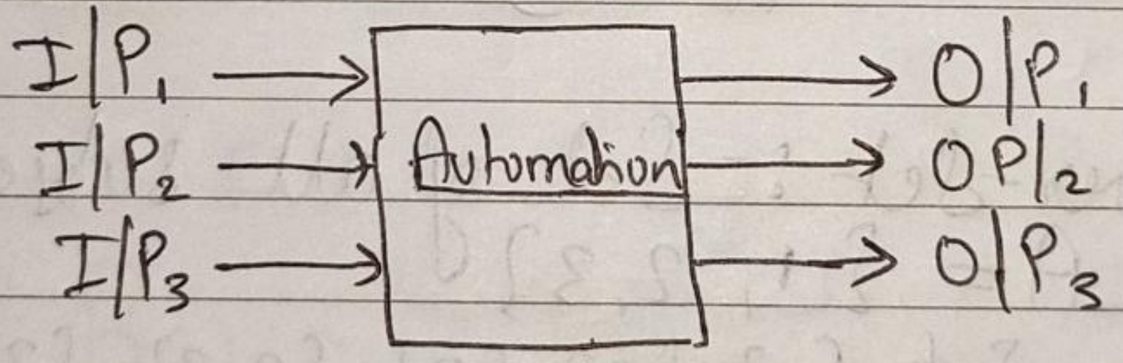
Automata

Unit - 1

Automata :- Automata is a kind of machine which takes some string as input and this input goes through a finit no. of states and may enter in the final state.

Automata :- m/c + sely + Calculation + Theory of computation.

↳ Automata is a machine which have the capability to do everything on its own and can do calculations with the help of theory of computation.



Sets

→ Set is a collection of object which can be finit or infinite.

OPERATIONS :-

① Union :- $A \cup B = \{n : n \in A \text{ or } n \in B\}$

$A = \{1, 2, 3\}$ $B = \{2, 3, 5, 6\}$

$A \cup B = \{1, 2, 3, 5, 6\}$

(2) Intersection :- $A \cap B = \{n : n \in A \text{ and } n \in B\}$

$A = \{1, 2, 3\}$ $B = \{2, 3, 4, 5\}$

$A \cap B = \{2, 3\}$.

(3) Difference :- $A - B = \{n : n \in A \text{ and } n \notin B\}$

$A = \{1, 2, 3\}$ $B = \{3, 4, 5\}$

$A - B = \{1, 2\}$

(4) Cartesian Product :-

$A = \{1, 2\}$ $B = \{3, 4\}$

$A \times B = \{(1,3), (1,4), (2,3), (2,4)\}$

(5) Subset :- A is subset of B if all element of A is present in that of B.

$A = \{1, 2\}$ $B = \{1, 2, 3, 4, 5\}$

$A \subset B$.

(6) Power-Set :- Set of all subset of A.

$A = \{1, 2, 3\}$

$P(A) = \{\phi, \{1\}, \{2\}, \{3\}, \{2,3\}, \{3,1\}, \{1,2\}, \{1,2,3\}\}$

Inductive Proof :-

Basically Mathematical Induction.

Q Proove $1 + 2 + 3 + 4 + \dots = \frac{n(n+1)}{2}$.

Ans → Step 1 :- Show that it is true for $n = 1$

$1 = \frac{1(1+1)}{2} = 1$

LHS = RHS

Step 2 :- Assume that it is true for $n = k$.

$\Rightarrow 1 + 2 + 3 + \dots - K = \frac{k(k+1)}{2}$

Step 3 :- Show that it is true for $k+1$.

$\Rightarrow 1 + 2 + 3 + \dots + k + k+1 = \frac{(k+1)(k+2)}{2}$

$\Rightarrow \frac{k(k+1)}{2} + (k+1)$

$= (k+1)\frac{(k+2)}{2} = RHS$

So, it is true for $n = k+1$. So By Mathematical Induction, $1 + 2 + 3 + \dots n = \frac{n(n+1)}{2}$ is true for all n.

Practice Question :-

1) Proove $1 + 3 + 5 + \dots (2k-1) = k^2$.

2) Proove $1 + 4 + 7 + \dots (3n-2) = \frac{n(3n-1)}{2}$

Basic Terminologies

* Symbols :- It is an entity or individual object
E.g. 1, a, b, #, etc.

* Alphabets :- Finite non-empty set of symbols.
E.g :- {a, b} {1, 0}, etc.

* Strings :- It is a finite collection of symbols from the alphabets. The string is denoted by w.

* The no. of symbols in a string w is called length of string. It is denoted by $|w|$.

$$w = 010$$
$$|w| = 3$$

* Language :- Language is a collection of appropriate string. A language formed over Σ can be finite or Infinite.

* Kleene Star :- Set of all string + NULL.

$$\Sigma^* = \Sigma^0 + \Sigma^1 + \Sigma^2 + \dots$$

e.g. $\Sigma = \{a\}$, $\Sigma^* = \{\epsilon, a, aa, aaa, \dots\}$

* Kleene Closure / Plus :- Set of all possible string of all length excluding ϵ.

$$\Sigma^+ = \Sigma^1 + \Sigma^2 + \Sigma^3 + \dots$$
$$\Sigma^+ = \Sigma^* - \{\lambda\}$$

* Power of an alphabet :- If Σ is an alphabet, we can express the set of all string of a certain length using an exponential notation. We denote it Σ^n to be set of string of length n.

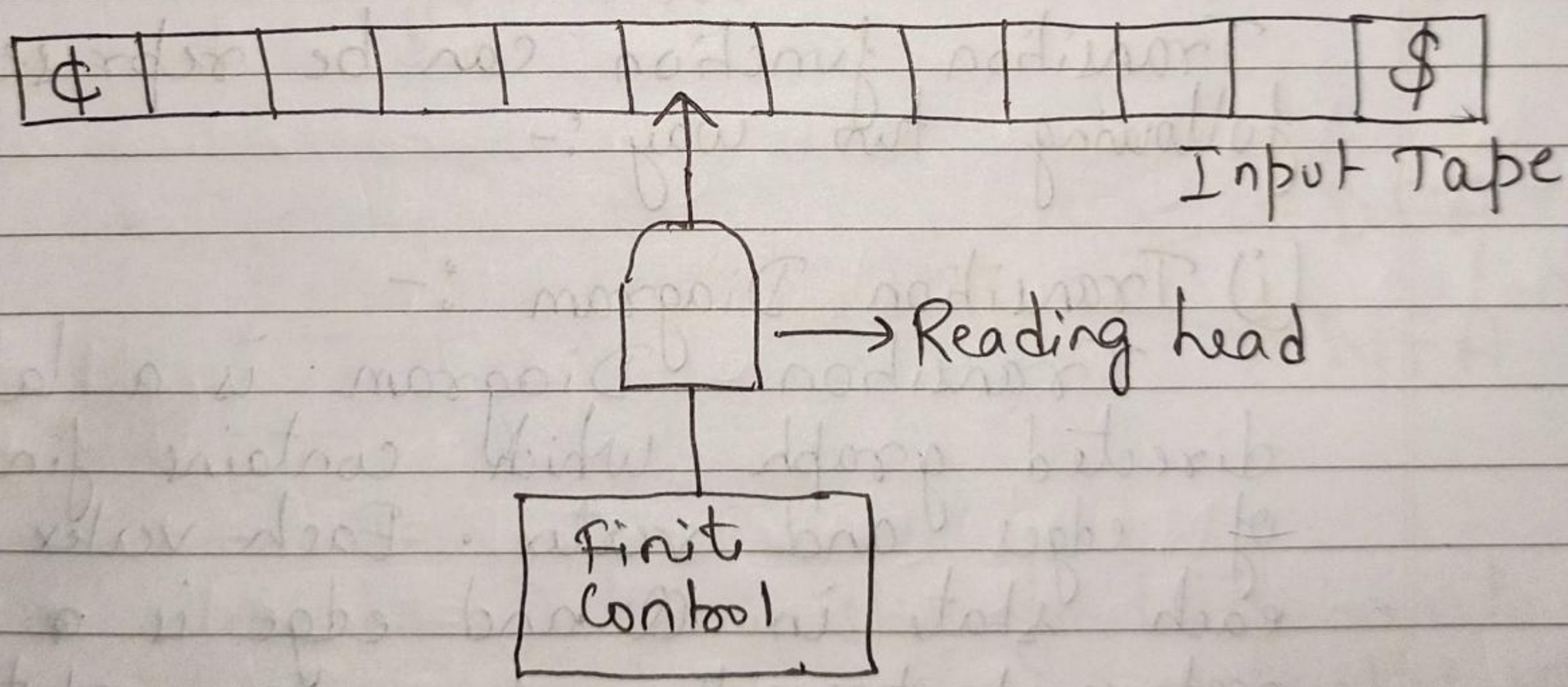
Eg. $\Sigma^0 = \{\epsilon\}$
$\Sigma^1 = \{0, 1\}$
$\Sigma^2 = \{00, 01, 10, 11\}$
.etc

* Transition :- Change of states.

Application of Automata :-

1) Used in Compiler design
2) Analog to digital
3) Lexical analyzer of a typical compiler.
   i.e The componet that

Block Diagram of Automata :-



(i) Input Tape :- It is a linear tape having numbers of cells. Each input symbols is placed in each cell. The left end contain end marker ¢ and right end contain $.

Absence of end markers → Infinite
Presence of end markers → Definite

(ii) Reading Head :- It reads the input data from the Input tape. It can be read from right to left as well as left to right. But it is preferable to read from left to right.

(iii) Finit Control :- It controls the transitions.

Transition function :-

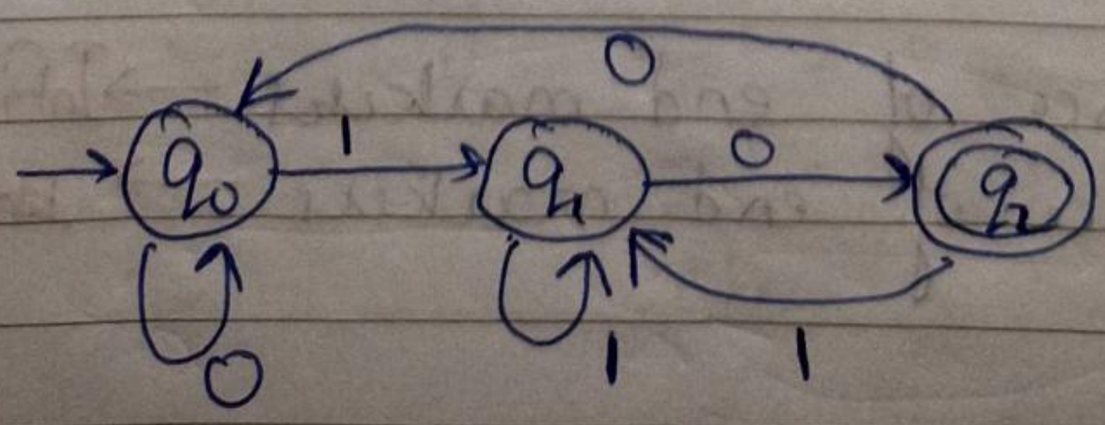
Transition function defines the movement of an automation from one state to another.

Transition function can be represented in following two way :-

(i) Transition Diagram :-

Transition Diagram is a labeled directed graph which contains finit no. of edges and vertex. Each vertex represent each state in Q and edge is used to represent transition from one state to another.

Start state is shown by an arrow with no source and final state is shown by double circle.



(ii) Transition Table :- A transition table is a conventional tabular representation of function like δ that takes the two argument and returns another state.

The rows correspond to the states & the colum corresponds to inputs.

	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_1

T.T for above Transition Diagram.

Finite Automata → DFA
→ NFA

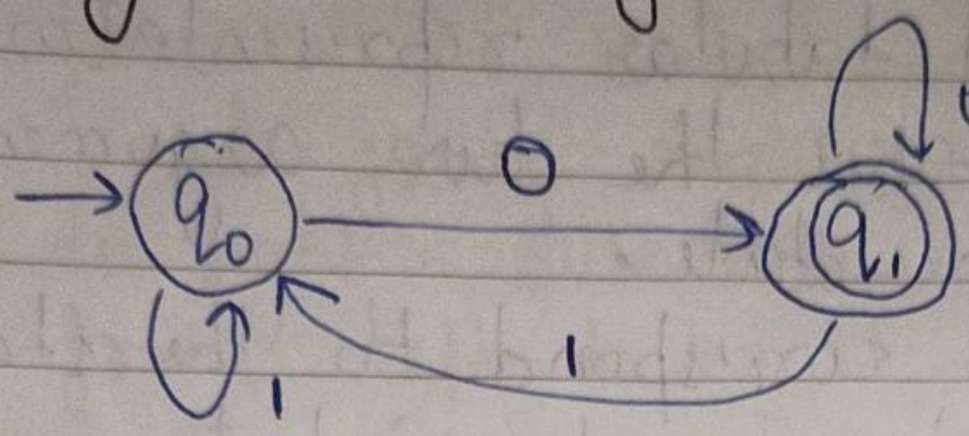
DFA :-

It refers to Deterministic finite automata. Deterministic refers to uniqueness of computations. In this, the machine goes to only one state for a particular input character. It do not accept null move. $\delta : Q \times \Sigma \longrightarrow Q$.

A DFA can be represented by a 5-tuples $(Q, \Sigma, \delta, q_0, F)$ where,

* $Q \rightarrow$ finit set of Strate
* $\Sigma \rightarrow$ finit set of Input symbols
* $\delta \rightarrow$ Transition function
* $q_0 \rightarrow$ initial State
* $F \rightarrow$ Final State

Q. Design DFA for string ending with 0.



How we did this?

For Ending with ⟶ n + 1

For starting | Contains ⟶ n + 2

formula for no. of state reqd

where n is lenght of string

Q. Design a DFA for string ending with 01.

Sol→ Here string is 01

n = 2 (lenght of string)

No. of state reqd = n + 1

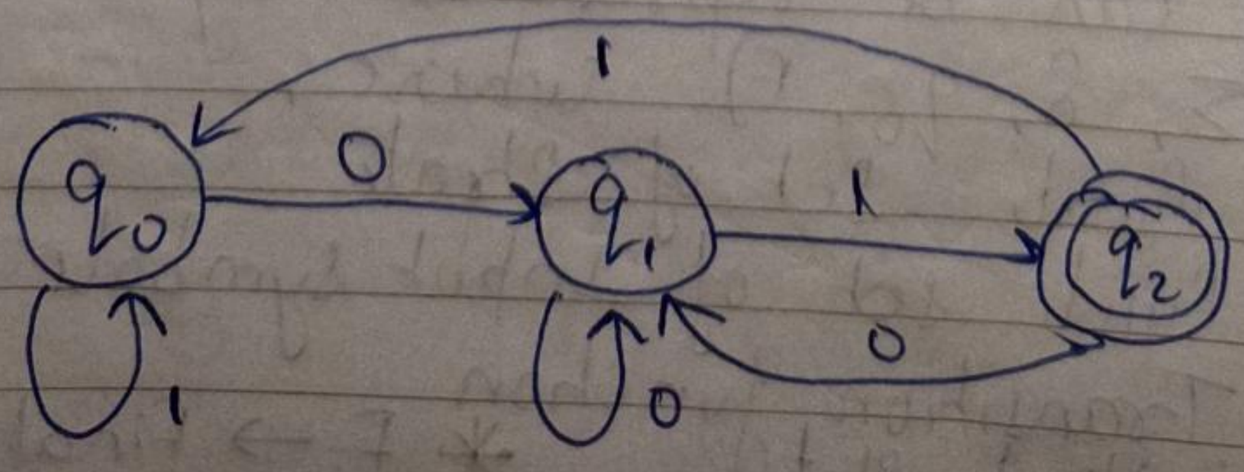
= 2 + 1

= 3.

Q = {q0, q1, q2}

Σ = {0, 1}

String = {01, 1001, 101, 001, 11110001}



Transition Table :-

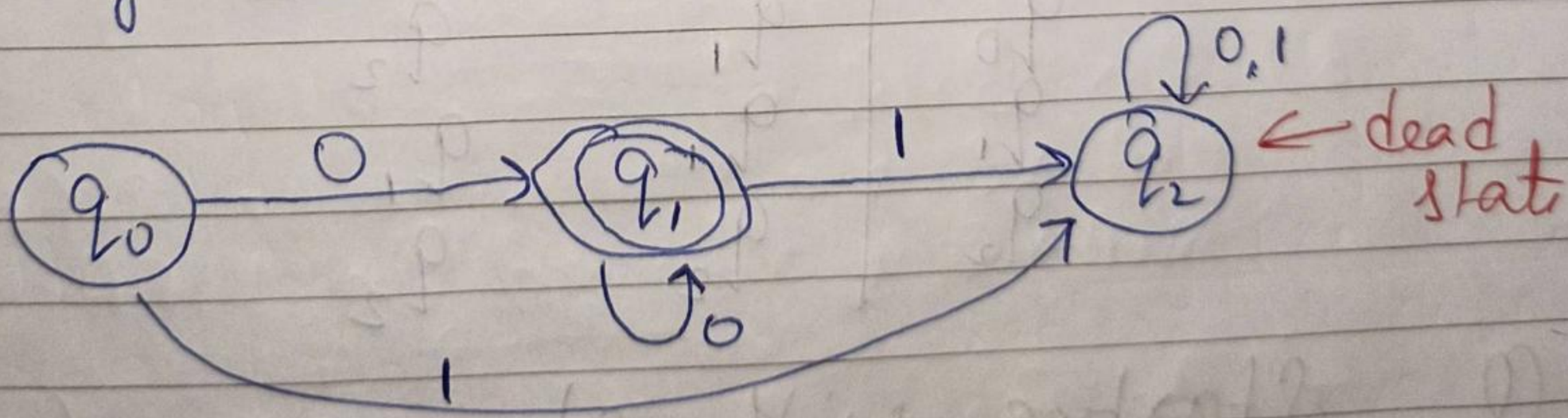
	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_0

Q Design a DFA for string accepting string 0 only.

