Advanced Algorithm

String Matching

String Matching Using Finite Automata

Example: {$\alpha \in \Sigma^* | \alpha$ is a binary number divisible by 4 }
(One Accepting State)

(0+1)*00

- Accepted

00
000
100
0000
0100
1000
1100

- Rejected

0
10
010
110

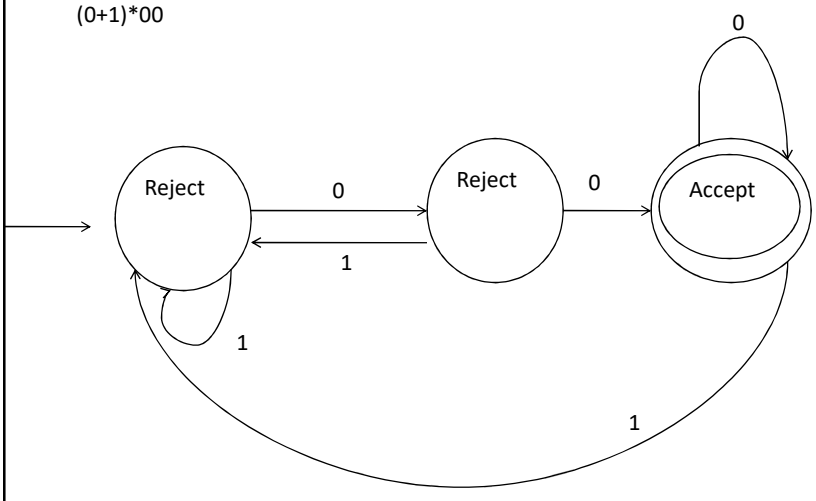
- Rejected

^
1
01
001
101

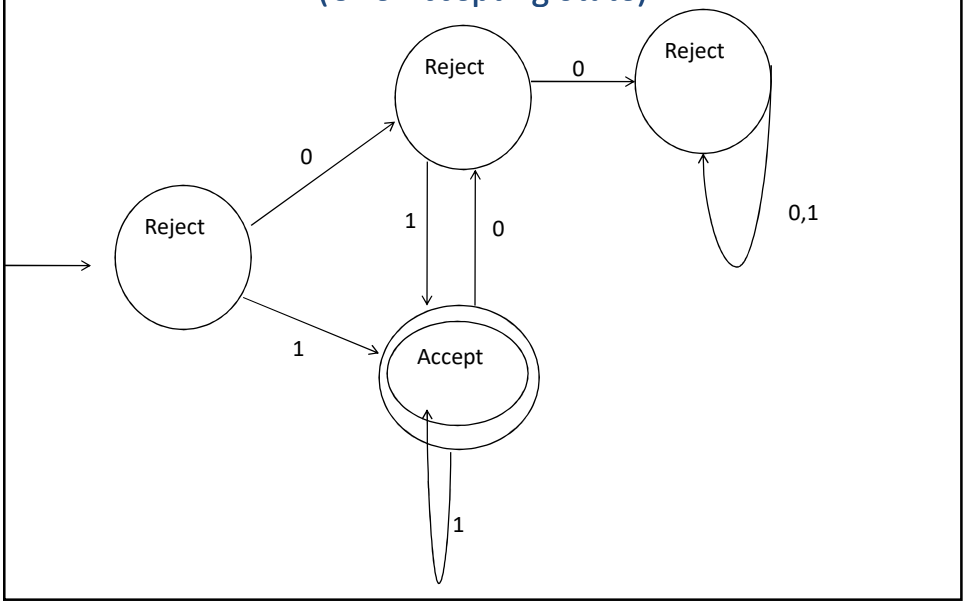
How many input symbols are required to change the state from “Accept” to “Reject” / “Reject” to “accept” ?

