Ex:- let ... matlab ...

Linkue

UNIT :-1

:- INTRODUCTION :-

Computation in theoritical way but we do practical but in all practical is made by some theory means concept means implementation is done by concept

Now, This is a subject that tell what computer works, and solve problems (konse prob. solve ho skte) and any problems that never be solve, chahe kitna powerful comp. ho.

- By me

Theory of computation or concept about computation for a machine i.e., behaviour of computation, type of computation, nature of computation, limitation and problem.

L A G { Language, Automata (machine), Grammar }

mealy moore conversion NFA to DFA min DFA

TERMINOLOGY :-

:- SYMBOL :- {a, b, c .... 0, 1, 2, 3 .. z}
a-z

(0) (1)

- ALPHABET :- (Σ) → Sigma A finite non empty set of symbols. {symbols kaa set} = }

Ex :- Σ = (a, b)

only make words that are of (a, b)

Σ = (a, b)

- Binary Alphabet (Σ) : Σ = (0, 1)

- STRINGS / WORDS (ω) :- → combination of i/p → words

A finite sequence of number defined on alphabet set {collection of Σ and sequence}

Ex :- Σ = (0, 1)

Σ, Σ, Σ = 0, 1, 00, 01, 10, 11

{finite seq. of symbols}

Ex:- let $\Sigma = (a,b)$ then on this alphabet we can defined number of strings which is the combination of a and b.

like $\omega_1 = a$, $\omega_2 = b$, $\omega_3 = aa$, $\omega_4 = ab$, $\omega_5 = bb$

$\omega_6 = ba$

:- LENGTH OF STRING (ω) :-

Number of positions of symbol in strings.
{:: count from 1}

Ex:- $|01101| = 5$

x means jo comb^n honge woh 5 hain

Ex:- $\Sigma = (a,b)$

length = 2

Total no of strings should be {aa, bb, ab, ba, aab, aaa, abb, abb, baa, bab, bba, bbb}

Alphabet is type of order or i/p that is used to made string

:- LANGUAGE :-

Collection of Strings. means- {set of strings}

Ex:- $\Sigma (a,b)$

L_1 = strings of length 3

so {aaa, aab, aba, baa, bb, abb, bab, bba,}

some L_2 = 1, 2, also 0.

so $\Sigma^0 = \epsilon$ (epsilon)

There are two types of languages

- finite :. means if there there given length=2

{a,b}

so the possible outcome is {means atleast 2 toh aane hi chahiye}

{aa, ab, ba, bb}

in case of finite.

- infinite :. means atleast one a.

so

lang^u => { a, aa, aaa, ... ab, abab, ba, baba. . . .

so this go ∞

Now we have to find this ~~lang~~ strings is a part of language.

- So we talk about finite we are able to find

ex:. {abab}

In this we able to find we make posible lang. and store and match it.

- but in infinite we are unable to do that bec. the lang. is ∞.

So there comes (AUTOMATA) and GRAMMAR

to a mathematical model and machine help to solve this problem. (It is an abstract model

:- POWER OF SIGMA (Σ) :- (alphabet)

empty string $|\Sigma| = 0$ kuch nhi toh length 0

Σ^0 = Set of all strings with length "0" {λ, ε} 2^0

Σ^1 = Set of all strings with length "1" {2'}

⋮

Σ^n so it is Σ^* → (star) set of all string with length ∞

It is also known as kleene closure (∞ lang.)

Σ^+ (positive closure) :- all possible strings but zero nhi hona.

$\rightarrow \Sigma^* = \Sigma^+ \neq \Sigma^\circ$

$\rightarrow \Sigma^* - \varepsilon$ or $\Sigma^\circ = \Sigma^+$

:- PROPERTIES OF STRING :-

~~GRAMMAR~~ !-

- Substring :- A substring is a itself is a string but it is a part of another string.

formal Defn :- if ω_1 and ω_2 are two strings then ω_2 is set to substring of ω_1, iff (if and only if) ω_1 contain atleast one occurence of ω_2

Ex :- $\omega_1 = aabab$

$\omega_2 = baa$

$\Sigma = (a, b)$

* In this order matter.

So this ω_2 is not part of ω_1. if ω_2 bab so this is a part.

Means order matter ω_2 is not substring ω_1.

$\omega_2 = aab$ now it is a part.

for every string there is exist atleast two Substring - Epsilon (ω) and second the string itself.

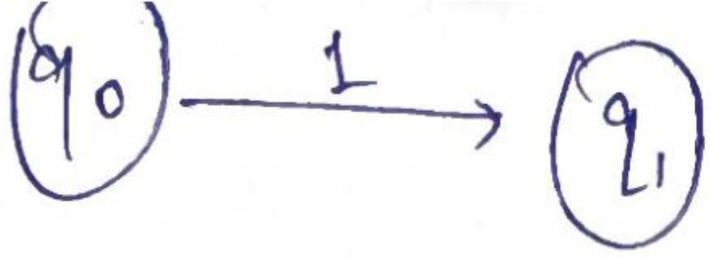
- TRANSPOSE OR REVERSE OF STRING :-

$\Sigma = (a, b)$

$\omega = aab$

it is represented by ω^R or ω^t

if we write a string from right to left is called reverse of the strings.



not go to other other
it is deterministic
but in NFA ek i/p kai
multiple state

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore
Shri Vaishnav Institute of Information Technology

3. Report (Answer to be submitted in written form only)

a) How does the process of creating a Paper Prototype contribute to the overall design and development cycle of a digital product?

b) Discuss the specific steps involved in creating a Paper Prototype, the benefits it offers in terms of usability testing and iterative design refinement, and how feedback gathered from Paper Prototypes can inform subsequent design iterations.

$\Sigma = (a, b)$

$\omega = aab$

reverse of ω^R or $\omega^t = baa\ (\omega)$

is $\omega = \omega^R$ (not necessary)

ex :- aba = aba

inside bracket not necessary but out.

- PALINDROME STRING :-

$\Sigma = (a, b)$

$\omega = aba$ (palindrome)

A string ω is said to be a palindrome iff. $\omega = \omega^R$

Ex = $\Sigma = (a, b)$

palindrome = (ε, a, b, aa, aaa, bbb, bab, aba)

:- OPERATIONS ON STRINGS :-

- CONCATINATION :- can be defined two or more strings

if ω_1 and ω_2 are two strings then concatination ω_1 with ω_2 produce a new string ω such that ω_2 always follows ω_1.

$\Sigma = (a, b)$ $\omega = \omega_1 . \omega_2$ (concat. oper)

$\omega_1 = aa$ = aabb

$\omega_2 = bb$

if there is not given so assume aa ∨ bb

→ mean if q_1 has 1 i/p it goes to next state

DETERMINISTIC FINITE AUTOMATA (DFA)

formal Definition: A finite automata is formally represented by five (5) tuples.

↳ taking one from then goes to another state

$$M = (Q, \Sigma, \delta, q_0, F)$$

↳ no. of State ↳ input ↳ final state

→ Q = non empty finite set of states. (set of all finite states)
(all transition means how much states are taken to go from one state to final state)

→ Σ = a finite input alphabet
$\Sigma = 0, 1$ (as)

→ δ (delta) = Transition function. (from one state to other state / second state)

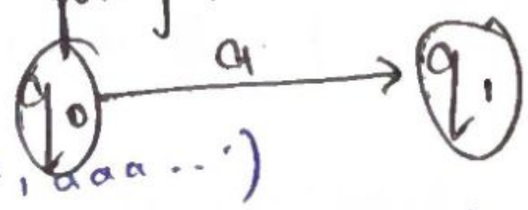
$$\delta = Q \times \Sigma \to Q$$
State state. (transition)

$q_0 \longrightarrow q_1 \longrightarrow$

→ q_0 = initial state and "$q_0 \in Q$" belongs (start state)

→ F = The set of final/accepting state
↳ final state. $F \subseteq Q$ ↳ subset
(There are only one final state but sometimes there are set of final state)

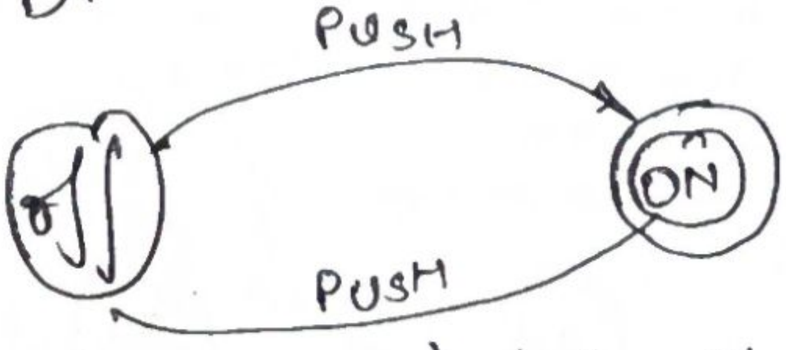
Deterministic means if there is one state with a alphabet like are going in another state it is going continously. but in this there is atleast one state so it is deterministic.



if string start with a (a, aa, aaa ...)

Ex:- of design states
→ (initial state) , ◎ Final state.

☆ In DFA in each state there is two I/P.



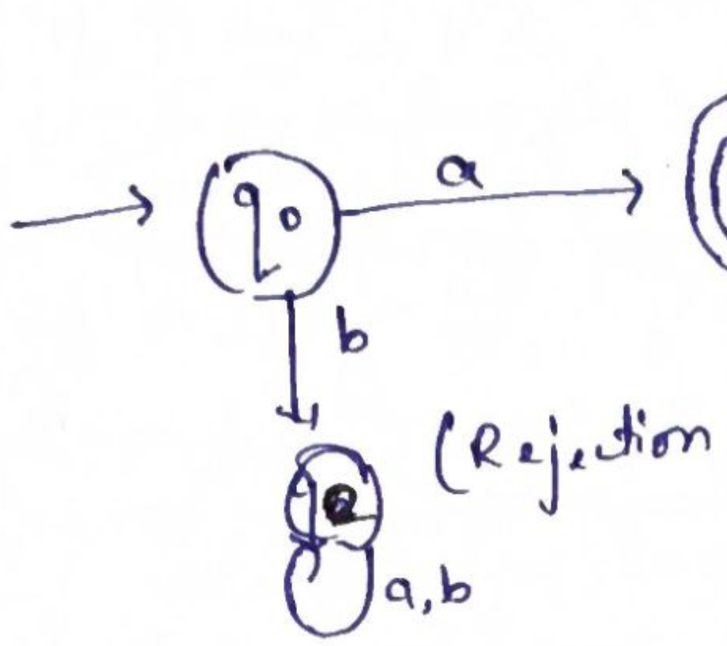
Q = (OFF, ON) :. string.
Lan :. Push, Pop

Ex1. if strings starts with a

so possible lang

{a, aa, aaa, ab, ..}

and Σ = (a, b) ← This is transition (δ)



now

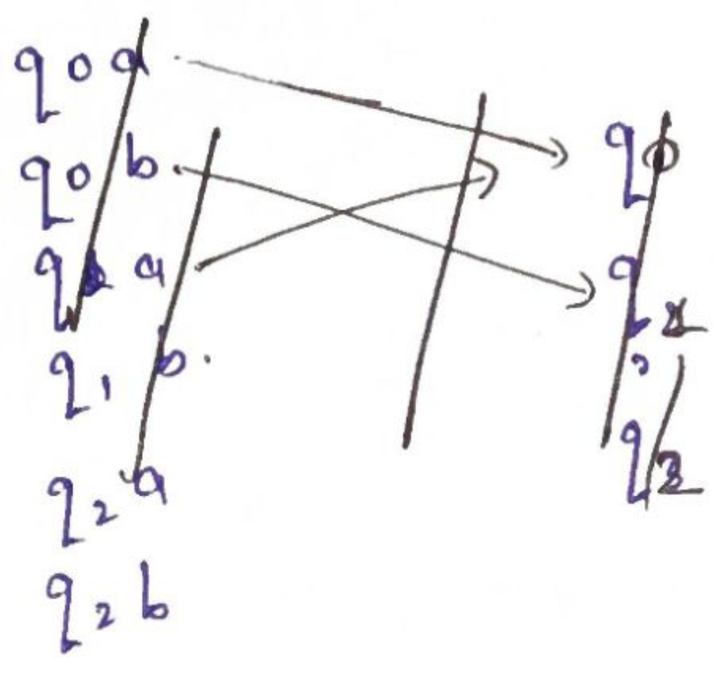
$\delta = Q \times \Sigma \longrightarrow Q$

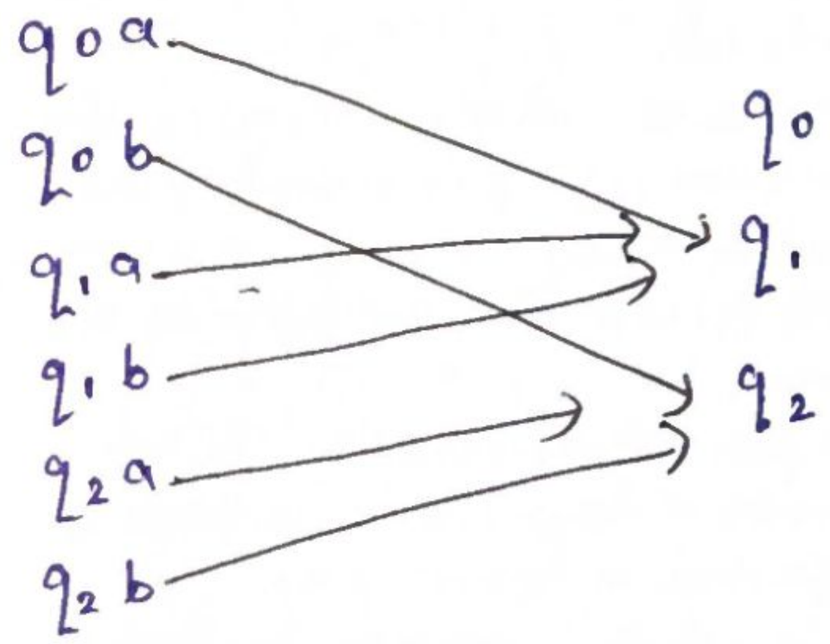
$q_0 \times (a, b)$ $q_0 a$ $q_0 b$

$q_1 \times (a, b)$ $q_1 a$ $q_1 b$

$q_2 \times (a, b)$ $q_2 a$ $q_2 b$

now

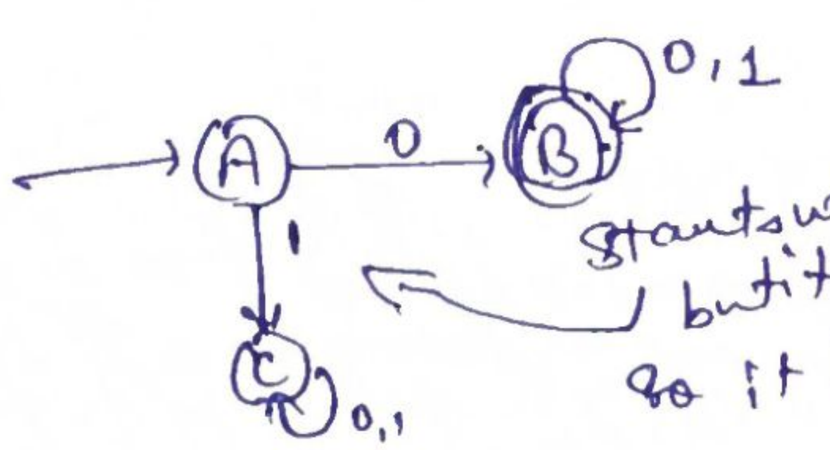




CONSTRUCT DFA which accept lang. of all strings.

L_1 :- Set of all strings starts with '0'

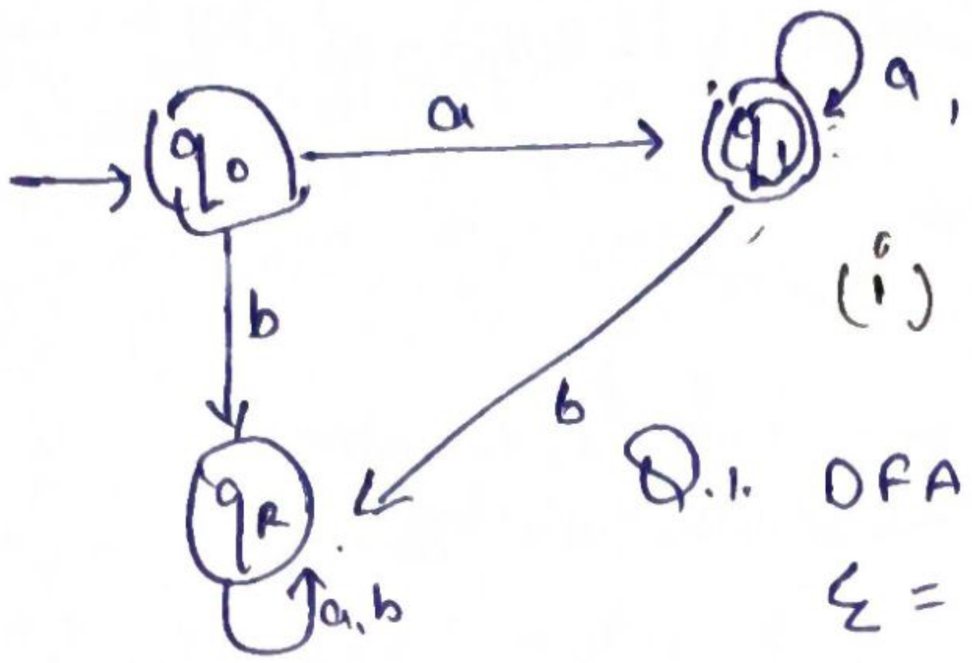
= {0, 00, 01, 000, 010, 011, 0011, ..}.



(i) Containing 'a'

Σ : a, b

$L = \{a, aa, aaa, aba, baa \ldots\}$.



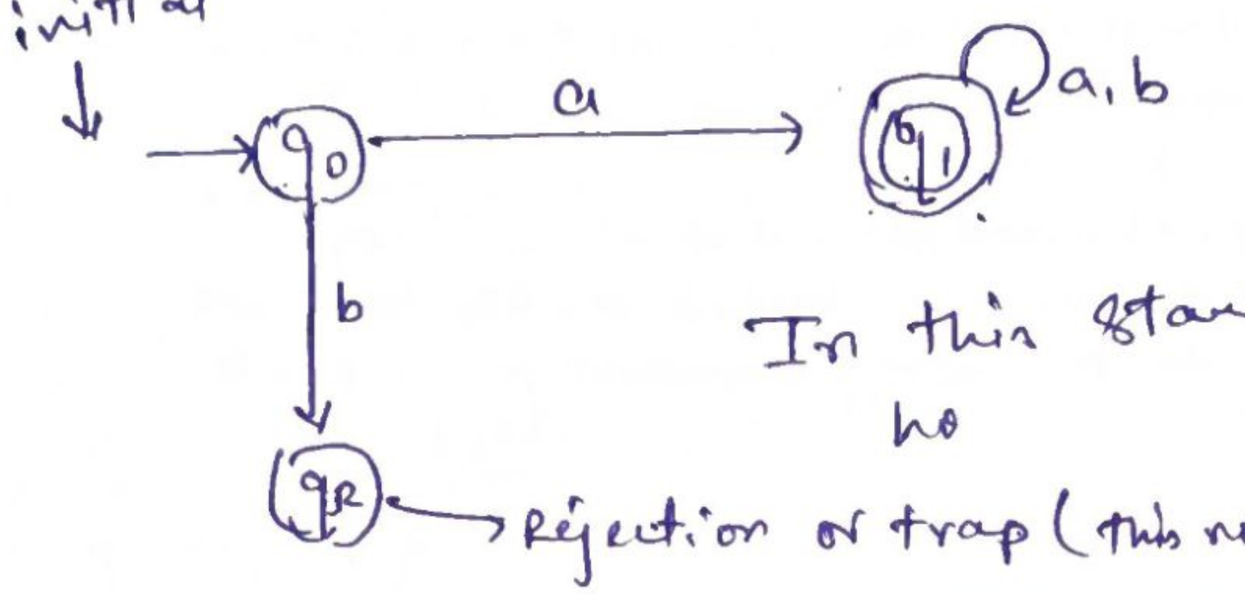
(i) Strings starts with pattern :-

Q.1. DFA which starts with over the i/p alphabet

$\Sigma = \{a, b\}$ (starts with a)

$L = (a, ab, abb, aab, \ldots)$.

we take minimal string so a ⟶ ~~one st~~ n+1 = 2 state.

