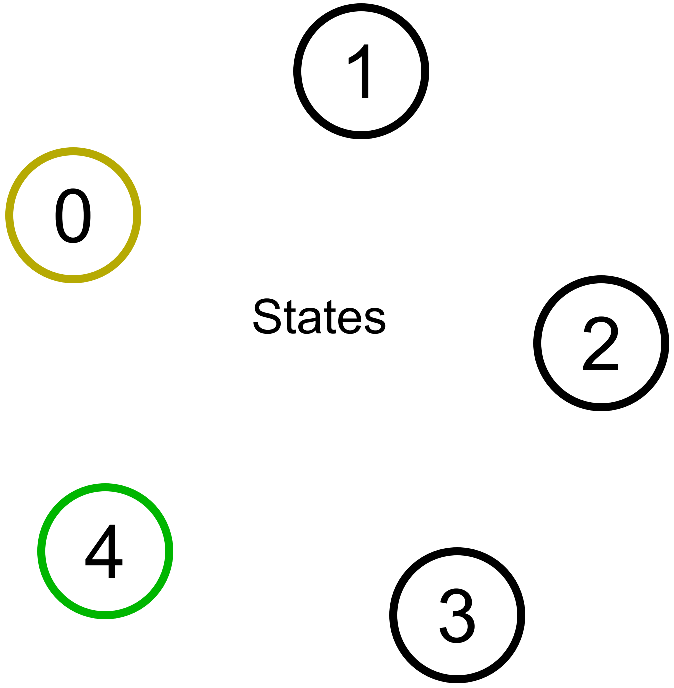
$q_0 \in \mathbf{Q}$ is the start state

Q is a set of accepting sates

Σ: input alphabet

δ: $Q \times \Sigma \rightarrow Q$: transition function

input state	a	b
0	1	0
1	1	2
2	1	3
3	4	0
4	1	2



Example (III)

Q is a finite set of states

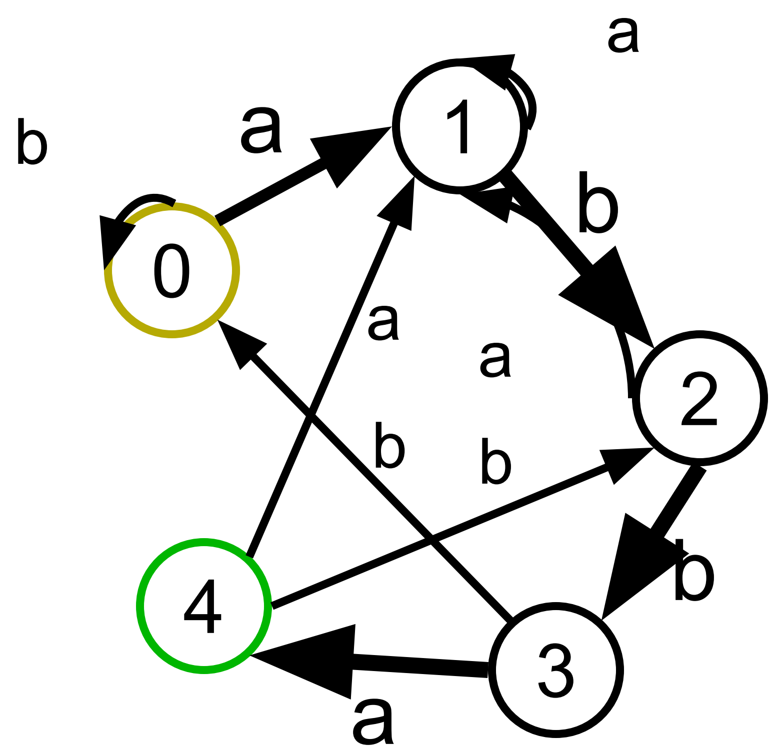
$q_0 \in \mathbf{Q}$ is the start state

Q is a set of accepting sates

Σ: input alphabet

δ: $Q \times \Sigma \rightarrow Q$: transition function

input state	a	b
0	1	0
1	1	2
2	1	3
3	4	0
4	1	2



Example (IV)