Example: {$\alpha \in \Sigma^* | \alpha$ is a binary number divisible by 4 }
(One Accepting State)

(0+1)*00



Example: String ending with '1' and doesn't contain 00 (One Accepting State)



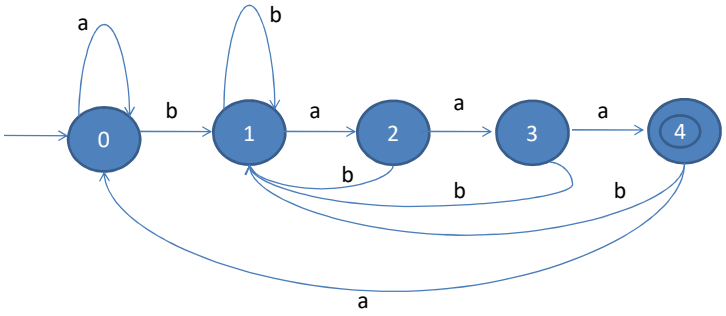
String Matcher using Finite Automata

Pattern : baaa

Prefix	Suffix
Null(ε)	Null(ε)
b	**a**
ba	**aa**
baa	**aaa**
baaa	baaa

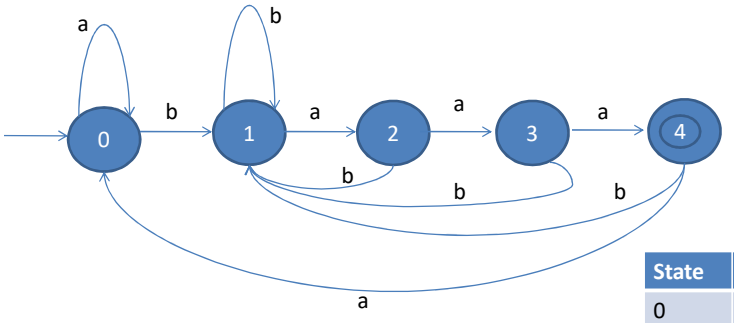
String Matcher using Finite Automata

Pattern : baaa



String Matcher using Finite Automata

Pattern : baaa



δ

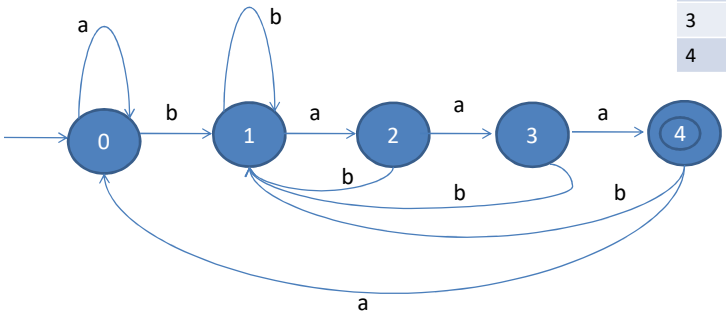
State	a	b
0	0	1
1	2	1
2	3	1
3	4	1
4	0	1

String Matcher using Finite Automata

Pattern : baaa

State	a	b
0	0	1
1	2	1
2	3	1
3	4	1
4	0	1

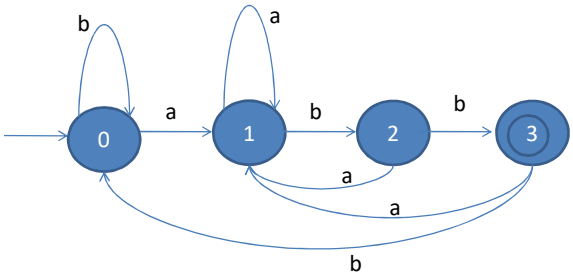
δ



T[i]	a	b	a	b	a	a	a	a
0	0	1	2	1	2	3	4	0

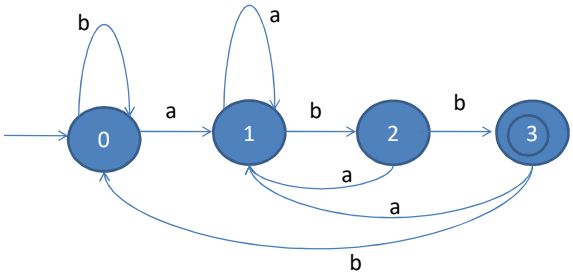
String Matcher using Finite Automata

Pattern : abb



String Matcher using Finite Automata

Pattern : abb

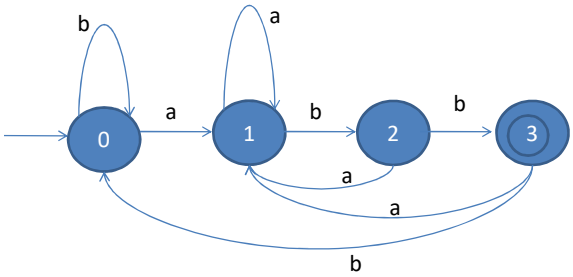


δ

State	a	b
0	1	0
1	1	2
2	1	3
3	1	0

String Matcher using Finite Automata

Pattern : abb



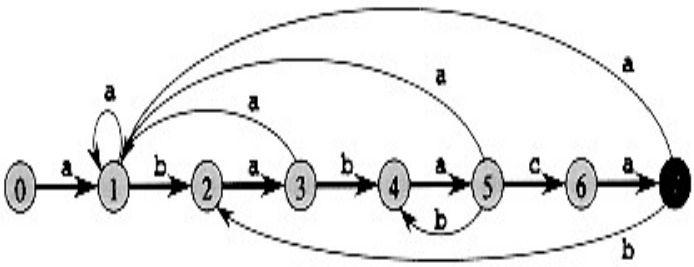
State	a	b
0	1	0