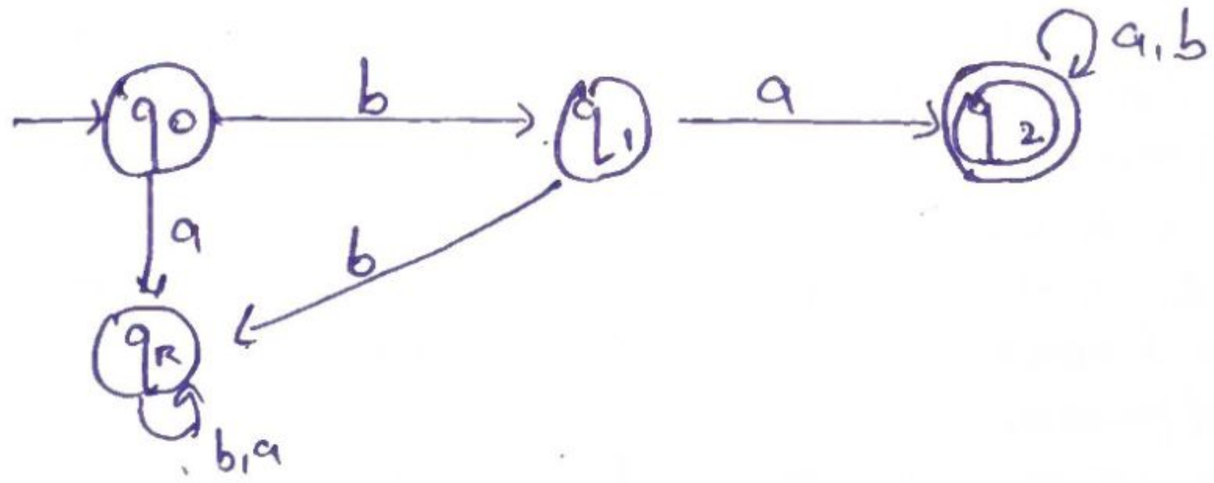


In this start matter ending kaisi bhi ho

Q.2 DFA which starts with ba

$\Sigma = \{a, b\}$, $L = (ba, baa, bab, baab \ldots)$

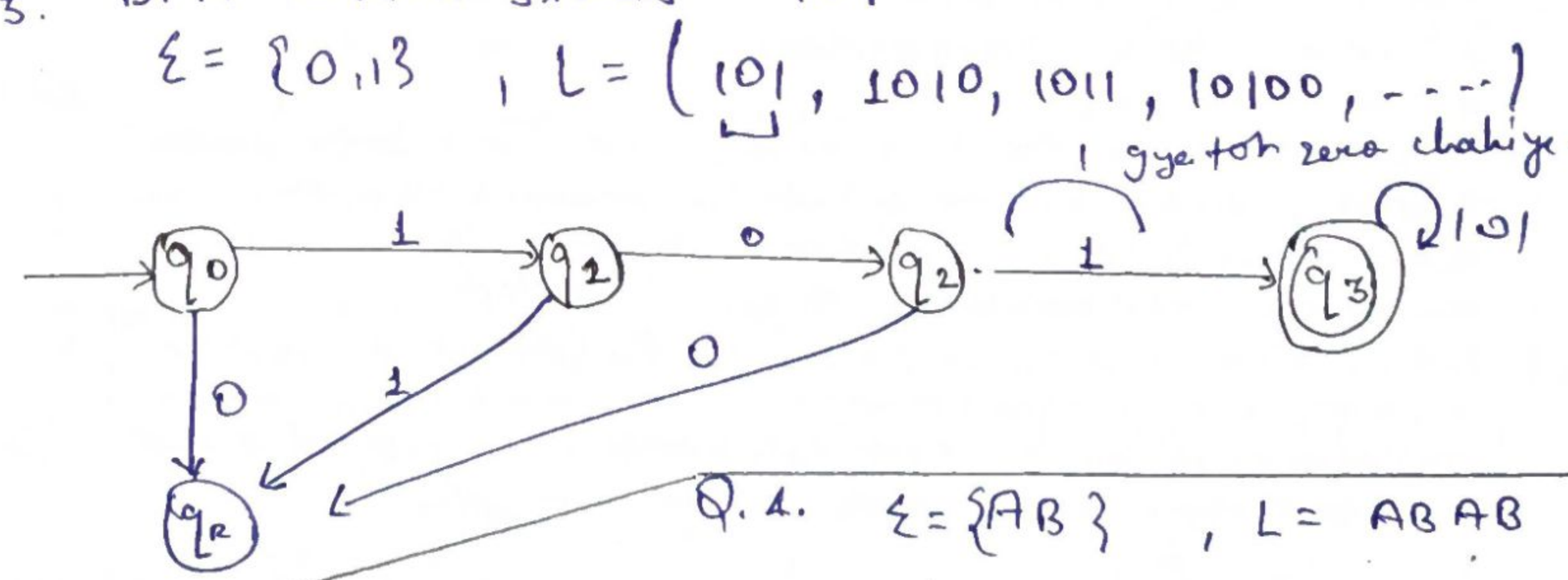
n+1 = 2+1 = 3



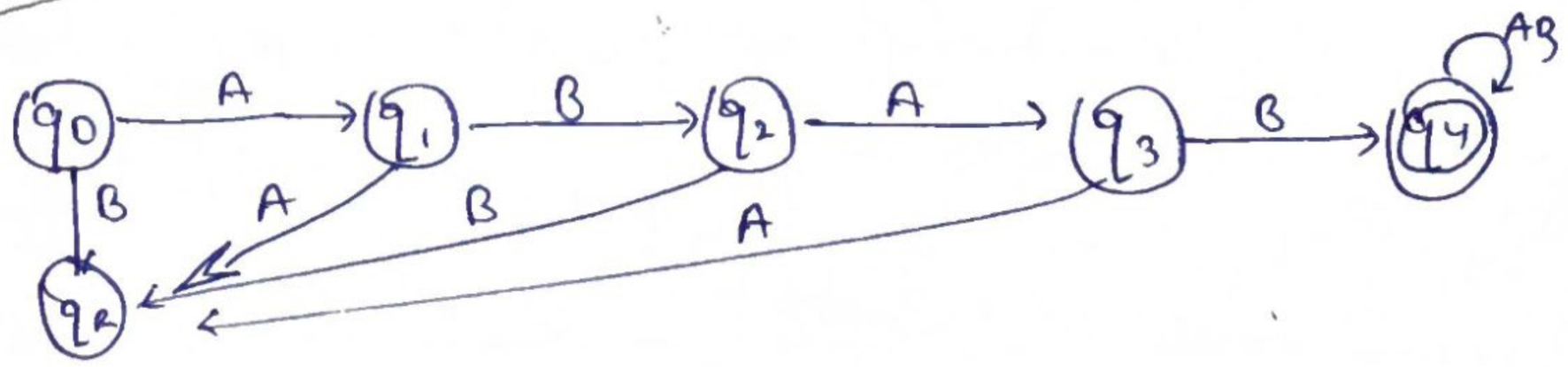
Q.3. DFA which starts 101

$\Sigma = \{0, 1\}$, $L = (101, 1010, 1011, 10100, \ldots)$

1 ya to zero chahiye.

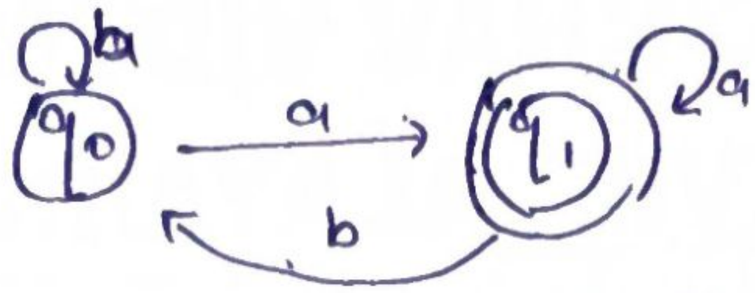


Q.4. $\Sigma = \{AB\}$, $L = ABAB$

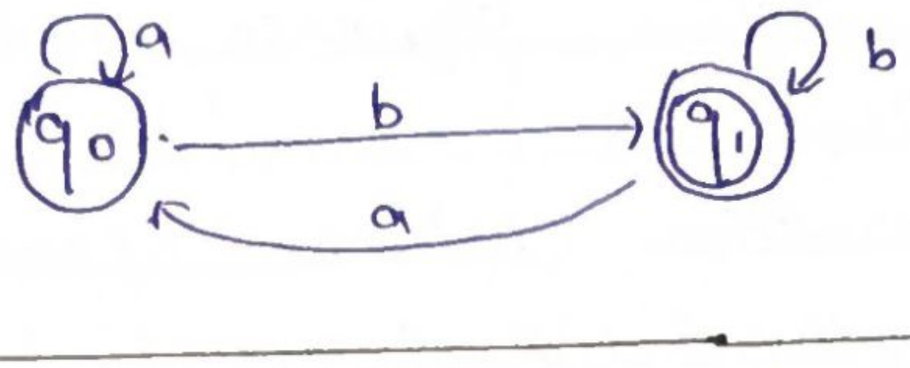


(ii) String ends with pattern (mean start not matter and matter

Q.1. end with a , $\Sigma = \{a, b\}$, L = (a, aa, ba, aaa, aab.....

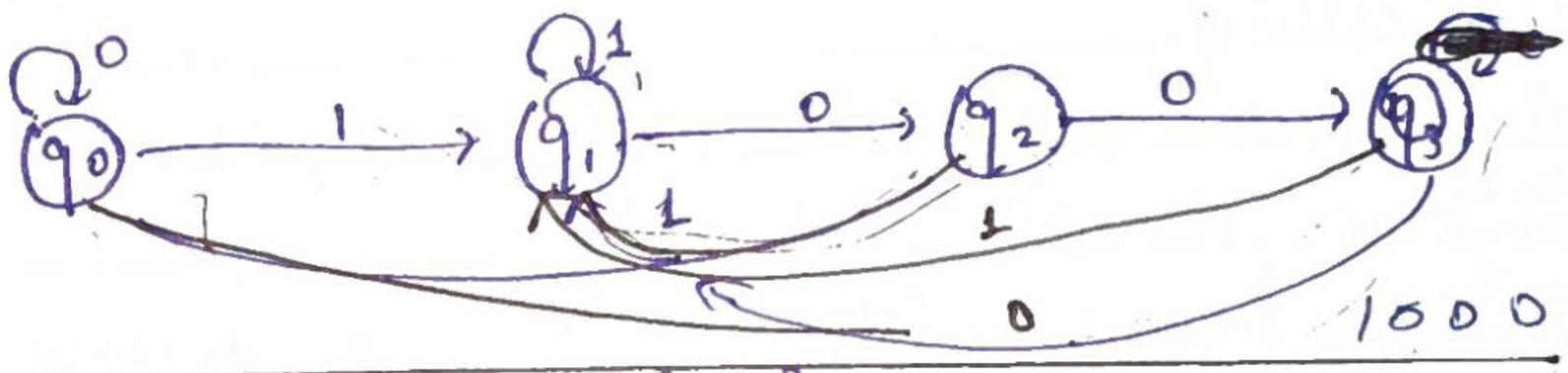


Q.2. end with b , $\Sigma = \{a, b\}$, L = (b, ab, aab, abab.....)

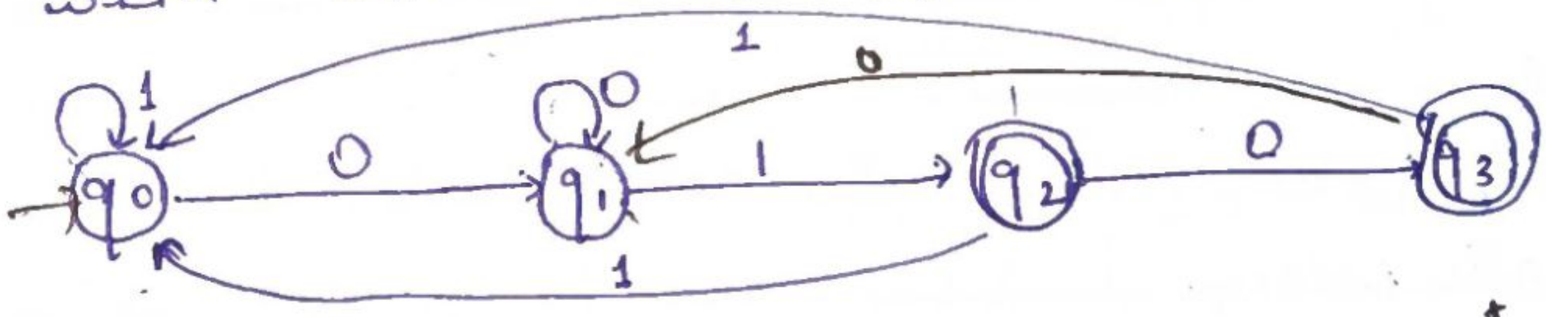


Q.3. ends with 01 , $\Sigma = \{0, 1\}$, L = (01, 001, 101, 010..

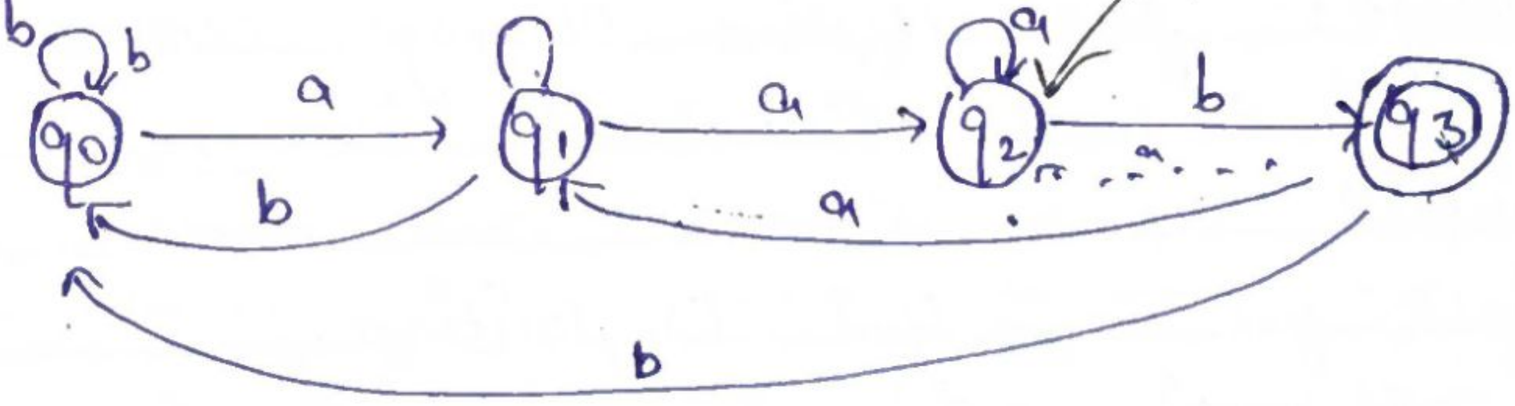
Q.4. end with 100 , $\Sigma = \{0, 1\}$, L = (100.....



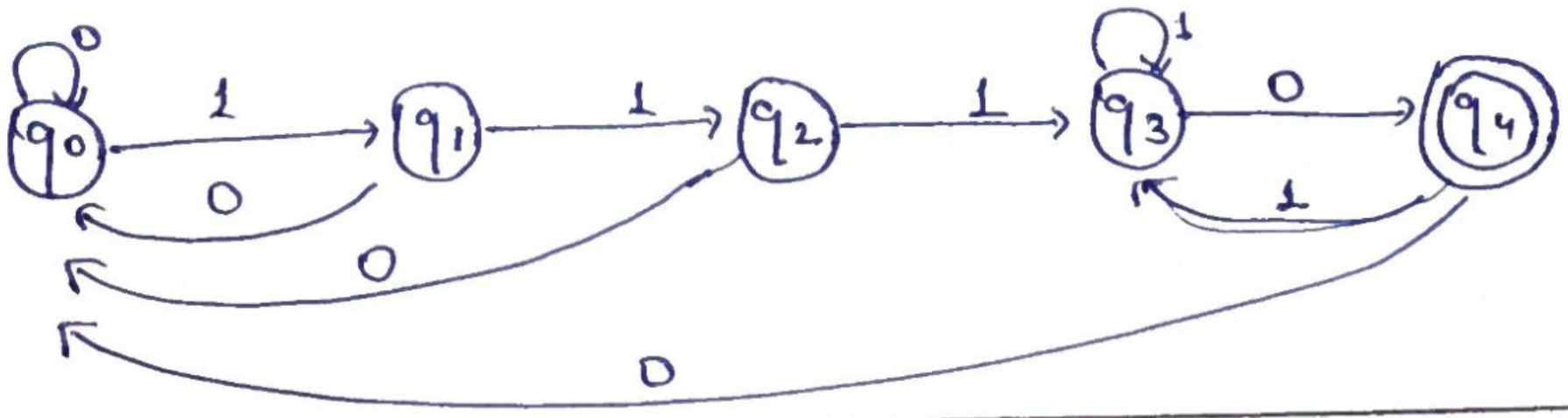
Q.5 end with 010 $\Sigma = \{0, 1\}$, L = {010...



Q.6. aab



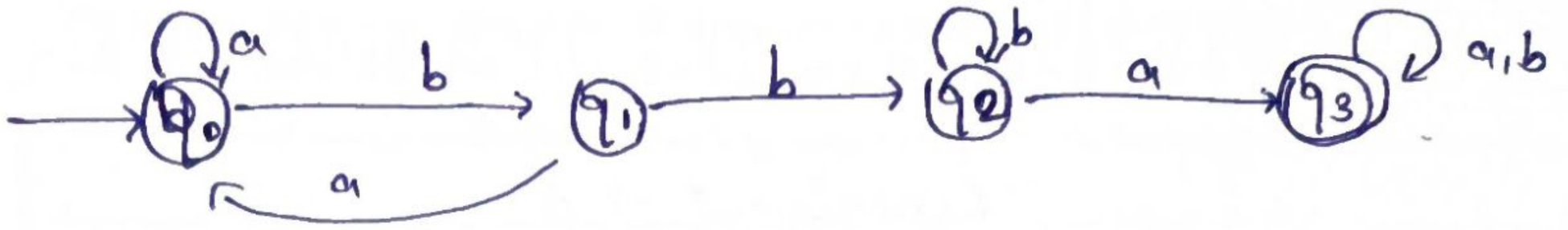
Q.7. 1110 , $\Sigma = \{0, 1\}$, L = (1110, 01110, 11110 ...



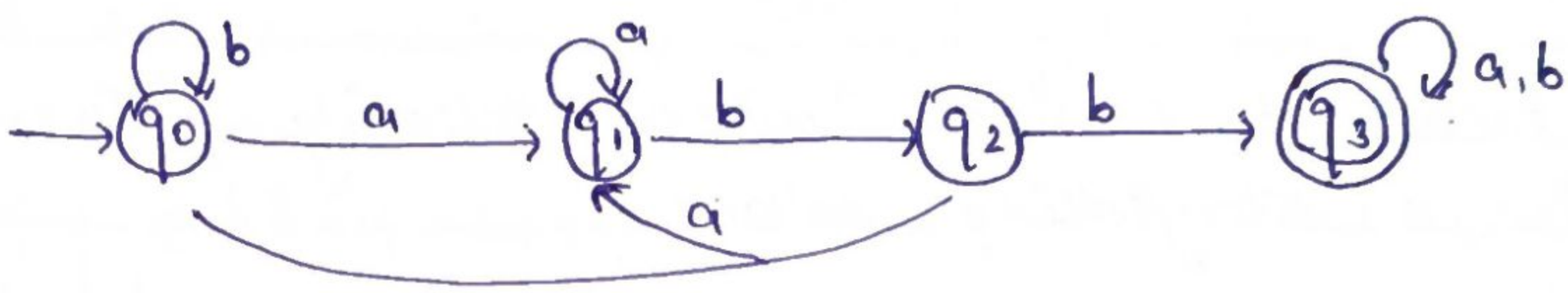
(iii) Sub String pattern (not start and end matter).