Q is a finite set of states

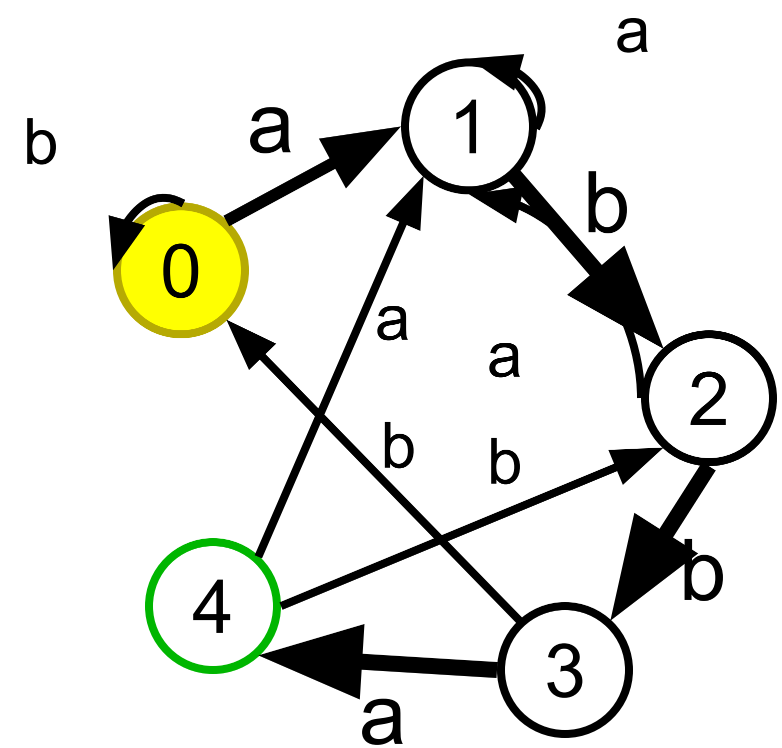
$q_0 \in \mathbf{Q}$ is the start state

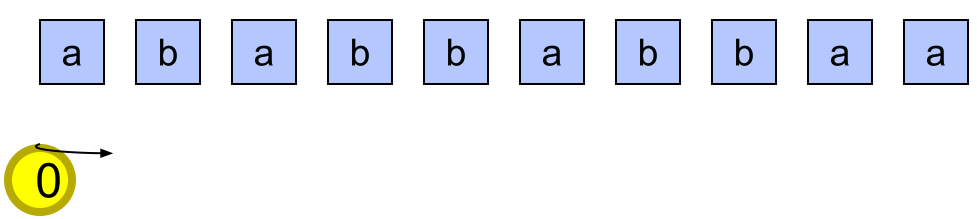
Q is a set of accepting sates

Σ: input alphabet

δ: Q × Σ → Q: transition function

input state	a	b
0	1	0
1	1	2
2	1	3
3	4	0
4	1	2





Example (V)

Q is a finite set of states

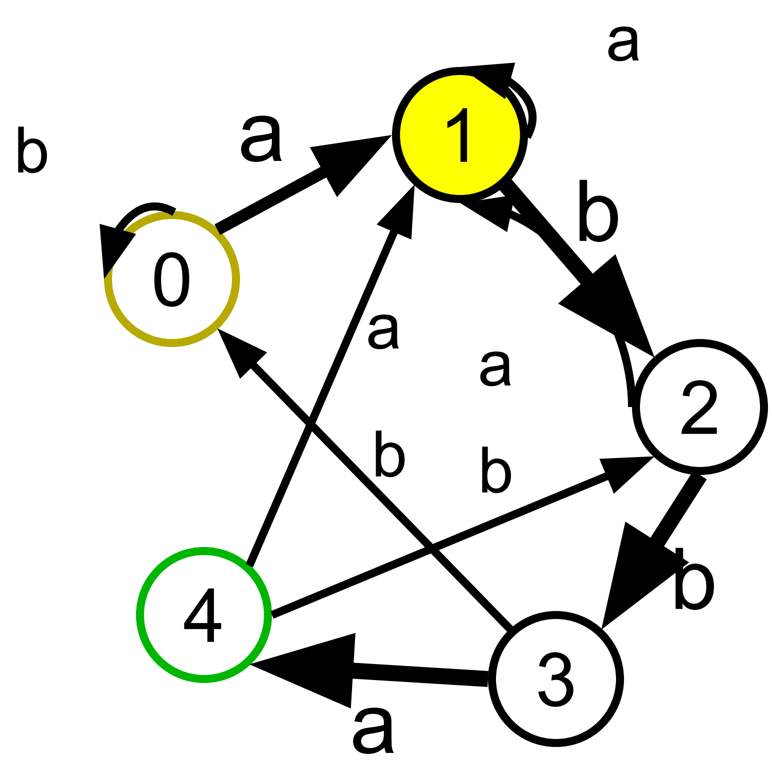
$q_0 \in \mathbf{Q}$ is the start state

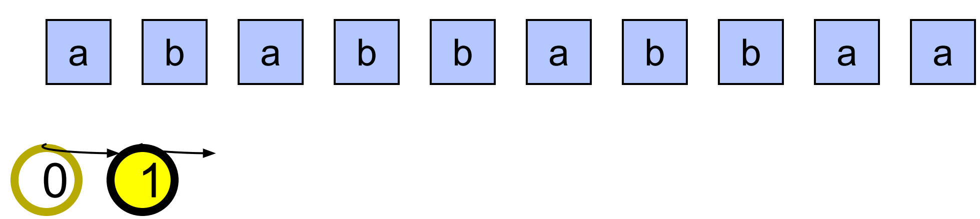
Q is a set of accepting sates

Σ: input alphabet

δ: $Q \times \Sigma \rightarrow Q$: transition function

input state	a	b
0	1	0
1	1	2
2	1	3
3	4	0
4	1	2





Example (VI)