Sol→ Here String = 0
n = 1

No. of stat = n+2 = 1+2
= 3

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
String = {0, 00, 000, 0000, ---}



Transition Table :-

	0	1
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_2	q_2

Q. String starting with 0.

Sol→ Here String = 0

n = 1 (lenght of string)

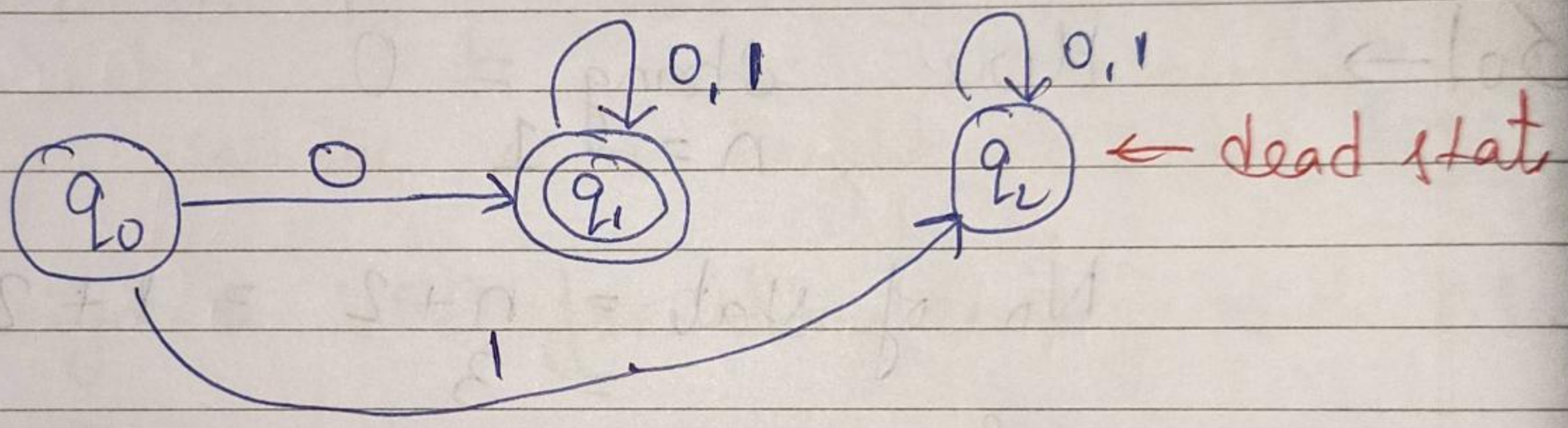
No. of states = n+2

= 3

Q = {q_0, q_1, q_2}

Σ = {0, 1}

String = {0, 01, 00, 011, 010, 001, - -



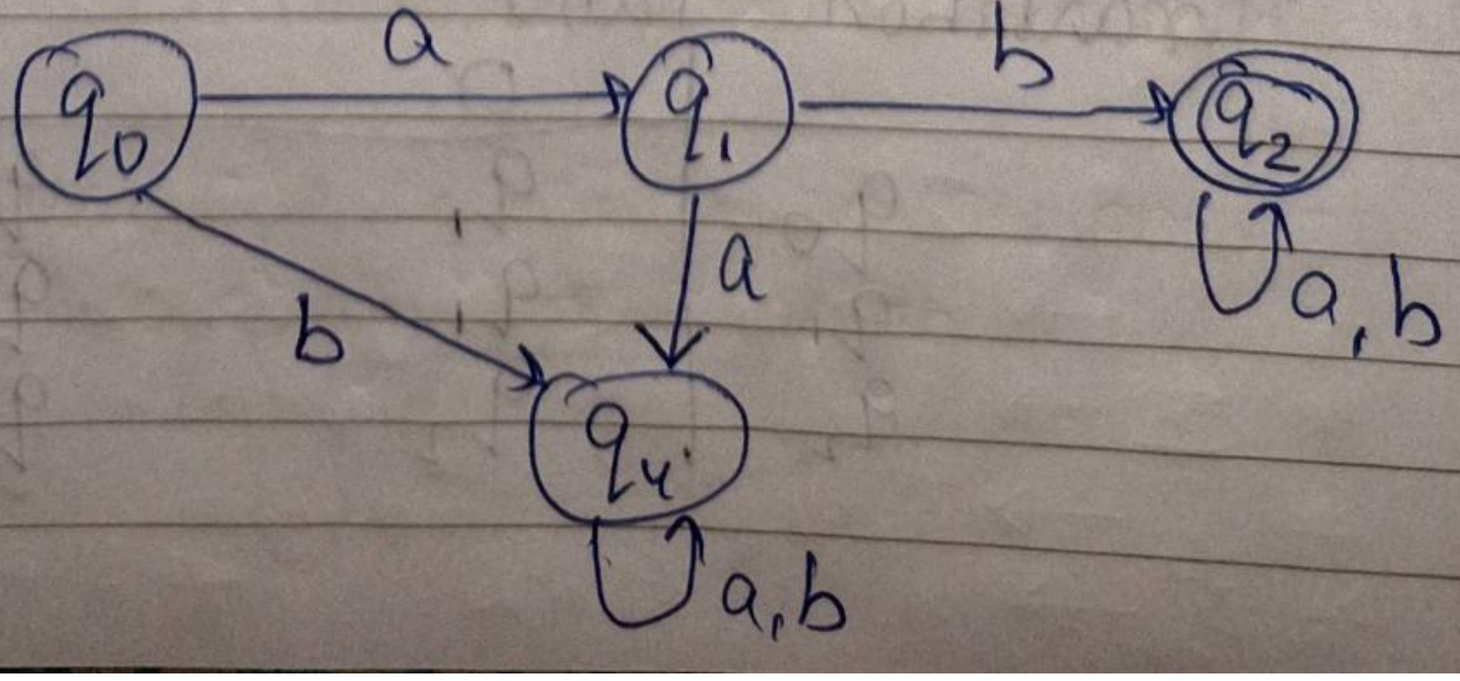
Transition Table :-

	0	1
q_0	q_1	q_2
q_1	q_1	q_1
q_2	q_2	q_2

Q Starting with ab in Σ = (a, b).

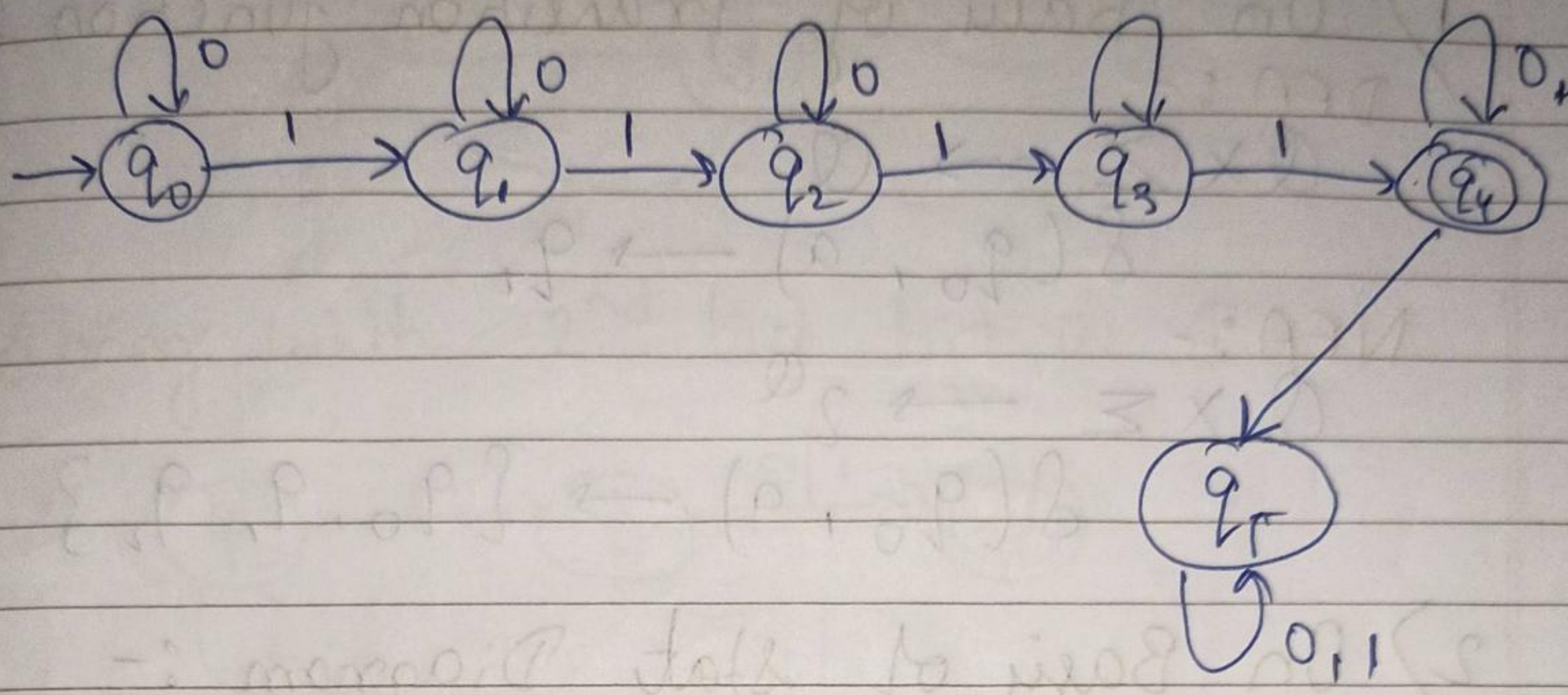
Sol→ n = 2

n+2 = 4



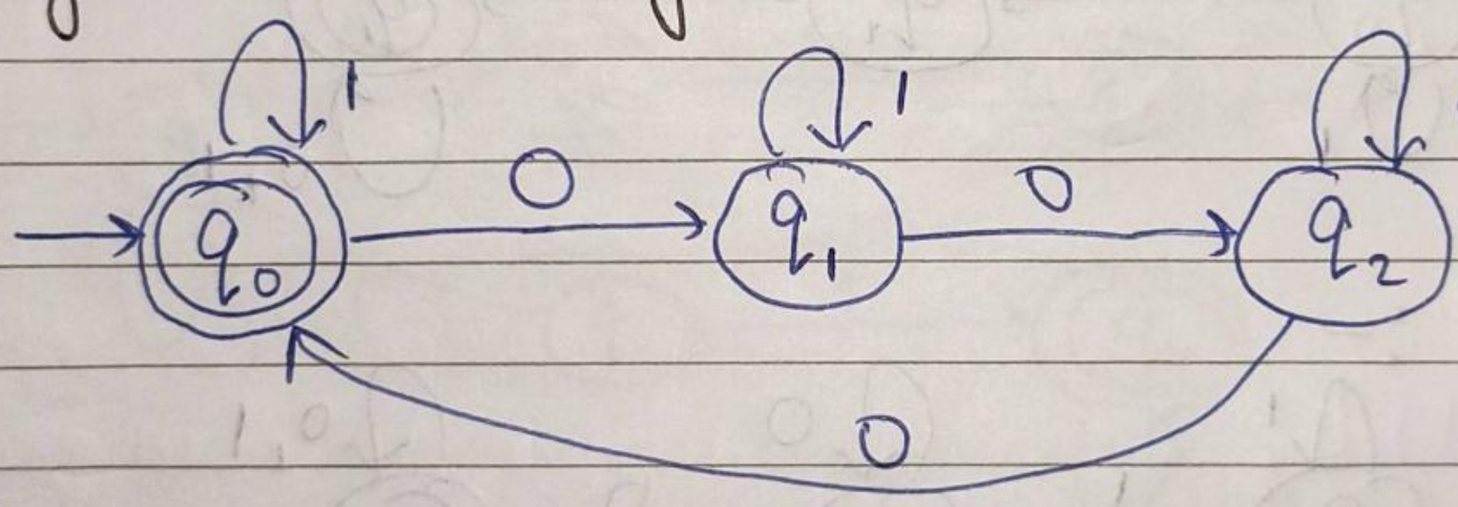
Q $\Sigma = \{0, 1\}$ and string contains exactly 4 1's.

Sol →



Q. String with No. of 0 is divisible by 3.

Sol →



NFA :-

NFA refers to Non-Deterministic finit Automata. It is used to transmit any no. of stat for a particular input. It can accept null Move.

$$\delta : Q \times (\Sigma \cup \epsilon) \dashrightarrow 2^Q$$

It has all same 5 tuple discussed in DfA.

Difference between NFA & DFA :-

1) On Basis of Transition function :-

DFA :-

$$Q \times \Sigma \longrightarrow Q$$

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

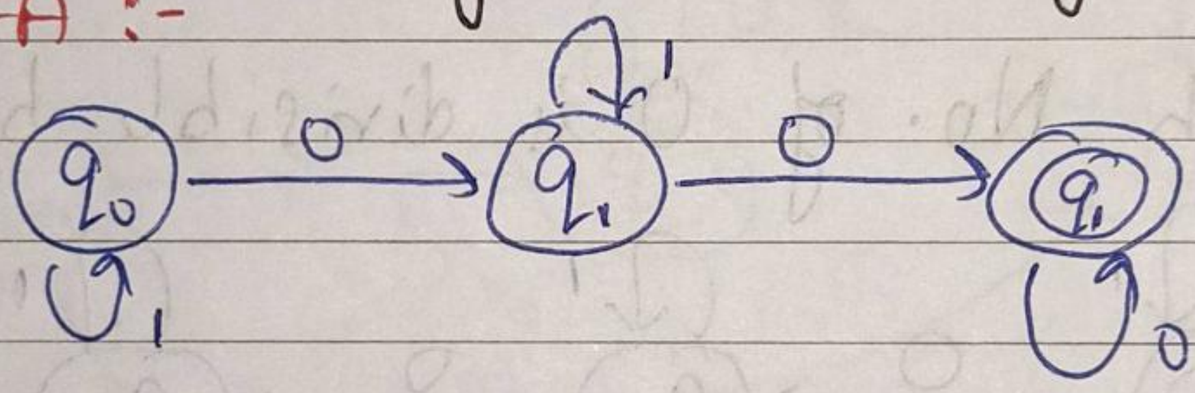
NFA :-

$$Q \times \Sigma \longrightarrow 2^Q$$

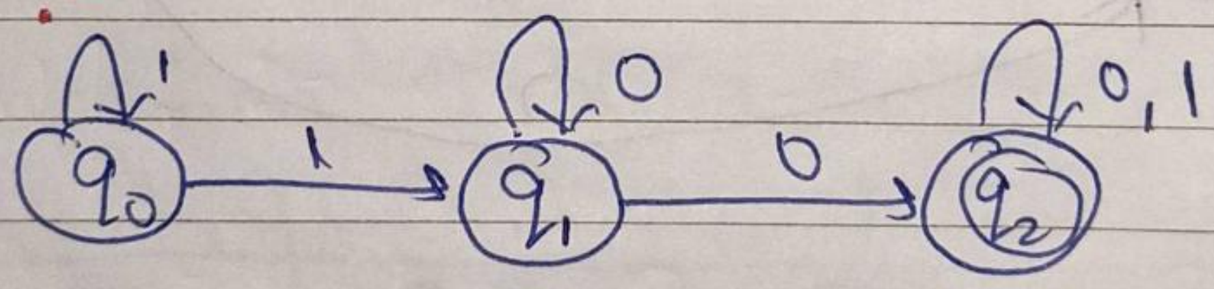
$$\delta(q_0, a) \longrightarrow \{q_0, q_1, q_2\}$$

2) On Basis of stat Diagram :-

DFA :-



NFA :-



3) On Basis of ϵ.

DFA → ϵ is not allowed in DFA

NFA → ϵ is allowed in NFA

4) On Basis of space :-

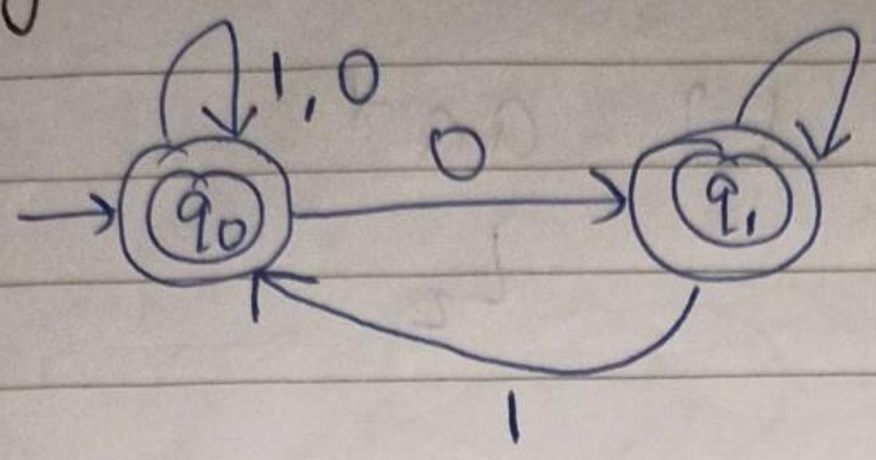
DFA takes less space compared to NFA

5) On Basis of difficulty in design :-

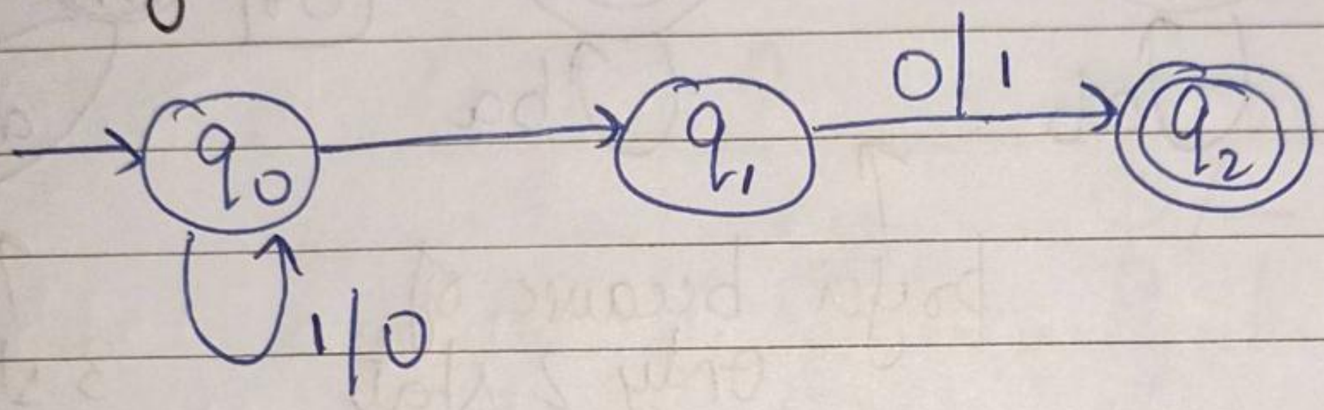
DFA is easy to design.

Q Design a NFA with string end with zero.

