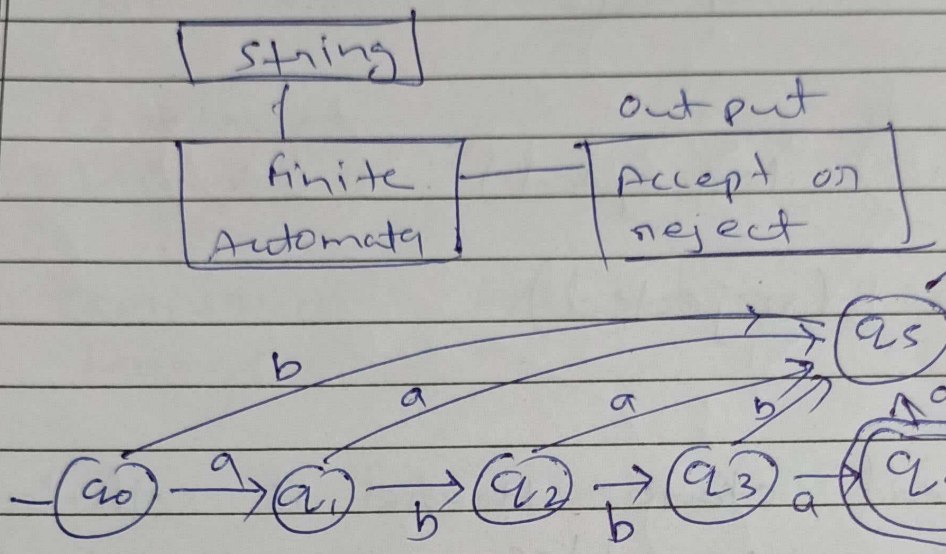
Page :
Date : / /

Finite Automation



Transition function :-

$\delta(q_0, a) \rightarrow q_1$

$\delta(q_0, b) \rightarrow q_5$

$\delta(q_4, a) \rightarrow q_5$

Page :
Date : / /

Dead state :-

In a perfect real machine dead state is must

a b b a (accepted)

a b a (rejected)

In dead state there is self loop of all variables.

If initial state is final state Null inputs is acceptable

Language is set of all acceptable string.

Page :
Date : / /

Construction of DFA

(i) Extend the language starting with smallest string.

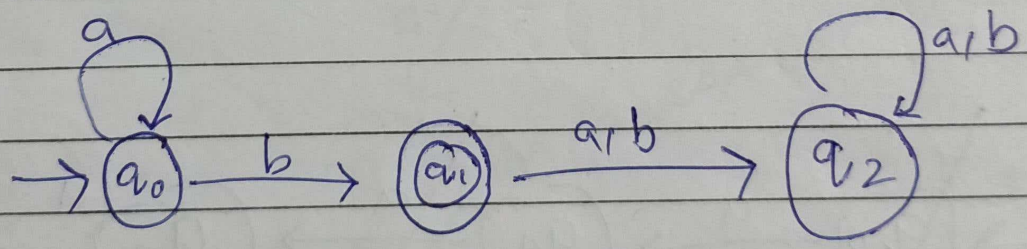
(ii) Construct FA for the smallest string.

(iii) Construct FA for remaining strings.