Q.1. Design DFA with consist of substring [bba] over the i/p $\Sigma = \{a, b\}$

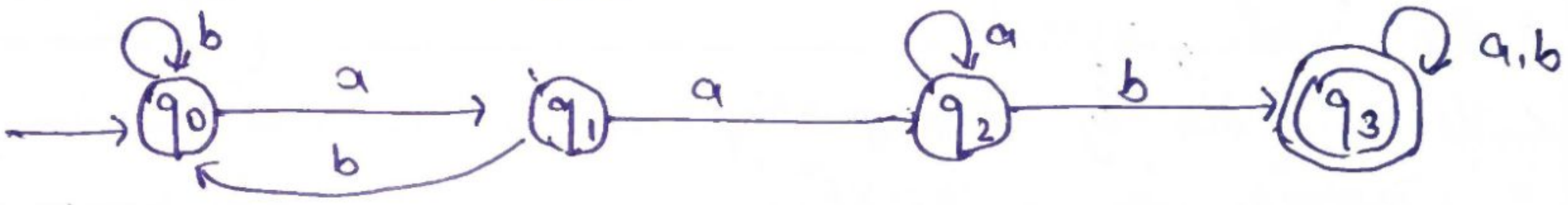
$\Sigma = \{a, b\}$, L = { bba, abba, bbab, abab .... }



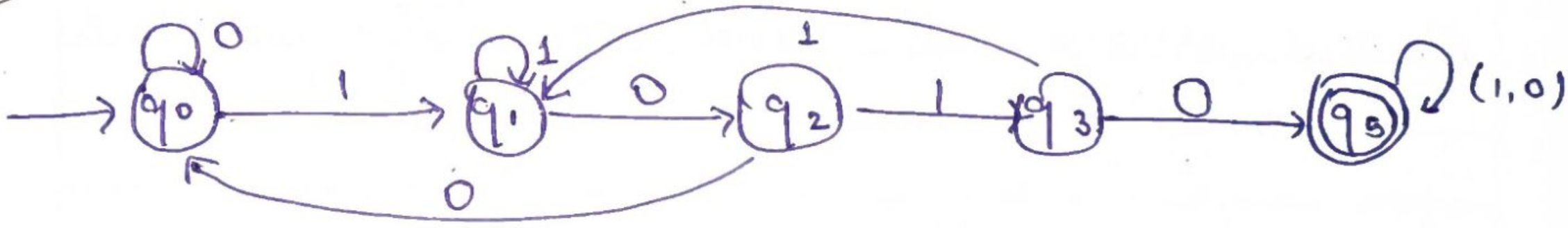
Q.2. $\Sigma = \{a, b\}$, L = { abb, aabb, babb, abba ... }



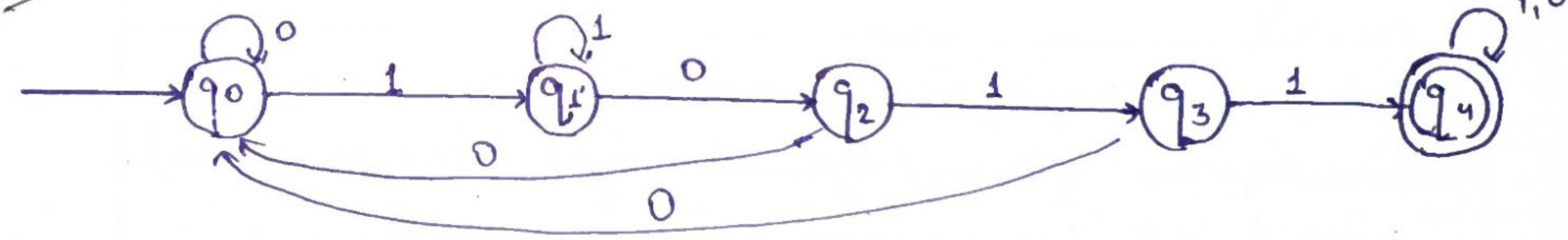
Q.3 $\Sigma = \{a, b\}$, L = { aab, baab, aaba ... }



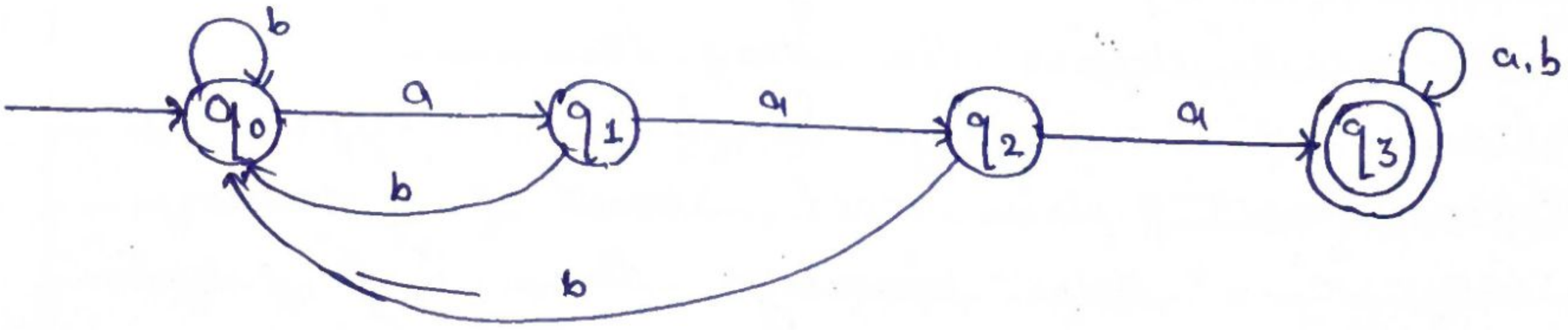
Q.4. 1010, $\Sigma = \{0, 1\}$, L = { 1010, 01010, 11010, ... }



Q.5. 1011, $\Sigma = (0, 1)$, L = { 1011, ... }

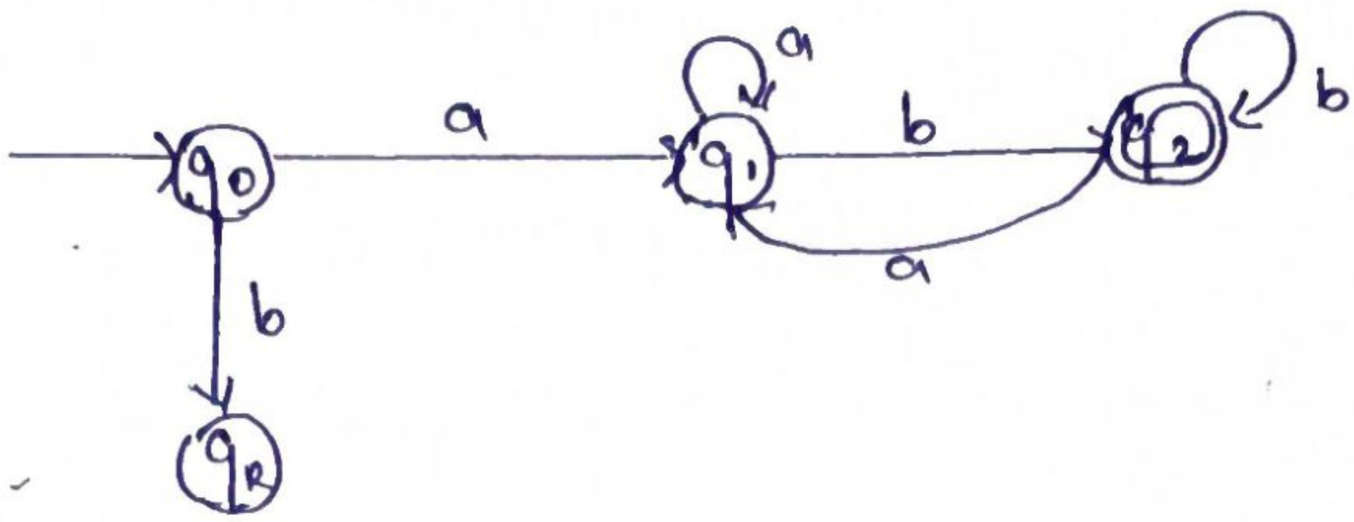


Q.6. aaa, $\Sigma = \{a, b\}$, L = { aaa, aaab, baaab .... }



Q.7 String start with a and end with b

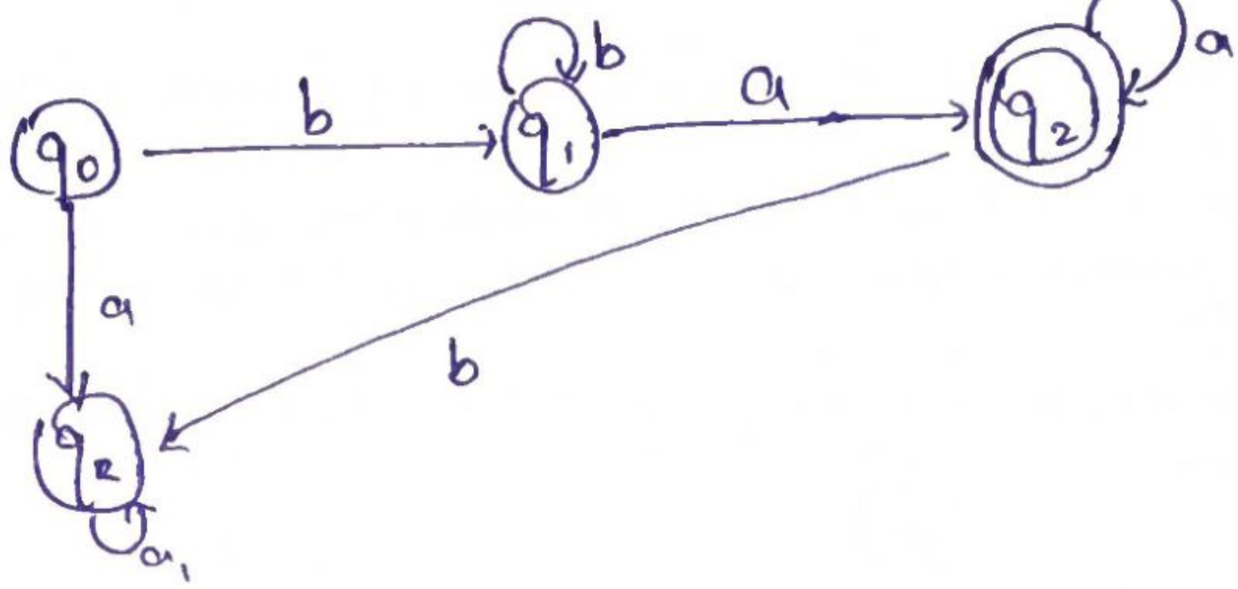
$\Sigma = \{a, b\}$, $L = \{ab, aab, abb, abbb, abab \ldots\}$



Q.8. • aa , $\Sigma = \{a, b\}$,

Q.9. Not start a and not end with b.

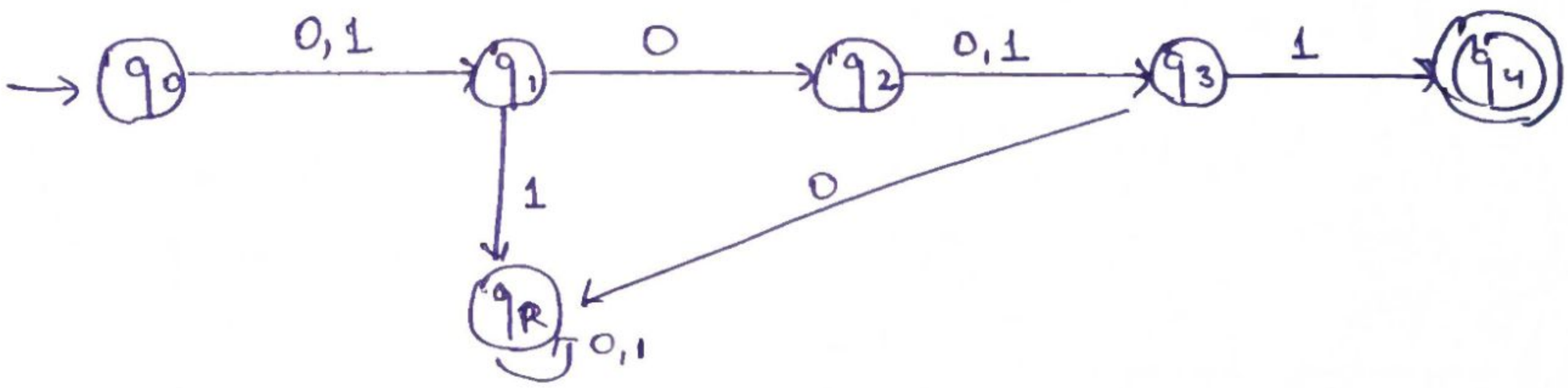
$\Sigma = \{a, b\}$, $L = \{ba, bba, baba, baa$



Equal

Q.10.:- DFA 2nd and 4th fin −0−1

$\Sigma = (0, 1)$, $L = \{1011, 0001, 0011, 1001 \ldots$



Divisibility Pattern Logic { in this always initial is final state }.

Q Design a finite automata accepting all strings over decimal i/p num system which is .

- Divisible by 2, 3, 4, 5 .

Q.. Divisible by 2 → Binary alphabet (num system binary hai. isliy)

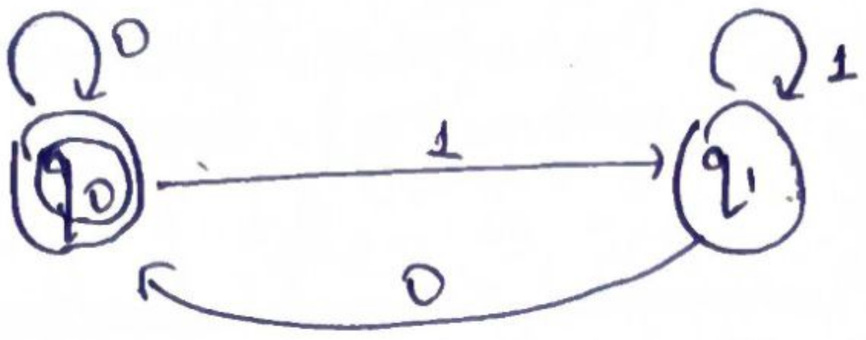
$\Sigma = \{0, 1\}$

~~Binary of Σ~~ = ~~0 1~~ 0 → now two q0 q1

Divisible by 2 = (0, 1) { 0 ÷ 2 → 0 , 10 ÷ 2 → 1)

Transition Table

	0	1
q_0	q_0	q_1
q_1	q_0	q_1



L = 0, 10, 100, 1000, 110

q_0 to final state

bcz zero is divisible by all.

Q.4

$Q \rightarrow q_0, q_1$ $\quad q \rightarrow q_0$

$\Sigma \rightarrow 0, 1$ $\quad f \rightarrow q_0$

$\delta \rightarrow q_0 \rightarrow q_1$

Q.2. Divisible by 3

Binary $\Sigma = \{0, 1\}$

Divisible by 3 = $\{0, 1, 2\}$ $q_0\ q_1\ q_2$

L = 0, 11, ...

0 0 0 — 0

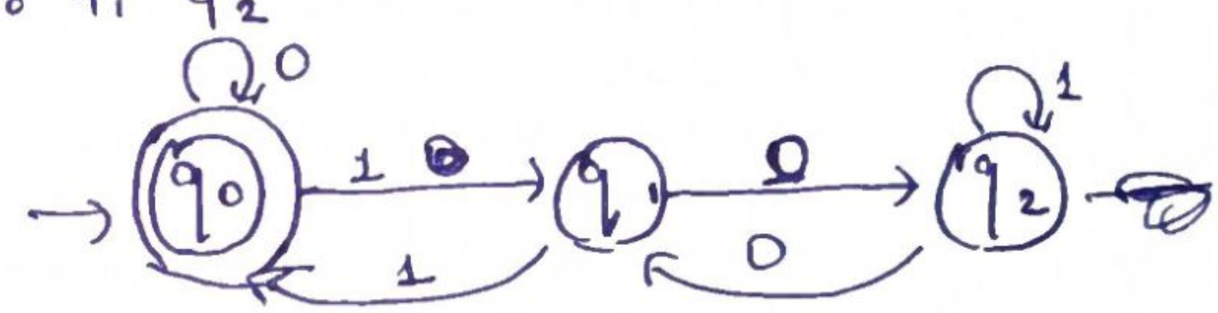
0 0 1 — 1

0 1 0 — 2

1 1 — 3

Transition Table

	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2



$Q \rightarrow q_0, q_1, q_2$

$\Sigma \rightarrow 0, 1$

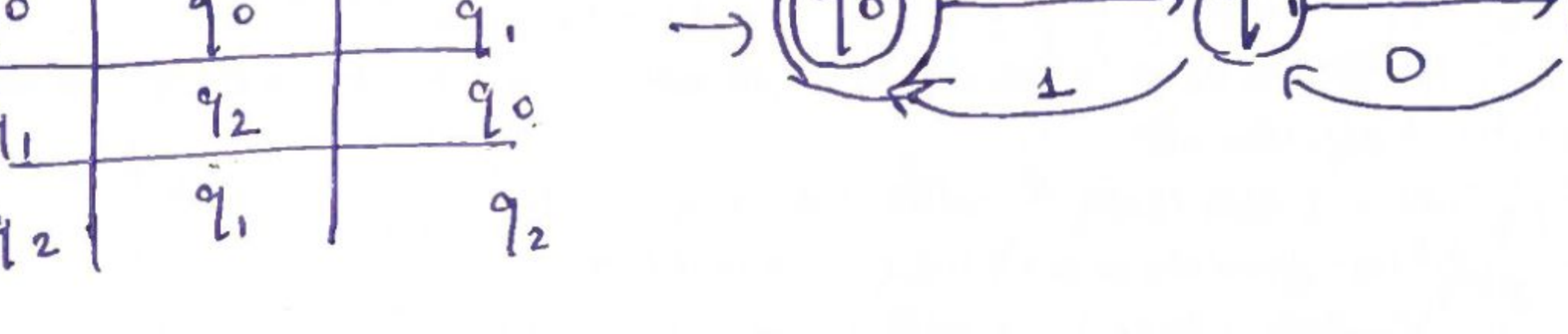
δ ⊛, $q \rightarrow q_0$, $f = q_0$

Q.3. Divisible by 4 $\quad L = \{0, 100, 1000, 1100 \ldots\}$

Binary $\Sigma = \{0, 1\}$, Divisible by 4 = $\{0, 1, 2, 3\}$ $q_0\ q_1\ q_2\ q_3$

Transition Table

	0	1
→ q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_2	q_3



verify $\quad w = 100$

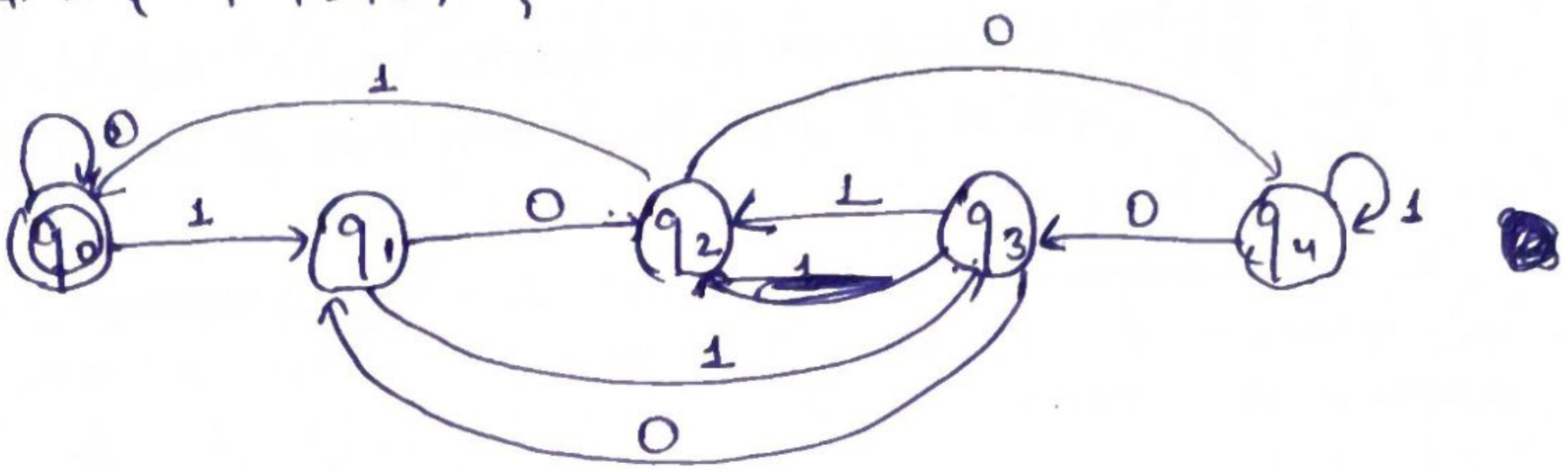
$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_0$

Q. Divisible by 5 :-

Binary $\Sigma = \{0,1\}$

$L = \{0, 0101, 1010, 1111, 10100 \ldots\}$ (0, 5, 10, 15, 20)

Divisible by 4 = $\{0,1,2,3,4\}$ ($q_0\ q_1\ q_2\ q_3\ q_4$)



δ	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_2	q_2
q_4	q_3	q_4
~~q_5~~	~~q_4~~	~~q_5~~

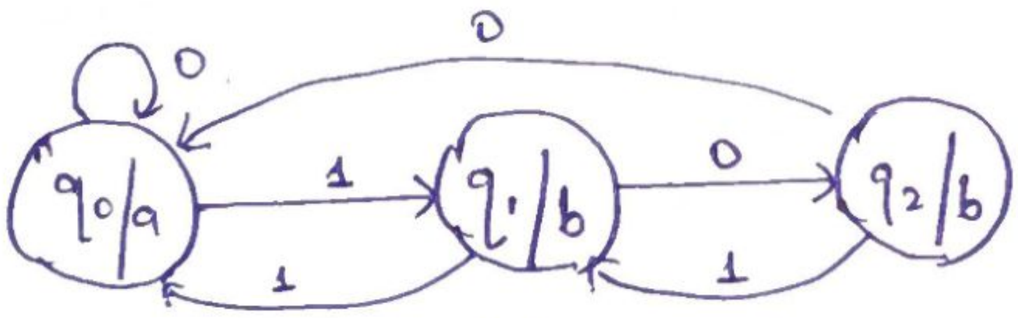
This is not main method plz anybody

MOORE MACHINE :- Defined with 6 tuples.