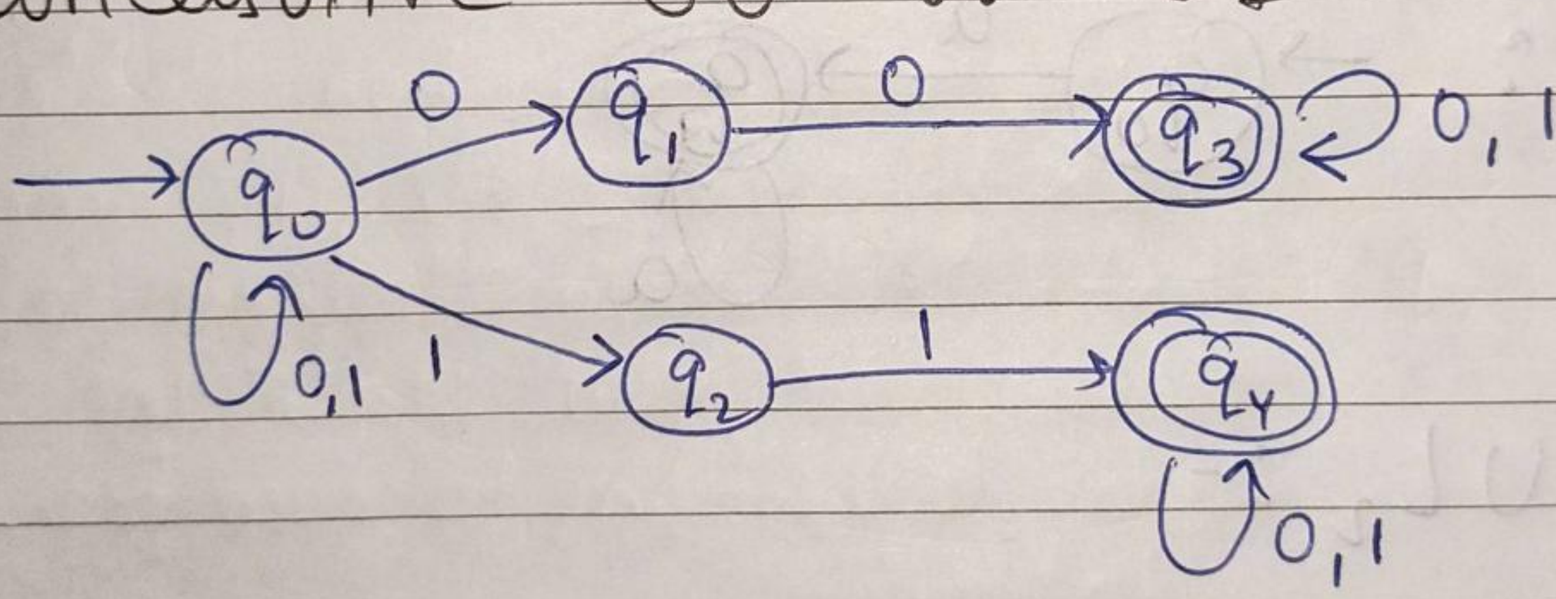
Sol →



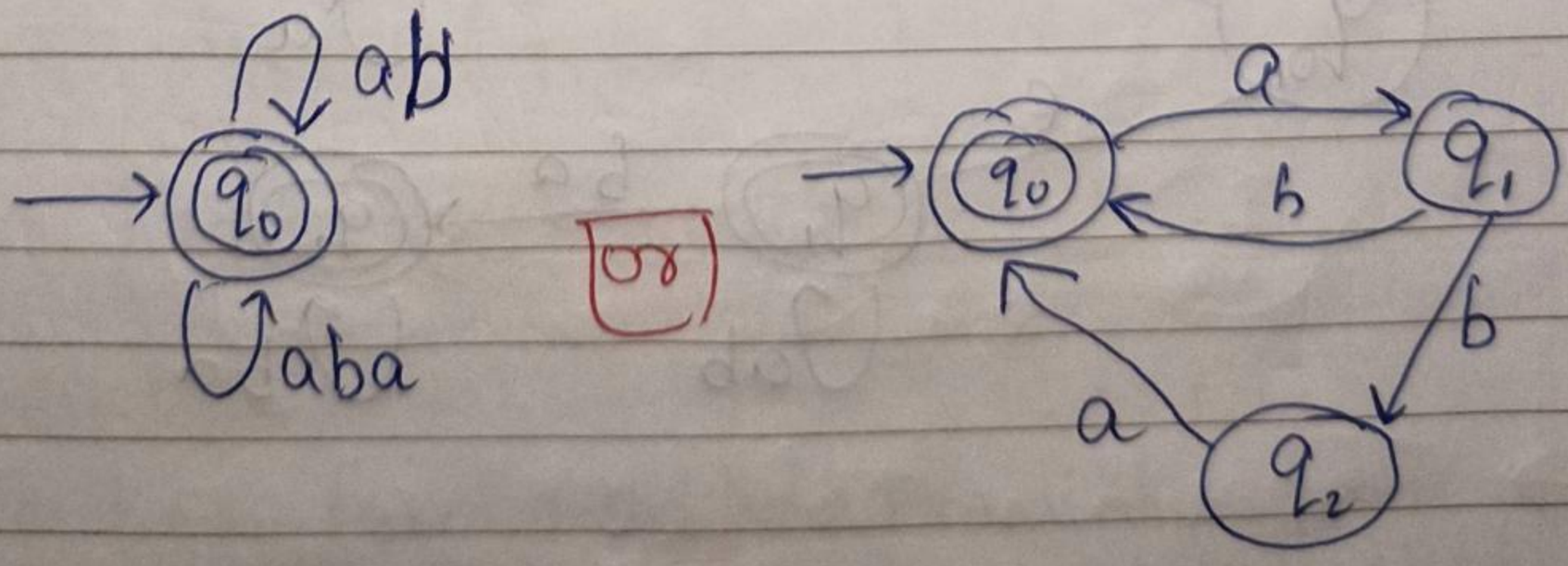
Q String with 2nd last bit as 1.



Q. Consecutive 00 or 11.



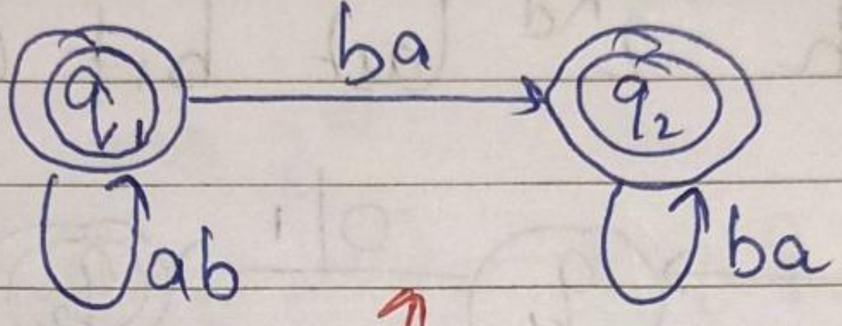
Q (ab ∪ aba)*

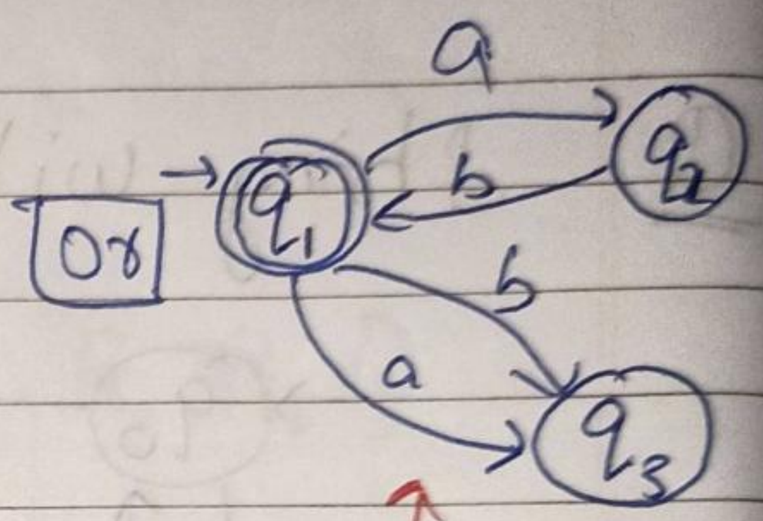


Q $(ab)^*(ba)^* \cup aa^*$

$\underbrace{(ab)^*(ba)^*}_{L_1} \cup \underbrace{aa^*}_{L_2}$

L_1 :

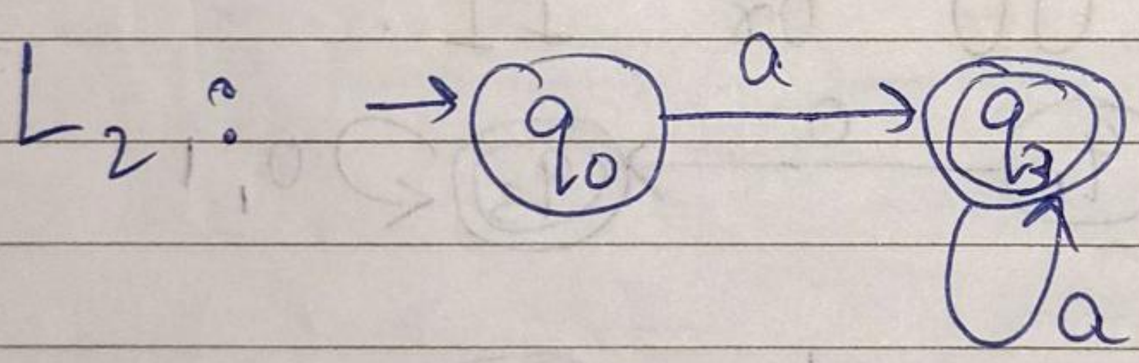




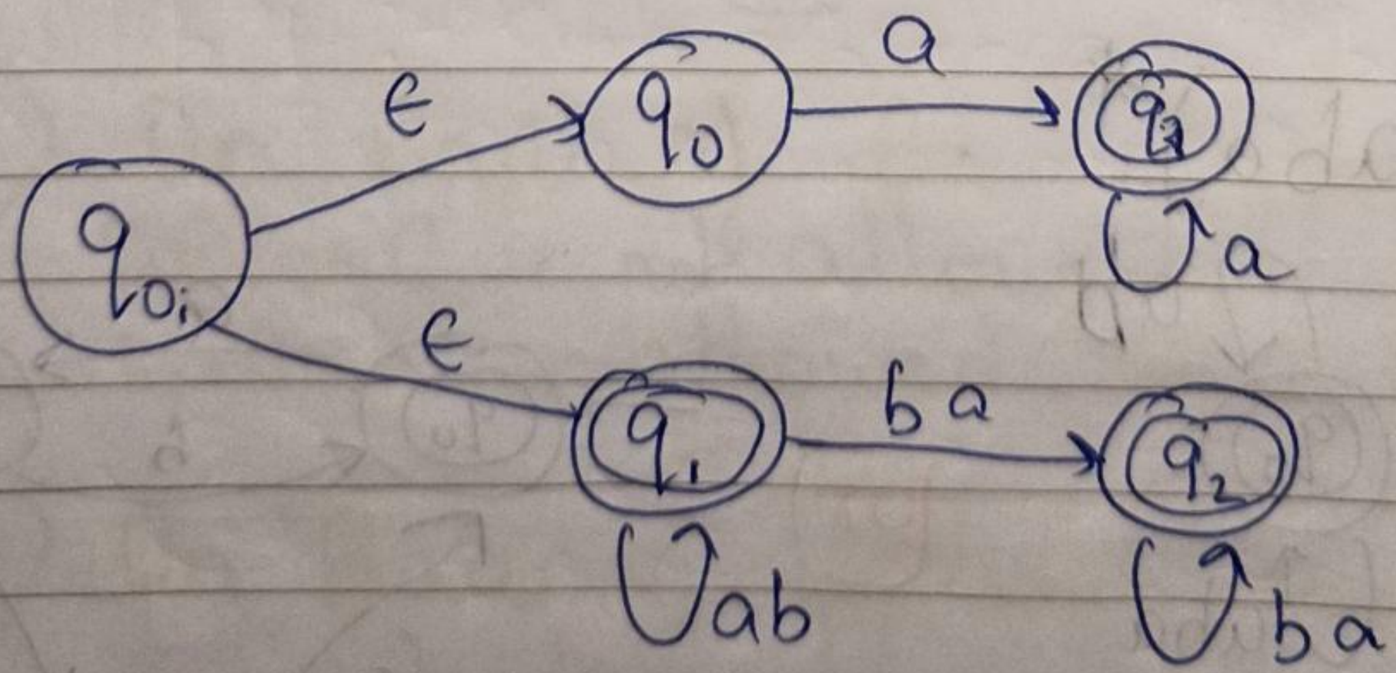
proper because of only 2 stat

3 stat so not preferable

L_2 :

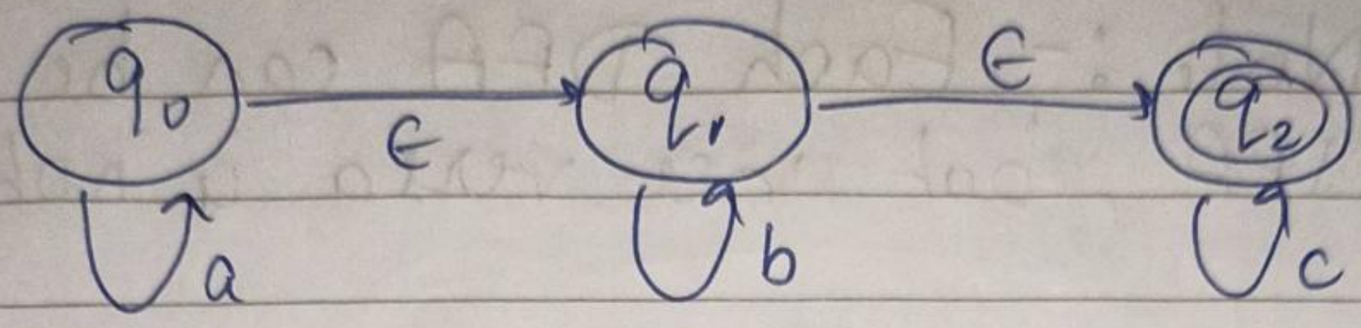


$L_1 \cup L_2$:-



ϵ - closure :-

It is only used for NFA with ϵ-transition.



T.T :-

	a	b	c	ϵ
q_0	q_0	ϕ/–	ϕ/–	q_1
q_1	ϕ/–	q_1/	ϕ/–	q_2
q_2	ϕ/–	ϕ/–	q_2	ϕ/–

ϵ - closure of a stat q is a set of states following by all transition of q that are labeled by ϵ.

ϵ-closure of q_0 = $\{q_0, q_1, q_2\}$
ϵ-closure (q_1) = $\{q_1, q_2\}$
ϵ-closure (q_2) = $\{q_2\}$

DFA vs NDFA

DFA	NDFA
→ Transition from a stat can be to only one stat.	→ Transition from a stat can be to multiple stat.
→ DFA does not permit empty string transition.	→ NDFA permits empty string transition
→ More space	→ less space

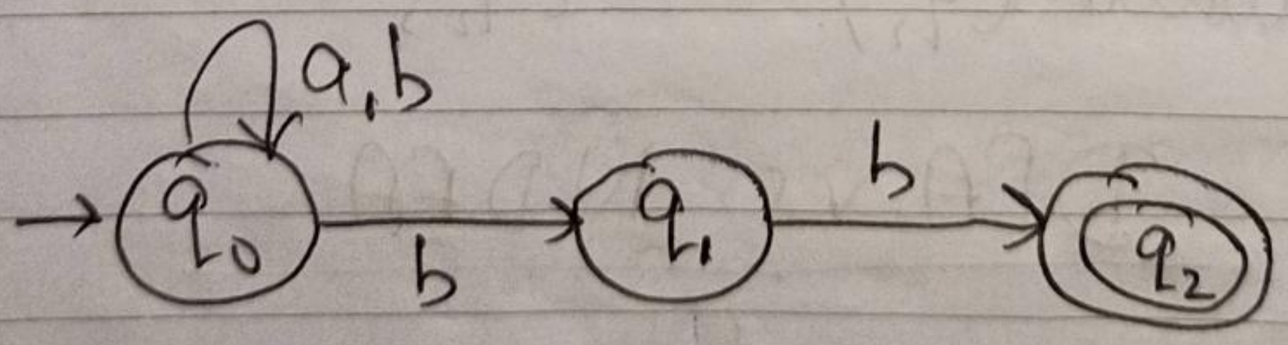
→ Backtracking is allowed in DFA → requires less repetition to get final stat.	→ Backtracking is not always possible. → require more repetition to get final stat.

→ Note :- Each DFA can be said to be NFA, but vice versa is not true.

Equivalence of NFA to DFA

Step 1 :- Draw Transition table.
Step 2 :- Identify initial state.
Step 3 :- Start with initial state to draw new transition table and check if new state is comming or not. If new state is present then add new state in Q' and add transition to new transition table.
Step 4 :- Keep repeating Step 3 untill no new state is present in T.T.

Q.



Sol → Transition Table :-

	a	b
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	ϕ	$\{q_2\}$
q_2	ϕ	ϕ

Initial state is q_0.

