Finite Automata (String Matching) → machine to find pattern

⇒ Time Complexity $O(n)$

because each character will be checked exactly once

⇒ It has a transition table

each table have 5 items

Q = set of states

Σ = set of alphabets

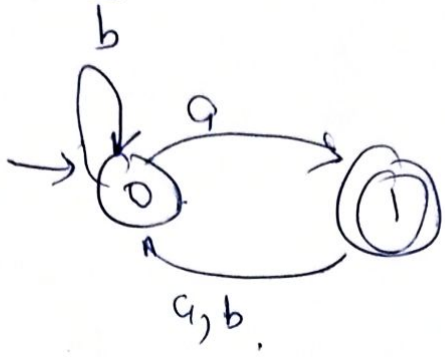
δ = transition mapping (b/w $Q \times \Sigma$)

$q_0 \in Q$ = initial state

$F \subseteq Q$ = final states

Initial state final state

Assume



State = Q = 1, 2

Σ, alphabet = a, b

$\delta(0, a) \rightarrow 1$

$\delta(1, a) \rightarrow 0$

} Transition example)

$q_0 = \{0\}$

$F = \{1\}$

Transition table

State ⟶

	a	b
0	1	0
1	0	0

⇒ Remember prefix & Suffix.

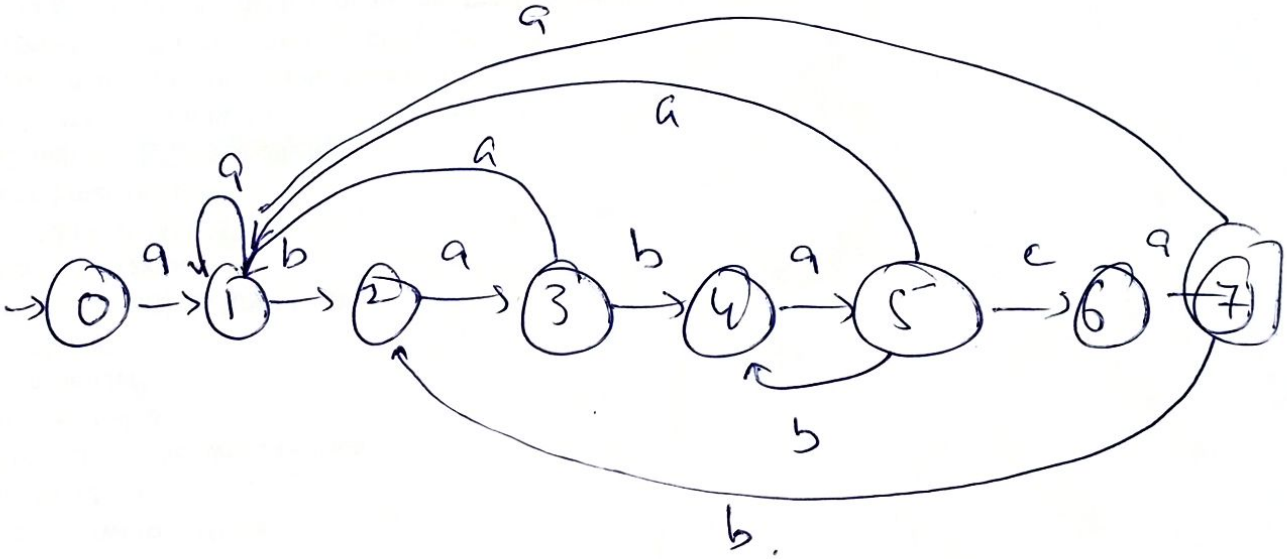
abcdabc.

a, ab, abc all are prefix & Suffix.