(iv) Convent FA into DFA

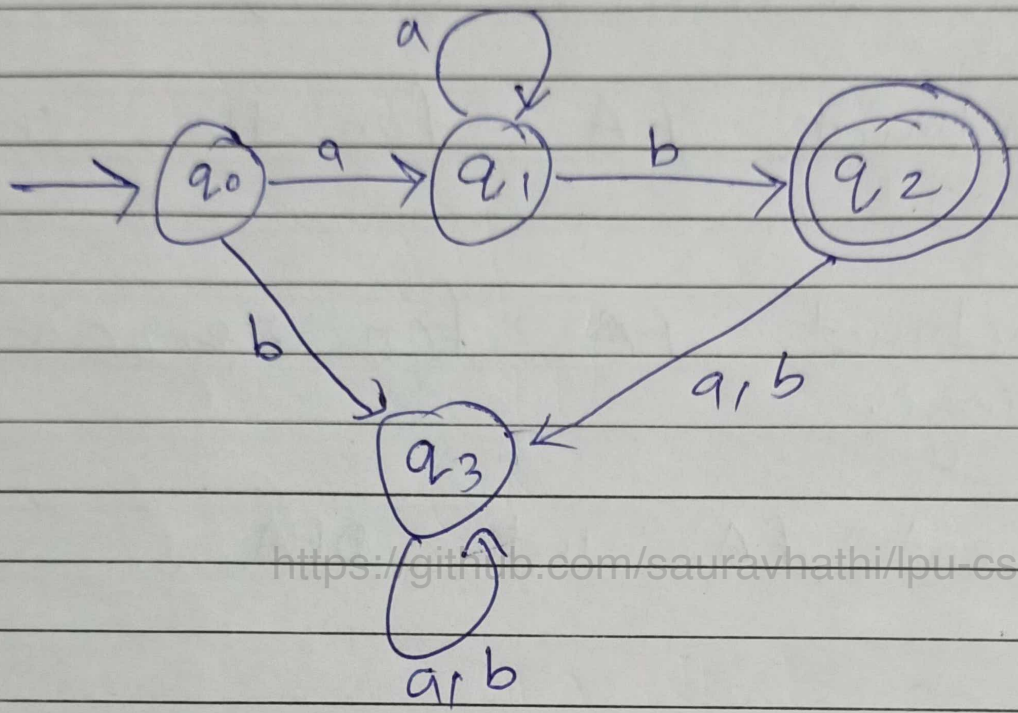
$L = a^n . b \quad ; \quad n \geq 0$

$= \{ b, ab, aab, aaab \ldots \ldots \}$



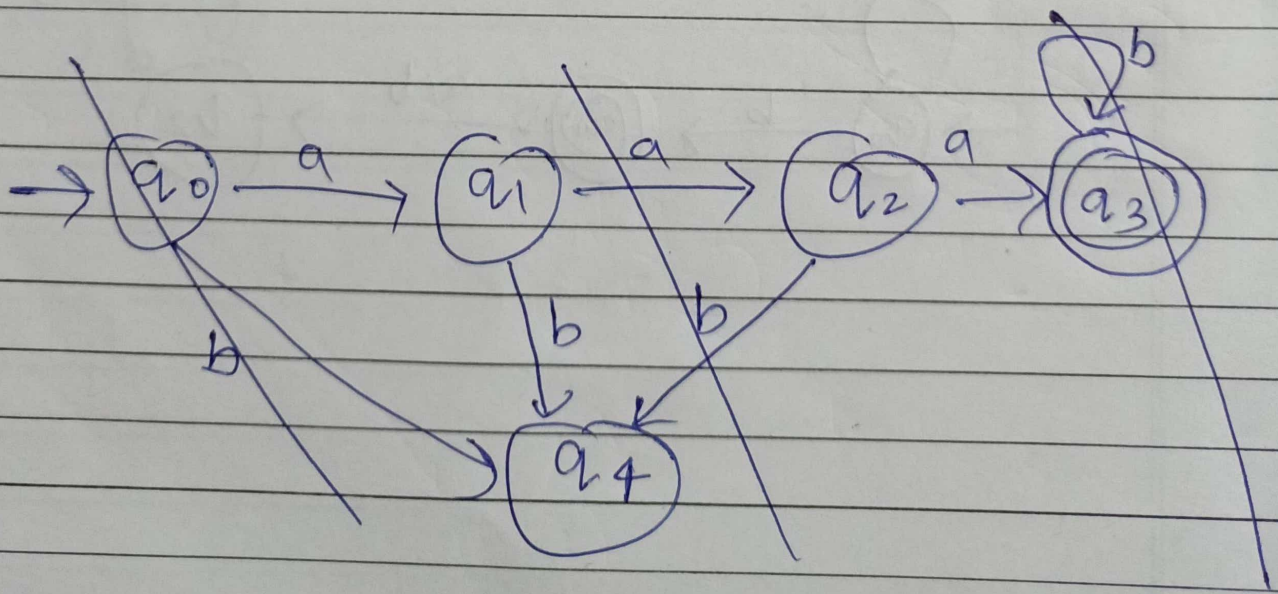
$L = a^n b \quad ; n > 0$

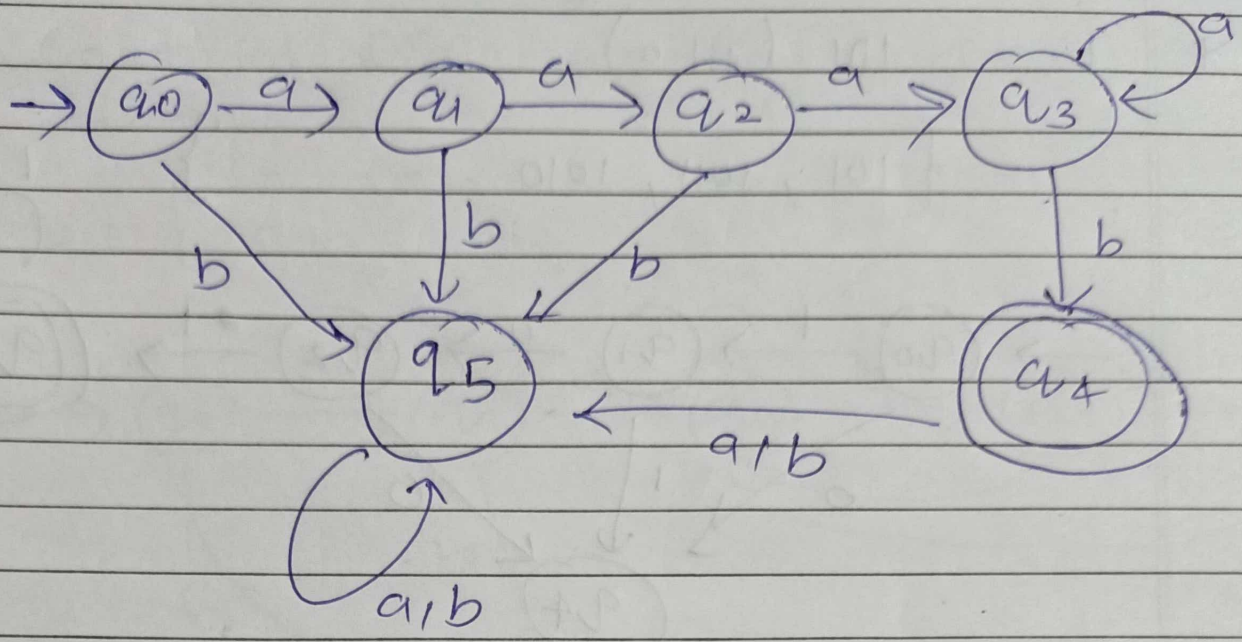
$= \{ab, aab, aaab \cdots\}$



$L = a^n b \quad ; n > 2$

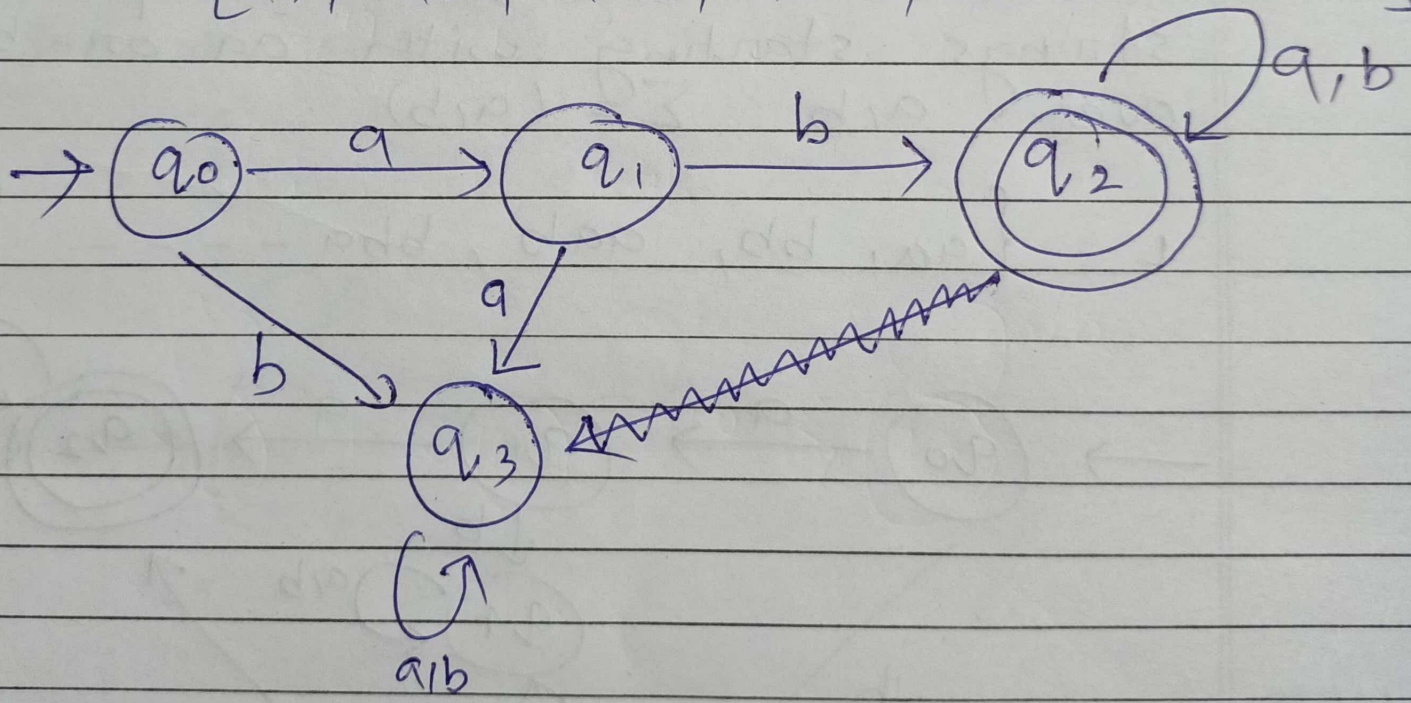
$= \{aaab, aaaab \cdots\}$





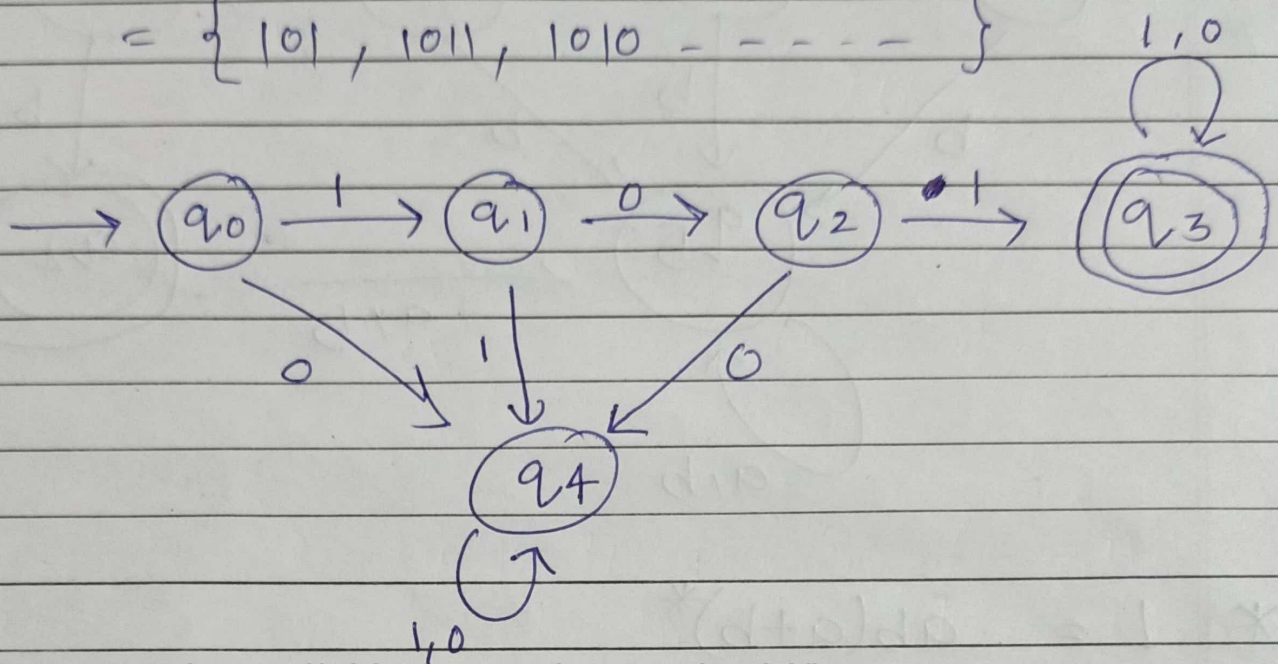