New $Q' = \{q_0\}$

New transition Table :-

	a	b
q_0	$\{q_0\}$	$\{q_0, q_1\}$

Now, new state is $\{q_0, q_1\}$

$\Rightarrow Q' = \{\{q_0\}, \{q_0, q_1\}\}$

Transition Table =

	a	b
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0, q_1, q_2\}$

Now, new state is $\{q_0, q_1, q_2\}$

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

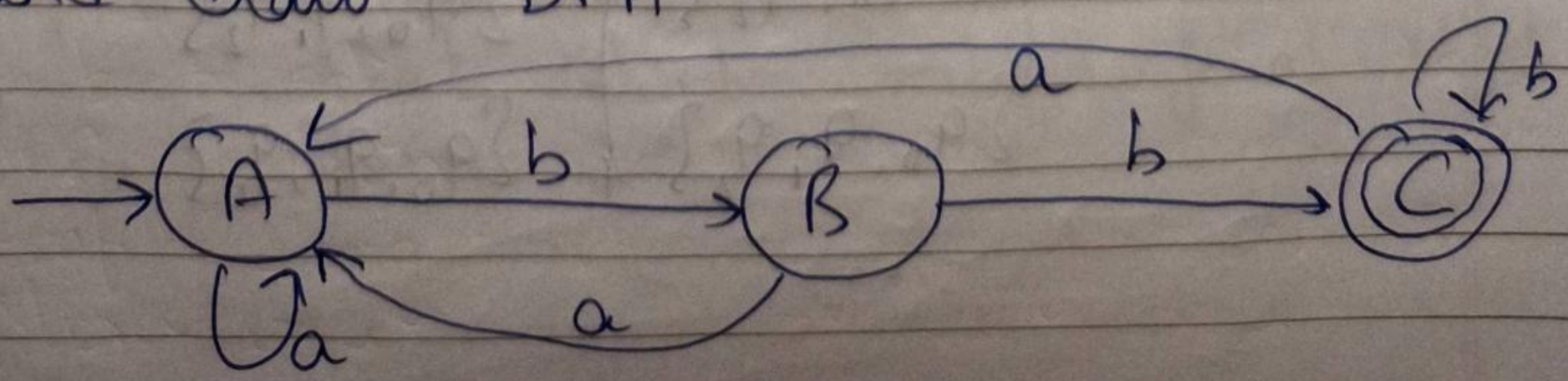
$TT' =$

	a	b
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0\}$	$\{q_0, q_1, q_2\}$

Since there is no new stat now, to we stop now and let assume

$A \longrightarrow \{q_0\}$ $B \rightarrow \{q_0, q_1\}$ $C \rightarrow \{q_0, q_1, q_2\}$

and draw DFA.

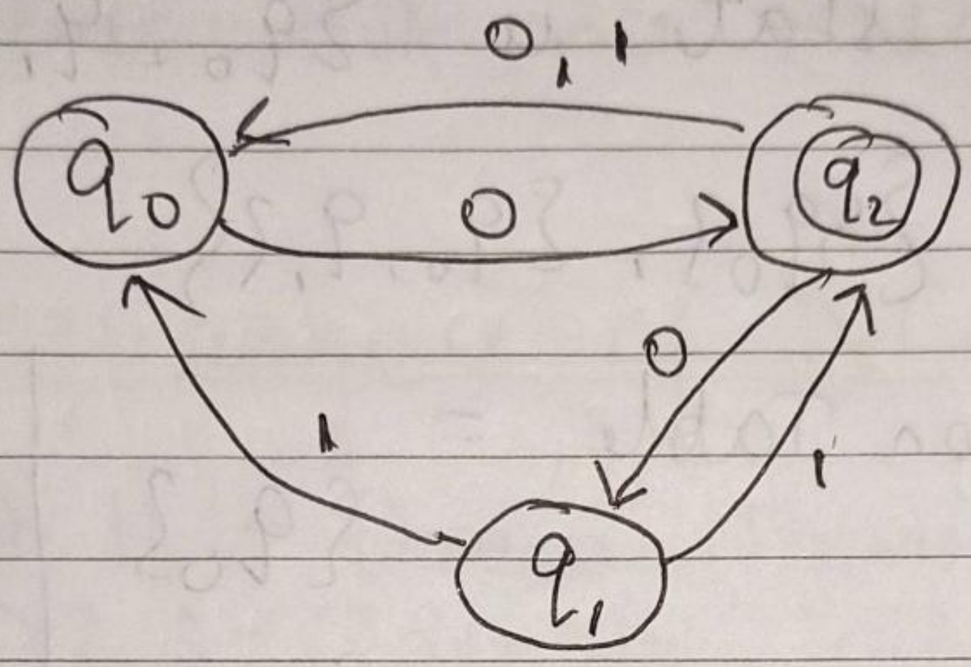


* Maximum No. of stat we can have while converting NFA to DFA

$$= 2^m$$

where $m \rightarrow$ No. of stat in NFA.

Q.



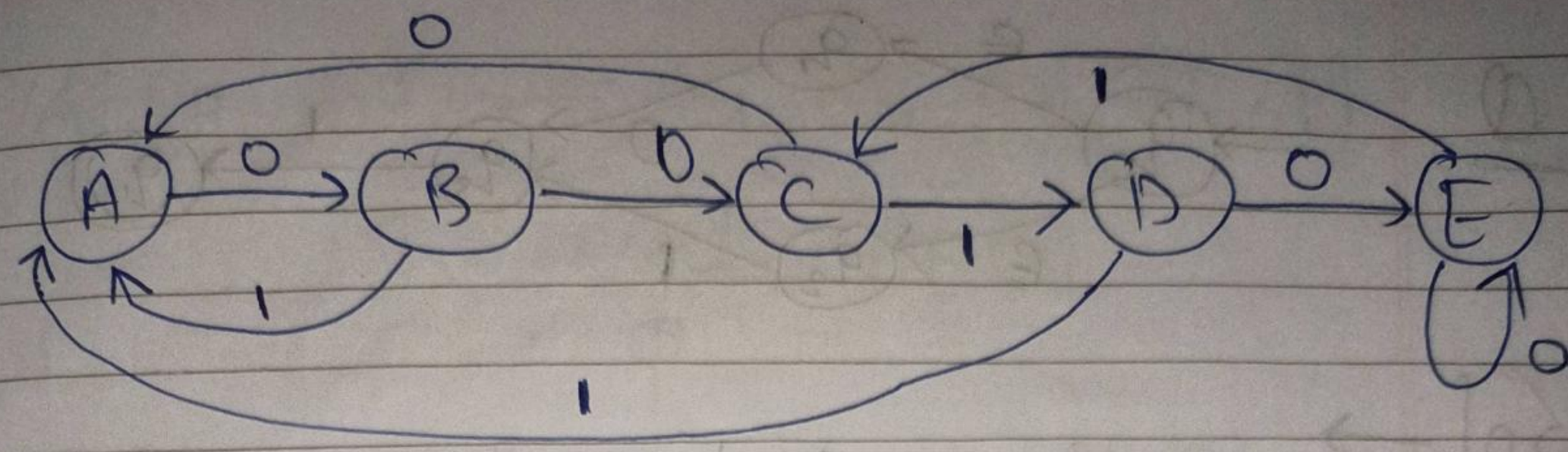
Sol → Q. T.T

	0	1
q_0	$\{q_2\}$	ϕ
q_1	ϕ	$\{q_0, q_2\}$
q_2	$\{q_1, q_0\}$	$\{q_0\}$.

$q_0 \rightarrow$ Initial

New. T.T

	0	1
$\{q_0\}$	$\{q_2\}$	ϕ.
$\{q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_2\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$



Conversion of NFA with ϵ to DFA :-

Step 1 → Take ϵ-closure for the starting stat of NFA as starting stat of DFA.

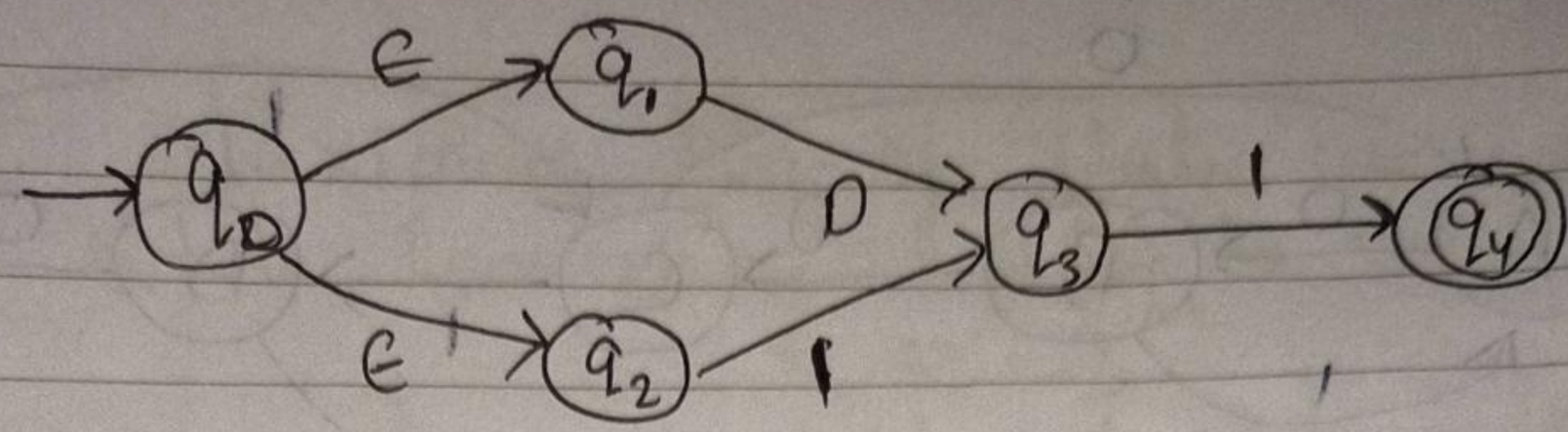
Step 2 → Find stat for each input symbol that can be traversed from the present. This mean union of transition value and their closure for each state of NFA present in current stat of DFA

Step 3 → If we find new state, take it as current step and repeat step 2.

Step 4 → Repeat step 2 and 3 until there is no new state present in transition table of DFA.

Step 5 → Mark the states of DFA as final state which contain final stat of NFA.

Q.



Sol →

ϵ-closure (q_0) :- $\{q_0, q_1, q_2\}$
ϵ-closure (q_1) :- $\{q_1\}$
ϵ-closure (q_2) :- $\{q_2\}$
ϵ-closure (q_3) :- $\{q_3\}$
ϵ-closure (q_4) :- $\{q_4\}$.

$A \rightarrow \{q_0, q_1, q_2\}$

$\delta'(A, 0) = \epsilon\text{-closure}(\delta(q_0, q_1, q_2), 0)$

$= \epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$
$= \epsilon\text{-closure}(\phi \cup q_3 \cup \phi)$
$= \epsilon\text{-closure}(q_3)$
$= \{q_3\}$.

$\delta'(A, 1) = \epsilon\text{-closure}(\delta(q_0, q_1, q_2), 1)$
$= \epsilon\text{-closure}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$
$= \epsilon\text{-closure}(\phi \cup \phi \cup q_3)$
$= \epsilon\text{-closure}(q_3)$
$= \{q_3\}$

$B \rightarrow \{q_3\}$

$\delta'(q_3, 0) = \epsilon\text{-closure}(\delta(q_3, 0))$

$= \epsilon\text{-closure}(\phi)$

$= \phi$

$\delta'(q_3, 1) = \epsilon\text{-closure}(\delta(q_3, 1))$

$= \epsilon\text{-closure}(q_4)$

$= \{q_4\}$

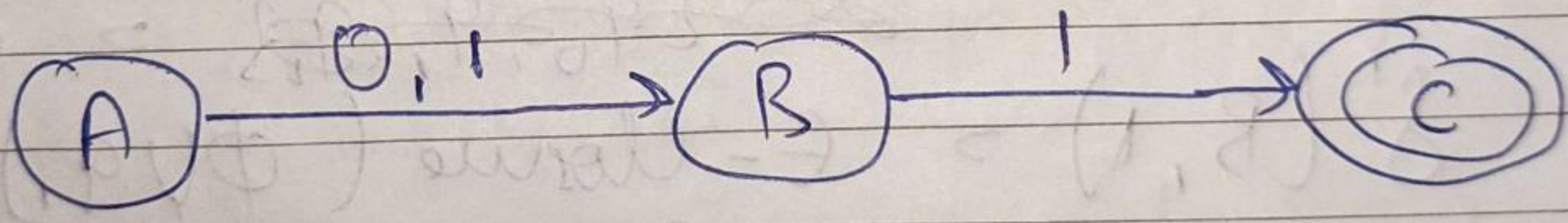
$C \rightarrow \{q_4\}$

$\delta(q_4, 0) = \epsilon\text{-closure}(\delta(q_4, 0))$

$= \phi$

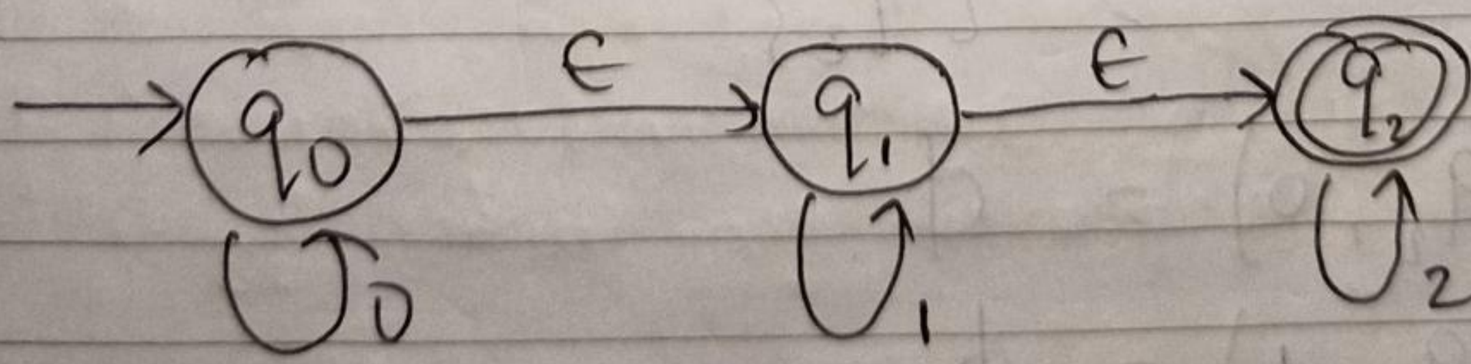
$\delta(q_4, 1) = \epsilon\text{-closure}(\delta(q_4, 1))$

$= \phi$.



Practice Question :-

Q.



Sol →

$\epsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$

$\epsilon\text{-closure}(q_1) = \{q_1, q_2\}$

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

$\delta'(A,0) = \epsilon\text{-Closure}(\delta(q_0,q_1,q_2),0)$

$= \epsilon\text{-clo}(q_0 \cup \phi \cup \phi)$

$= \epsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$

$\delta'(A,1) = \epsilon\text{-closure}(\delta(q_0,q_1,q_2),1)$

$= \epsilon\text{-Closure}(\phi \cup q_1 \cup \phi)$

$= \epsilon\text{-closure}(q_1) = \{q_1, q_2\}$.

$\delta'(A,2) = \epsilon\text{-closure}(\delta(q_0,q_1,q_2),2)$

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

$= \{q_2\}$

$\delta'(B,0) = \epsilon\text{-closure}(\phi \cup \phi)$

~~$= \{q_0, q_1, q_2\}$~~ $= \phi$

$\delta'(B,1) = \epsilon\text{-closure}(\phi \cup q_1)$

$= \{q_1, q_2\}$

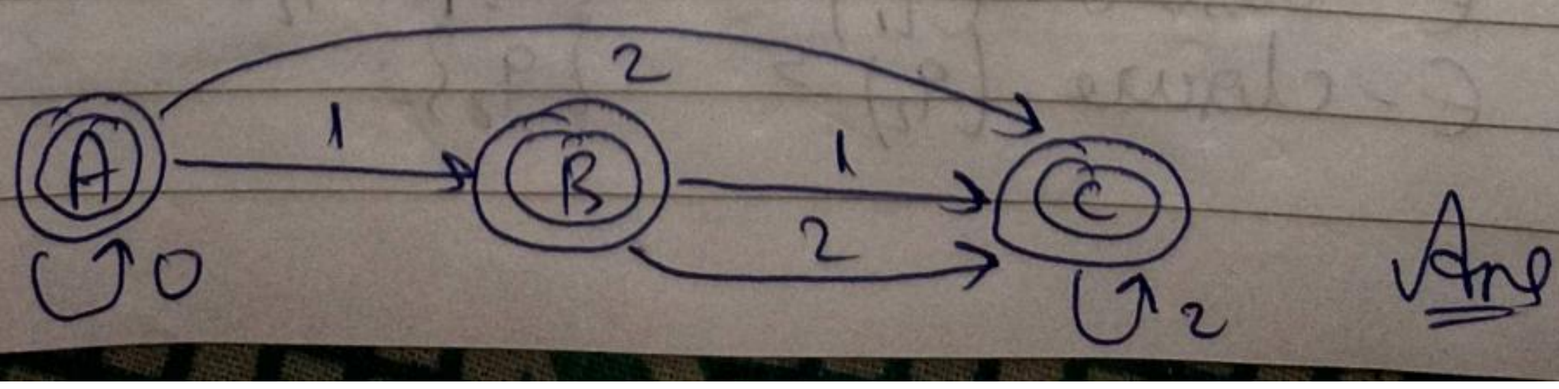
$\delta'(B,2) = \{\phi, q_2\}$

$= \{q_2\}$.

$\delta'(q_2,0) = \phi$

$\delta'(q_2,1) = \phi$

$\delta'(q_2,2) = \{q_2\}$



Ans

Limitations of DFA :-

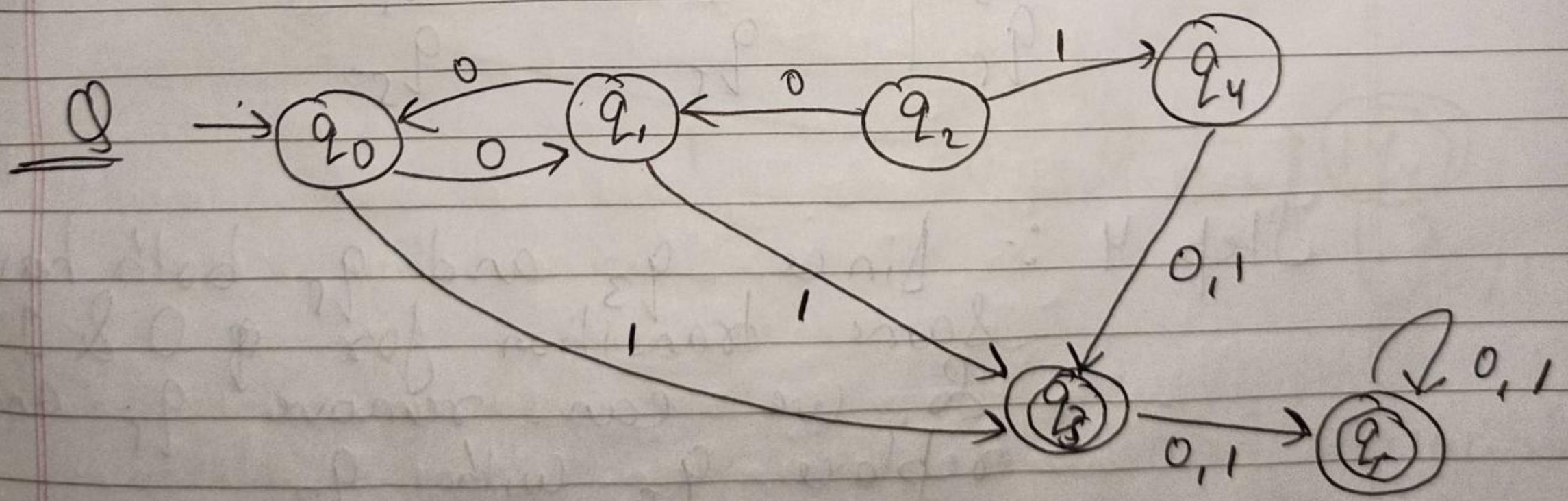
- Limited memory
- String Comparision
- Linear Power

a^{2^n} , a^{n^2} , $(a^m)^n$ $m>n$ $m<n$ $m=n$, $a^n b^m$ $n>m$ $n<m$ $n=m$,

Minimization of DFA :-

The process of reducing a given DFA to its minimal form is called as minization of DFA. It Contain minimum no. of states. The DFA in its minimal form is called as Minimal DFA.