Q is a finite set of states

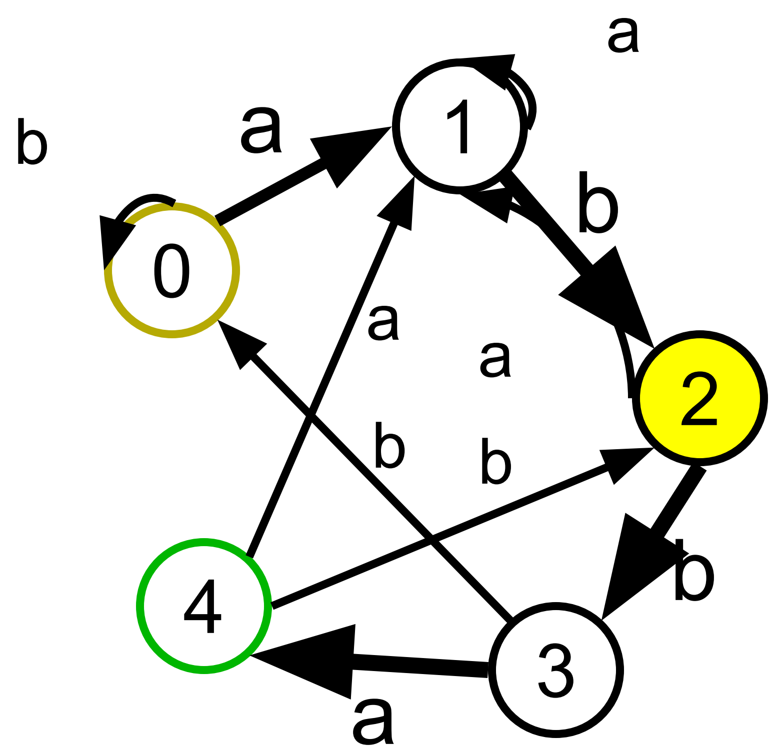
$q_0 \in \mathbf{Q}$ is the start state

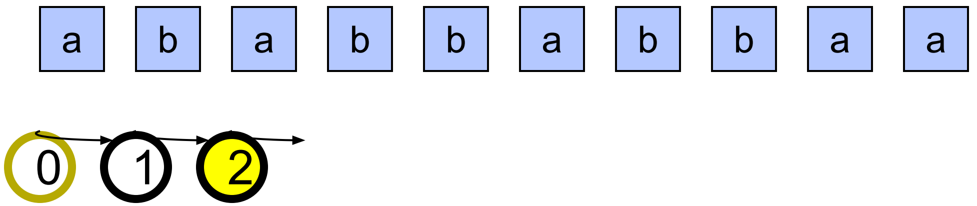
Q is a set of accepting sates

Σ: input alphabet

δ: $Q \times \Sigma \rightarrow Q$: transition function

input / state	a	b
0	1	0
1	1	2
2	1	3
3	4	0
4	1	2





Example (VII)