* $L = ab(a+b)^*$

$= \{ab, aba, abb, abaa, abab \text{ ---- }\}$



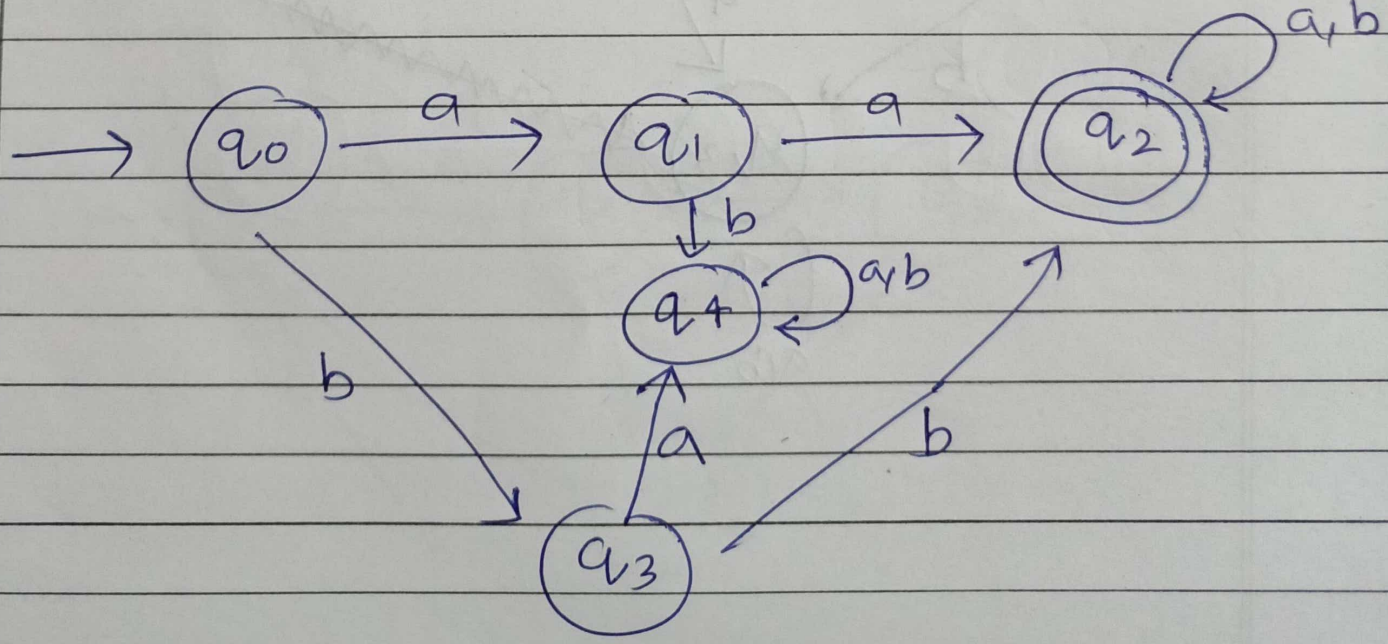
* $L = 101(1+0)^*$

$= \{101, 1011, 1010 ----- \}$



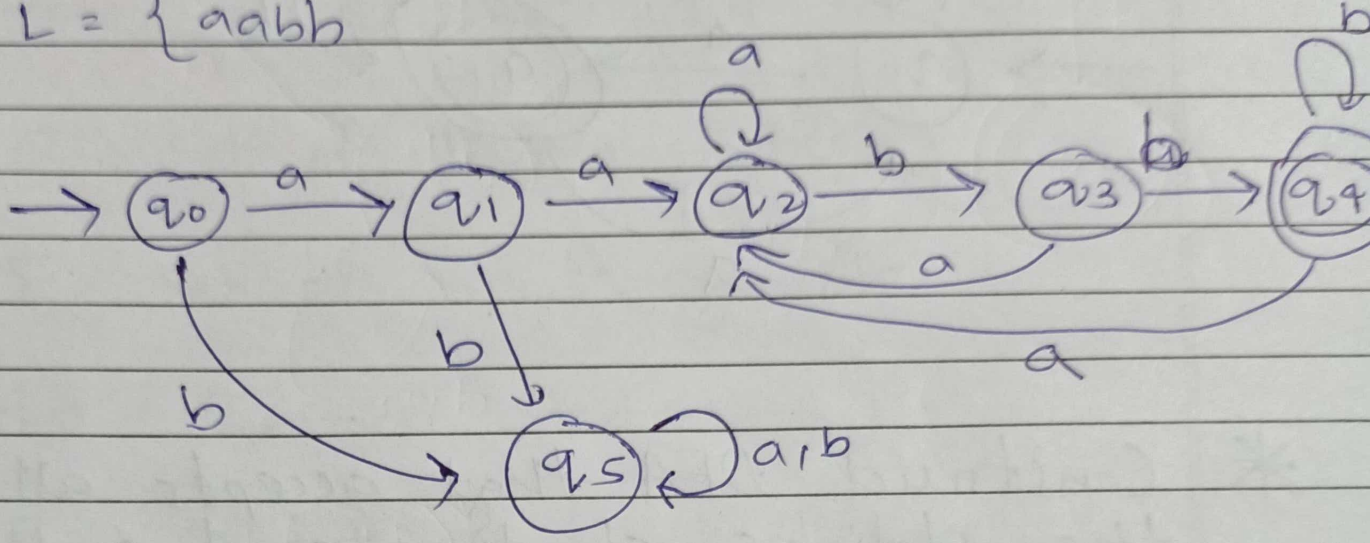
* Construct a DFA that accepts all strings starting with aa or bb over a,b $\Sigma = (a,b)$

$L = \{aa, bb, aab, bba ----- \}$



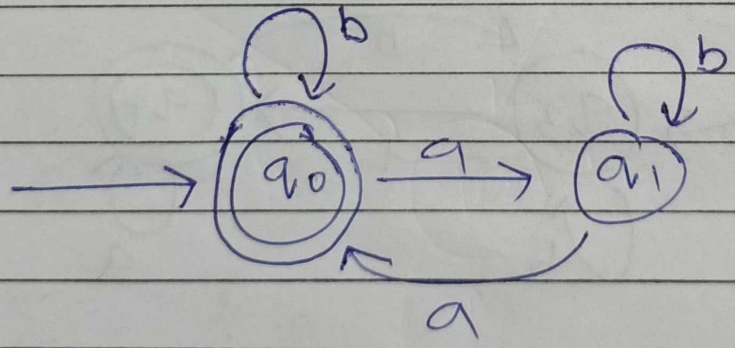
* Construct DFA starting with aa ending with bb