Moore machine is like DFA but with O/P.

DFA - only checks if string is accepted or not. (it only checks validity)

but in moore machine har each state pe ek O/P hota hai. means jab bhi ko machine kisi state pai pohachti toh O/P deti hai



in this no. of i/p $n = n+1 = 6$

20110 (5) aaabaa (6)

Transition TABLE

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

Defined as :

$M = (Q, q_0, \Sigma, \Delta, \delta, \lambda)$

$Q \rightarrow$ State.

$q_0 \rightarrow$ initial state

$\Sigma \rightarrow$ i/p alph → symbol

$\Delta \rightarrow$ O/P of (a,b) symbol

$\delta \longrightarrow$ Transition f^n ($\delta = Q \times \Sigma \rightarrow Q$)

$\lambda \longrightarrow$ O/P f^n ($Q \Longrightarrow \Delta$)

$q_0 \longrightarrow a$

$q_1 \longrightarrow b$

$q_2 \longrightarrow b$

* Output depends only on current state, not input.

O/P timing → after state change (means state change krne kai

Real world use :- Elevator control, vending machine (O/p after full i/p)

MEALY MACHINE :-

This machine (FSM) finite state machine, that give i/p with O/p and O/p depends on current state + i/p symbol.

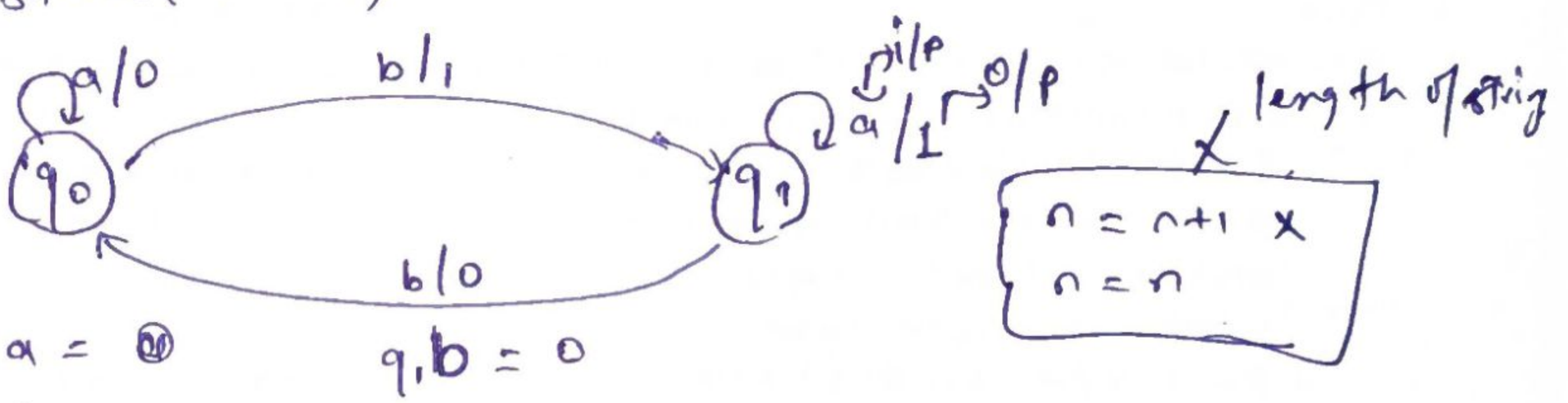
As compare to moore it is fast and reactive

$M = (Q, \Sigma, \delta, \Delta, \lambda, q_0)$ all same to more

but $\lambda = Q \times \Sigma \longrightarrow \Delta$

means depends on current state + i/p symbol

O/p is generated during transition ek dum fast not after reaching state (moore).



Transit

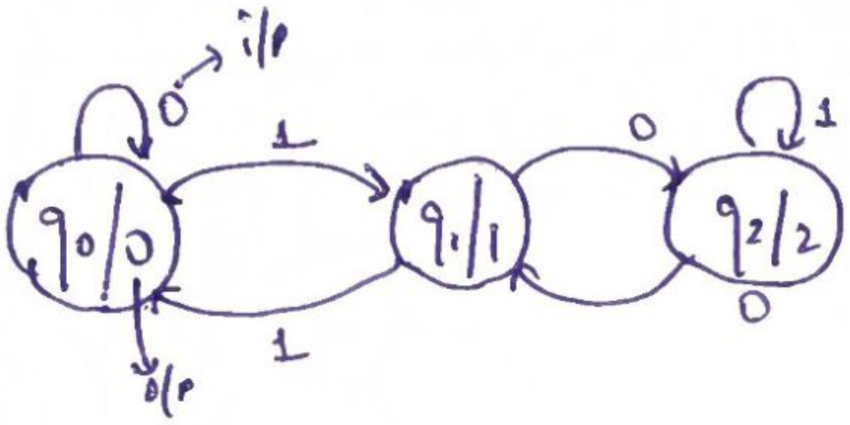
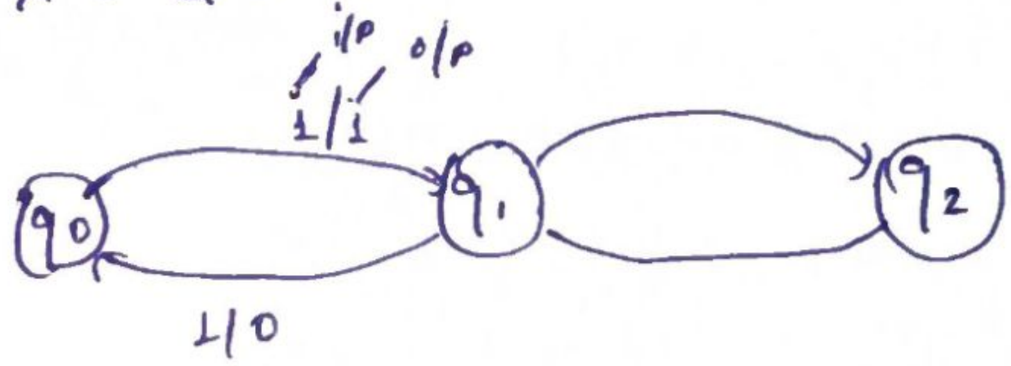
Current State	Next state.			
	I/p a		I/p b	
	State	O/p	State	O/p
q_0	q_0	0	q_1	1
q_1	q_1	0	q_0	0

- faster respons, fewer states needed compared to moore.
- Ideal for real time systems like network protocols.

Ex :- moore press button → machine change state → then give O/p

mealy same → machine instant react give O/p

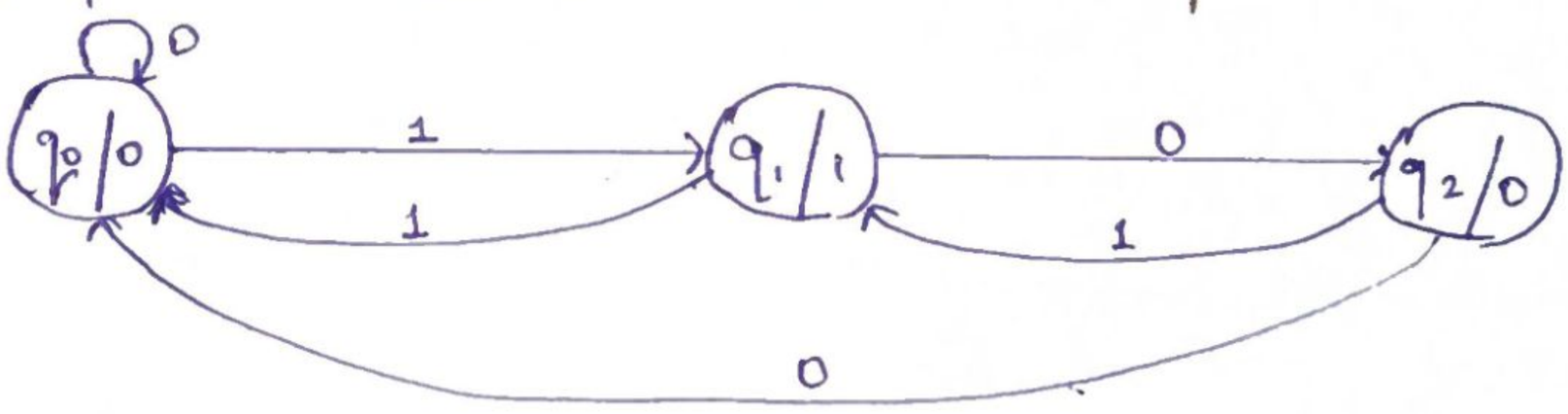
Difference between moore and mealy :-

MOORE MACHINE (if moore me O/p inside state)	MEALY MACHINE (If mealy me O/p is outside the state.
- Output only depends on present state, not i/p.	Output only depends on present state and present input.
- I/p string length 'n' → O/p string 'n+1' eg i: 1101	I/p string string of length 'n' → O/p lengths is also 'n'
- $\lambda : Q \rightarrow \Delta$	$\lambda : Q \times \Sigma \rightarrow \Delta$
	
- O/p is synchronous with clock.	Mealy O/p is asynchronous.

MOORE TO MEALY CONVERSION :-

(inside the state o/p)

Q.



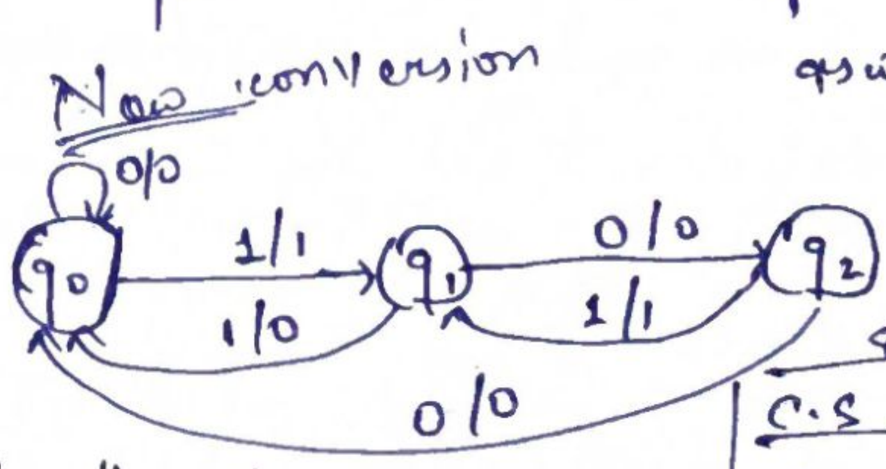
Now hum jaante ki moore main O/p inside the state rehta hain but mealy main outside the state rehta i/p and O/p rehta hain toh kuch nhi tension bs (q0) 0/0, 1/1 → new q0 kaa O/p zero hain q_1 kaa 1 toh 0/0, 1/1 jidhar arrow toh waala O/p.

Transition of moore.

C.S.	Next State		O/p
	0	1	
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_0	q_1	0

jis direction main jaa rhaa usska O/p jiska O/p like $q_0 \rightarrow 0$

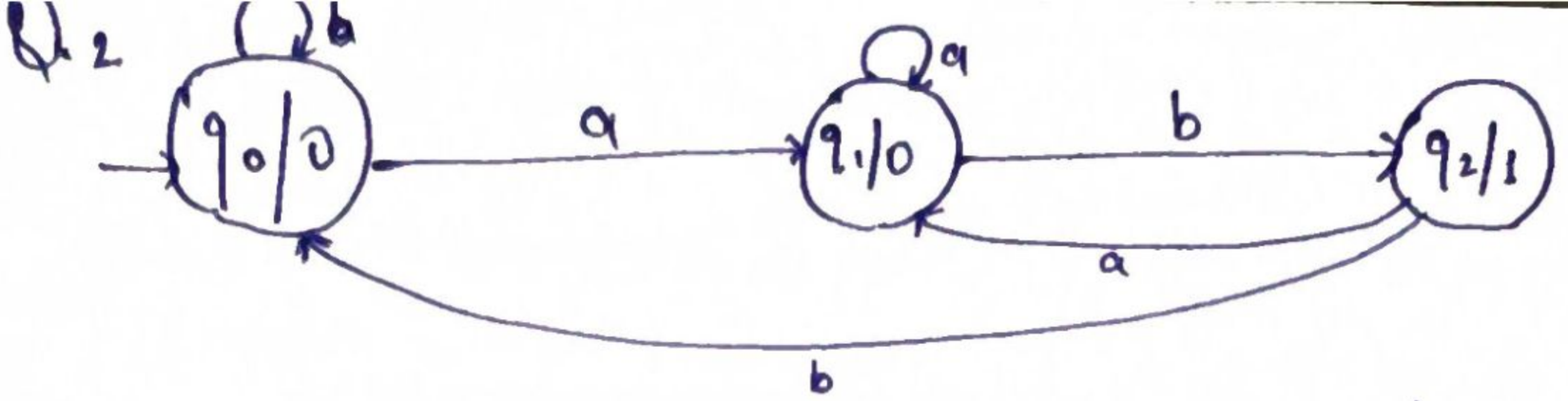
New conversion



as we talk about

Transit Table same as moore.

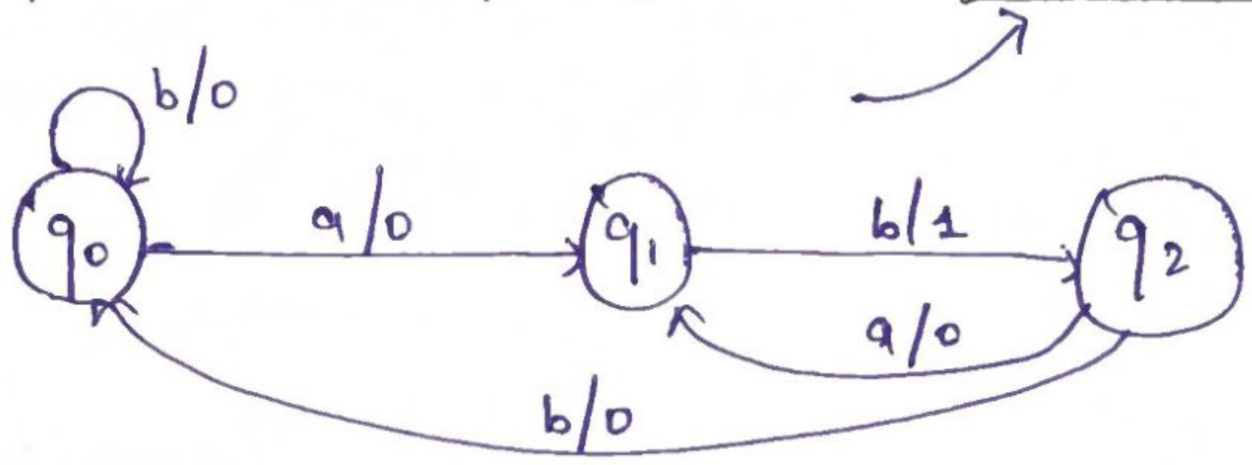
C.S.	Next State	
	0	1
q_0	q_0/0	q_1/1
q_1	q_2/0	q_0/0
q_2	q_0/0	q_1/1



Moore Transi ⟶ Mealy transition (same moore

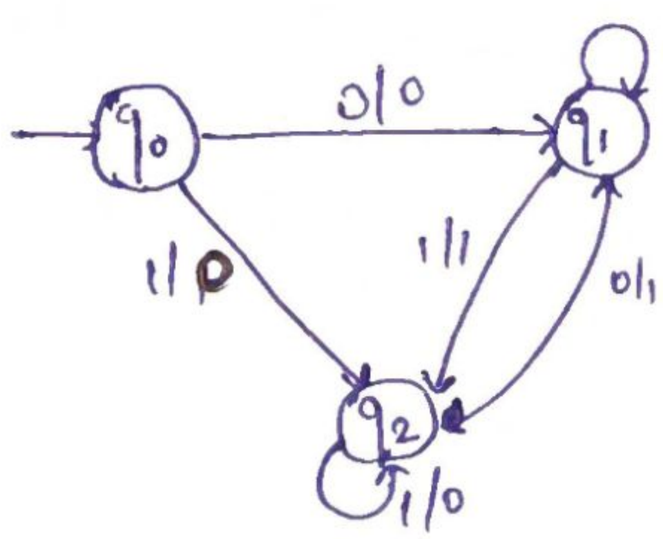
Current State	Next State		O/P
	b	a	
q_0	q_0	q_1	0
q_1	q_2	q_1	0
q_2	q_0	q_1	1

Current State	Next State	
	b	a
q_0	$q_0/0$	$q_1/0$
q_1	$q_2/1$	$q_1/0$
q_2	$q_0/0$	$q_1/0$

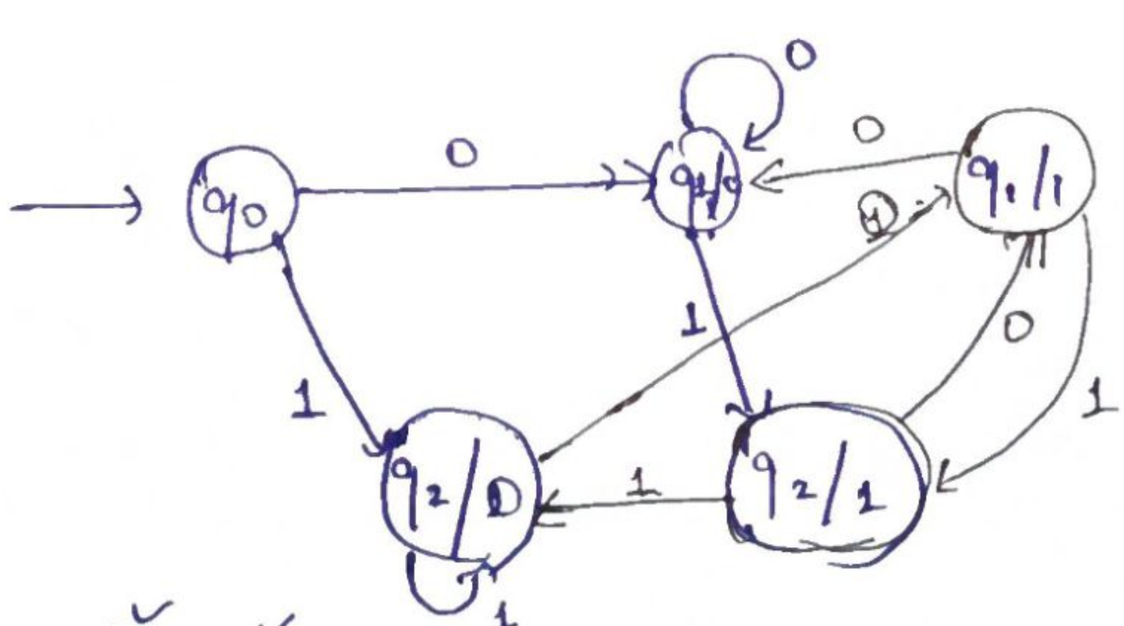


easy.