Q is a finite set of states

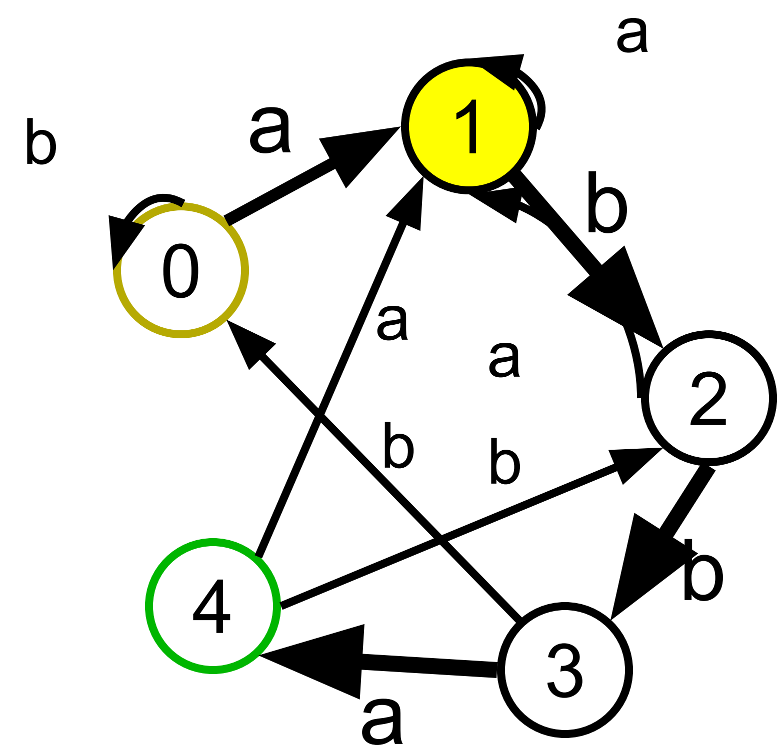
$q_0 \in \mathbf{Q}$ is the start state

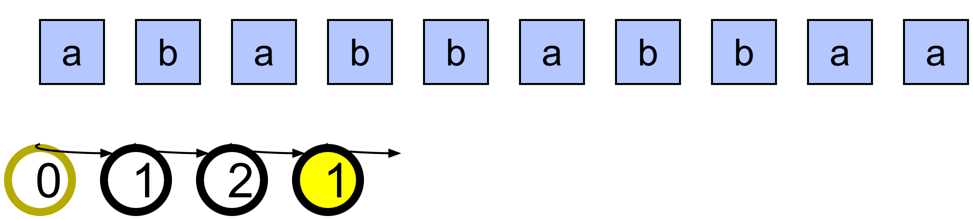
Q is a set of accepting sates

Σ: input alphabet

$\delta: Q \times \Sigma \rightarrow Q$: transition function

input state	a	b
0	1	0
1	1	2
2	1	3
3	4	0
4	1	2





Example (VIII)

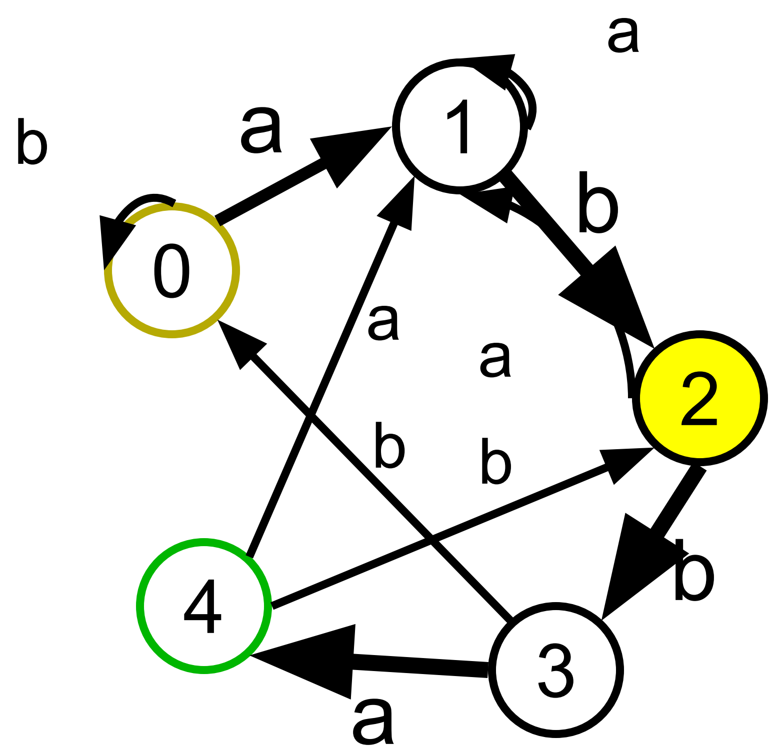
Q is a finite set of states

$q_0 \in \mathbf{Q}$ is the start state

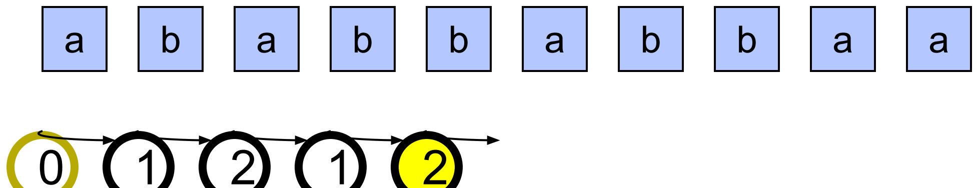
Q is a set of accepting sates

Σ: input alphabet

δ: $Q \times \Sigma \rightarrow Q$: transition function



input / state	a	b
0	1	0
1	1	2
2	1	3
3	4	0
4	1	2



Example (IX)