L = {aabb

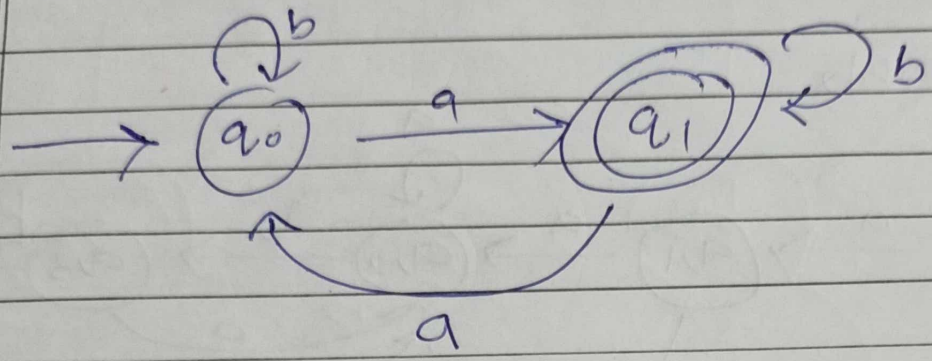


* Construct DFA that accepts even no of a's

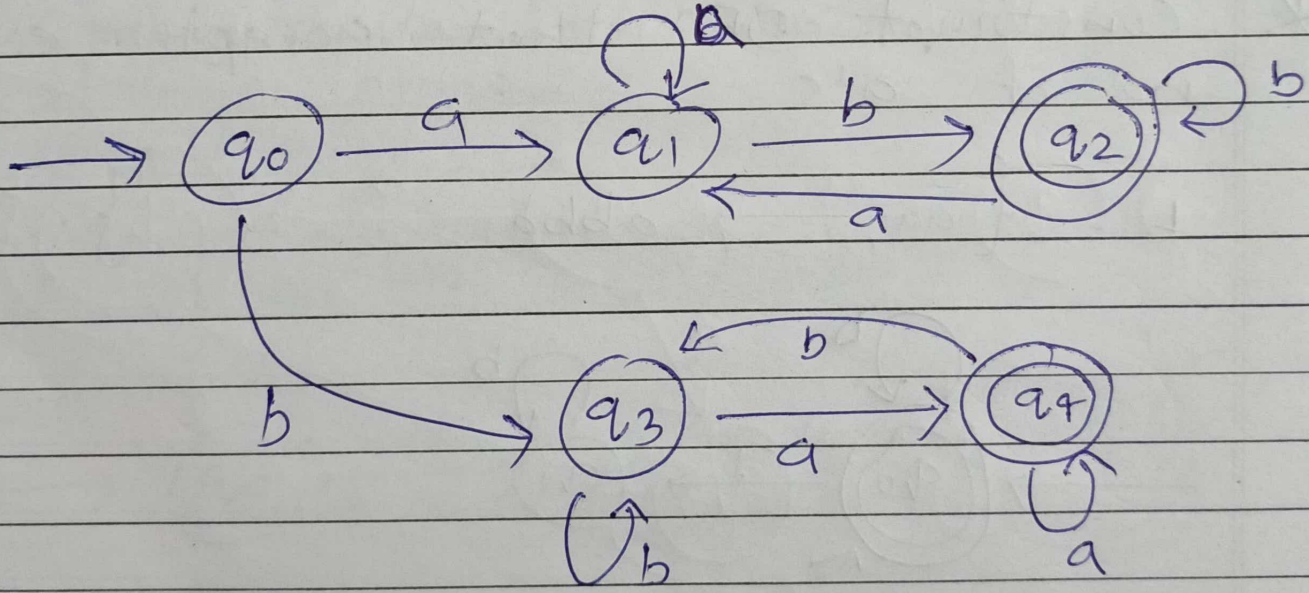
L = { aa, , abba ---- }



* Construct DFA that accepts odd no of a's.



* Construct DFA that accepts all the strings starting and ending with different characters.



* Non-Deterministic Finite Automata (NFA)

There are 5 tupples namely Q Σ δ q_0 F

Q : Set of all the states

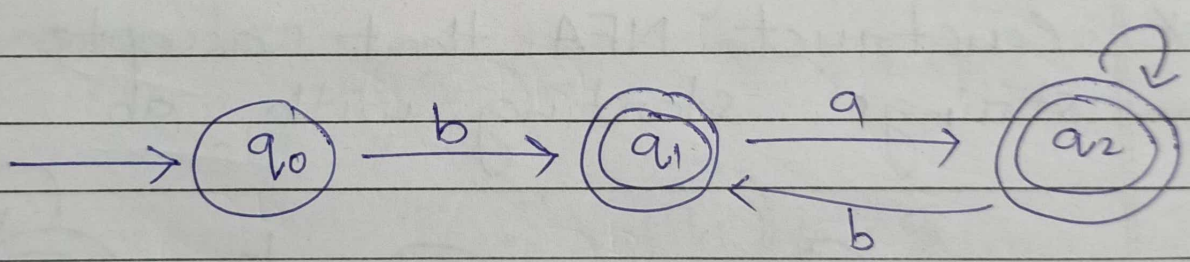
Σ : Set of input symbols

q_0 : Initial state

F : A set of final states

δ : Transition function, defined as

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

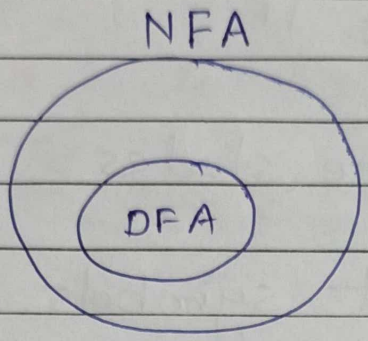


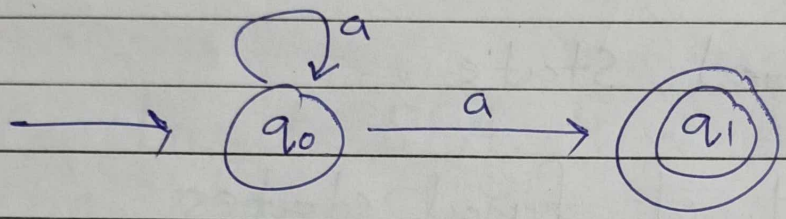
$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$ $\quad q_0 = q_0$

$F = \{q_1, q_2\}$

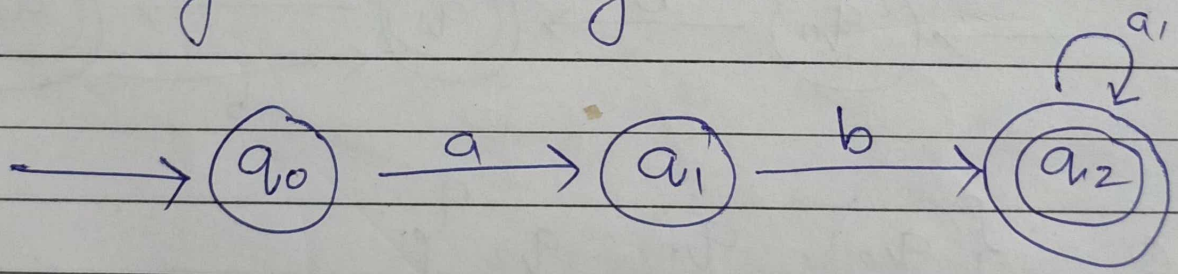
Note ⇒ Every DFA is NFA but every NFA is not DFA