1	1	2
2	1	3
3	1	0

δ

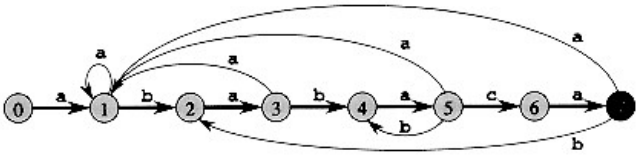
T[i]	a	b	a	b	b	a	b	b
0	1	2	1	2	3	1	2	3

String Matcher using Finite Automata

Pattern : abababaca

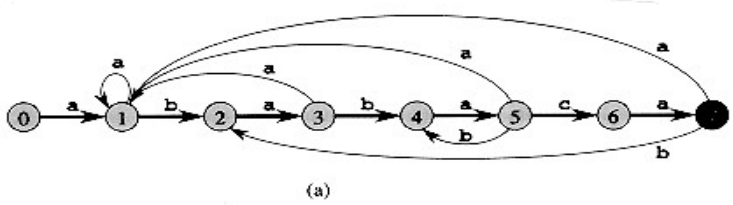


String Matcher using Finite Automata



	input			
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	

String Matcher using Finite Automata



(a)

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	

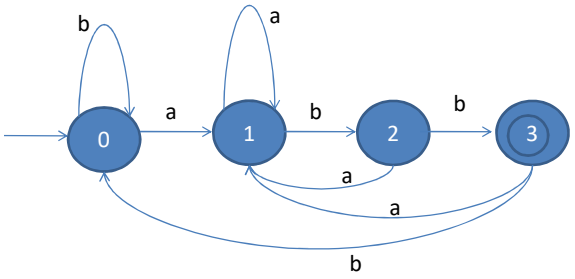
(b)

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
state $\phi(T_i)$	0	1	2	3	4	5	4	5	6	7	2	3

(c)

String Matcher using Finite Automata

Pattern : abb



State	a	b
0	1	0
1	1	2
2	1	3
3	1	0

δ

δ

T[i]	a	b	a	b	b	a	b	b
0	1	2	1	2	3	1	2	3

String Matcher using Finite Automata

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

T[i]	a	b	a	b	b	a	b	b
0	1	2	1	2	3	1	2	3

δ

State	a	b
0	1	0
1	1	2
2	1	3
3	1	0

String Matcher using Finite Automata

Pattern : abba

- Construct Finite Automata for above pattern using prefix and suffix concept
- Using Finite Automata, extract transition table 'δ'
- Match the pattern with the below given string

bbaabbabbbaaabba