Q is a finite set of states

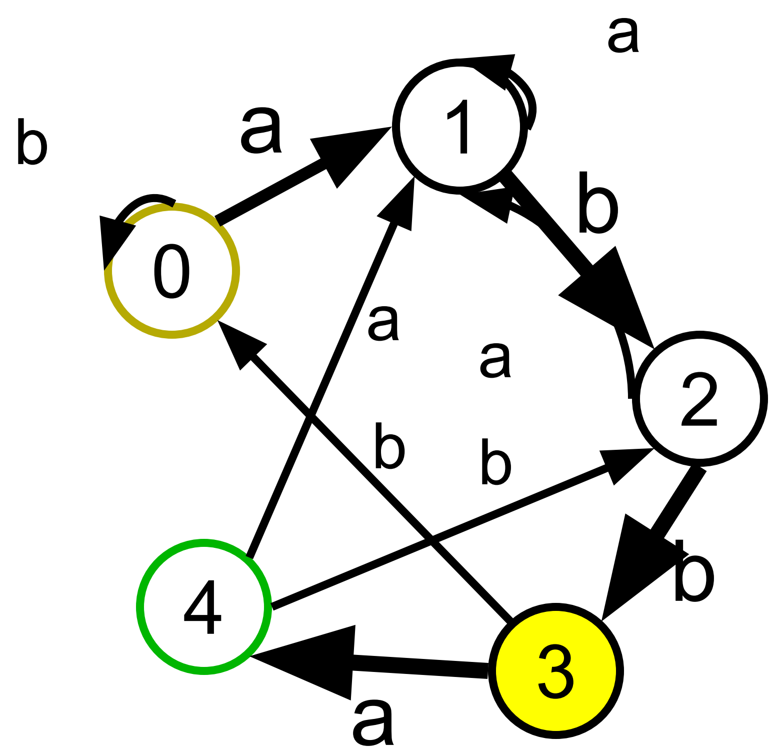
$q_0 \in \mathbf{Q}$ is the start state

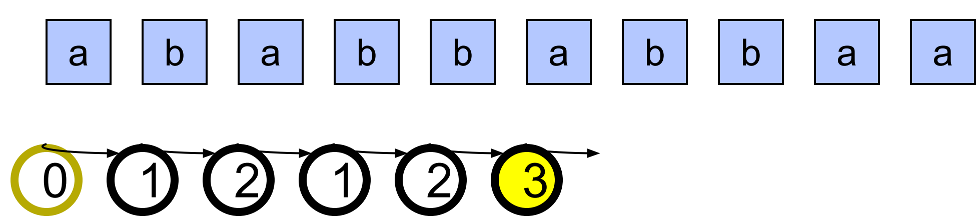
Q is a set of accepting sates

Σ: input alphabet

$\delta: Q \times \Sigma \rightarrow Q$: transition function

input state	a	b
0	1	0
1	1	2
2	1	3
3	4	0
4	1	2





Example (X)