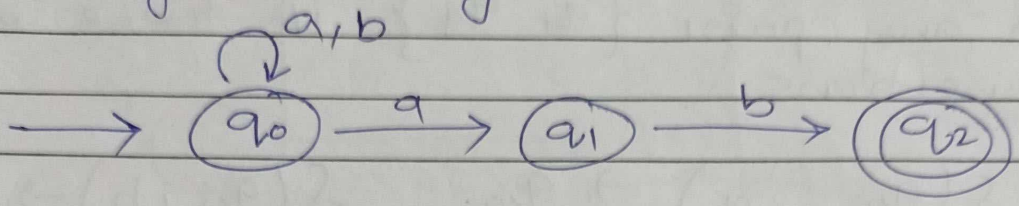


Construction of NFA :-

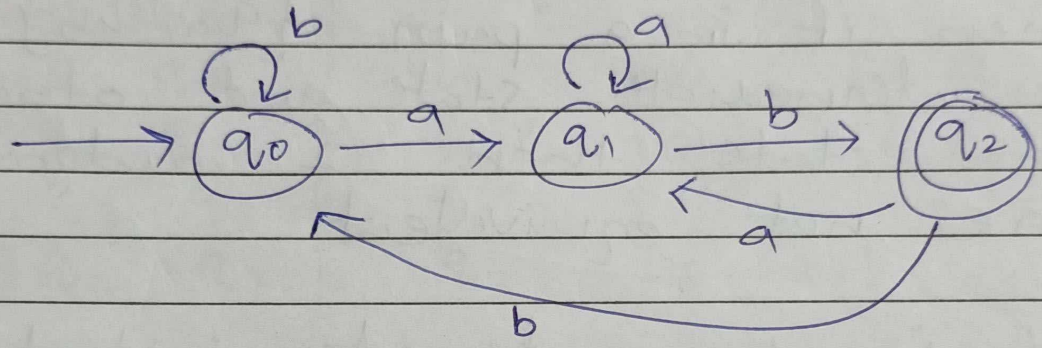
* Construct NFA that accepts all strings starting with ab



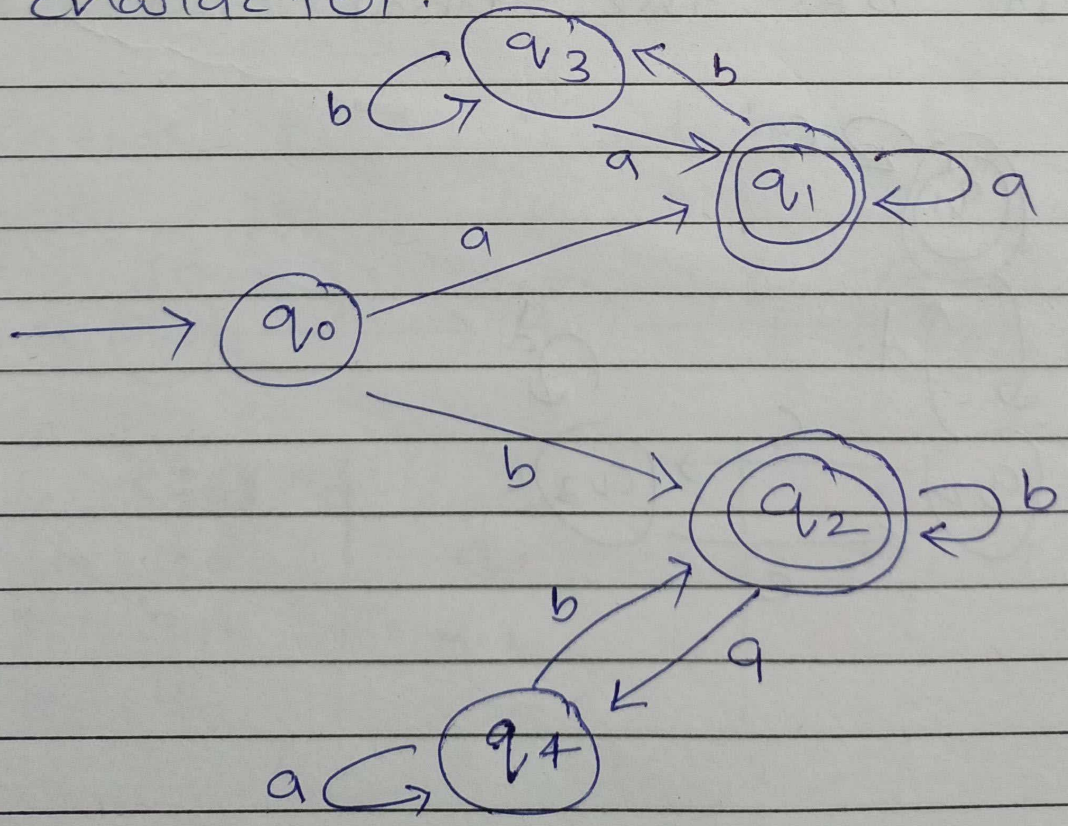
* Construct NFA that accepts all the strings ending with ab



DFA



* Construct DFA: starting, ending same character.



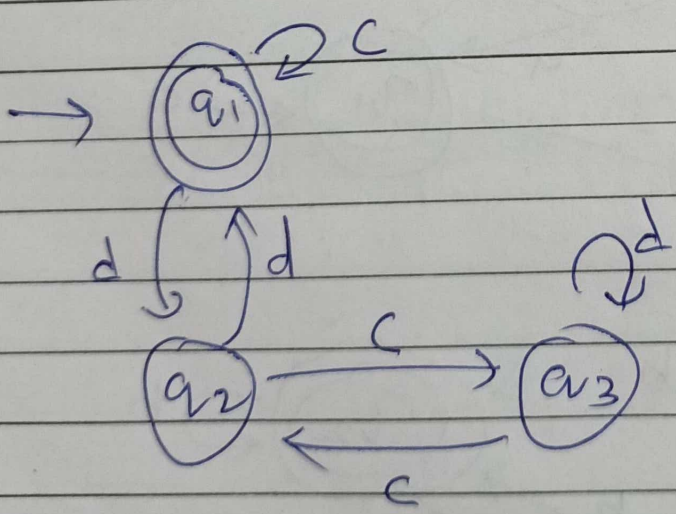
Equivalence of NFA and DFA :-

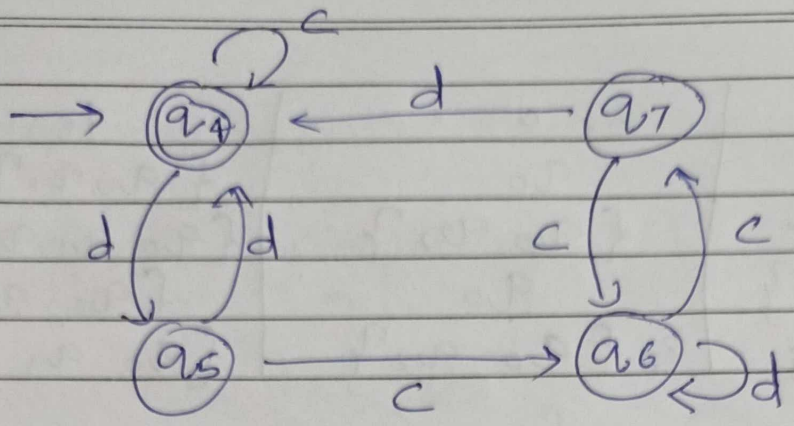
→ For any pair $\{q_i, q_j\}$ if there exist transitions

$$\delta(q_i, a) \to q_a \qquad \delta(q_j, b) \to q_b$$

then if in a pair $\{q_a, q_b\}$ one is intermediate state and other is final state then 2 automatas are not equivalent.

→ If in 1 automata initial state is the final state then in 2nd automata also initial state must be the final state.