Q)



Sol → Step 1 :- Identify Dead state & Inaccessible state.

Here, there is no dead state
But there is 2 inaccessible state.

Step 2 :- Draw Transition table with all the states except dead & inaccessible one

	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
$*q_3$	q_r	q_r
$*q_5$	q_r	q_5

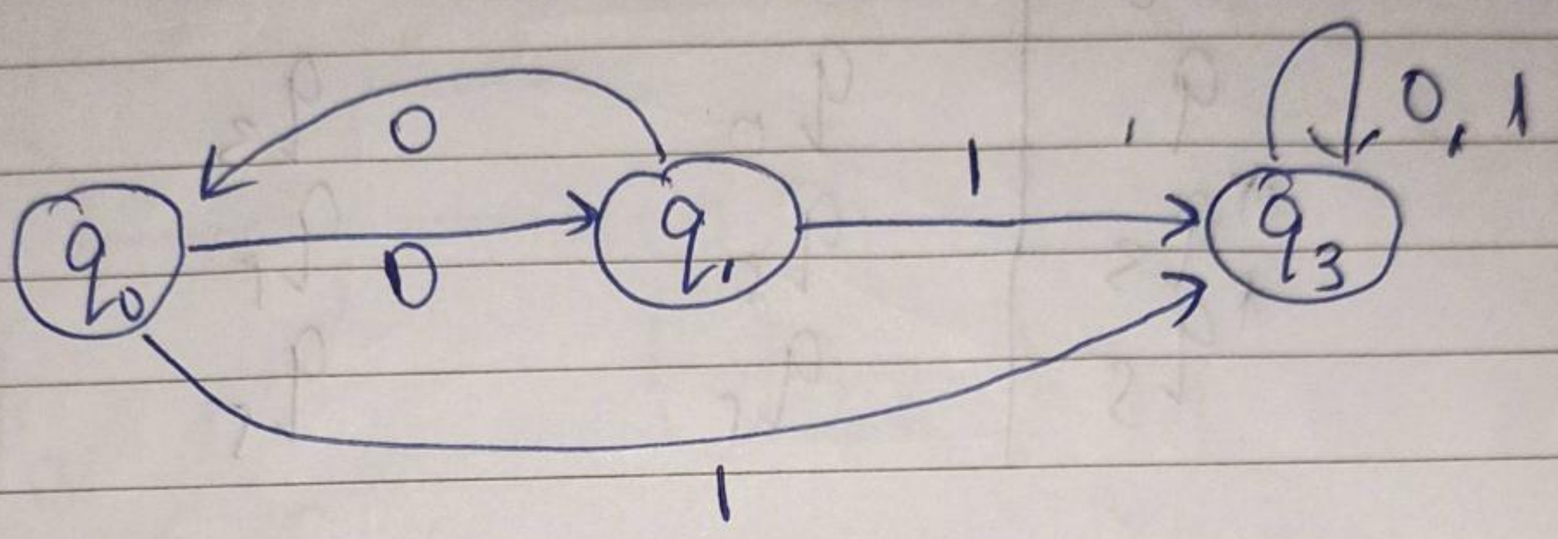
Step 3 :- Creat partition between final state and all other states

	0	1
q_0	q_1	q_3
q_1	q_0	q_3
q_3	q_r	q_r
q_r	q_5	q_5

Step 4 :- Since q_3 and q_r both have same transition for @ 0 & 1. So, we can remove q_r and replace q_5 with q_3.

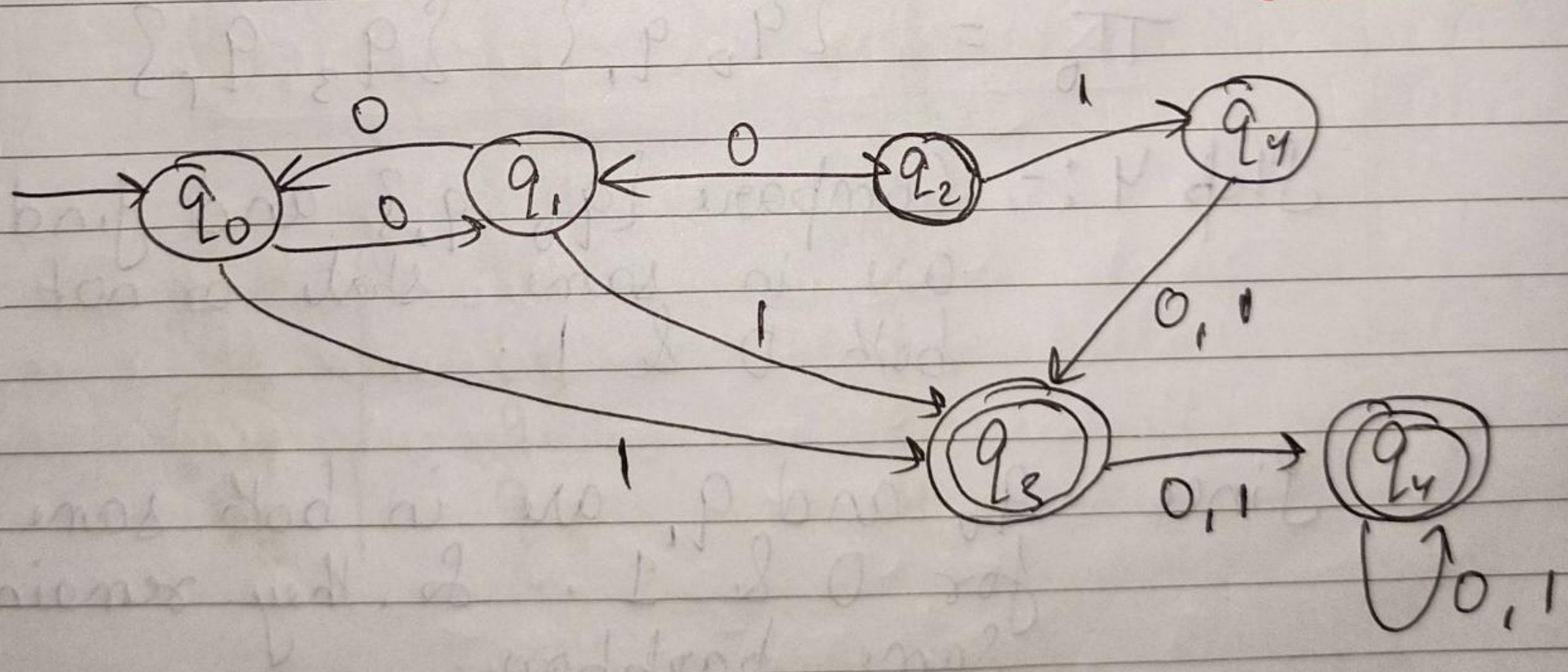
	0	1
q_0	q_1	q_3
q_1	q_0	q_3
q_3	q_3	q_3

Step 6 :- Now since there is no other pair of state with same transition, so we stop and draw DFA for current transition table.



Minimization of DFA using Equivalence Theorem

Q.



Sol → Step 1 :- Identify dead state and inaccessible state

Here there is no dead state but there is 2 inaccessible state.

Step 2 :- Draw T.T for all states except dead state and inaccessible state.

	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
$*q_3$	q_5	q_5
$*q_5$	q_5	q_5

Step 3 :- Make partition with one partition full of final state and other partition with all other state.

$$\pi_0 = \{q_0\ q_1\}\ \{q_3\ q_5\}$$

Step 4 :- Compare $[q_0, q_1]$ and find if they are in same state or not for both 0 & 1.

Since q_0 and q_1 are in both same state for 0 & 1. So, they remain in same partition.

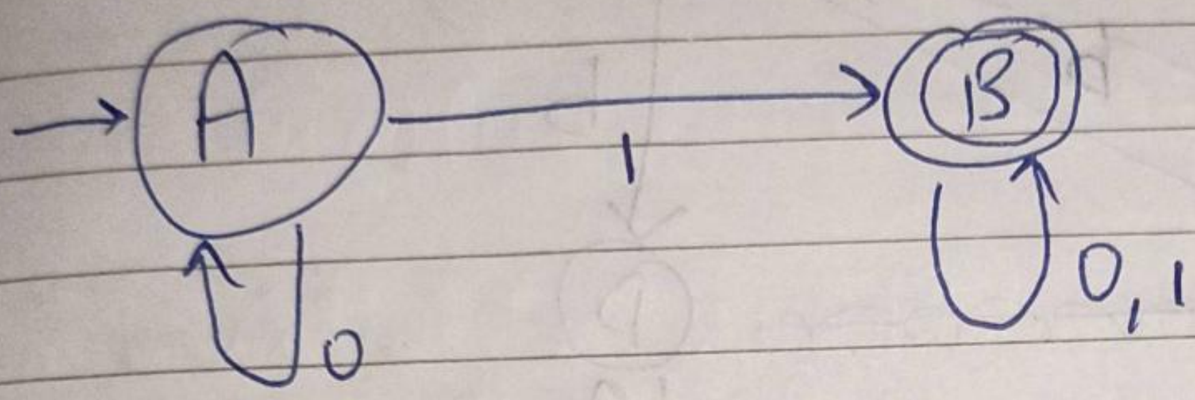
Also, q_3 & q_5 are also in same state so they remain in same state too.

So,

$$\pi_1 = \{q_0\ q_1\}\ \{q_3\ q_5\}$$

Since $\pi_0 = \pi_1$ so, we stop.

From Π, we can see $\{q_0, q_1\}$ are equivalent and similarly $\{q_3, q_5\}$ are equivalent and can be merged together as A and B respectively.



Q Minimize DFA with following T.T :

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_3	q_4
q_2	q_3	q_5
~~q_3~~	~~q_5~~	~~q_5~~
$*q_3$	q_3	q_4
$*q_4$	q_4	q_5
$*q_5$	q_3	q_4

Sol → $\pi_0 = \{q_0, q_1, q_2\}\ \{q_3, q_4, q_5\}$

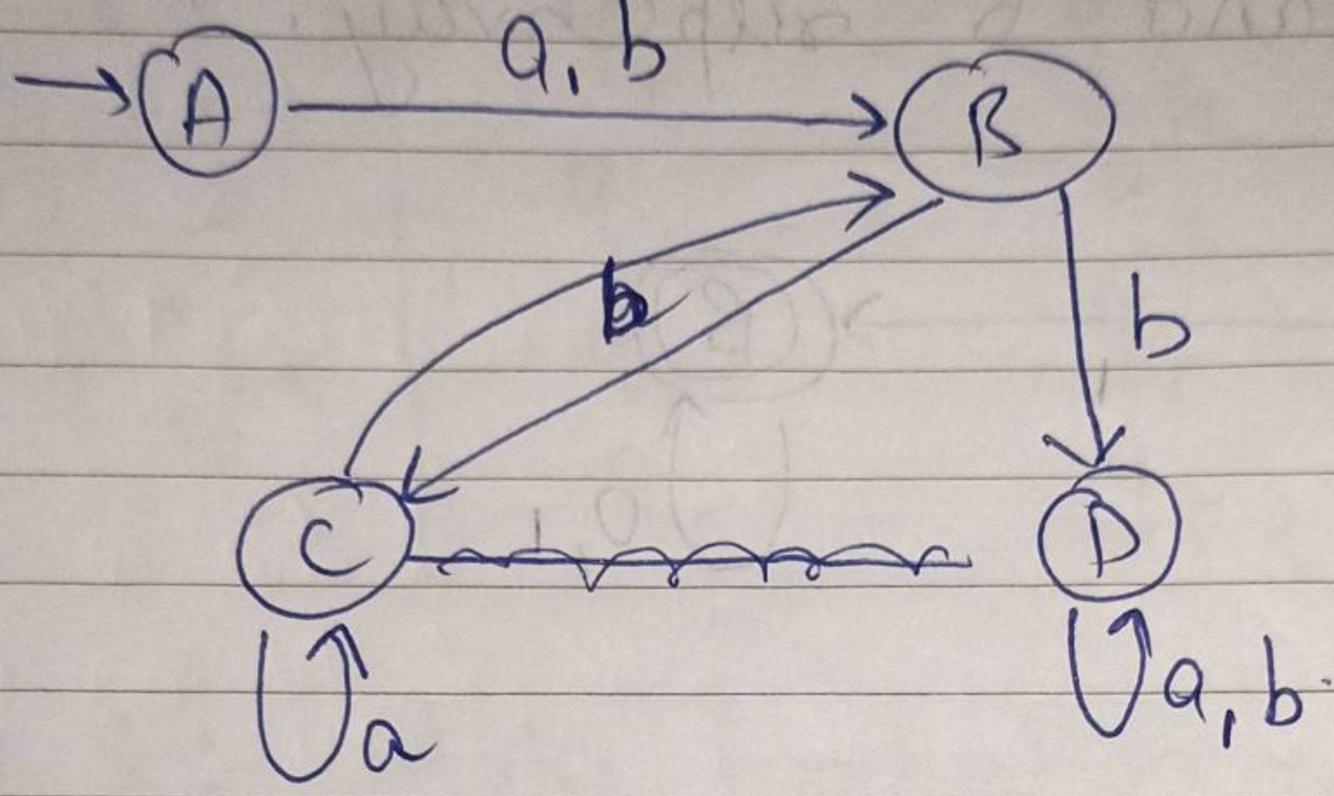
$\pi_1 = \{q_0\}\ \{q_1, q_2\}\ \{q_3\}\ \{q_4\ q_5\}$

$\pi_2 = \{q_0\}\ \{q_1, q_2\}\ \{q_3\}\ \{q_4\ q_5\}$

$\pi_1 = \pi_2$

$A \rightarrow \{q_0\}$ $\quad C \rightarrow \{q_3\}$

$B \rightarrow \{q_1, q_2\}$ $\quad D \rightarrow \{q_4, q_f\}$



Regular Expression

The language accepted by finite automata can be easily described by simple expression called regular expression. It is most effective way to ~~ex~~ represent any language.

a , a^* , a^+ , aa^*

$(a+b)^*$ → same any combination of either a or b

$(aa+ab)$ → either aa or ab.

→ a is a R.E
→ a* is a R.E
→ ϕ, $\in$ is also a R.E
→ a^+ (excluding null)
→ ab (concatenation) {ab}
→ a+b (union) {a} or {b}.

Examples :-

Q ending with 0 $\Sigma = \{0, 1\}$

$(0+1)^* 0$

Q Writ R.E for string starting with 0 with $\Sigma = [0, 1]$

$0(0+1)^*$.

Q Subset of 01 is there.

$(0+1)^* 01 (0+1)^*$

Q Subset of 101 ending with 0.

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

Q. R.E for $L = \{a^m b a^n \quad m>0, n>0\}$.

$aa^* baa^*$ or $a^+ b a^+$

Q $L = \{a^n b^m \quad (n+m) \text{ is even}\}$

$= ((aa)^*(bb)^* + a(aa)^* b(bb)^*)$.

Q 10^{th} digit from right is 1.

$(0+1)^* 1 (0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)$

Q. R.E in which any no. of 0's is followed by any no. of 1 then followed by any no. of 2.

$0^* 1^* 2^*$

Q. all string of (0,1) with atleast two consequtive 0's.

$(0+1)^* 00 (0+1)^*$

Q. First symbol 1, last symbol 0.

$1(1+0)^* 0$

Q. $\Sigma = \{a, b\}$, even no. of a followed by odd no. of b.

$(aa)^* b(bb)^*$.

Identities of Regular Expression :-

1) $\phi + r = r$

2) $\phi r = \phi$

3) $\in r = \in$

4) $(r^*)^* = r^*$

5) $rr^* = r^+$

6) $\in^* = \in \qquad \phi^* = \in$

7) $r^+ r^* = r^*$

8) $\in + rr^* = r^*$

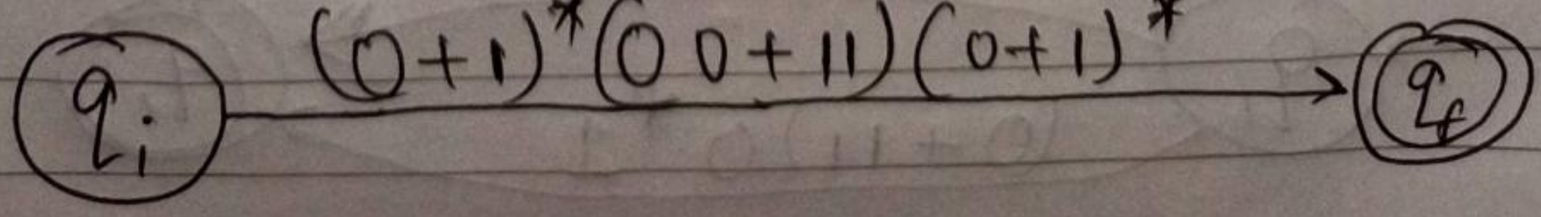
9) $(pq)^* p = p(qp)^*$

10) $(p+q)^* = (p^* + q^*)^* = (p^* q^*)^*$.