Q is a finite set of states

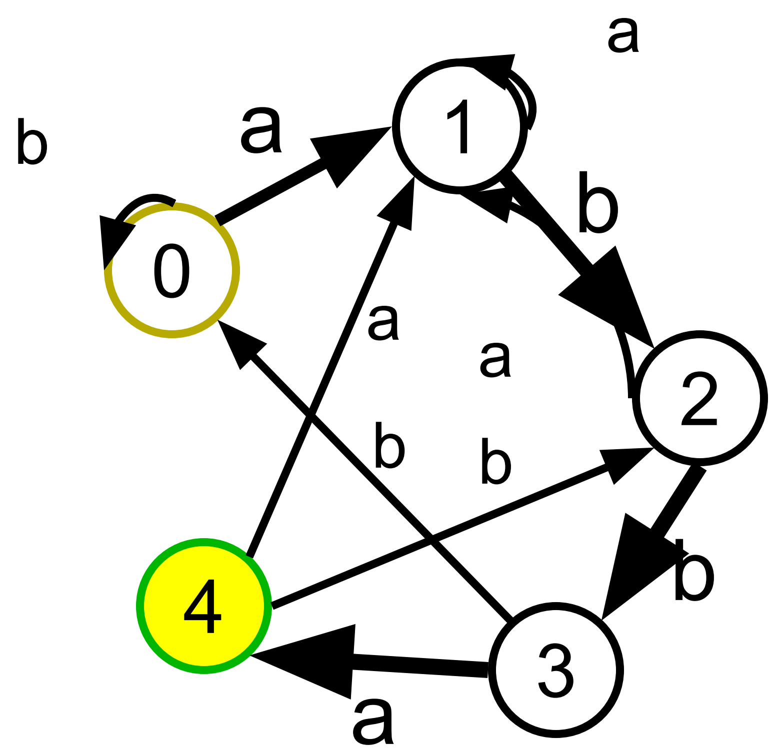
$q_0 \in \mathbf{Q}$ is the start state

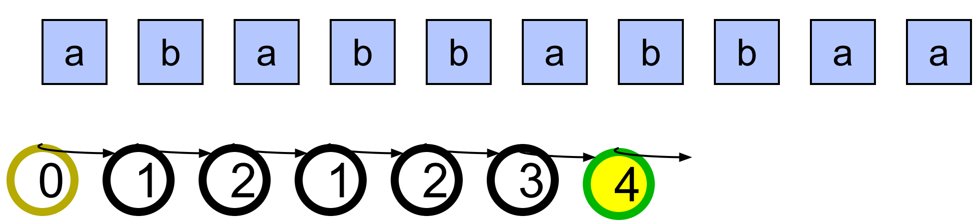
Q is a set of accepting sates

Σ: input alphabet

δ: $Q \times \Sigma \rightarrow Q$: transition function

input / state	a	b
0	1	0
1	1	2
2	1	3
3	4	0
4	1	2





Example (XI)