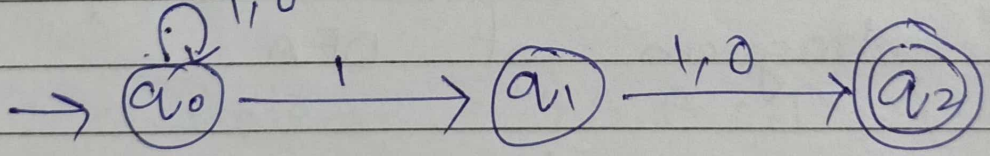


states	c		d	
q_1 q_4	q_1	q_4	q_2	q_5
q_2 q_5	q_3	q_6	q_1	q_4
q_3 q_6	q_2	q_7	q_3	q_6
q_2 q_7	q_3	q_6	q_1	q_4

Equivalent automatas

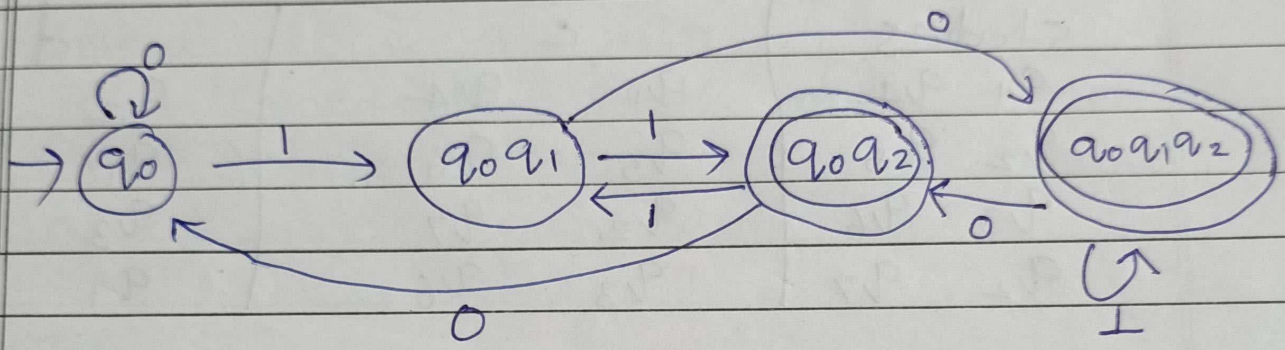
Conversion NFA to DFA :-

* Create NFA : second last bit is 1



State	0	1
→ q_0	q_0	q_0, q_1
q_1	q_2	q_2
q_2	—	—

States	0	1
→ q_0	q_0	$\{q_0 q_1\}$
$\{q_0 q_1\}$	$\{q_0 q_2\}$	$\{q_0 q_1 q_2\}$
$\{q_0 q_2\}$	q_0	$\{q_0 q_1\}$
$\{q_0 q_1 q_2\}$	$\{q_0 q_2\}$	$\{q_0 q_1 q_2\}$



Minimization of DFA :-

Process -

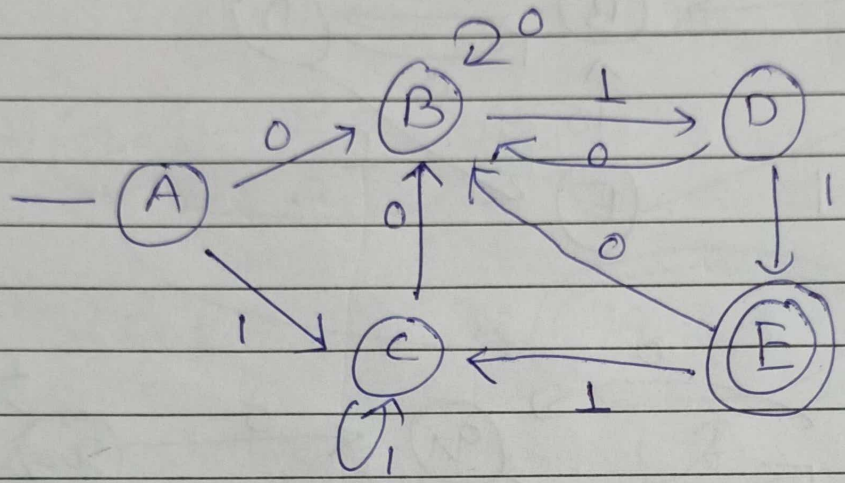
→ Make the transition table from state diagram of DFA

→ Make 0-equivalence by making 2 pairs one for final state and one for non-final state.

→ Try separating the above pairs

according to their equivalencies.

→ Repeat step 3 until 2 equivalencies become same.



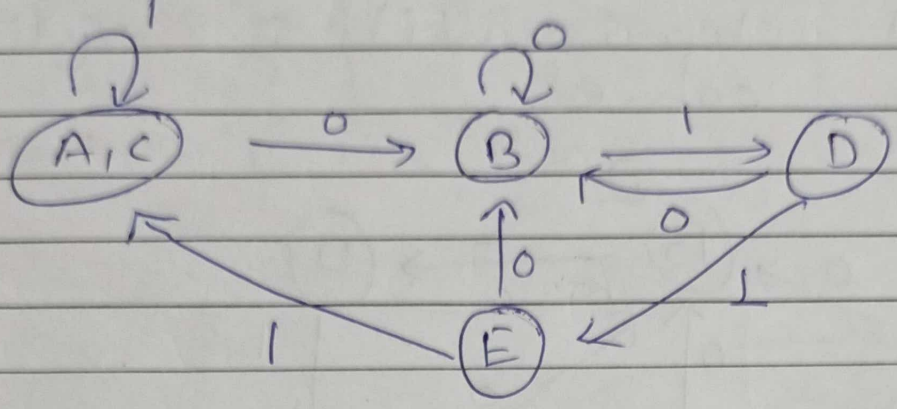
state	0	1
→ A	B	C
B	B	D
C	B	C
D	B	E
(E)	B	C