MEALY TO MOORE CONVERSION :



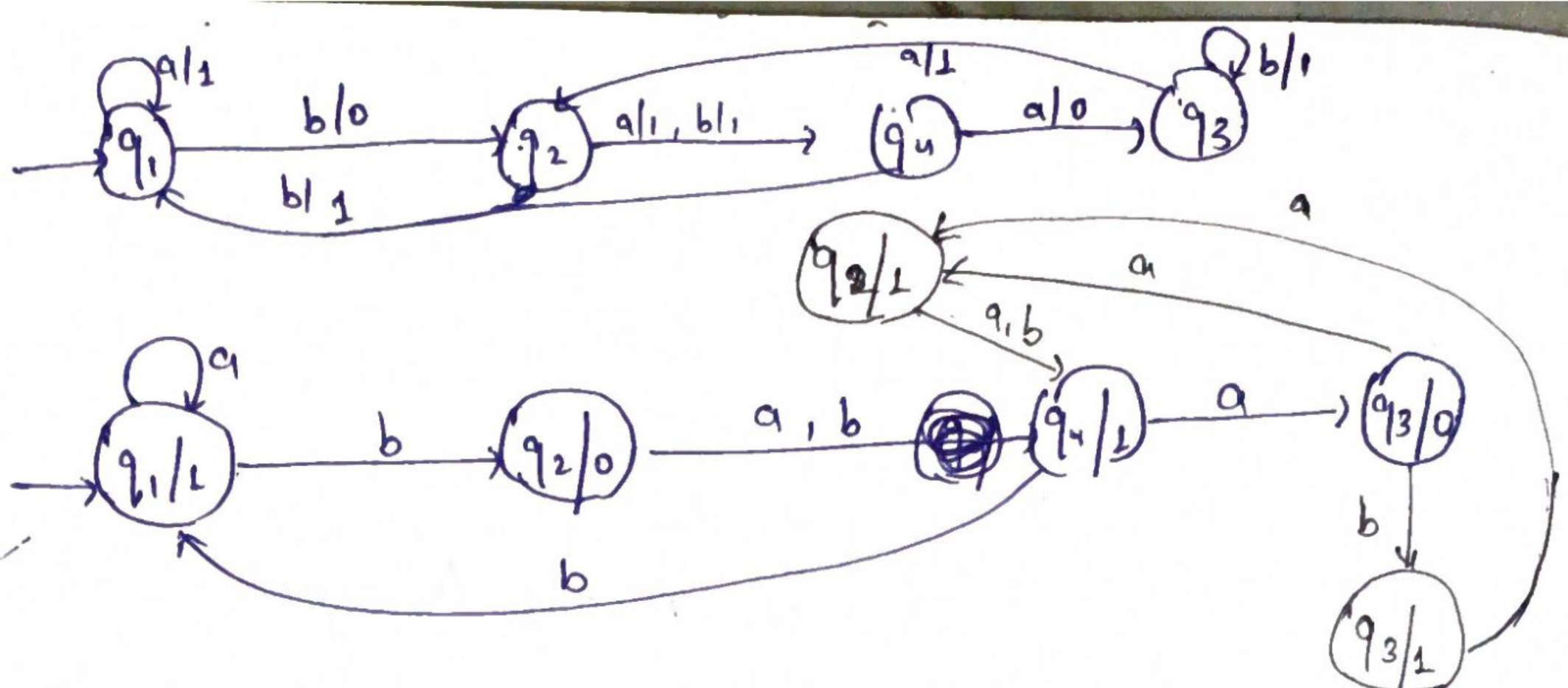
- initialize, q_0 gain initialize
- Now this in we have to see which side arrow go.
- q_0 have 0 and 1 i/p goes to q_1 and q_2
- we ~~too~~ make and write o/p inside next to
- then q_0 done now, q_1
- q_1 has O/P(0) and 0 inside the state also same so it is done easily if not match make new state.
- now q_1 1 i/p go to q_2 but no match



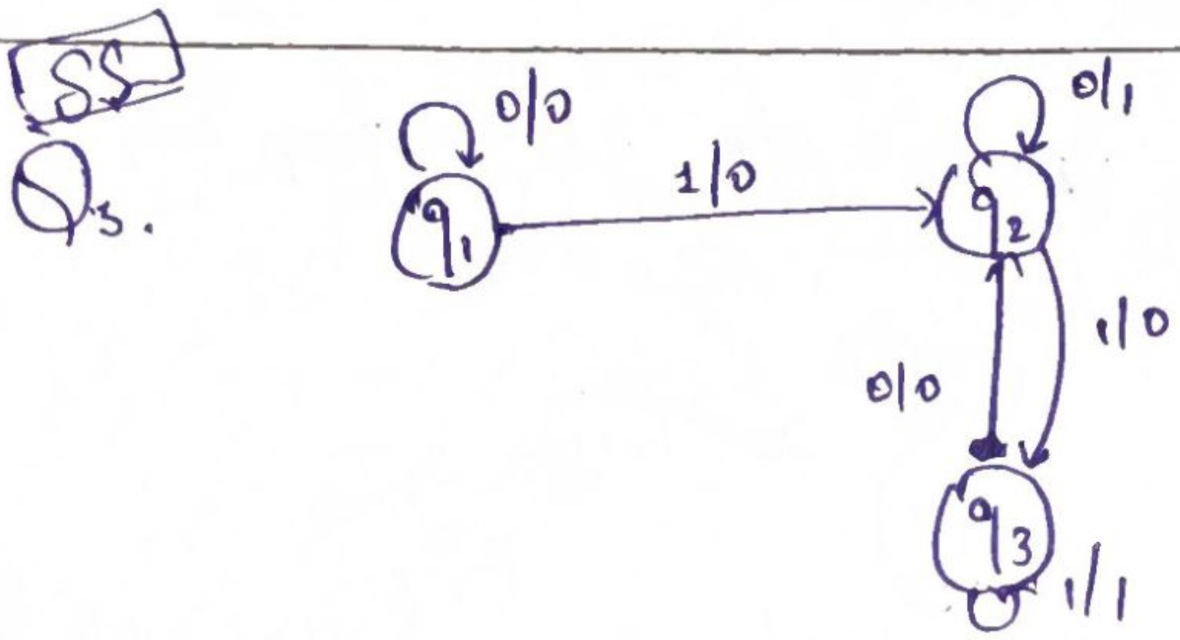
In this we make dummy states.

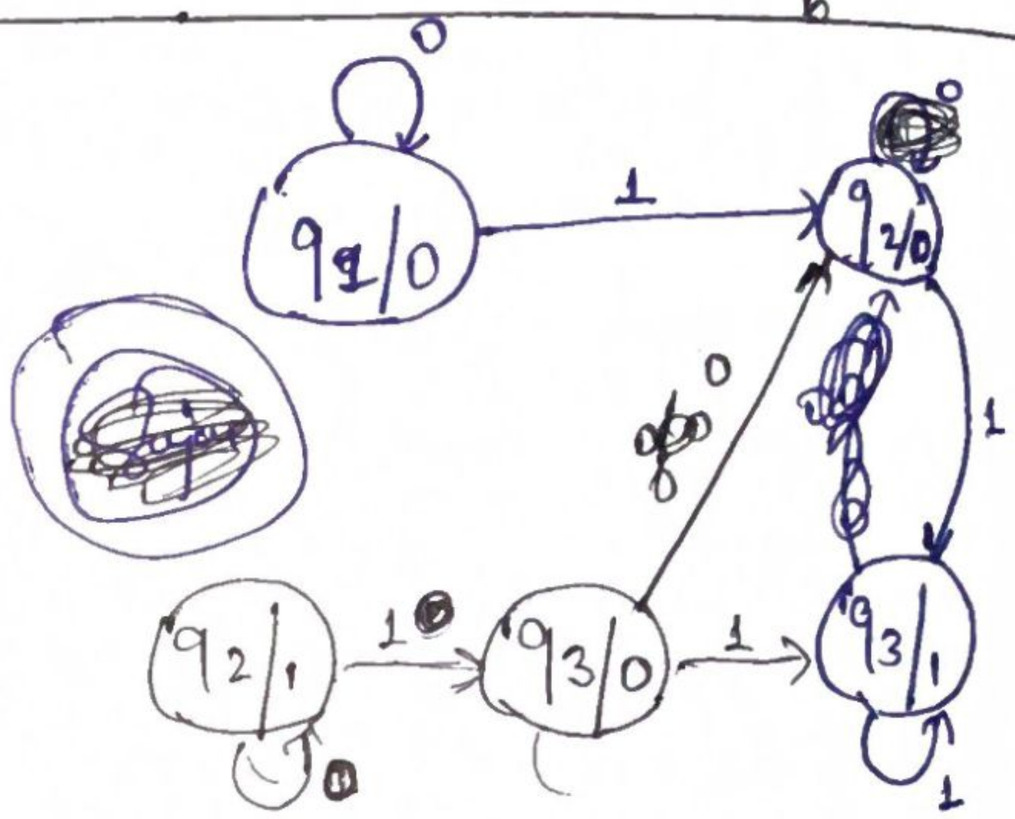
now mealy ⟶ ~~no~~ n state, n O/P
current O/P
n x n =
3 x 2 = 6 states

Q.2



Q.3

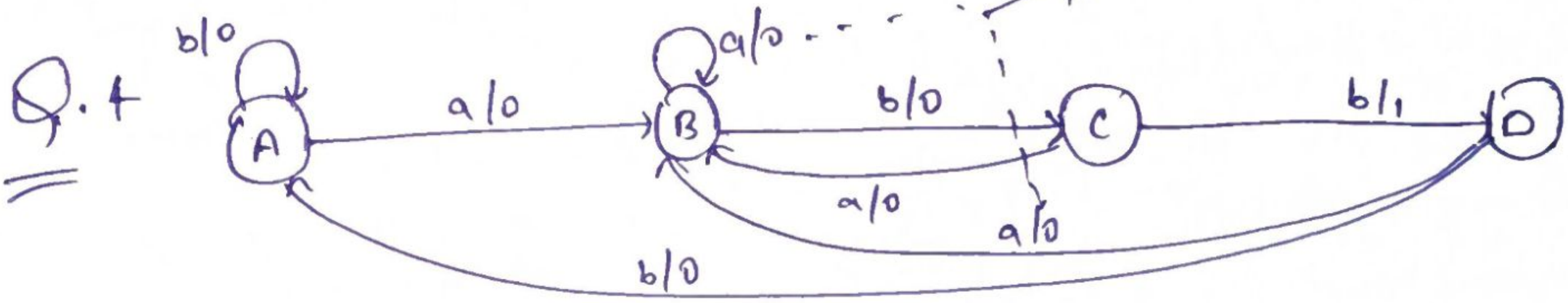


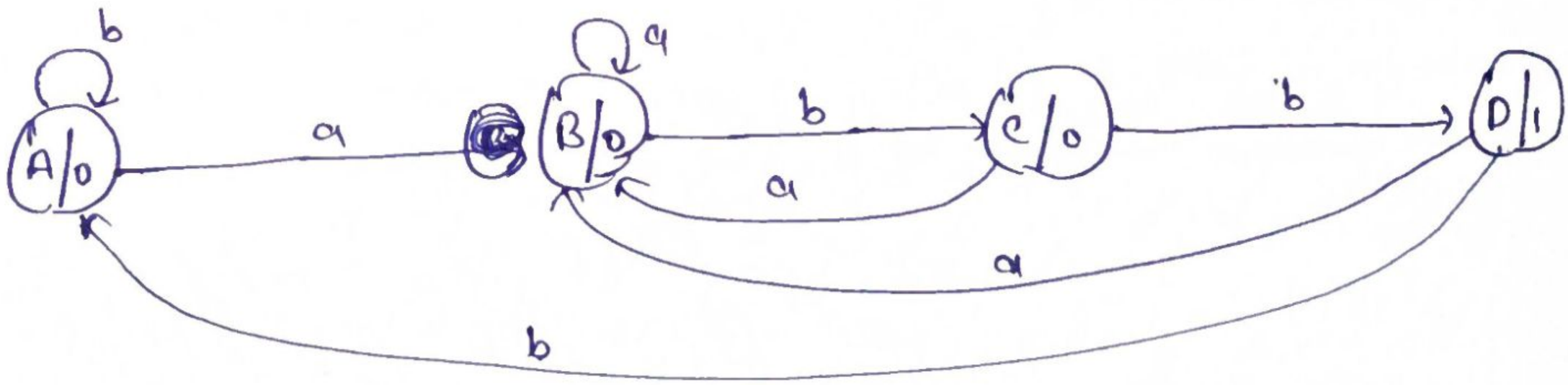


check for the relation last O/P

(0/0)

Q.4

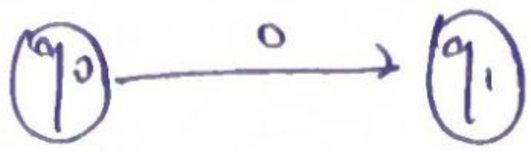




NON DETERMINISTIC FINITE AUTOMATA :-
(choices of moves)

multiple moves
same tuples {NDFA(Q, Σ, q0, F, δ)

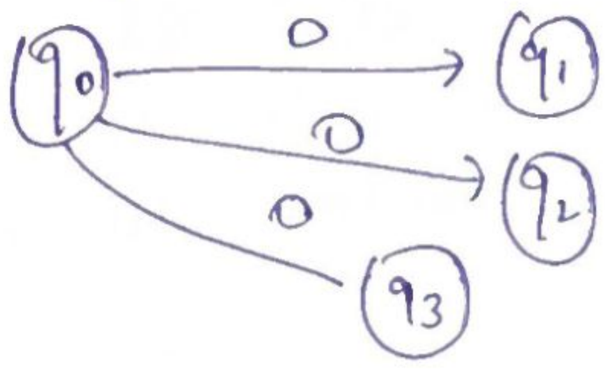
as we know DFA is deterministic means i/p ka O

so  not O ka their single state.

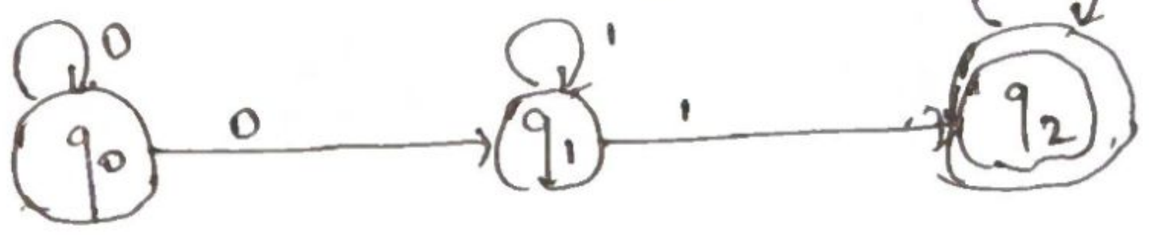
state not same.

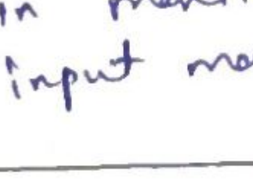
But in NFA has multiple choices means

ek i/p kai diff. states your choice where to go



Ex:



DFA	NFA
- Dead configuration is not allowed means  { hr state main 2 i/p hona chahiye but hum DFA main define krna pdta hain mention krna pdta hai Σ input main	Allowed jhaa mention krna ki jhanjhat nhi hain.
- Multiple choice not available 	are available (ek hi state main same i/p
- ε - move not allowed (null transition) 	allowed (without i/p) 
- Digital computer are deterministic	not
- design difficult to understand	easy.

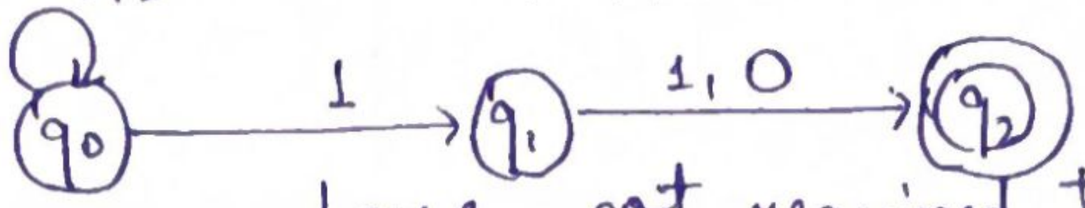
NFA designing :-

Q. NFA all all binary strings in which 2nd last bit 1.

$\Sigma = 0, 1$

$L = \{ 10, 11, 110, 111, \ldots \}$ mean starting sein funkchi patta.

0,1 ← 10, 11 both



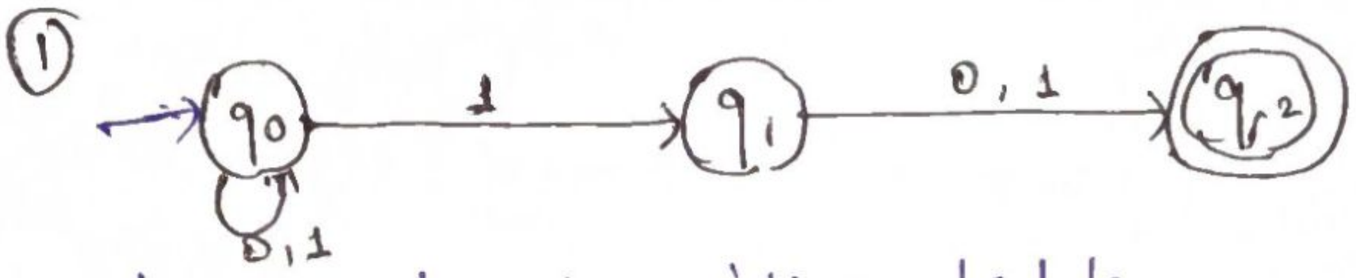
we have not required to complete last state o/p allowed beco we kenow it is dead configuration nhi kra ton bhi chlta hain.

CONVERSION OF NFA to DFA

$\Sigma = 1, 0$

same asprevious.

(1)



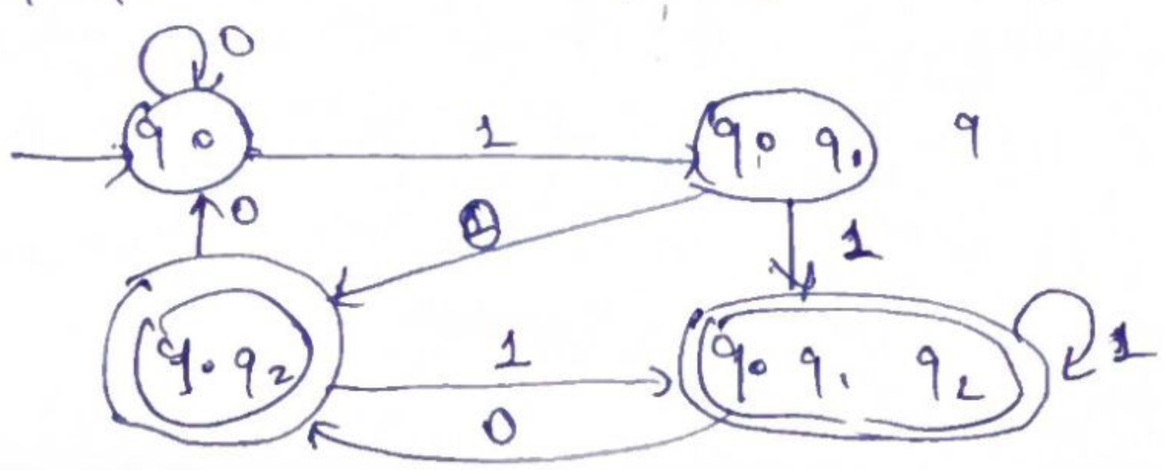
first make transition table. NFA

δ	0	1
→ q_0	q_0	q_0, q_1
q_1	q_2	q_2
Final q_2	-	-

now transition DFA

δ	0	1
expand → q_0	q_0	$q_0 q_1 2$
$q_0 q_1$	$q_0 q_2$	$q_0 q_1 q_2$
$q_0 q_2$	q_0	$q_0 q_1 q_2$
$q_0 q_1 q_2$	$q_0 q_2$	$q_0 q_1 q_2$

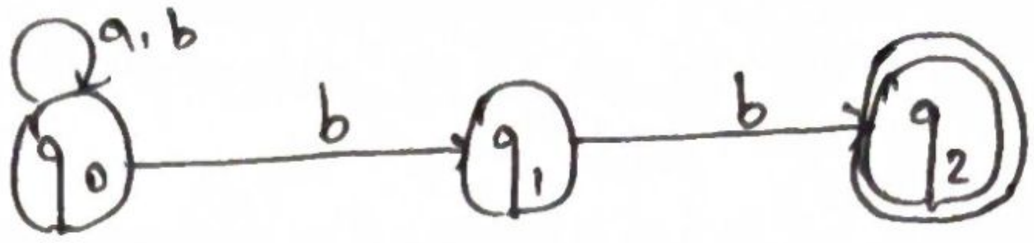
in this DFA trans first write same as first row of NFA then expand $q_0 q_2$ now after $q_0 q_2$ expand now check if repetion or not of expand jaise q_0 kara kiya then $q_0 q_1$ baad $q_0 q_2$ usme q_0 phir aaya toh usgai nhi krna 4 state



two final because q_2 is final in NFA

check karna repeat hua kiya.