Q is a finite set of states

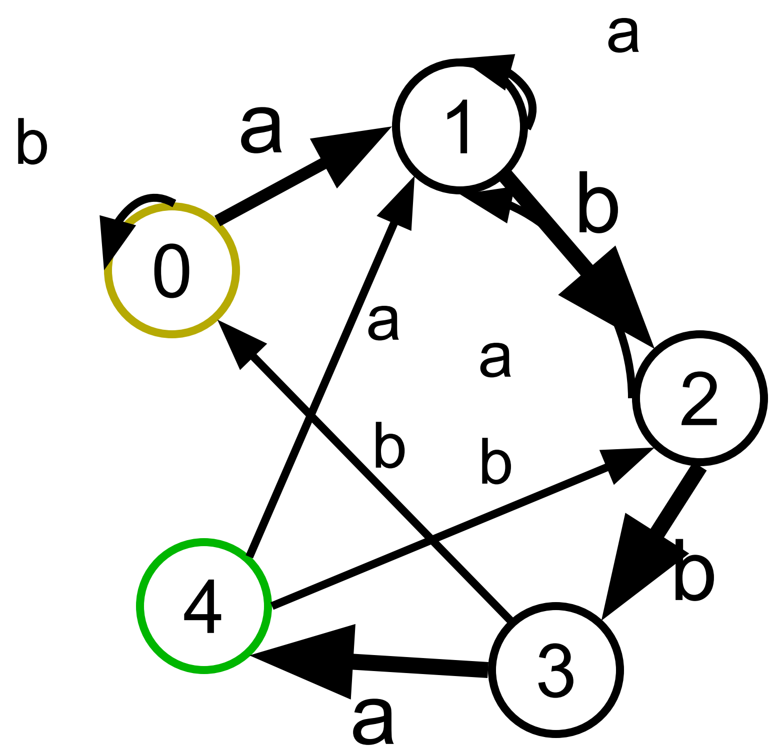
$q_0 \in \mathbf{Q}$ is the start state

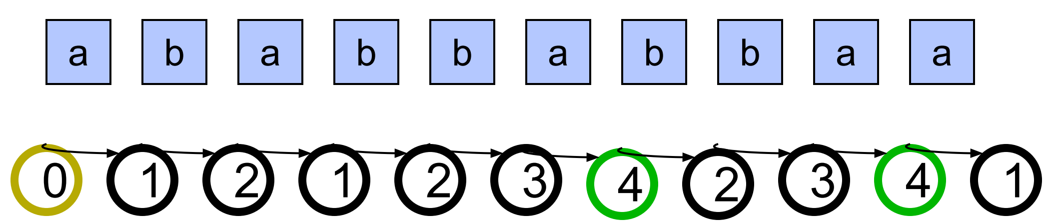
Q is a set of accepting sates

Σ: input alphabet

δ: $Q \times \Sigma \rightarrow Q$: transition function

input state	a	b
0	1	0
1	1	2
2	1	3
3	4	0
4	1	2





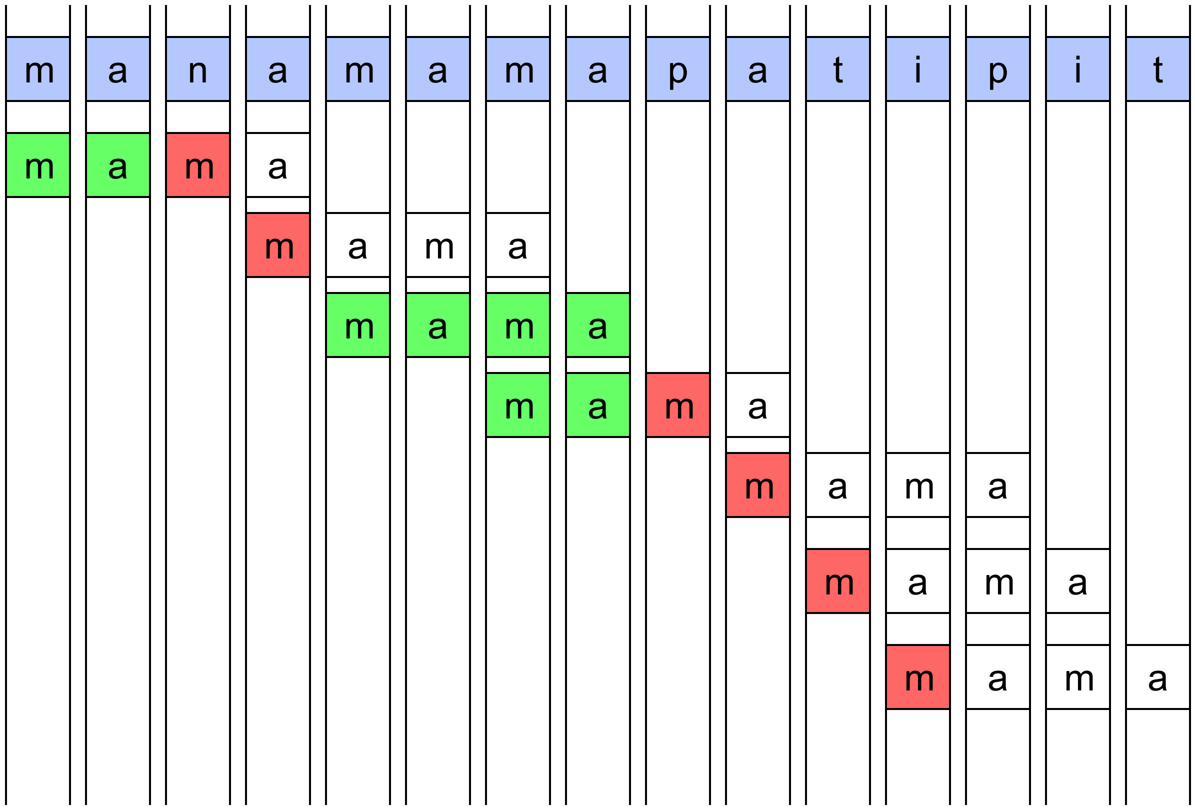
Finite-Automaton-Matcher

- **The example automaton accepts at the end of occurrences of the pattern abba**
- **For every pattern of length m there exists an automaton with m+1 states that solves the pattern matching problem with the following algorithm:**

Finite-Automaton-Matcher(T,δ,P)

1. n ← length(T)
2. q ← 0
3. for i ← 1 to n do
4.   q ← δ(q,T[i])
5.   if q = m then
6.   s ← i - m
7.   return "Pattern occurs with shift" s

Computing the Transition Function: The Idea!



How to Compute the Transition Function?

- A string *u* is a **prefix** of string *v* if there exists a string *a* such that: *ua* = *v*
- A string *u* is a **suffix** of string *v* if there exists a string *a* such that: *au* = *v*
- Let P_k denote the first k letter string of P

Compute-Transition-Function(P, Σ)

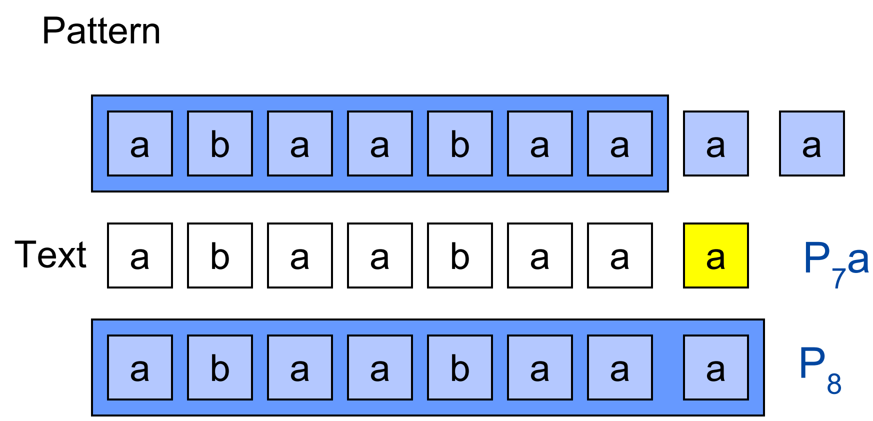
1. **m $\leftarrow$ length(P)**
2. **for q $\leftarrow$ 0 to m do**
3. **for each character a $\in \Sigma$ do**
4. **k $\leftarrow$ 1+min(m,q+1)**
5. **repeat**
   k $\leftarrow$ k-1
6. **until P_k is a suffix of $P_q a$**
7. **δ(q,a) $\leftarrow$ k**