0-equivalence = {A, B, C, D} {E}

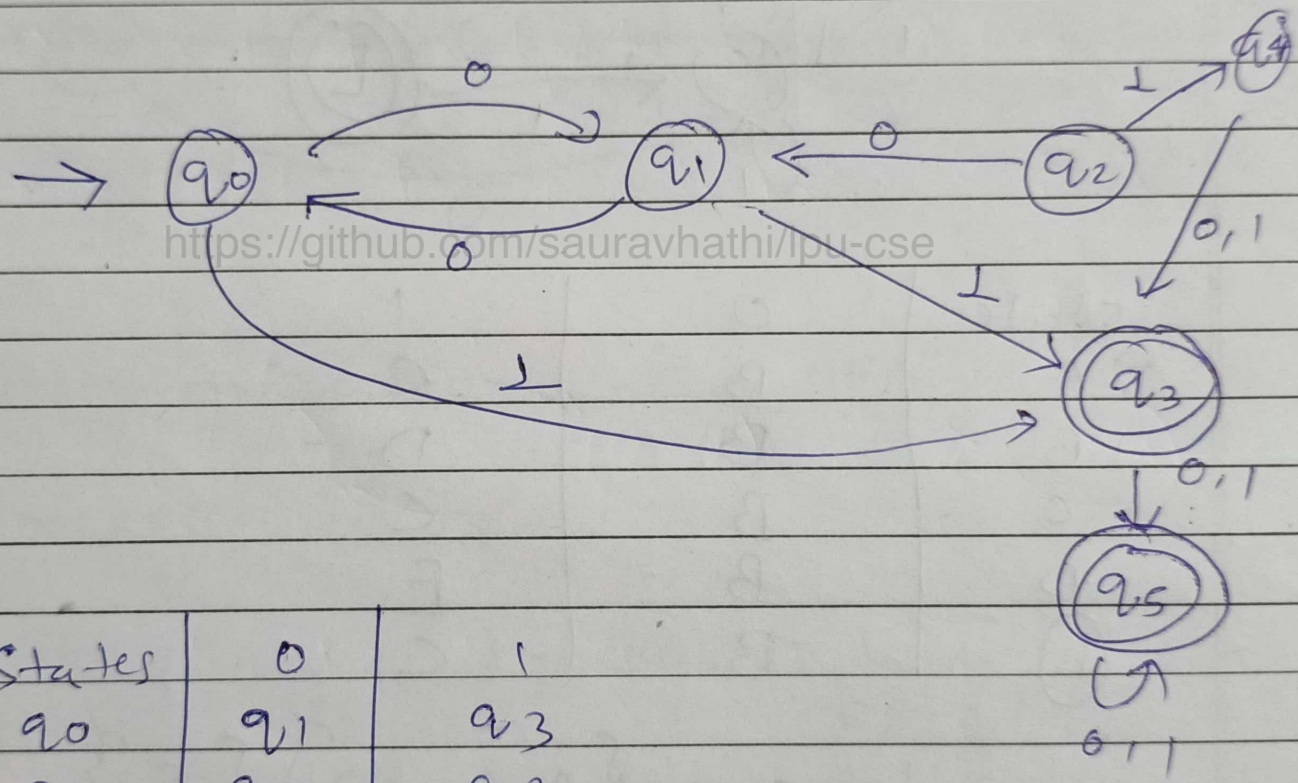
1-equivalence = {A, B, C} {D} {E}

2-equivalence = {A, C} {B} {D} {E}

3-equivalence = {A,C}{B}{D}{E}



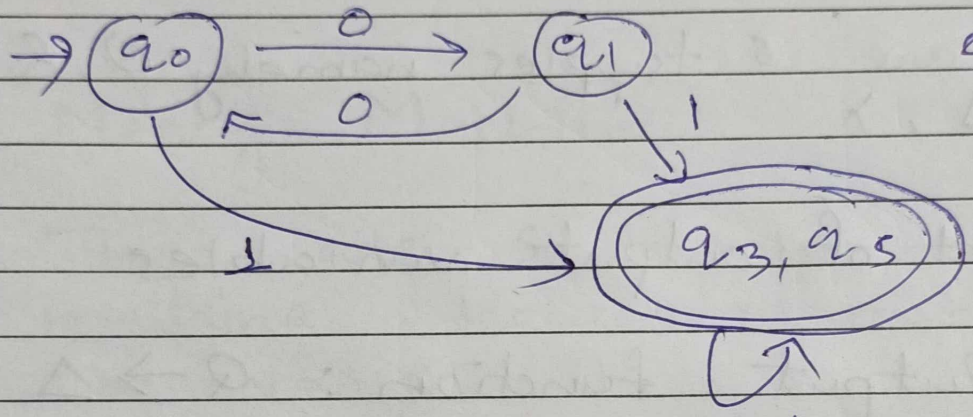
(c)



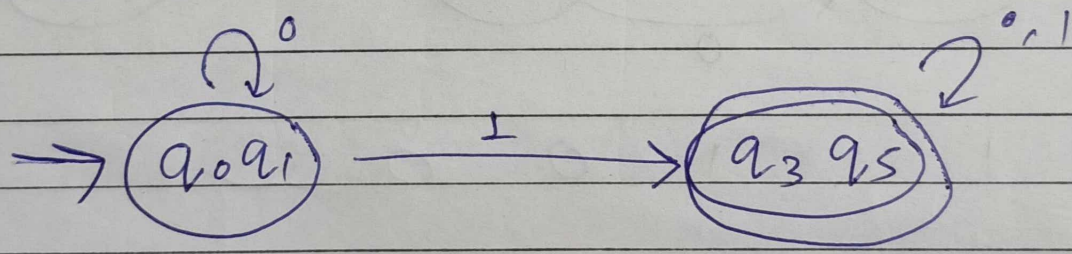
States	0	1
q_0	q_1	q_3
q_1	q_0	q_3
q_2	q_1	q_4
q_3	q_5	q_5
q_4	q_3	q_3
q_5	q_5	q_5

$\{q_3, q_5\}$ $\{q_0, q_1, q_2, q_4\}$

$\{q_3, q_5\}$ $\{q_0\}$ $\{q_3\}$ $\{q_1\}$ $\{q_2\}$



Note :- Eliminate unreachable states



Finite automata with output :-

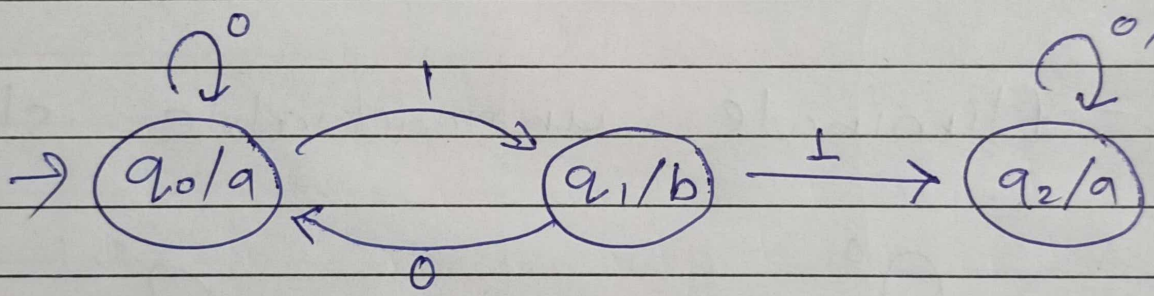
Moore and Mealy machine -

Moore Machine

There are 6 tupples namely $Q, \delta, \Sigma, q_0, \Delta, \lambda$

Δ = set of output variables

λ = Output function : $Q \rightarrow \Delta$



1 1 1 0 0

q_0 q_1 q_2 q_2 q_2 q_2

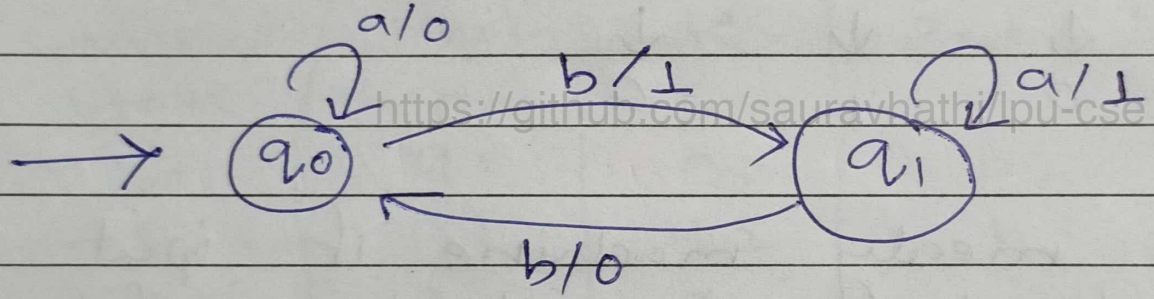
a b a a a a

If input length is n output length is n+1 in moore machine

Current State	Next state		Output
	0	1	
→ q_0	q_0	q_1	a
q_1	q_0	q_2	b
q_2	q_2	q_2	a

Mealy Machine

There are 6 tupple as in moore machine, however output function (λ) varies



$\lambda : Q \times \Sigma \rightarrow \Delta$

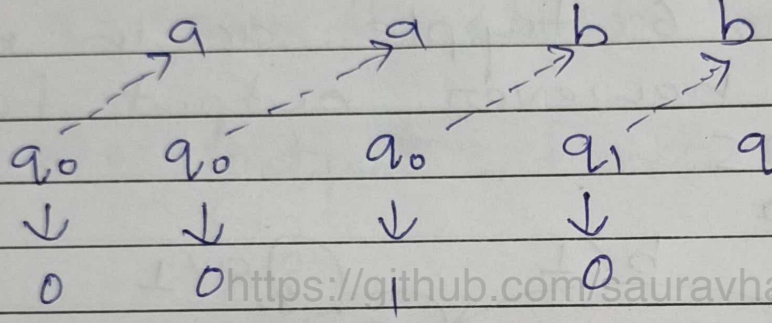
In moore machines state represents output, in mealy machines state and input represents output i.e., transition represent outputs.