(2)



NFA

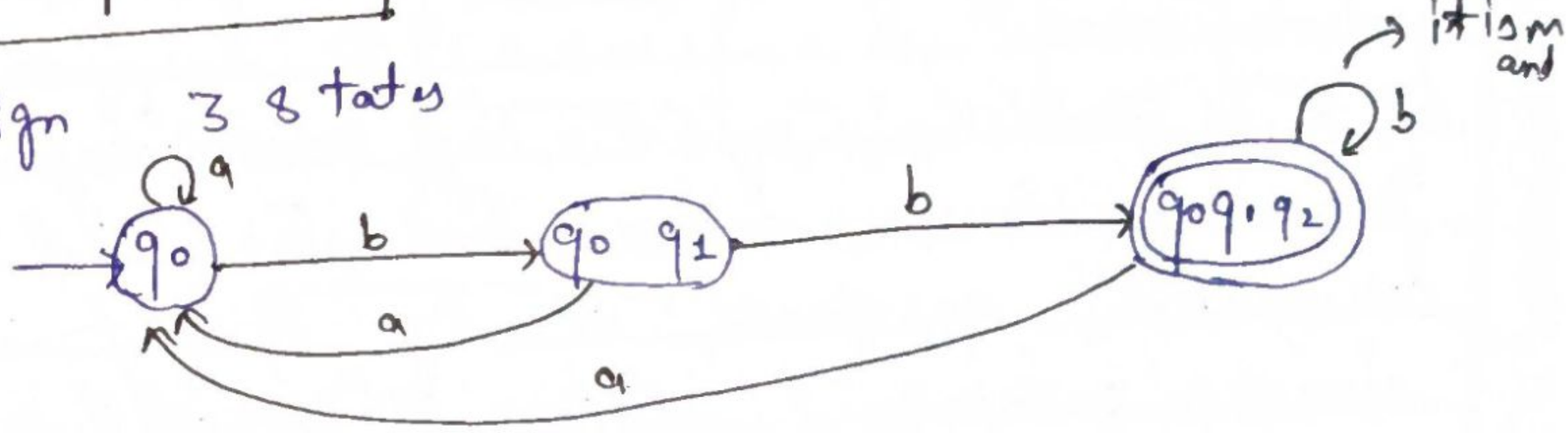
δ	a	b
→ q0	q0	q0, q1
q1	-	q2
(q2)	-	-

DFA

	δ	a	b
→	q0	q0	q0 q1
	q0 q1	q0	q0 q1 q2
	(q0 q1 q2)	q0	q0 q1 q2

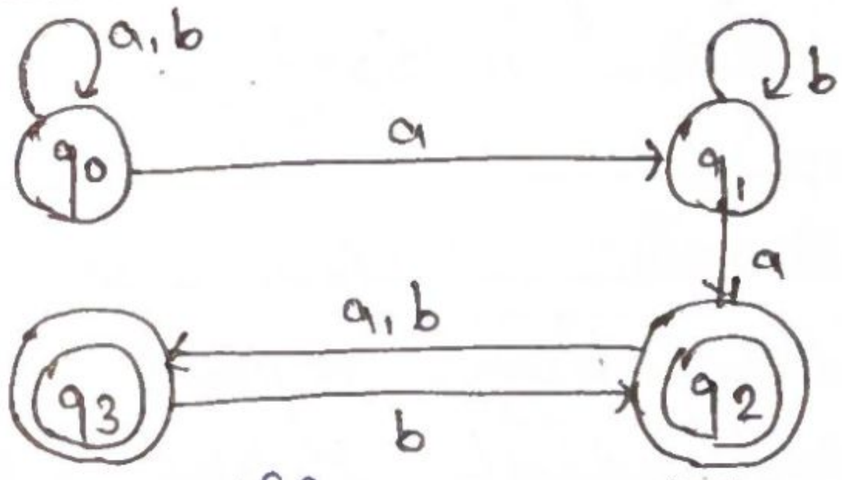
Q.4 →

New Design 3 states



(3)

AE =



AG =

NFA

δ	a	b
→ q0	q0, q1	q0
q1	q2	q1
(q2)	q3	q3
(q3)	-	q2

CH =

Now

BF =

DFA

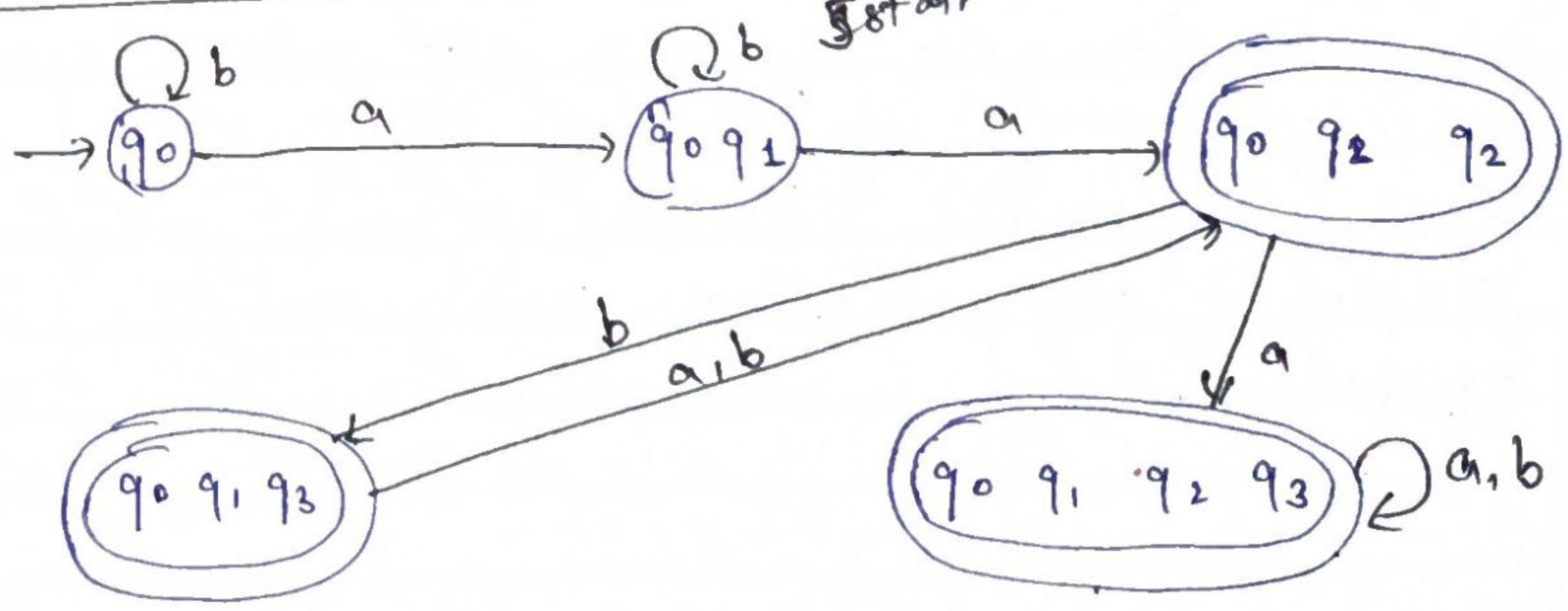
	δ	a	b
1	→ q0	q0 q1	q0
2	q0 q1	q0 q1 q2	q0 q1
3	(q0 q1 q2)	q0 q1 q2 q3	q0 q1 q3
4	(q0 q1 q3)	q0 q1 q2	q0 q1 q2
5	(q0 q1 q2 q3)	q0 q1 q2 q3	q0 q1 q2 q3

5 states

BH =

Stq

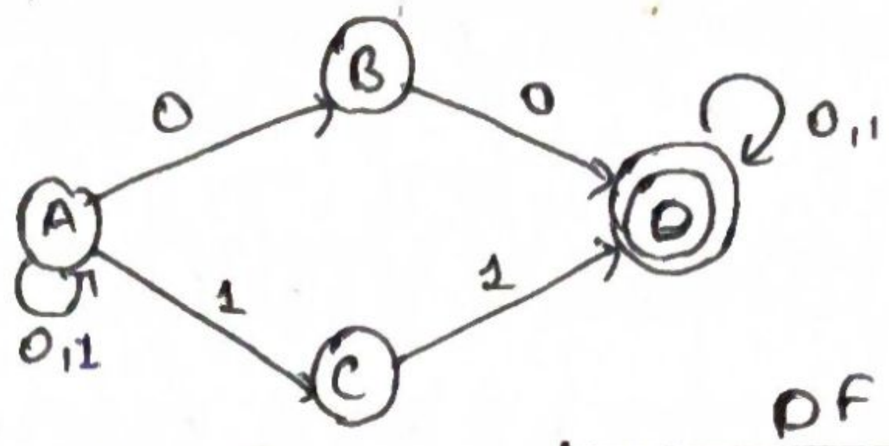
AB =



CE

*

Q.4



no comma

NFA

δ	0	1
→ A	A,B	A,C
B	D	-
C	-	D
(D)	D	D

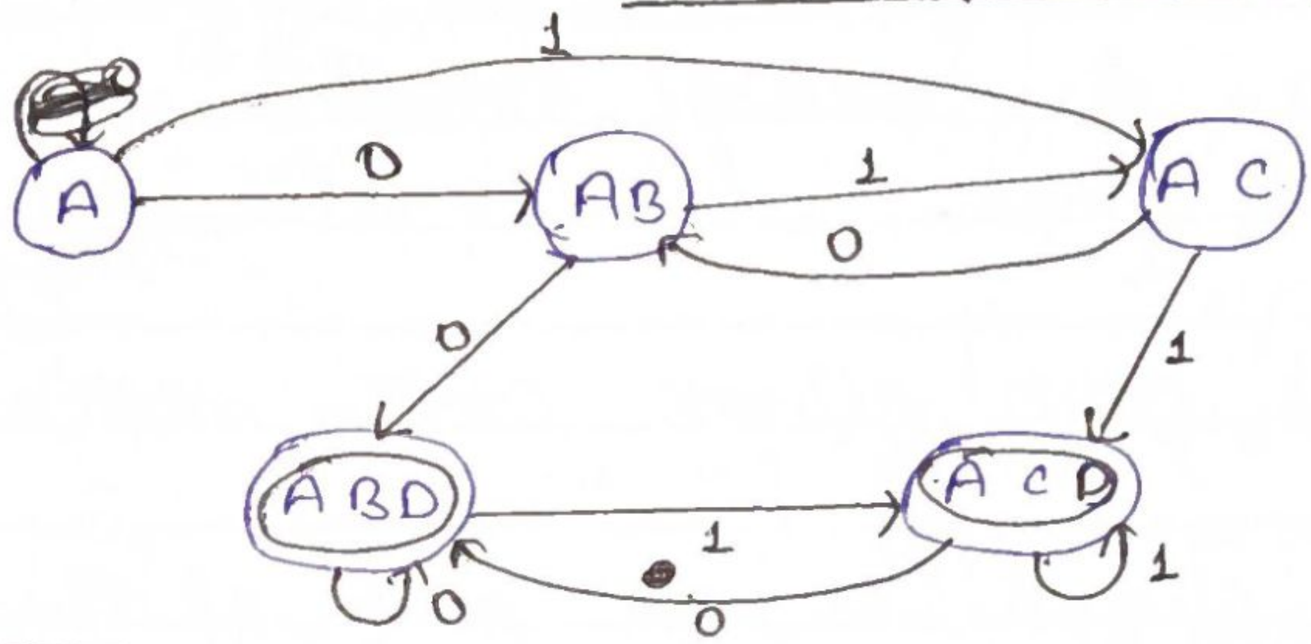
DFA

	δ	0	1
1 →	A	[A,B]	[A,C]
2	[A,B]	[A,B,D]	[A,C]
3	[A,C]	[A,B]	[A,C,D]
4	[A,B,D]	[A,B,D]	[ACD]
5	[A,C,D]	[A,B,D]	[ACD]

5 states

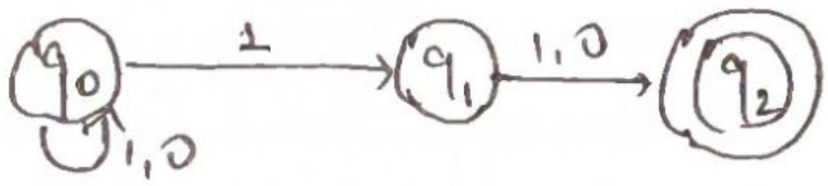
Q.4

→



AE =

Q.5



AG =

NFA

δ	0	10
→q0	q0	q0q1
q1	q2	q2
(q2)	-	-

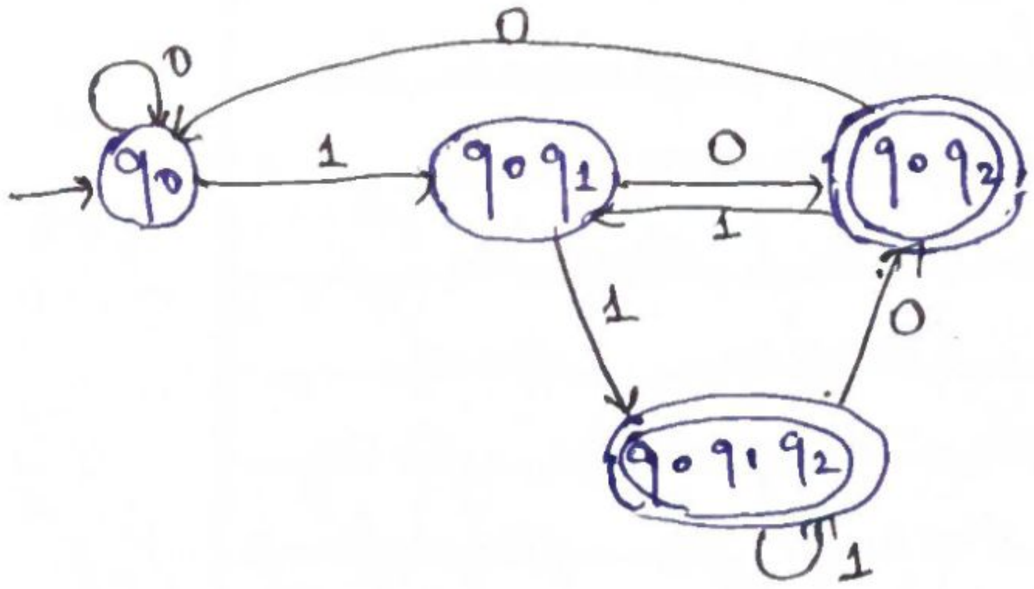
DFA

δ	0	1
→ q0	q0	q0q1
q0q1	q0q2	q0q1q2
(q0q2)	q0	q0q1
(q0q1q2)	q0q2	q0q1q2

CH =

DOW

BF =



BH =

80

STY

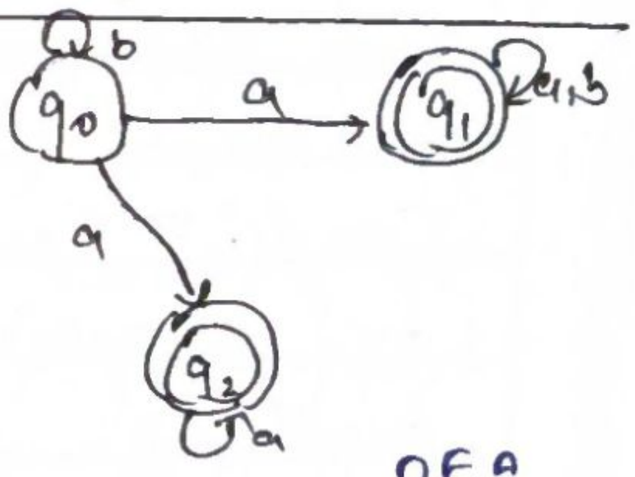
AB =

//

CE

*

Q.6

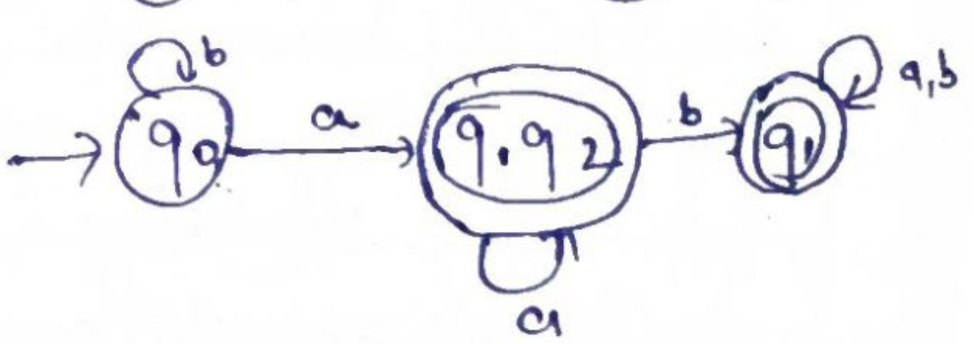


NFA

δ	a	b
→q0	q0,q2	q0
(q1)	q1	q1
(q2)	q2	-

DFA

δ	a	b
→ (q0)	q0,q2	q0
(q0,q2)	q0,q2	q0,q1
(q0,q1)	q0,q2	q0,q1



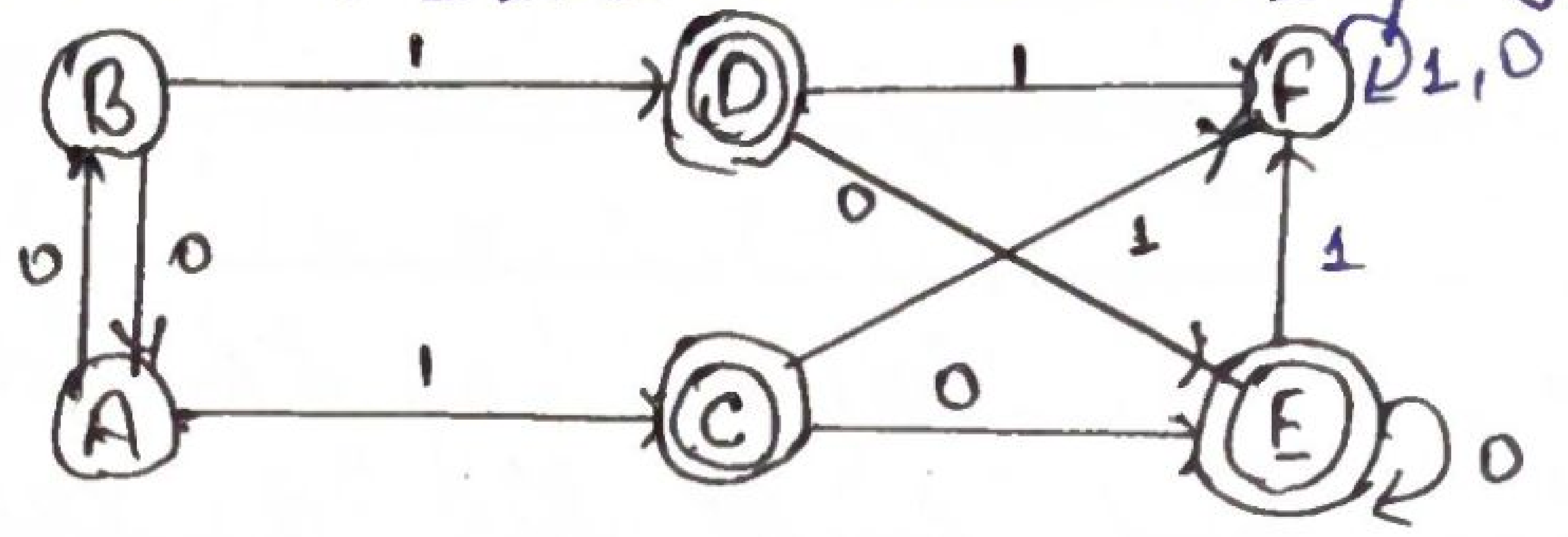
MINIMIZATION of DFA :- (means agar 5 ki jugh, 3 state se hi kaam ho)

There are two method of minimization of DFA :-

- Partition method
- Table filling method or
- My Hill Nerode Theorem

(i) TABLE FILLING METHOD / My Hill Nerode theorem :-

Q.



final state :- D, C, E

Step(i) if ~~Automata~~ given so find final state or not there given or given final

Step(ii) Make a table from automata (start with 3 first step) (last step) in this table with final not two of final (C E X

	A	B	C	D	E
B	✓				
C	✓	✓			
D	✓	✓			
E	✓	✓			
F	✓		✓	✓	✓

Step(iii) from automata mark in table where final state comes C, D, E

Step (v) write unmarked pairs.
AB, CD, CE, DE, AF, BF

Step (v) Now make Iteration of ~~up~~ unpair

AB = $\delta(A,0) = B$, $\delta(B,0) = A$] (AB)] done
$\delta(A,1) = C$, $\delta(B,1) = D$] (CD)

CD = $\delta(C,0) = E$, $\delta(D,0) = E$] (E,E)
$\delta(C,1) = F$, $\delta(D,1) = F$] (F,F)
unmark

CE = $\delta(C,0) = E$, $\delta(E,0) = E$] (E,E)
$\delta(C,1) = F$, $\delta(E,1) = F$] (F,F)
unmark

DE = $\delta(D,0) = E$, $\delta(E,0) = E$] (E,E)
$\delta(D,1) = F$, $\delta(E,1) = F$] (F,F
unma

AF = $\delta(A,0) = B$, $\delta(F,0) = F$]
$\delta(A,1) = C$, $\delta(F,1) = F$]
(B,F) and (C F)
done

BF = $\delta(B,0) = A$, $\delta(F,0) = F$] (AF) ~~unmar~~
$\delta(B,1) = D$, $\delta(F,1) = F$] (D F) done

so in first iteration four unmarked :: Q why C,D
- AB, CD, CE, DE

Condⁿ in both pair any one should be final and another not final so marked ☆

2nd Iteration

AB = δ(A,0) = B] (B,A)] unmarked
δ(B,0) = A]
δ(A,1) = C] (C,D)
δ(B,1) = D

CD = δ(C,0) = E] (E,E) unmarked
δ(D,0) = E]
δ(C,1) = F] (E,F)
δ(D,1) = F

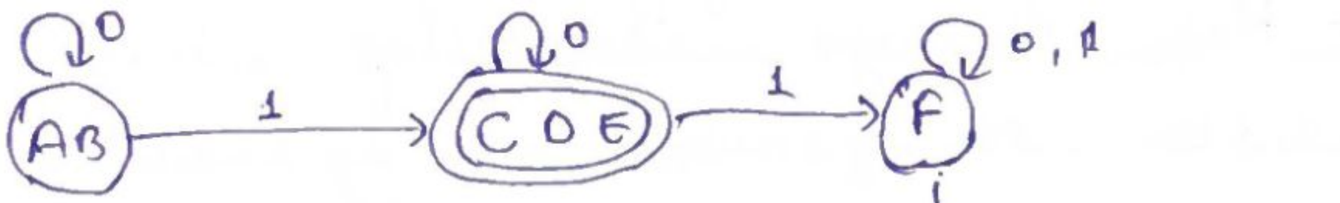
EC = δ(E,0) = E] unmarked
δ(C,0) = E]
δ(E,1) = E]
δ(C,1) = F

ED = also unmarked.