Example

- A string *u* is a **prefix** of string *v* if there exists a string *a* such that: *ua* = *v*
- A string *u* is a **suffix** of string *v* if there exists a string *a* such that: *au* = *v*
- Let P_k denote the first k letter string of P

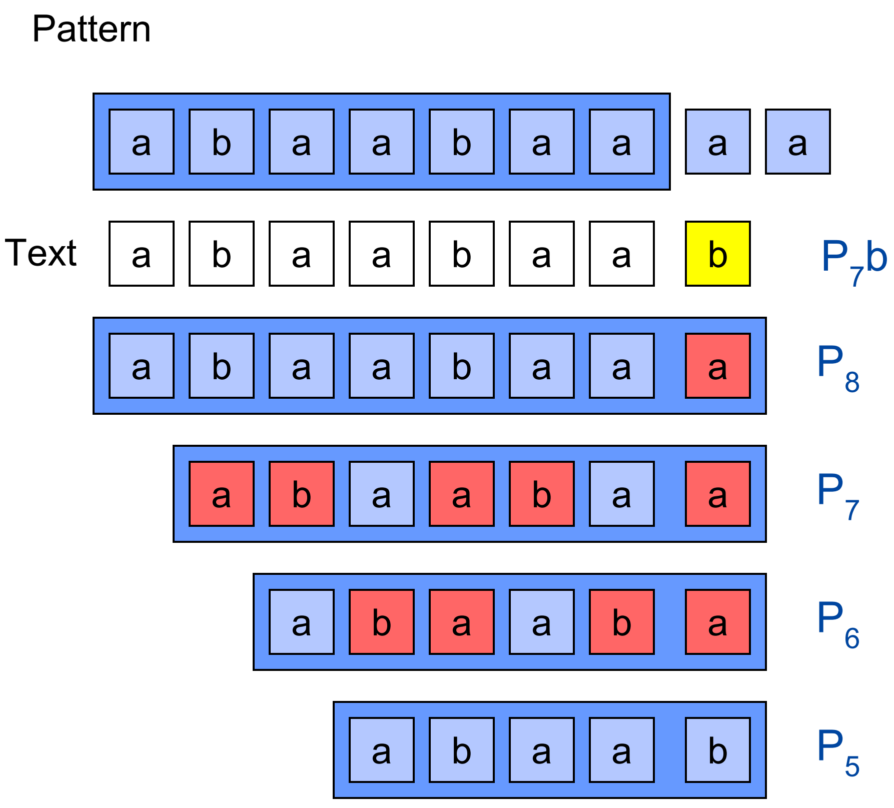
Compute-Transition-Function(P, Σ)

```
m ← length(P)
for q ← 0 to m do
  for each character a ∈ Σ do
        k ← 1+min(m,q+1)
        repeat
              k ← k-1
        until P_k is a suffix of P_q a
         δ(q,a) ← k
```



Example

- A string *u* is a **prefix** of string *v* if there exists a string *a* such that: *ua* = *v*
- A string *u* is a **suffix** of string *v* if there exists a string *a* such that: *au* = *v*
- Let P_k denote the first k letter string of P



Compute-Transition-Function(P, Σ)

1. **m ← length(P)**
2. **for q ← 0 to m do**
3. **for each character a ∈ Σ do**
4. **k ← 1+min(m,q+1)**
5. **repeat**
k ← k-1
6. **until P_k is a suffix of P_qa**
7. **δ(q,a) ← k**
8.

Running time of Compute Transition-Function

- A string *u* is a **prefix** of string *v* if there exists a string *a* such that: *ua* = *v*
- A string *u* is a **suffix** of string *v* if there exists a string *a* such that: *au* = *v*
- Let P_k denote the first k letter string of P

```
Compute-Transition-Function(P, Σ)
m ← length(P)
for q ← 0 to m do
  for each character a ∈ Σ do
        k ← 1+min(m,q+1)
        repeat
            k ← k-1
        until P_k is a suffix of P_q a
         δ(q,a) ← k
```

Factor: m+1

Factor: |**Σ**|

Factor: m

Time for check of equality: m

Running time of procedure: $O(m^3 |\Sigma|)$