Convert R.E to F.A

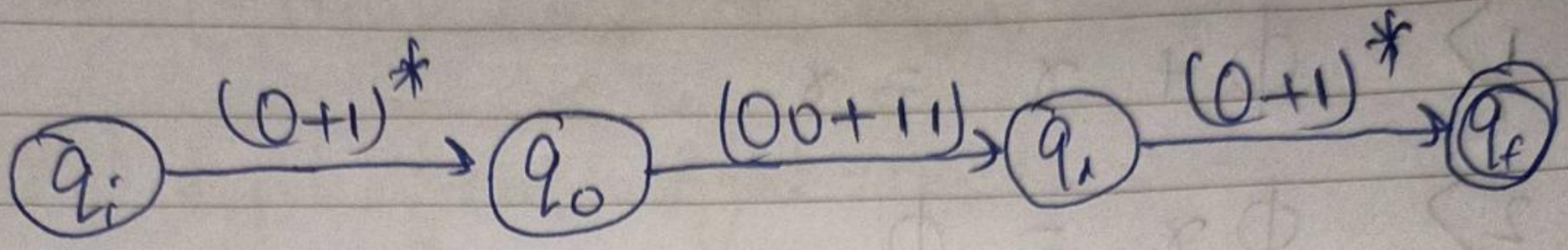
Q $(0+1)^* (00+11) (0+1)^*$.

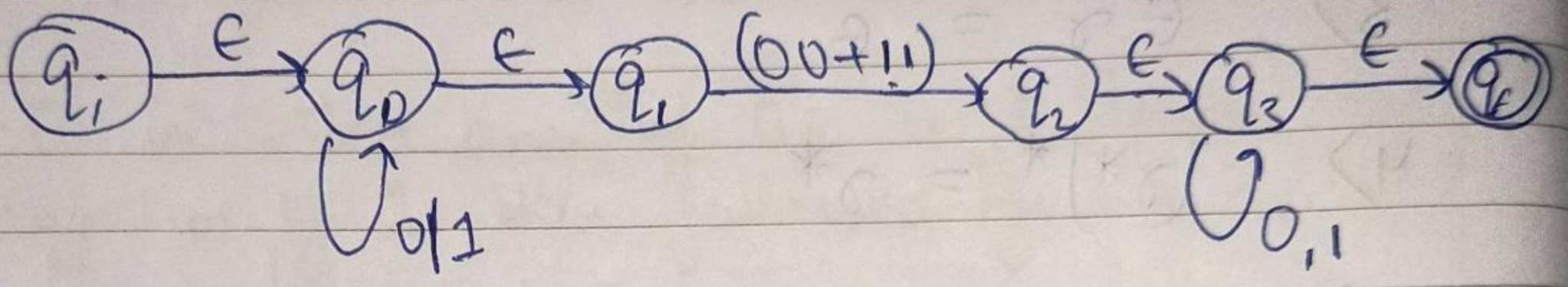
Sol →

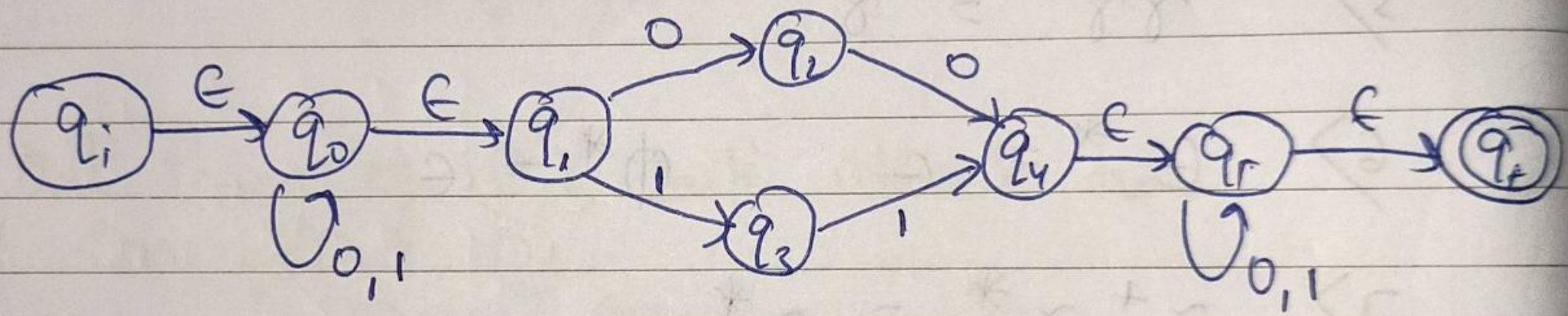


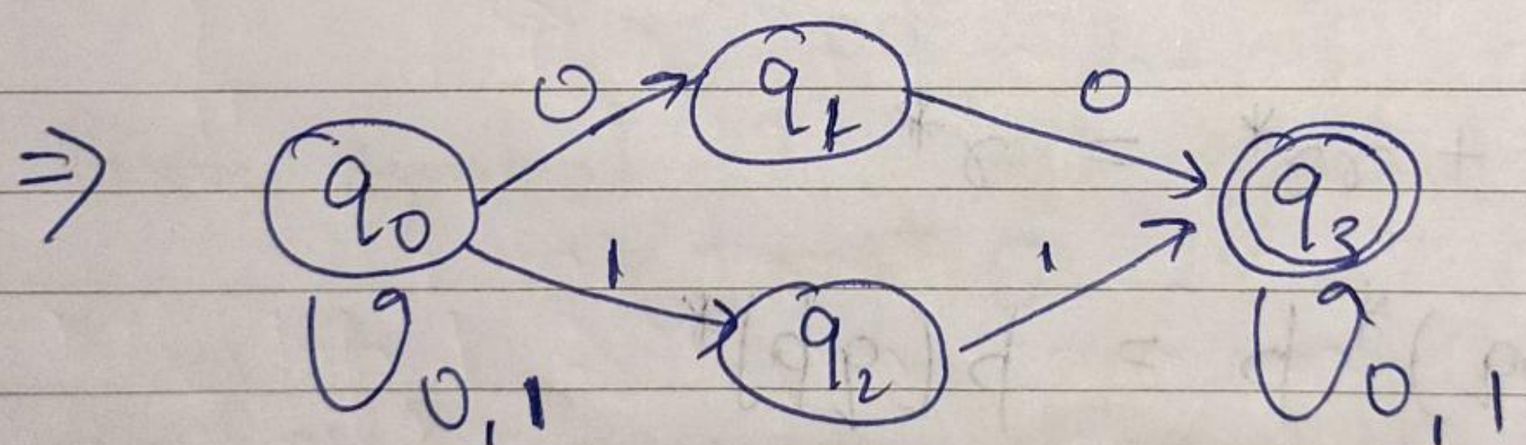
$(0+1)^*$ $[00+11)$ $(0+1)^*$

A B C.



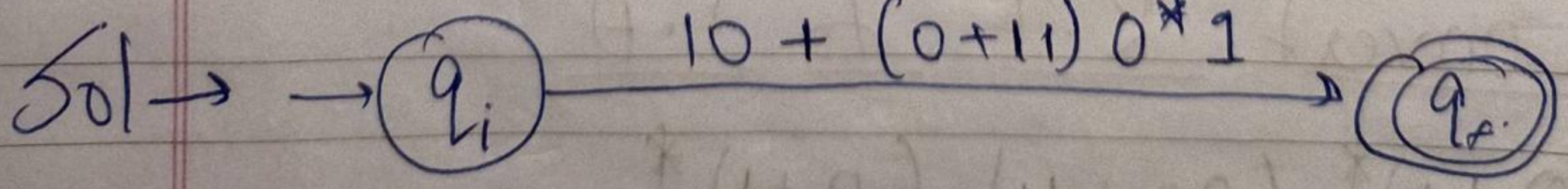


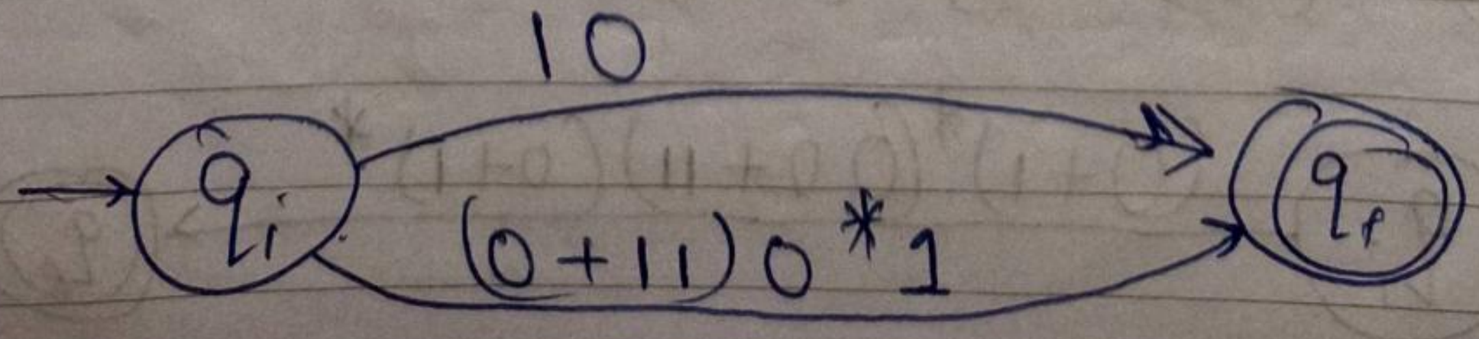


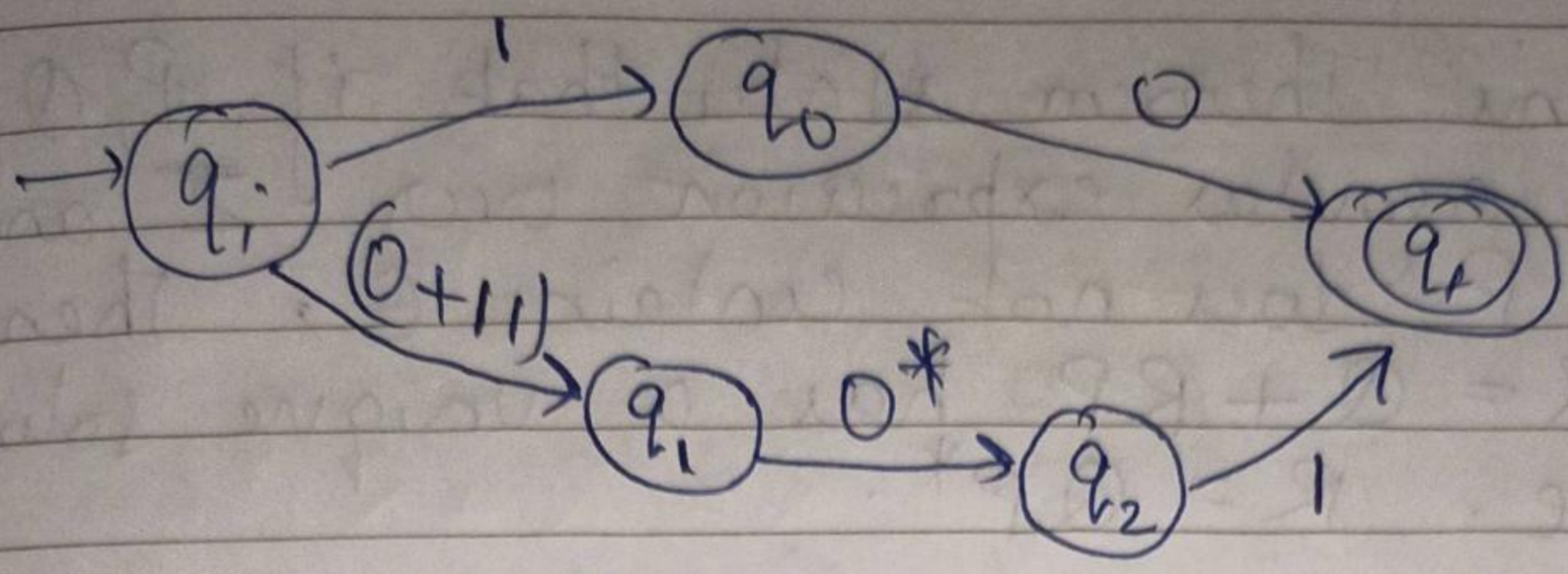


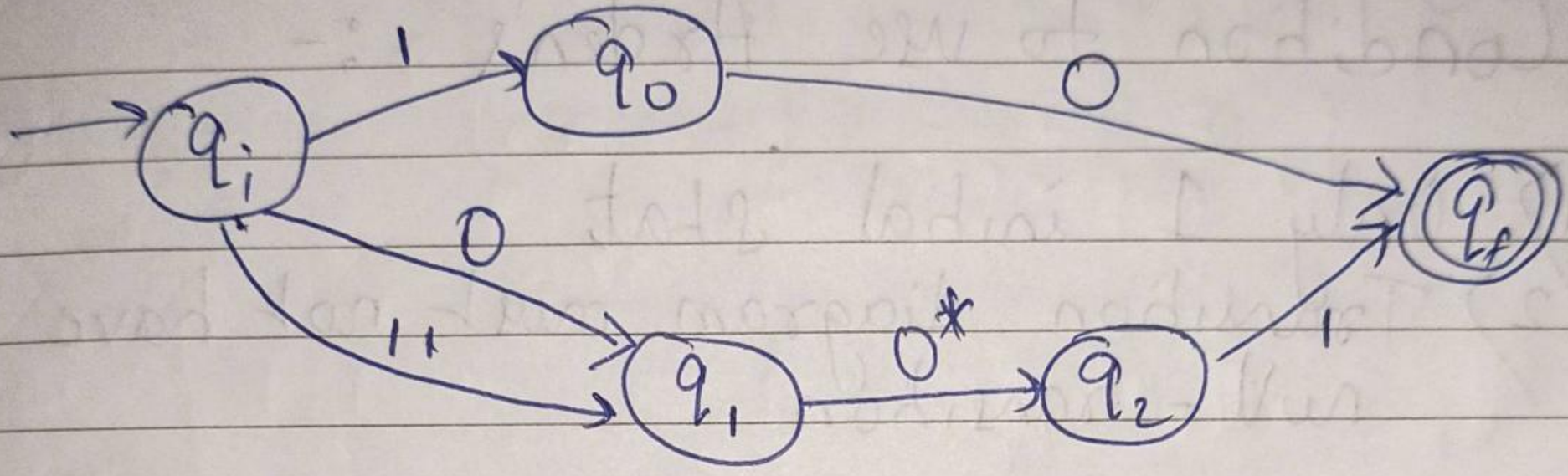
$=$

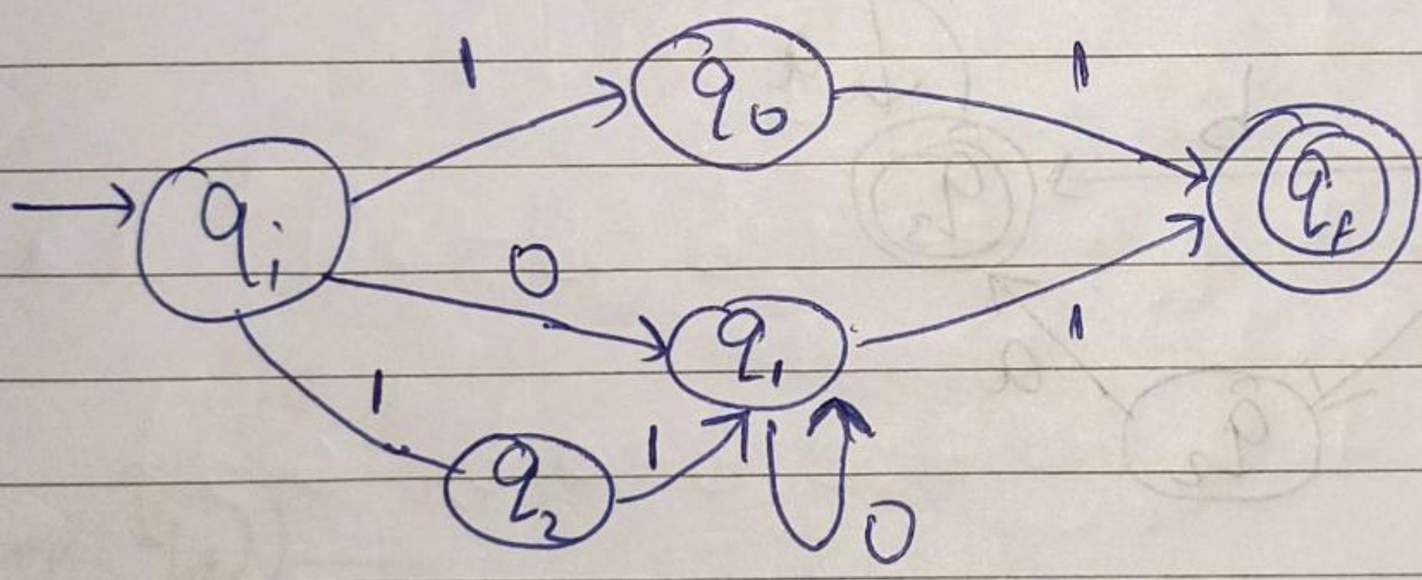
Q. $10 + (0+11)\,0^*1$.











Ans

DFA to R.E

Arden's Method

Stat Elimination Method.

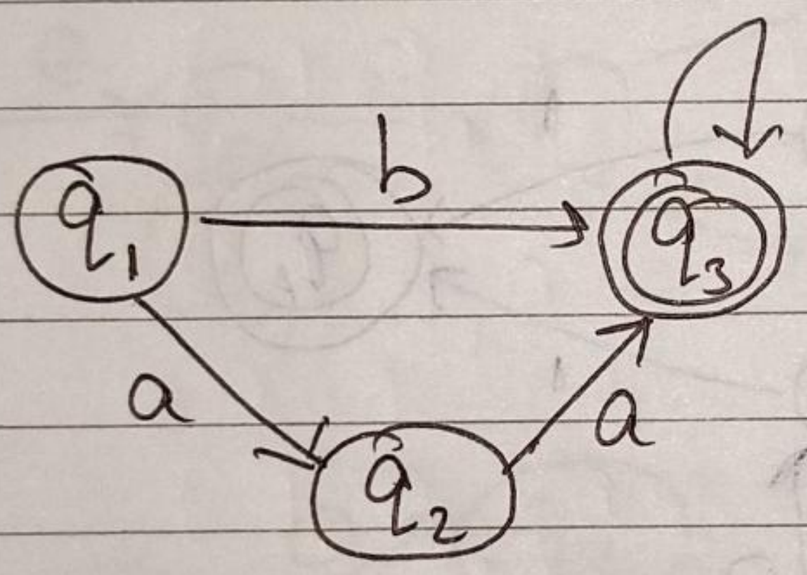
Arden's Method :- It is Based on Arden's Theorem which is popularly used to convert a given DFA to its regular expression.