Now same pairs come so stop ~~abc~~ BC

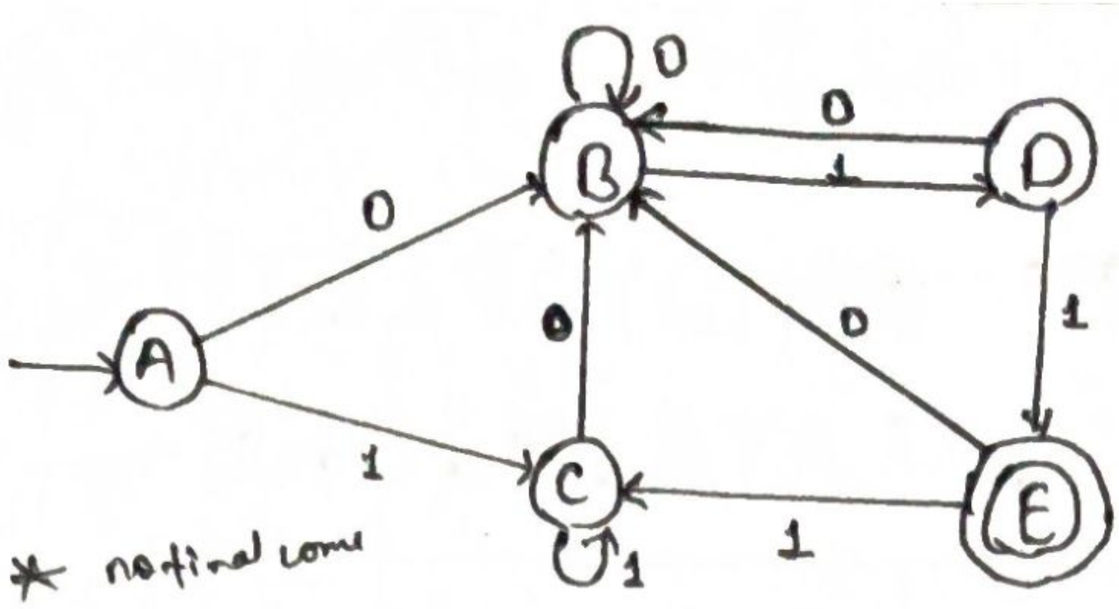
Now write unmarked (AB, CD, CE, DE) AB AC BC ~~find common~~ there is missing F ((to b state this s hagyi minimization)

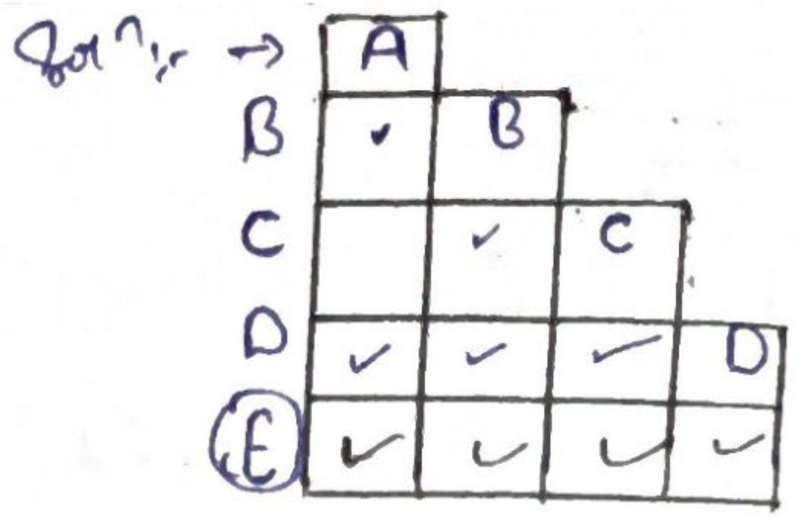
common (AB, CDE CDE



why it take Question

Q.2. Minimize the following DFA using table filling method





where E

Unmarked pairs :- AB, AC, BC, AD, BD, CD

Now 1st Iteration -

Now unmarked are -

AB, AC, BC.

2nd iteration

now again check 1st iteration mein agar

Q.4 AB = $\delta(A,0) = B$, $\delta(B,0) = B$] (B,B)
$\delta(A,1) = C$, $\delta(B,1) = D$] (C,D) unmarked

→ * nothing come so

AB = (B,B) * (C,D) found

· AC = $\delta(A,0) = B$, $\delta(C,0) = B$, $\delta(A,1) = C$, $\delta(C,1) = C$ unm

· * Bc = $\delta(B,0) = B$, $\delta(C,0) = B$, $\delta(B,1) = D$, $\delta(C,1) = C$

AC = (B,B) = (C C) *

AE =

AD = $\delta(A,0) = B$, $\delta(D,0) = B$] (B,B)
✓ $\delta(A,1) = C$, $\delta(D,1) = E$] (C,E)

BD = $\delta(B,0) = B$, ✓ $\delta(D,0) = B$, $\delta(B,1) = D$, $\delta(D,1) = E$

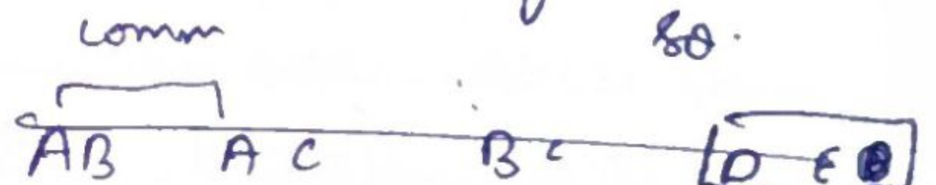
BC = (BB) * (Dc) Bc found

AG = (

so it again comes so.

CD = $\delta(C,0) = B$, $\delta(D,0) = B$] ignore ✓, $\delta(C,1) = C$, $\delta(D,1) = E$] final

CH =



Now

BF =

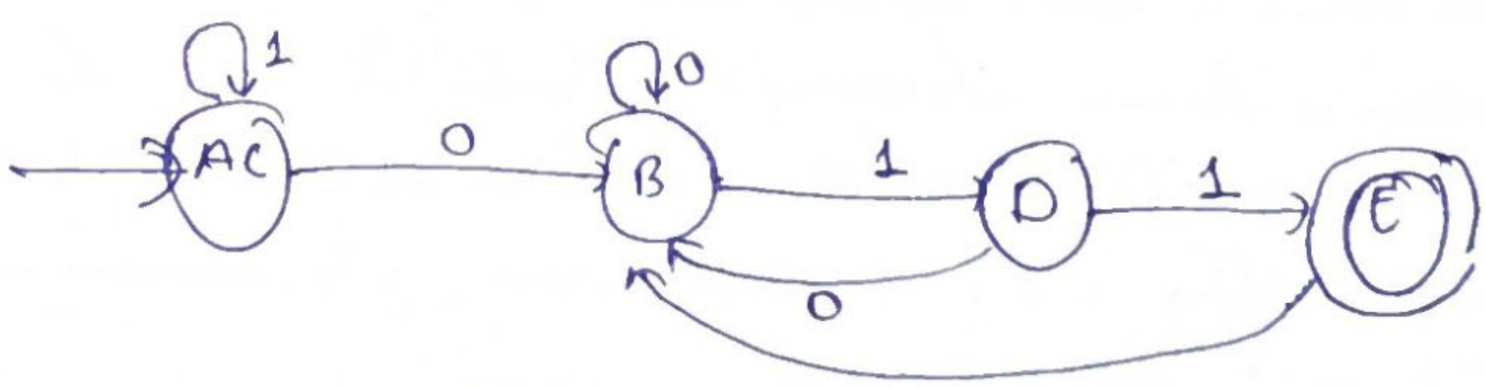
BH =

No so unpair is AC now missing cross check. & states is.

A C ⇝ B D E

so

Stq



AB=

✓

3 state come on to 4 state

CE

*

Q.3. Obtain the distinguisable table for the automata and then minimize state of DFA.

δ	a	b
A	B	F
B	G	C
C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

C* final

Q.4 →

	A						
B	✓	B					
C	✓	✓	C				
D	✓	✓	✓	D			
E	✓	✓	✓	✓	E		
F	✓	✓	✓	✓	✓	F	
G	✓	✓	✓	✓	✓	✓	G
H	✓	✓	✓	✓	✓	✓	✓

✓ marked

* unmarked

Two pairs one final and non final ✓ marked

Now unmarked -

AB, AD, BD, AE, BE, ED, AF, BF, DF, EF, AG, BG, DG, EG, FG.

AH, BH, DH, EH, FH, GH

1st iteration

AB = δ(A, a) = B] (BG)
(B, a) = G

AD = δ(A, a) = B] (B, C)
(D, a) = C
(A, b) = F] (FG)
(D, b) = G

BD = δ(B, a) = G C
(D, a) C
(B, b) C
(D, b) G

C, C
G G

AE = δ(A, a) = B
δ(E, a) = H
(A, b) = F
(E, b) = F

BE = δ(B, a) = G]
(E, a) = H
(B, b) = C
(E, b) = H

ED = δ(E, a) = H]
(D, a) = C
(E, b) = H
(D, b) = G

AF = δ(A, a) = B
(F, a) = C
(A, b) = F
(F b) = G

AG = (A, a) = B]
(G, a) = G
(A, b) = F]
(G, b) = E

AH =

BF = (B, a) = G]
(F a) = C
(B b) = C]
(F b) = G

DF = δ(D, a) = C]
(F, a) = C
(D, b) = G]
(F, b) = G

EF = δ(E, a) = H]
(F, a) = C
(E, b) = H]
(F, b) = G

Now
BF =

BH =

BG = B, a = G]
G, a = G
B, b = C]
G, b = E

DG = δ(D, a) = C]
G, a) = G
D, b) = G]
G, b) = E

EG = δ(E, a = H]
G, a = G
E, b = H]
G, b = E

FG = δ(F, a) = E]
(G, a) = G
(F, b) = G]
(G, b) = E

So

AB =

~~So unmarked (*) → {BD, AE, BF, DF, EG}~~

CE
*

AH = A, a = B]
H, a = H
A, b = F]
H b = C

BH = B, a = G]
H, a = G
B, b = C]
H, b = C

DH = D, a = C]
H, a = G
D, b = G]
H, b = C

EH = E, a = H]
H, a = G]
E, b = H]
H, b = E

FH = F, a = C
H, a = G
F b = G
H b = C

So unmarked = { AE, DF, AG, EG, BH, DH, FH }

2nd iteration

~~BD = δ(B,a) = G~~ ~~(D,a) = C~~ ~~(B,b) = C~~ ~~(D,b) = G~~

AE = δ(A,a) = B
(E,a) = H
(A,b) = F
(E,b) = F

{FH marked in first iterat

~~This all again~~ come So ~~Stop~~

So unmarked = {

DF = δ(D,a) = C
(F,a) = C
(D,b) = G
(F,b) = G

AG = δ(A,a = B
G,a = G
A,b = F
G,b = E
{B ~~marked~~}

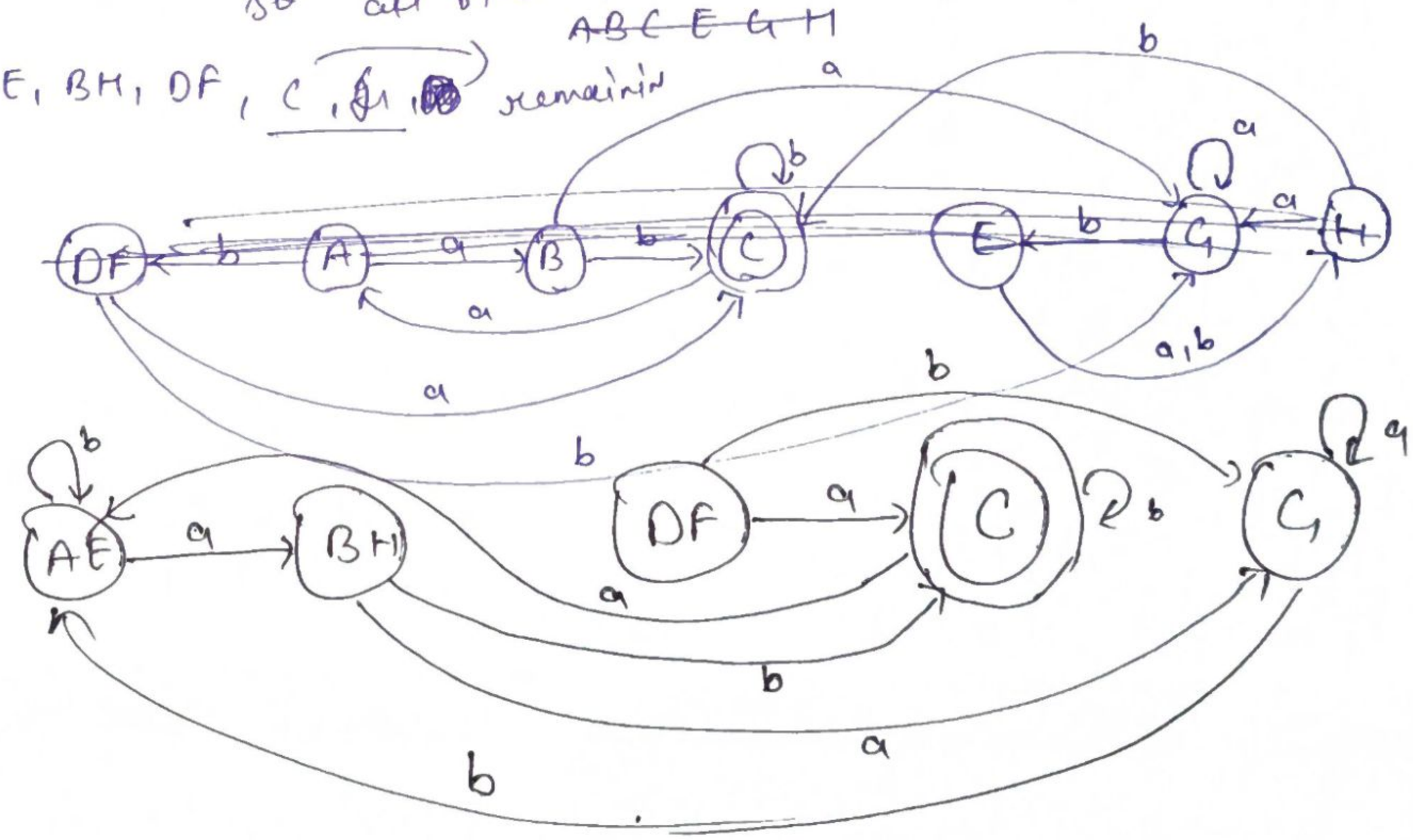
EG = δ(E,a) = H
δ(G,a) = G
δ(E,b) = H
δ(G,b) = E
{HE marked}

BH = δ(B,a) = G] (G,G)
δ(H,a) = G
δ(B,b) = G] (E,C)
δ(H,b) = C
{C & BH marked}

Now in 2nd iteration unmarked is DF, BH, E

So all other → ~~A B C E G H~~

AE, BH, DF, C, FH, E remaining



Date-	Day-	Time-

Meal	Type of food thrown	Category A. Cooked B. Uncooked C. Inedible D. Beverage	How much thrown (gram or litre or count)	Why not eaten
Breakfast				
Lunch				
Dinner				
In-between meals (if any)				

4

→

f	D	1
A	B	A
B	A	C
C	D	B
D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

	A						
B	~	B					
C	✓	✓	C				
D	✓	✓	✓	D			
E	✓	✓		✓	E		
F	~		✓	✓	✓	F	
G		✓	✓	✓	✓	~	G
H	✓	✓	✓	✓	✓	✓	✓

Step(i) we mark final pairs acc. to table that is

D. Now write unmarked.

AB, AC, BC, BE, BE, CE, AF, BF, CF, EF

AG, BG, CG, EG, FG, AH, BH, CH, EH

FH, GH

Step(ii) I st iteration check all.

AB = { A D = B, B D = A, A I = A, B I = C } , AC = B D A B , BC = A D C B

AE = B D A F , BE = A D C F , CE = D D B F , AF = B G A E , BF = A G C E , CF = D G B E , EF = D G F E ,

AG = { B F A G , BG = A F G G , CG = D F B G , EG = D F F G , FG = G F E G , AH = B G A D , BH = A G C D

CH = D G G B D , EH = D G F D , FH = G G E D , GH = F G G D

Now check → AB = ✱, AC = ✓, BC = ✓, AE = ✓, BE = ✓, CE = ✱, AF = ✱

BF = ✱, CF = ✓, EF = ✓, AG = ✱, BG = ✱, CG = ✓, EG = ✓, FG = ✱, AH = ✓

BH = ✓, CH = ✓, EH = ✓, FH = ✓, GH = ✓. (I st iteration mark [-])

So unmarked → ~~AB, CE, BF, AF~~ AB, AD, AF, BF, CE, AG, BG, FG

Step(iii) 2nd iteration

AB = B A ✓ A C] marked in first

AF = B G ✓ A E] marked in 1st

BF = A G ✱ C E] — not in table

CE = D D B F] ignore

✱ B F] — unmarked in 1st

AG = B F A G]] both unmarked in 1st iter

✱ 2

BG = A F C G] comes in 1st

✓

Convert the following ε NFA to DFA

ε ghab mai

initial → first bcoz ye first isko ε closure

$A = \varepsilon\text{-closure}(q_0) = \{q_0\ q_1\ q_2\} \longrightarrow A$

$\delta(A,a) = \varepsilon\text{-closure}(\delta(q_0\ q_1\ q_2), a) = E\text{-clos}(q_0) = A$

$\delta(A,b) = \varepsilon\text{-closure}(\delta(q_0\ q_1\ q_2), b) = \varepsilon\text{-clos}(q_1) = \{q_1, q_2\} = B$

$\delta(A,@c) = \varepsilon\text{-closure}(\delta(q_0 q_1 q_2), c) = \varepsilon\text{-clos}(q_2) = \{q_2\} = C$

A

$A = \{q_0 q_1 q_2\}$ $B = \{q_1, q_2\}$ $C = \{q_2\}$

$\delta(B,a) = \varepsilon\text{-clos}(\delta(q_1,q_2), a) = \varepsilon\text{-clos}(\phi) = D$ $q_1 \xrightarrow{a} q_2$

q_1 a lekar q_2

$\delta(B,b) = \varepsilon\text{-clos}(\delta(q_1,q_2), b) = \varepsilon\text{-clos}(q_1) = B$

$\delta(B,c) = \varepsilon\text{-clos}(\delta(q_1,q_2), c) = \varepsilon\text{-clos}(q_2) = C$

Continue Jab tak new state or ~~set~~ nhi mil jaata

∴

$\delta(D,a) =$ ~~δ~~ $\delta(D,b) = \delta(D,c) = D$

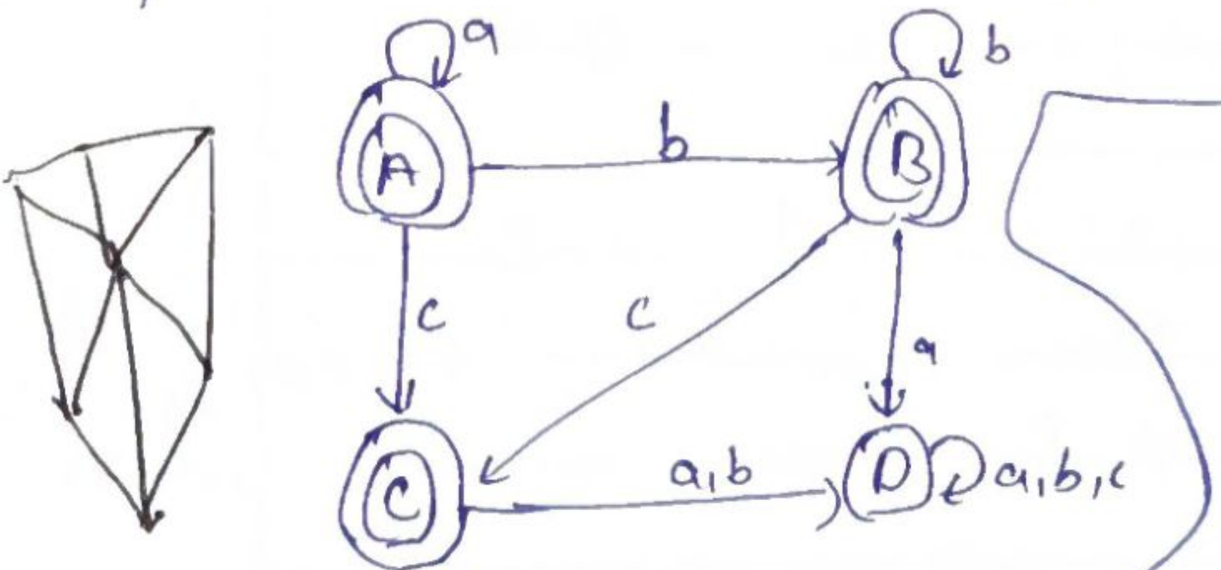
$\delta(C,a) = \varepsilon\text{-clos}(\delta(q_2,a)) = \phi = D$

$\delta(C,b) = \varepsilon\text{-clos}(\delta(q_2,b)) = \phi = D$

$\delta(C,c) = \varepsilon\text{-clos}(\delta(q_2,c)) = \varepsilon\text{-clos}(q_2) = C$

- first initial where to go with ε

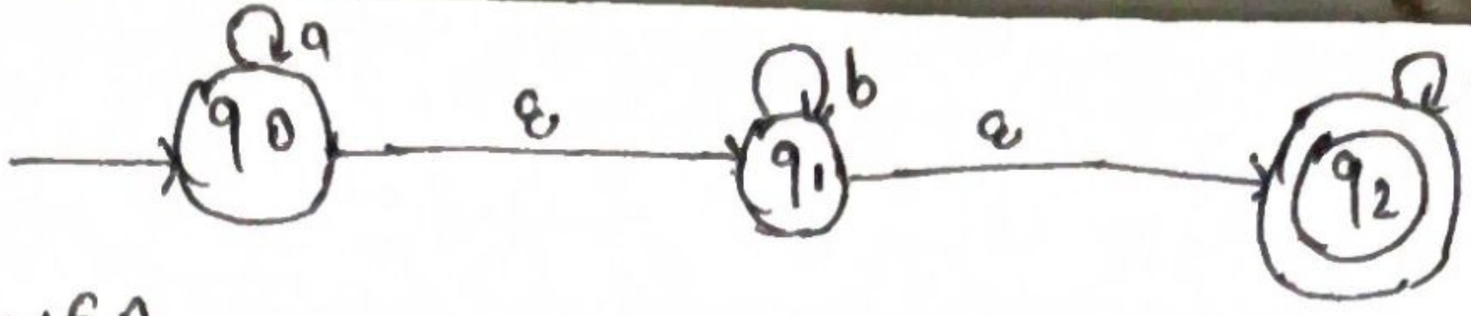
jis jis mai q_2 wo find.



	a	b	c	e
q_0	$q_0 q_1 q_2$	x	-	$q_0 q_2$
q_1	x ~~$q_0 q_1 q_2$~~	$q_1 q_2$	-	~~$q_0 q_1 q_2$~~ q_2
q_2	x	x	q_2	x

←	a	b	c
$q_0 q_1 q_2$	$q_0 q_1 q_2$	$q_1 q_2$	q_2
$q_1 q_2$	x	$q_1 q_2$	q_2
q_0	-	-	q_2

Q.