Now

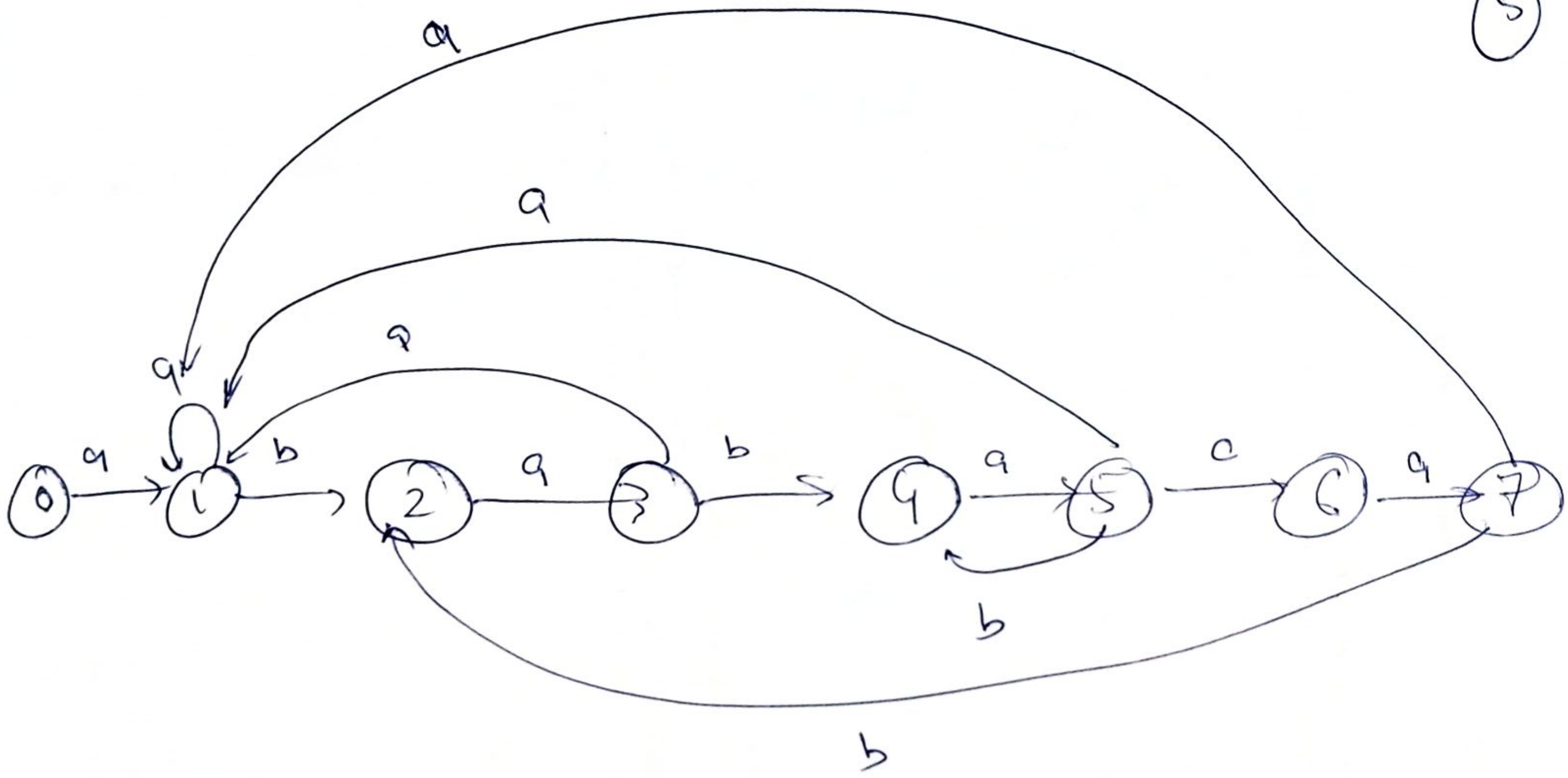
T = a b a b a b a c a b a

P = a b a b a c a



⇒ Transition Table

State	a	b	c	Pattern
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	


0
1
2
3
4
5
6
7
a
b
c

Naive

Algo

Transition Table

Text, pattern size

finite auto [T, δ, m]

n = length(T)

zero state // ← q = 0

for i ← 1 to n

do q = δ(q, T[i]) ⇒ 1

if q = m

print found at i − m + 1

⇒ match with transition table

T = a b a b a b a c a b a

state 1 2 3 4 5 4 5 6 7 2 3

q

eg

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

~~O(m)~~ $O(m|\Sigma|)$

↓

no of characters

e.g. P = n a n o

T = b a n a n a h o n a

0 0 1 2 3 2 3 4 1 2.

	n	a	o
0	(1)	0	0
1	1	(2)	0
2	(3)	0	0
3	1	2	(4)
4	1	0	0