Arden's Theorem states that if P, Q, R are regular expression over Σ and if P does not contain $\in$; Then $R = Q + RP$ has a unique solution i.e. $R = QP^*$.

Condition to use Arden's :-

1) Only 1 initial stat
2) Transition diagram must not have null - transition.

Q



Sol → Step 1 : Form equation with all the stat considering transition coming from other stat towards that stat. Add $\in$ in initial stat equation.

$$q_1 = \in + q_2 a \quad \text{---} \quad (1)$$

$$q_2 = q_1 a \quad \text{---} \quad (2)$$

$$q_3 = q_1 b + q_2 a + q_3 a \quad \text{---} \quad (3)$$

Step 2 :- Bring final stat in form $R = Q + RP$ to get required expression.

$$q_3 = q_1 b + q_2 a + q_3 a$$

$$q_3 = q_1 b + (q_1 a) a + q_3 a \quad (\text{Using ii})$$

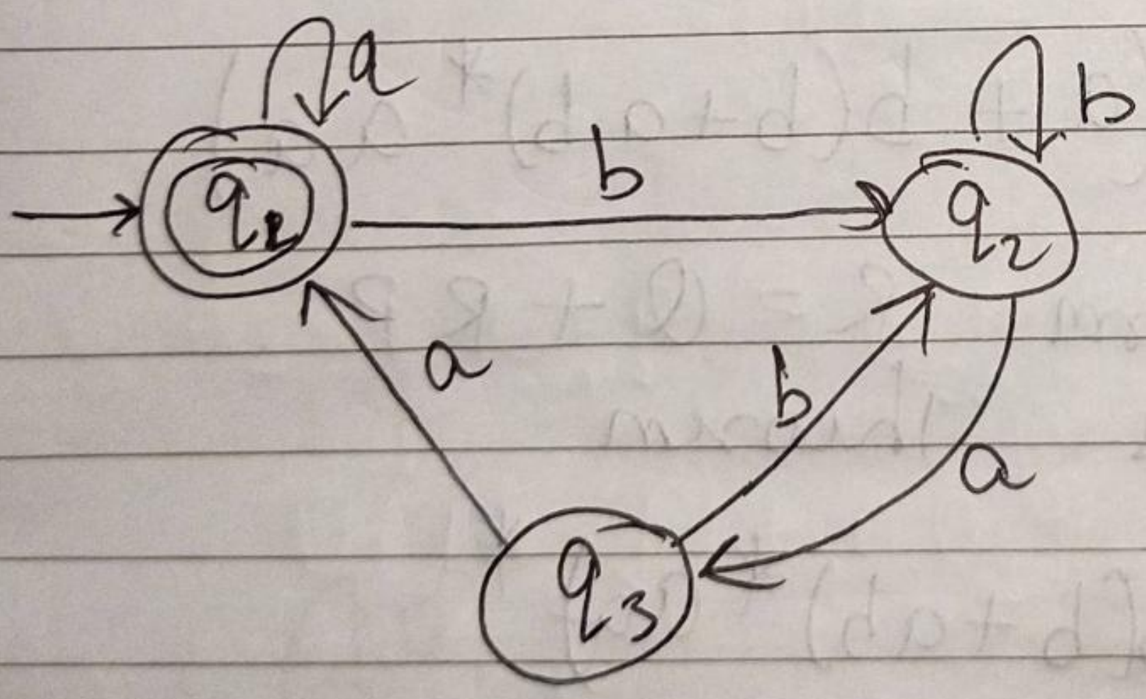
$$q_3 = (b + aa) q_1 + q_3 a$$

$$q_3 = (b + aa) + q_3 a \quad (\text{Using i}).$$

This is in form $R = P + QP$

$$\Rightarrow q_3 = (b + aa) a^* \quad \underline{\text{Ans}}$$

Q.



Sol →

$$q_1 = \epsilon + q_1 a + q_3 a \quad -- (i)$$

$$q_2 = q_1 b + q_2 b + q_3 b \quad -- (ii)$$

$$q_3 = q_2 a b$$

$$q_2 = q_1 b + q_2 b + (q_2 a) b \quad \text{(using iii)}$$

$$q_2 = q_1 b + q_2 (b + ab)$$

This is in form $R = Q + RP$

So, by Arden's Theorem:

$$q_2 = (q_1 b)(b + ab)^* \quad \text{--- (iv)}$$

$$q_1 = \epsilon + q_1 a + q_3 a \quad \text{---}$$

$$q_1 = \epsilon + q_1 a + q_2 aa$$

$$q_1 = \epsilon + q_1 a + q_1 b (b + ab)^* aa \quad \text{(using ii)}$$

$$q_1 = \epsilon + q_1 (a + b(b + ab)^* aa)$$

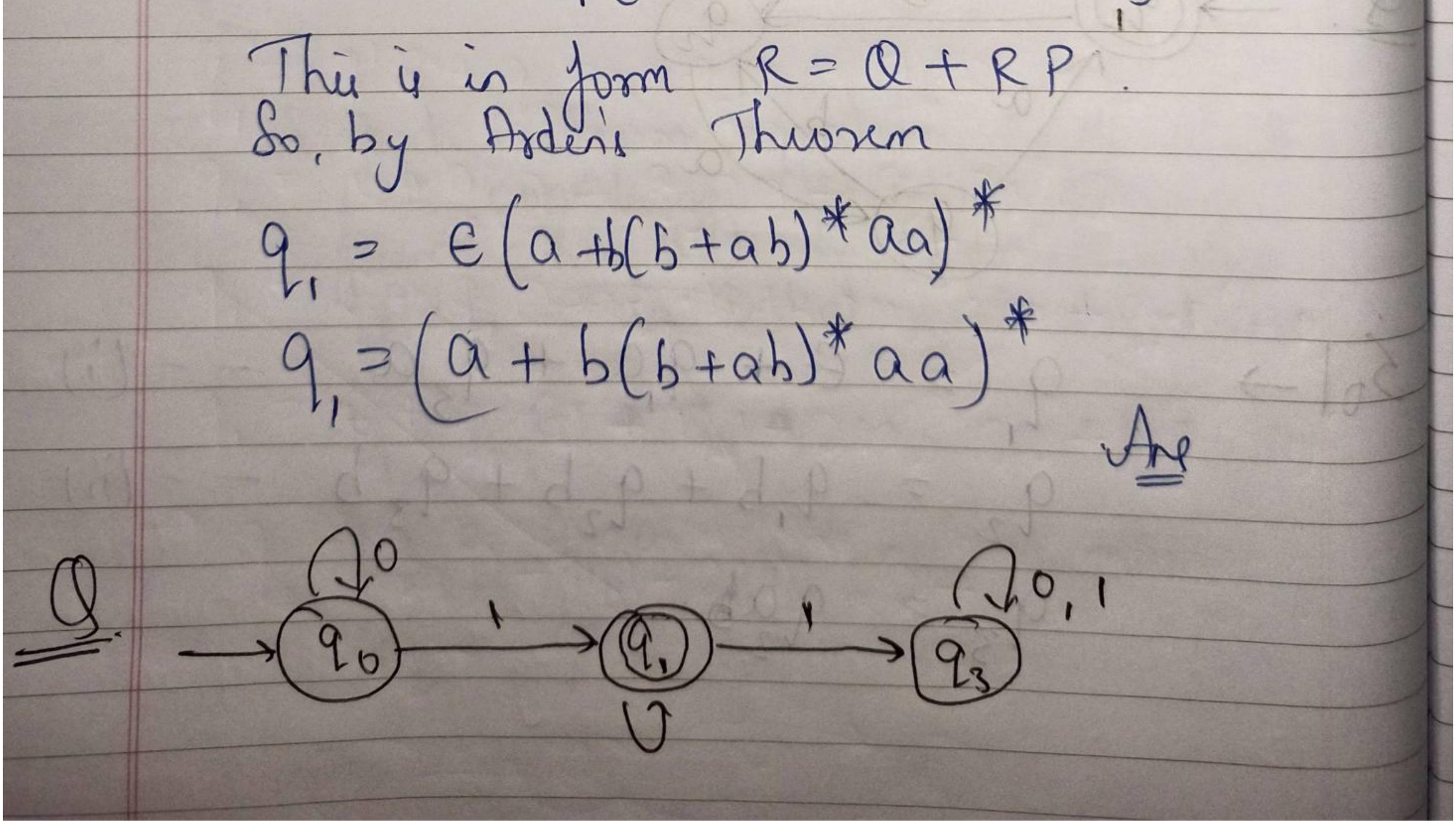
This is in form $R = Q + RP$.

So, by Arden's Theorem

$$q_1 = \epsilon (a + b(b + ab)^* aa)^*$$

$$q_1 = (a + b(b + ab)^* aa)^*$$

Ans



Sol → $q_0 = \epsilon + q_0 \cdot 0$ — (i)

$q_1 = q_0 \cdot 1 + q_1 \cdot 0$ — (ii)

$q_3 = q_1 \cdot 1 + q_3 \cdot 0 + q_3 \cdot 1$ — (iii)

$q_0 = \epsilon + q_0 \cdot 0$

This is in form $R = Q + RP$ so by Ardens

$q_0 = 0^*$ — (iv)

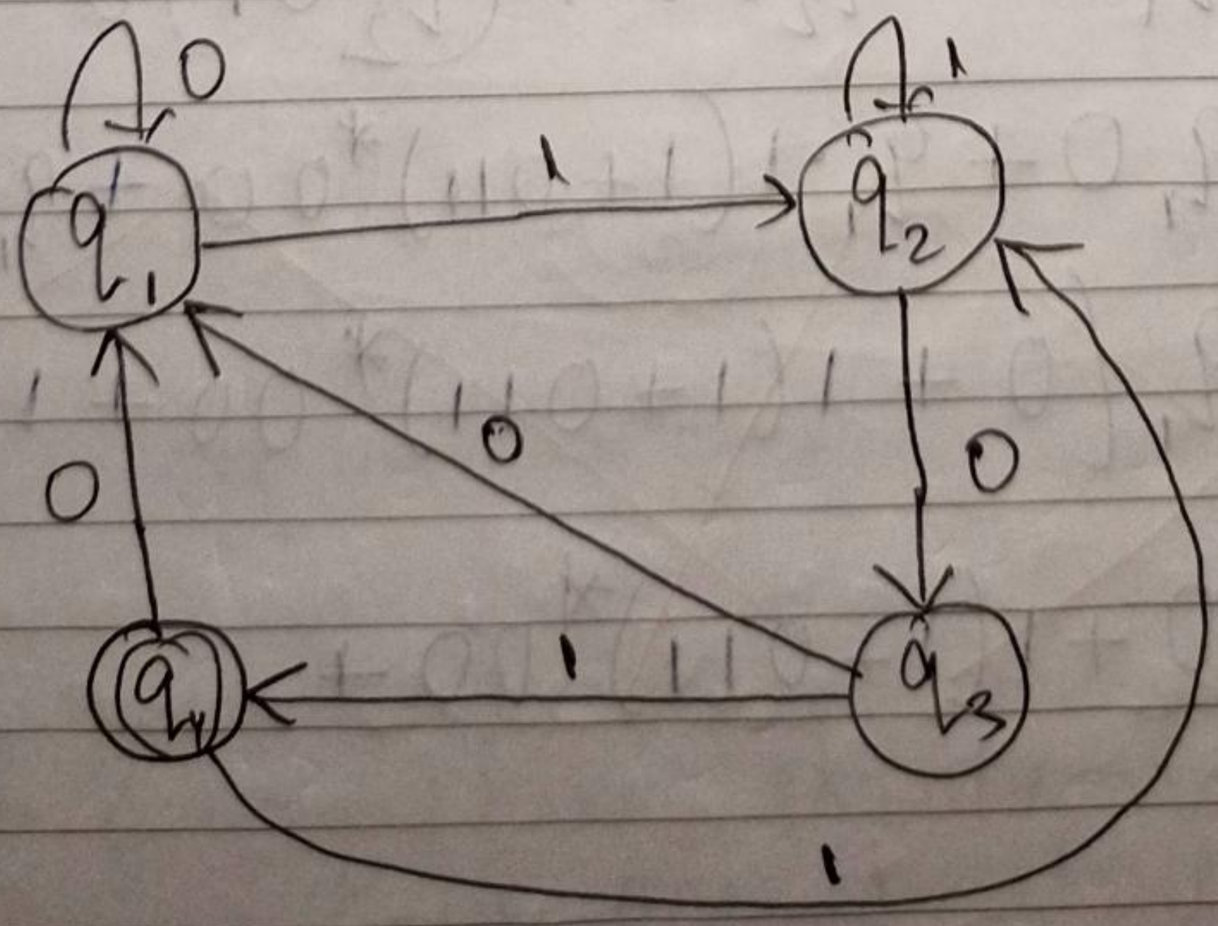
Put this in (ii)

$q_1 = 0^* 1 + q_1 \cdot 0$

This is in form $R = Q + RP$ so, by Arden's Theorem

$q_1 = 0^* 1 0^*$ Ans

Q.



Sol → $q_1 = \varepsilon + q_1 0 + q_3 0 + q_4 0$ — (i)

$q_2 = q_1 \cdot 1 + q_2 \cdot 1 + q_4 \cdot 1$ — (ii)

$q_3 = q_2 \cdot 0$ — (iii)

$q_4 = q_3 \cdot 1$ — (iv)

$q_2 = q_1 \cdot 1 + q_2 \cdot 1 + q_4 \cdot 1$ —— v

$q_2 = q_1 \cdot 1 + q_2 \cdot 1 + (q_2 \cdot 01)1$ $\left[\because q_4 = q_2 01\right]$

$q_2 = \underset{R}{\underline{q_1 \cdot 1}} + \underset{Q}{\underline{q_2}}\underset{P}{\underline{(1 + 011)}}$

$q_2 = q_1 \cdot 1 \ (1 + 011)^*$

$q_1 = \varepsilon + q_1 0 + q_3 0 + q_4 0$

$q_1 = \varepsilon + q_1 0 + q_2 00 + (q_2 01)0$

$q_1 = \epsilon + q_1 0 + q_2 (00 + 010)$

$q_1 q_1 = \epsilon + q_1 0 + q_1^2 \cdot 1 (1 + 011)^* (00 + 010)$.

$q_1 = \epsilon + q_1 (0 + 1 (1 + 011)^* (00 + 010))$.

$q_1 = (0 + 1 (1 + 011)^* (00 + 010))^*$

$q_4 = q_3 \cdot 1$

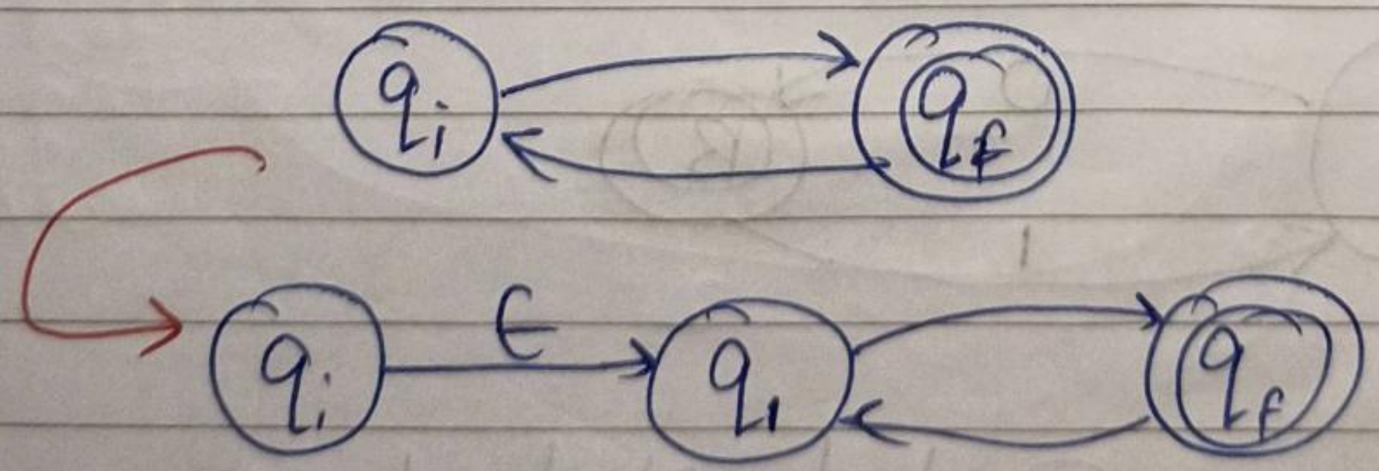
$= q_2 \cdot 01$

$= q_1 \cdot 1 (1 + 011)^* 01$

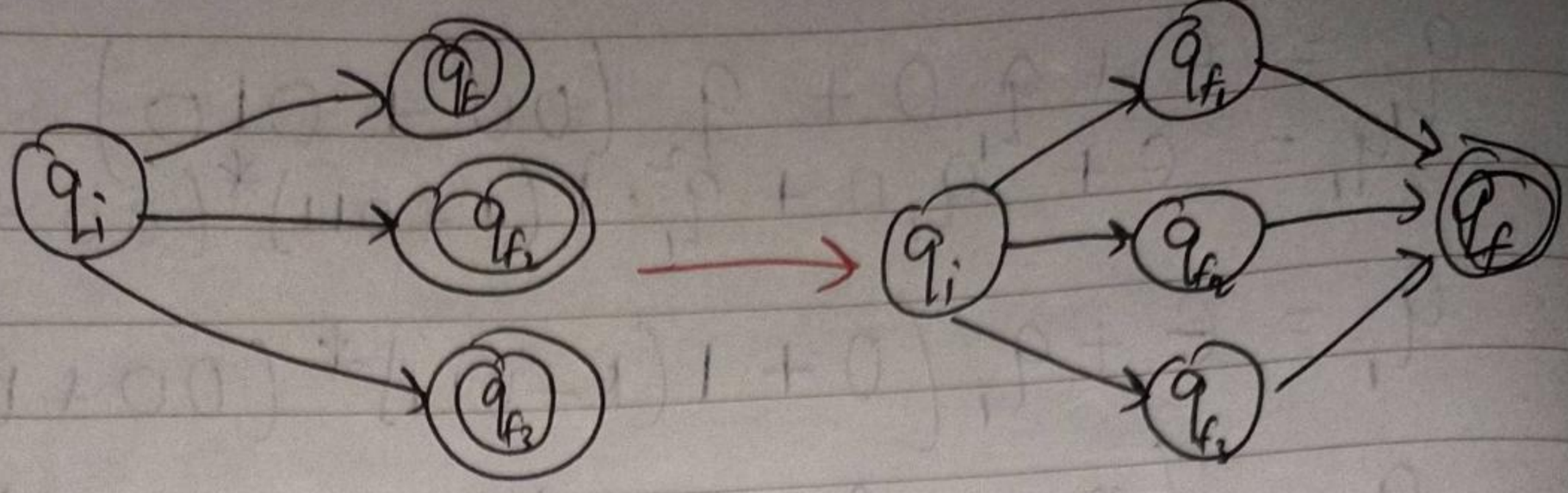
$q_4 = (0 + 1 (1 + 011)^* (00 + 010))^* 1 (1 + 011)^* 01$