ε-NFA

δE	a	b	c	ε
q0	{q0}	ϕ	ϕ	{q1}
q1	ϕ	{q1}	ϕ	{q2}
q2	ϕ	ϕ	{q2}	ϕ

ε-closure (q0) = {q0 q1, q2}

ε-closure (q1) = {q1 q2}

ε-closure (q2) = {q2}

DFA

δD	a	b	c
→q0 q1 q2 *	q0 q1 q2 ✓	q1 q2	q2
q1 q2 *	ϕ	q1 q2 ✓	q2
q2 *	ϕ	ϕ	q2

union

(1)

$\delta_D(\{q_0 q_1 q_2\}, a) = \varepsilon\text{-closure}(\delta_E(\{q_0 q_1 q_2\}, a))$

$= \varepsilon\text{-closure}(\delta_E(q_0, a) \cup \delta_E(q_1, a) \cup \delta_E(q_2, a))$

$= \varepsilon\text{-closure}(\{q_0\} \cup \phi \cup \phi)$

$= \varepsilon\text{-closure}(q_0)$

$= \{q_0 q_1 q_2\}$

(2)

$\delta_D(\{q_0 q_1 q_2\}, b) = \varepsilon\text{-closure}(\delta_E\{q_0 q_1 q_2\}, b)$

$= \varepsilon\text{-closure}(\delta_E(q_0, b) \cup \delta_E(q_1, b) \cup \delta_E(q_2, b))$

$= \varepsilon\text{-closure}(\phi \cup \{q_1\} \cup \phi)$

$= \varepsilon\text{-closure}(q_1)$

$= \{q_1 q_2\}$

$\delta_D(\{q_0 q_1 q_2\}, \varepsilon) =$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup q_2)$

$= \varepsilon\text{-closure}(q_2)$

$= \{q_2\}$

New (3)

q_2

$\delta_D(\{q_2\}, a) = \phi$

$\delta_D(\{q_2\}, b) = \phi$

$\delta_D(\{q_2\}, c) = q_2$

Now

$q_1 q_2$

$\delta_D(\{q_1 q_2\}, a) = (\phi \cup \phi)$

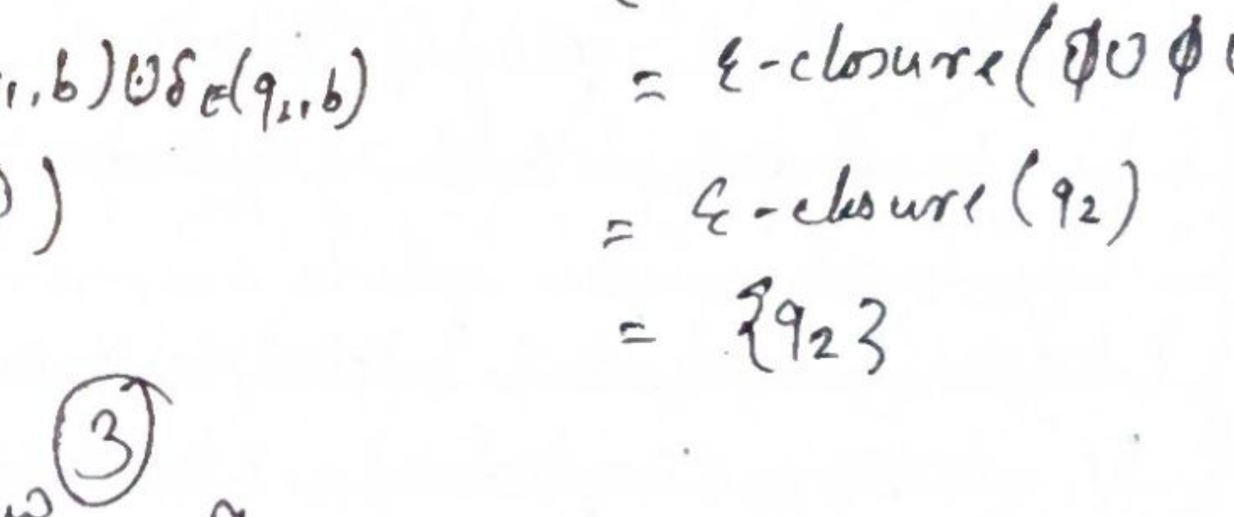
$= \phi$

$\delta_D(\{q_1 q_2\}, b) = (q_1 \cup \phi)$

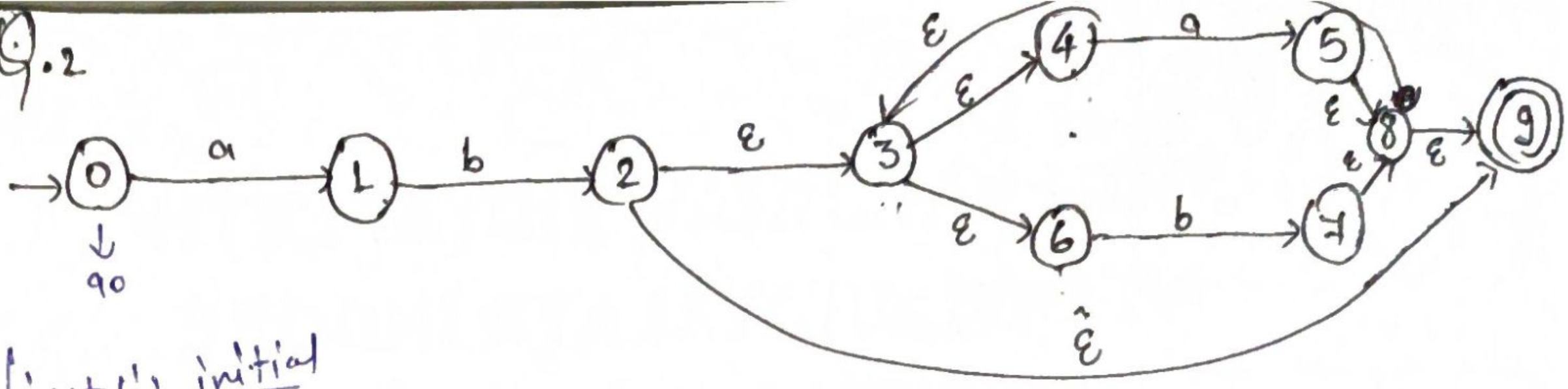
$= q_1 q_2$

$\delta_D(\{q_1 q_2\}, c) = (\phi \cup q_2)$

$= q_2$



Q.2



first(i) initial

$q_0 = \varepsilon\text{-clos}(q_0)$

$= \varepsilon\text{-clos}(0)$

$= \{0\} \longrightarrow A$

row (a, b)

$\delta(A,a) = \varepsilon\text{-clos}(\delta(A,a))$

$= \varepsilon\text{-clos}(0,0)$

$= \{1\} \longrightarrow B$

$\delta(A,b) = \varepsilon\text{-clos}(\delta(A,b))$

$= \varepsilon\text{-clos}(0,b)$

$= \phi$

check and continously make table

		a	b
→	A	B	–
	B	–	C
*	C	D	E
go to final	*	D	E
come in this.	*	E	E

$\delta(B,a) = \varepsilon\text{-clos}(\delta(B,a))$

$= \varepsilon\text{-clos}(1,a)$

$= \phi$

$\delta(B,b) = \varepsilon\text{-clos}(\delta(B,b))$

$= \varepsilon\text{-clos}(1,b)$

$= \varepsilon\text{-clos}(2)$

epsilon of 2 $= \{2,3,4,6,9\} \longrightarrow C$

$\delta(C,a) = \varepsilon\text{-clos}(\delta(C,a))$

$= \varepsilon\text{-clos}(\delta\{2,3,4,6,9\},a)$

$= \varepsilon\text{-clos}\{5\}$

new state epsilon $= \{5,8,9,3,4,6\} \longrightarrow D$

$\delta(C,b) = \varepsilon\text{-clos}(\delta(C,b))$

$= \varepsilon\text{-clos}(\{2,3,4,6,9\},b)$

$= \varepsilon\text{-clos}\{7\}$

$= \{7,8,9,3,4,6\} \longrightarrow E$

$\delta(0,a) = \varepsilon\text{-clos}(\delta(0,a))$

$= \varepsilon\text{-clos}(\{5,8,9,3,4,6\},a)$

$= \varepsilon\text{-clo}\{5\}$

$= \{3,4,5,6,8,9\} \longrightarrow D$

$\delta(D,b) = \varepsilon\text{-clos}(\delta(D,b))$

$= \varepsilon\text{-clos}(\{5,8,9,3,4,6\},b)$

$= \varepsilon\text{-clos}[7]$

$= \{7,8,9,3,4,6\} \longrightarrow E$

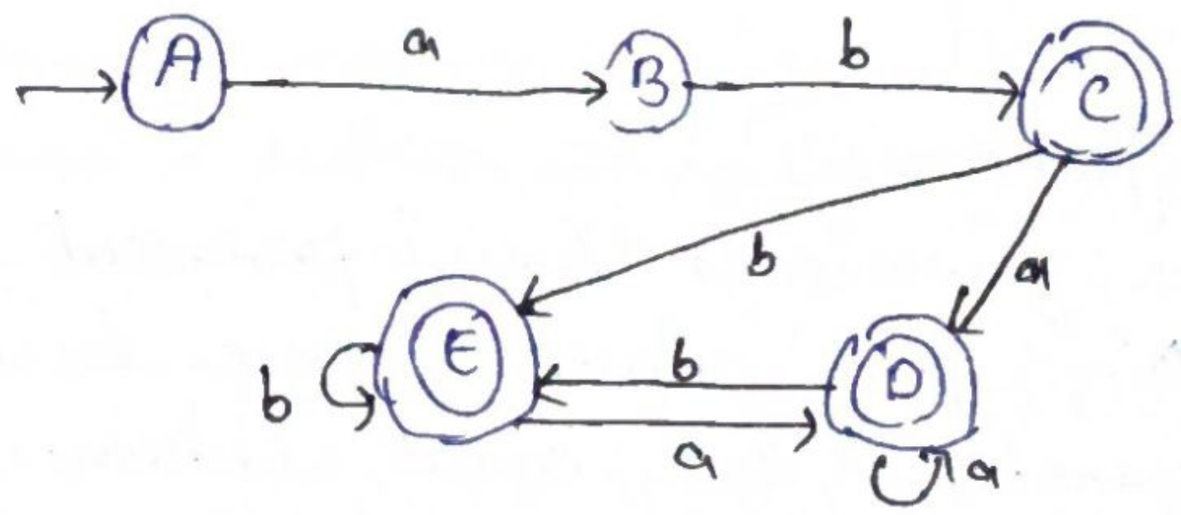
$\delta(E,a) = \varepsilon\text{-clos}(\delta\{E,a\}) = \varepsilon\text{-clos}(\{5,6,9,3,4,6\},a)$

$= \varepsilon\text{-clos}\{5\} = \{3,4,5,6,8,9\} \longrightarrow D$

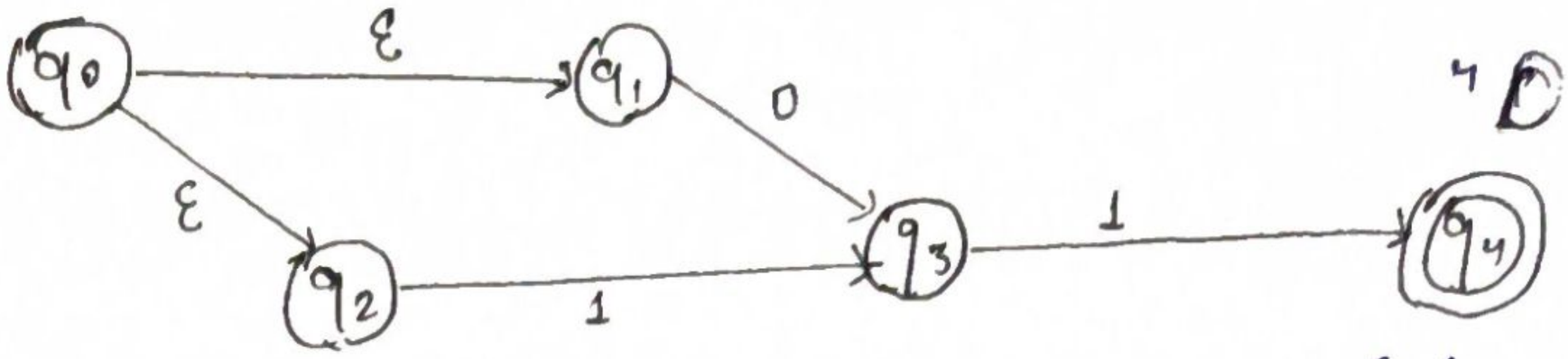
$\delta(E,b) = \varepsilon\text{-clos}(\delta(E,b)) = \varepsilon\text{-clos}(\{7,8,9,3,4,6\},b)$

$\varepsilon\text{-clos}\{7\} = \{7,8,9,3,4,6\} \longrightarrow E$

now same repeat step



Q.3.



for initial

$A = \varepsilon\text{-clos}(q_0) = \{q_0, q_1, q_2\} \longrightarrow A$

for A

$\delta(A,0) = \varepsilon\text{-clos}(\delta(q_0 q_1 q_2), 0) = \varepsilon\text{-clos}(q_3) = q_3 \rightarrow B$

$\delta(A,1) = \varepsilon\text{-clos}(\delta(q_0 q_1 q_2), 1) = \varepsilon\text{-clos}(q_3) = q_3 \rightarrow B$

for B

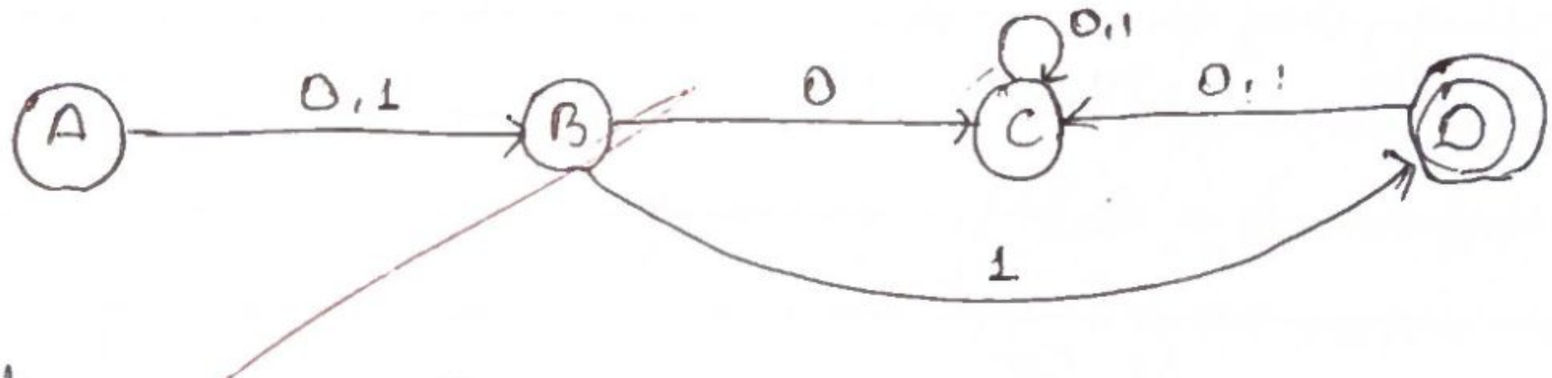
$\delta(B,0) = \varepsilon\text{-clos}(\delta(q_3), 0) = \varepsilon\text{-clos}(\phi) \Rightarrow C$

$\delta(B,1) = \varepsilon\text{-clos}(\delta(q_3), 1) = \varepsilon\text{-clos}(q_4) \Rightarrow D$

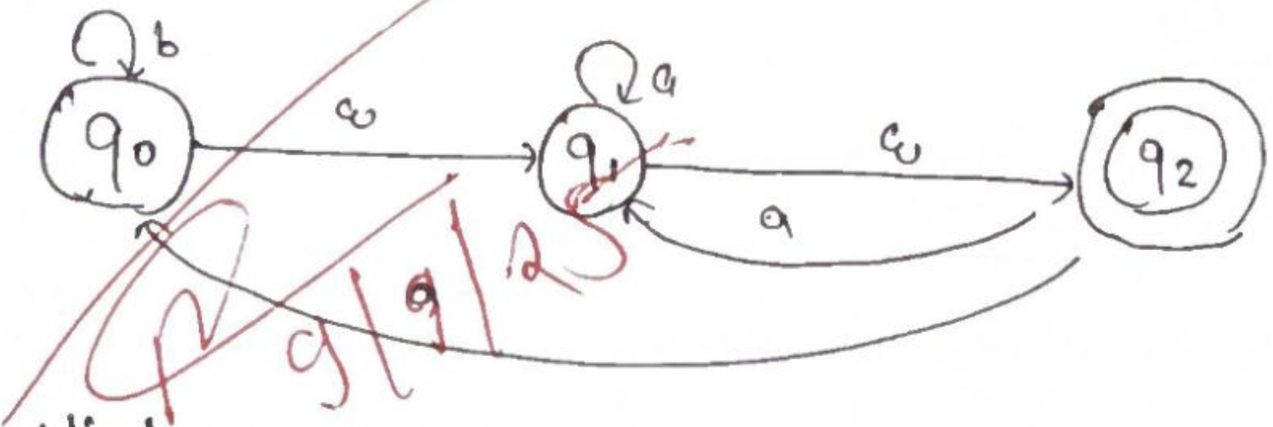
for $\delta(C,0) = (C,1) = \phi$

for C: $\delta(D,0) = \varepsilon\text{-clos}(\delta(q_4), 0) = \varepsilon\text{-clos}(\phi) = C$

$\delta(D,1) = \varepsilon\text{-clos}(\delta(q_4), 1) = \varepsilon\text{-clos}(\phi) = C$ (q_4 final.)



Q.4.



δ	a	b
A	B	C
B	B	D

for Initial

$A = \varepsilon\text{-clos}(q_0) = \{q_0, q_1, q_2\} = A$

$(A,a) = \varepsilon\text{-clos}(A,a) = \varepsilon\text{-clos}\{q_0 q_1 q_2 (a)\} = \{q_1, \} = B$

$(A,b) = \varepsilon\text{-clos}(A,b) = \varepsilon\text{-clos}(q_0 q_1 q_2 (b)) = \{q_0\} = C$

$(B,a) = \varepsilon\text{-clos}(B,a) = \varepsilon\text{-clos}\{(q_1,), \} = \{q_1\} = B$

$(B,b) = \varepsilon\text{-clos}(B,b) = \varepsilon\text{-clos}\{(q_1), b\} = \{\phi\} = D$