DFA ⇒ $\delta : Q \times \Sigma \to Q$

NFA ⇒ $\delta : Q \times \Sigma \to 2^Q$

Moore ⇒ $\delta : Q \times \Sigma \to Q$; $\lambda : Q \to \Delta$

Mealy ⇒ $\delta : Q \times \Sigma \to Q$; $\lambda : Q \times \Sigma \to \Delta$



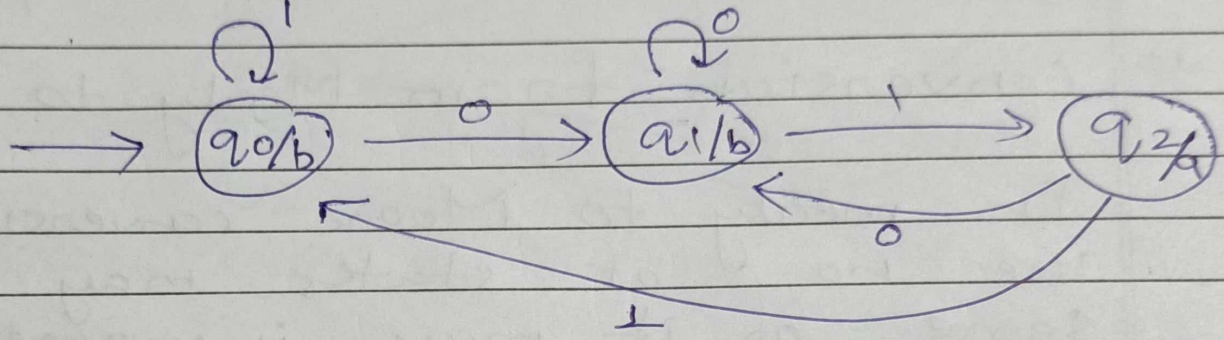
→ In mealy machine if input is of length n, output is of length n

Current State	Next State	
	a ; o/p	b ; o/p
→ q_0	q_0 ; 0	q_1 ; 1
q_1	q_1 ; 1	q_0 ; 0

Page :
Date : / /

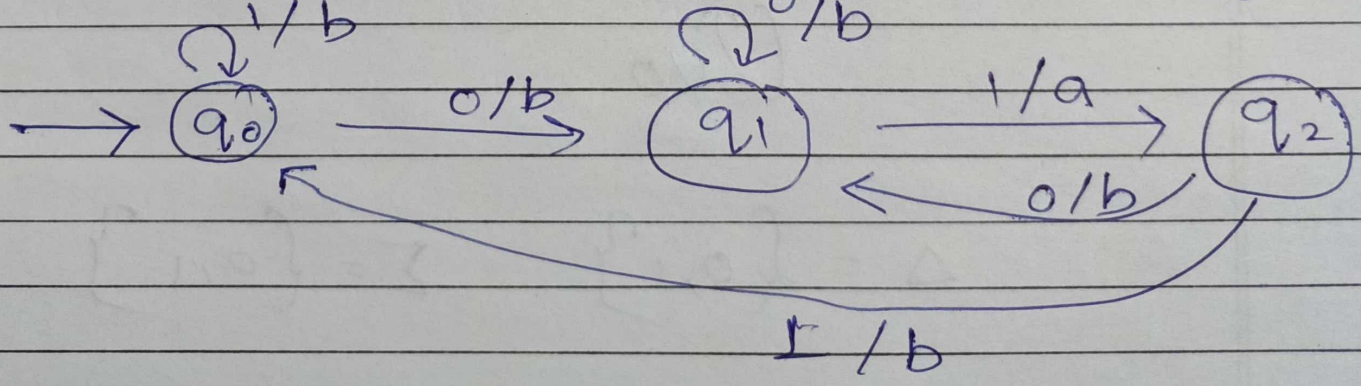
Q) Construct a moore machine that gives output as 'a' on every occurence of '01'.

⇒ DFA that ends with '01'



Current State	Next State		Output
	0	1	
→ q_0	q_1	q_0	b
q_1	q_1	q_2	b
q_2	q_1	q_0	a

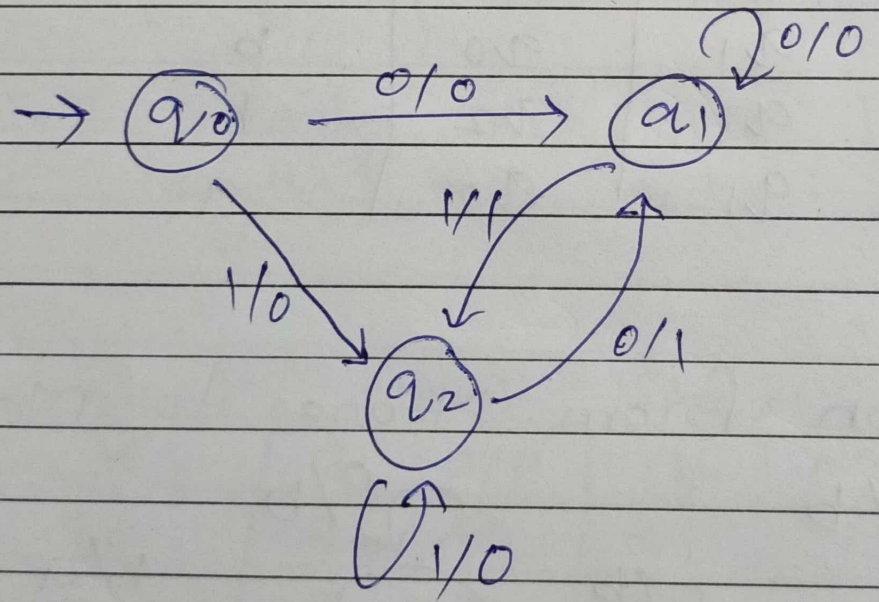
* Conversion from Moore to Mealy :



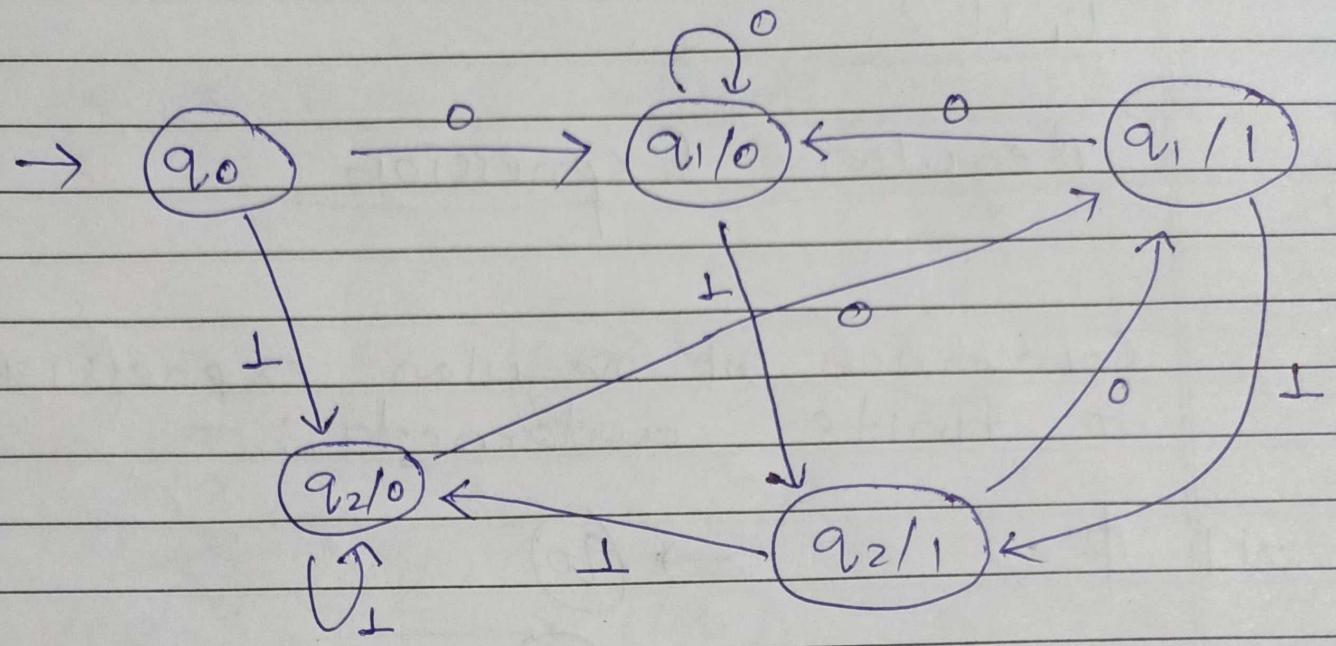
State	0		1	
	Next State	O/P	Next State	O/P
→ q_0	q_1	b	q_0	b
q_1	q_1	b	q_2	a
q_2	q_1	b	q_0	b

* Conversion from Mealy to Moore :

In mealy to Moore conversion the no of states may remain same or it may increase, however in moore to Mealy no of states remain same.



$$\Delta = \{0,1\} \qquad \Sigma = \{0,1\}$$



Max
Total no of State in Moore Machine = No of states in Mealy X No of inputs