State Elimination Method :-

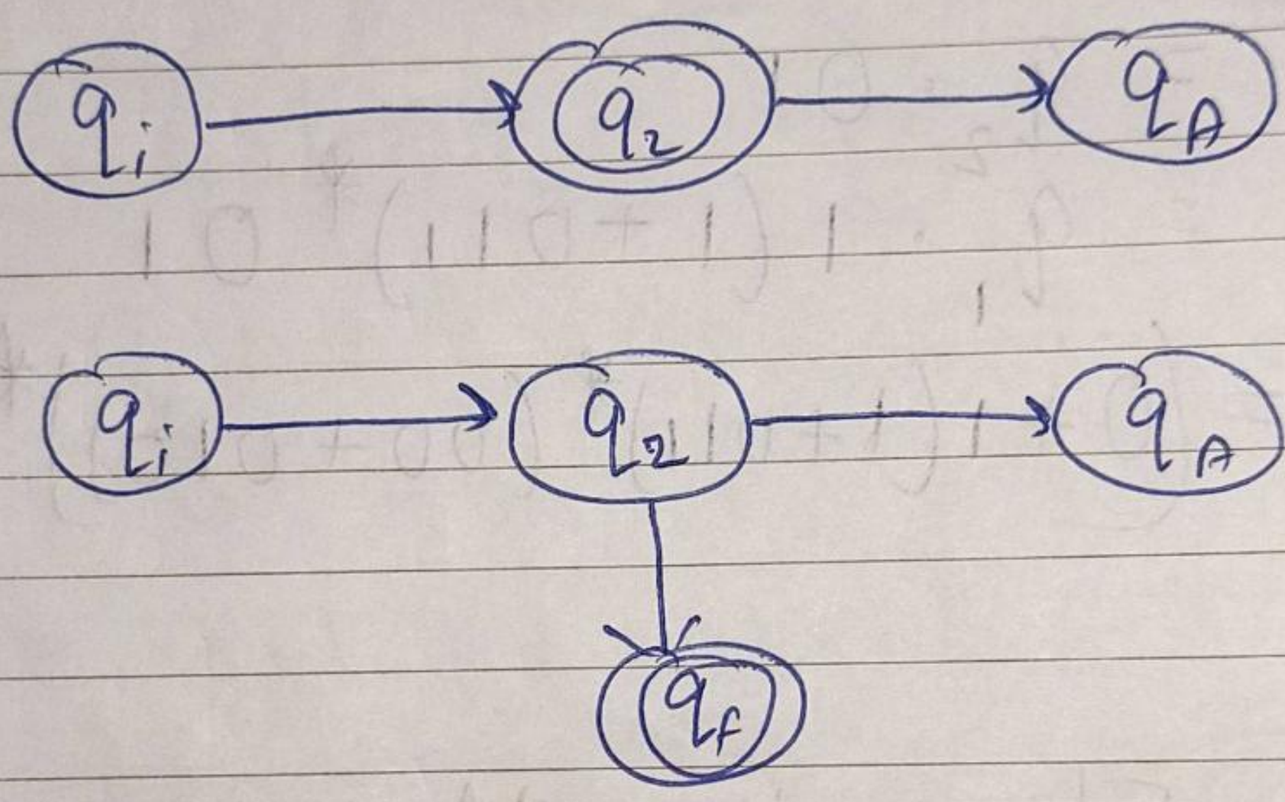
Step 1 :- There should be no incoming edge to initial state.



Step 2 :- There should not be more than one final state.

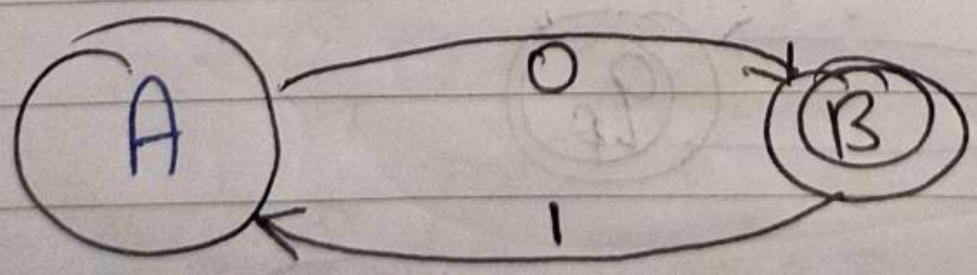


Step 3 :- There should not be outgoing edge from final stat

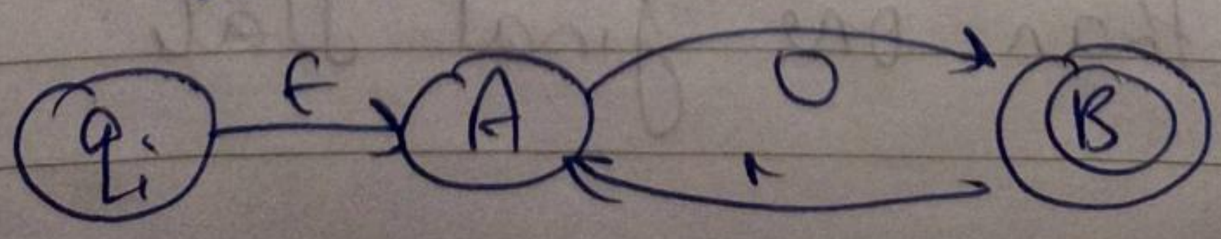


Step 4 :- Eliminat all intermediat stat one by one.

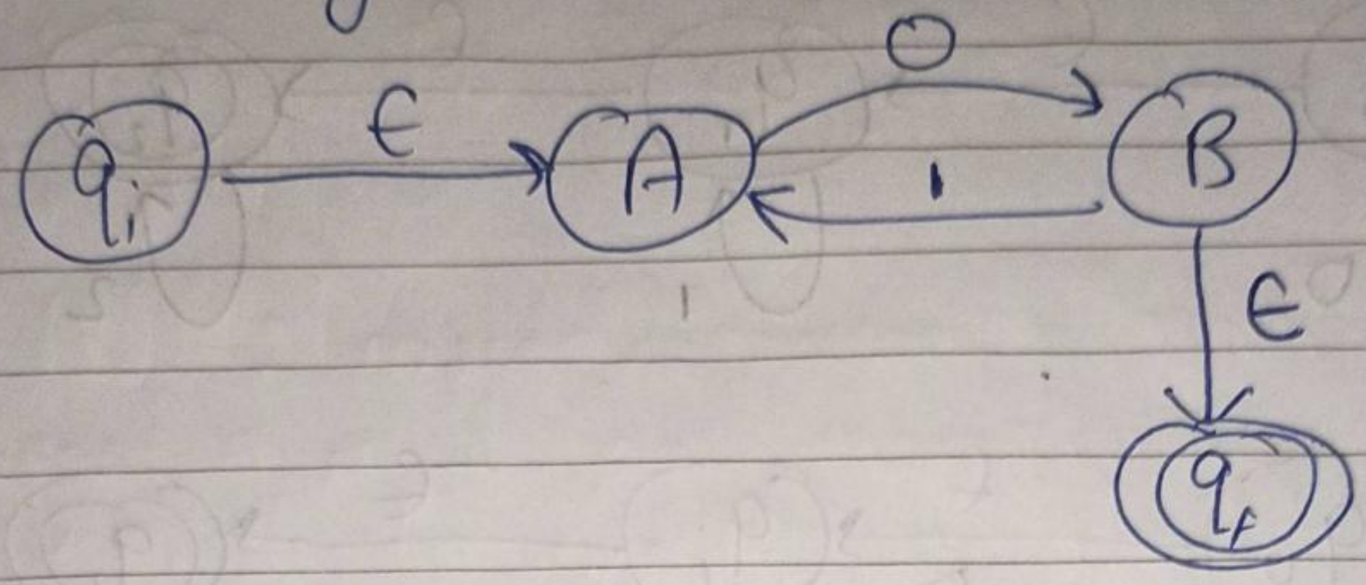
Q



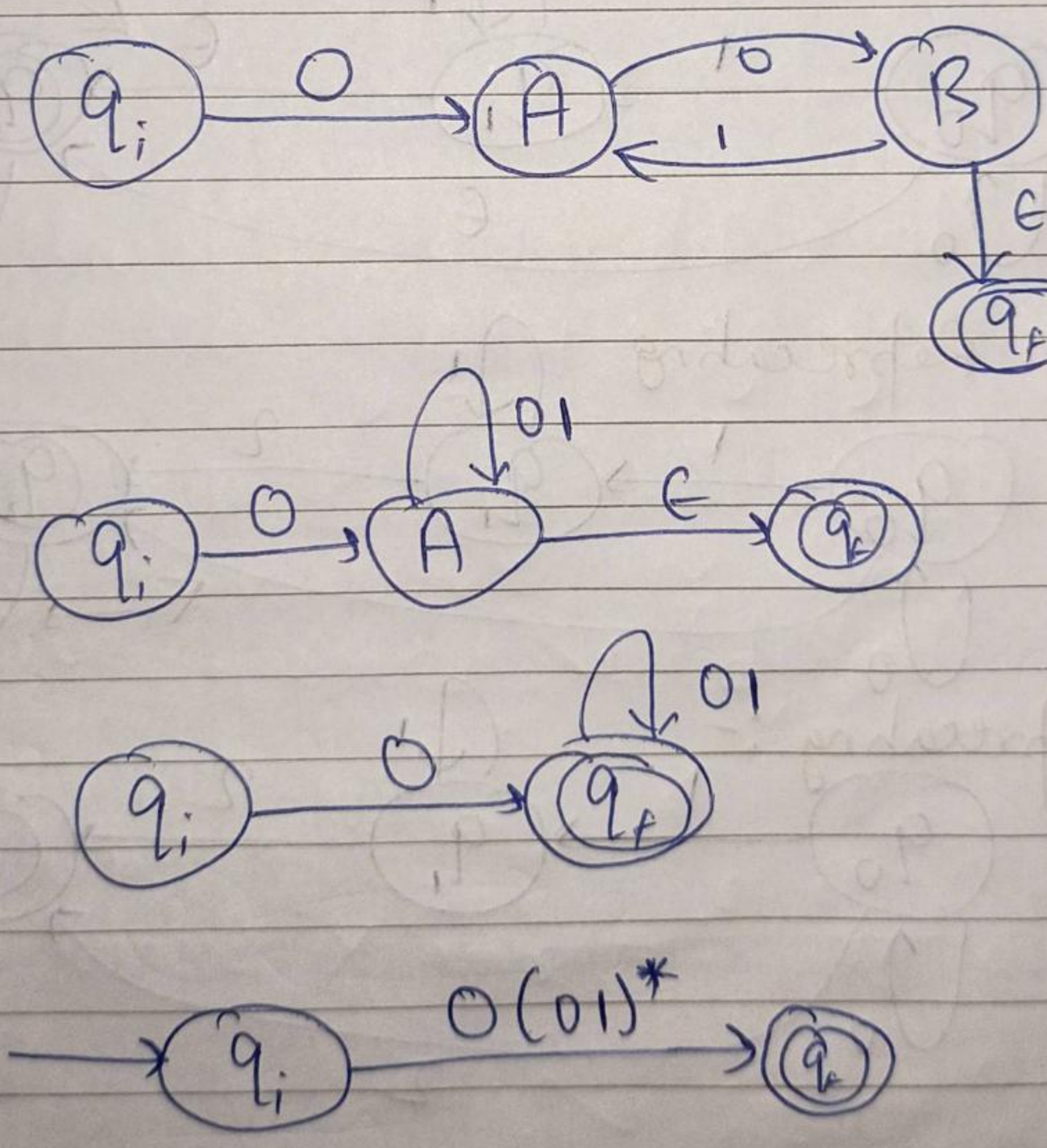
Sol → Step 1 :- Initial stat has incoming edge so. we creat new stat q_i



Step 2 :- Final stat has ~~incoming~~ outgoing edge. So, we add new stat q_f.



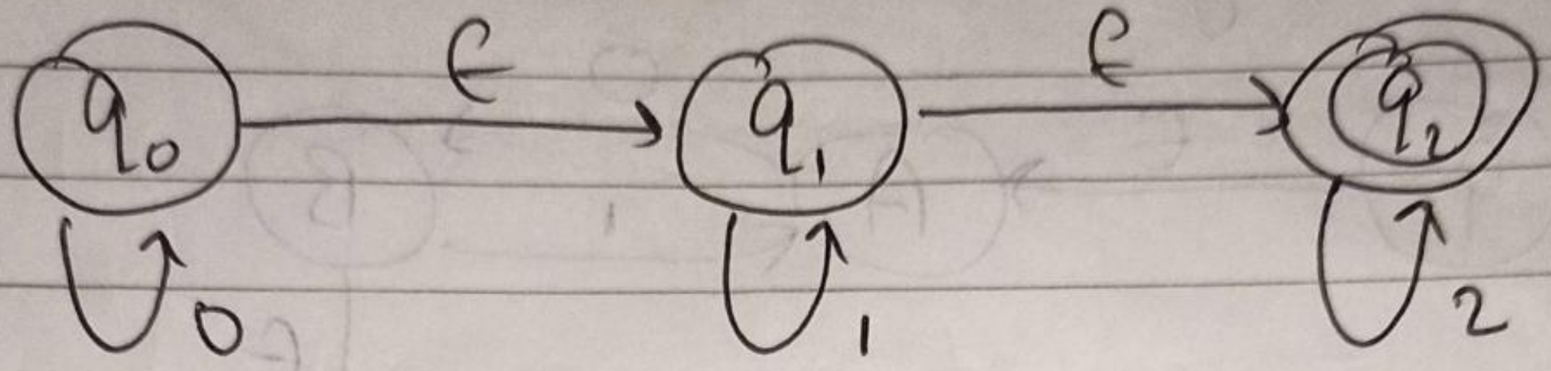
Step 3 :- Now, we start eliminating intermediat stat.



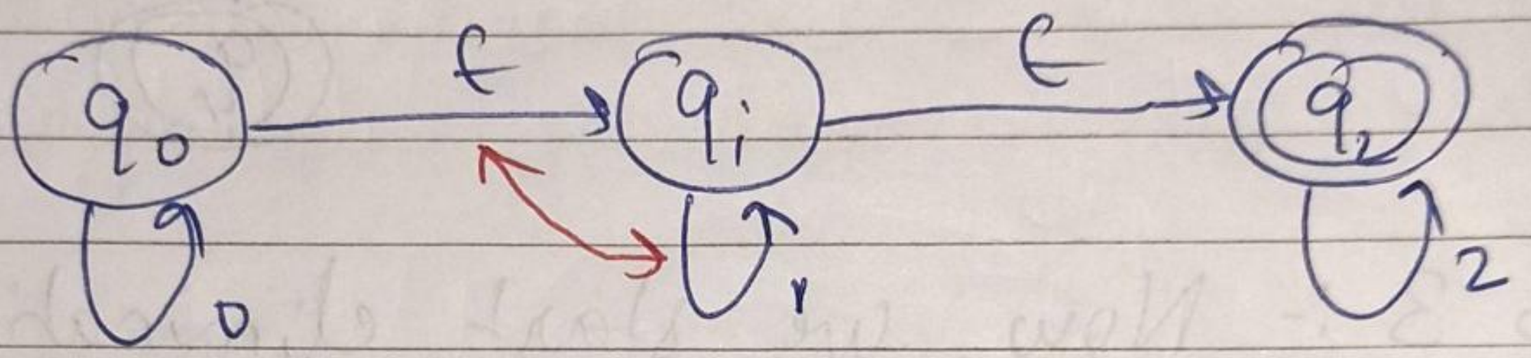
So, regular expression → $0(01)^*$.

Conversion of NFA with ϵ to without ϵ.

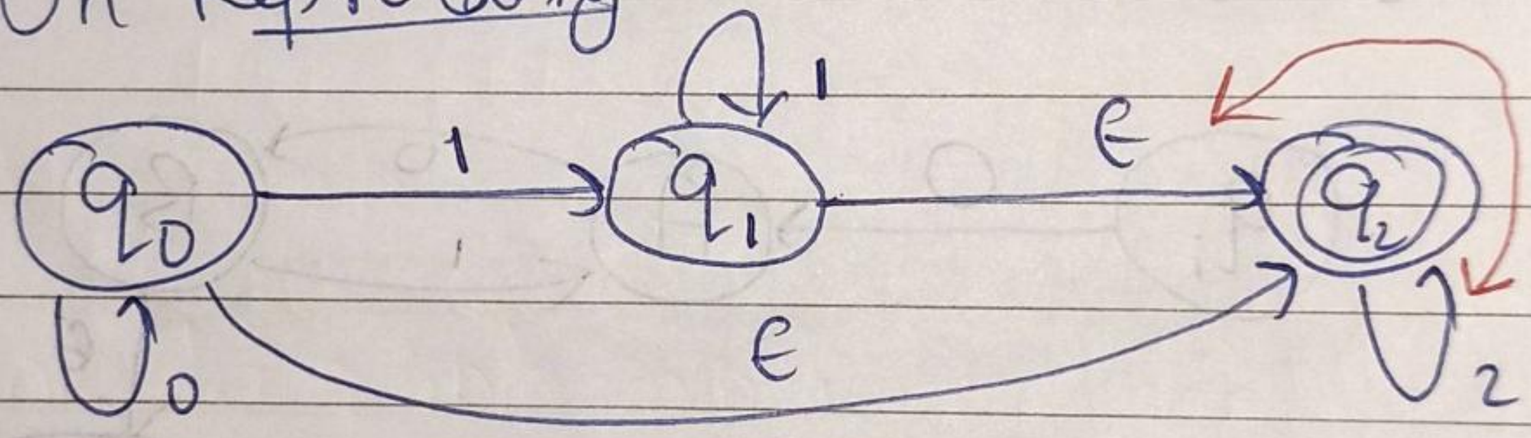
Q.



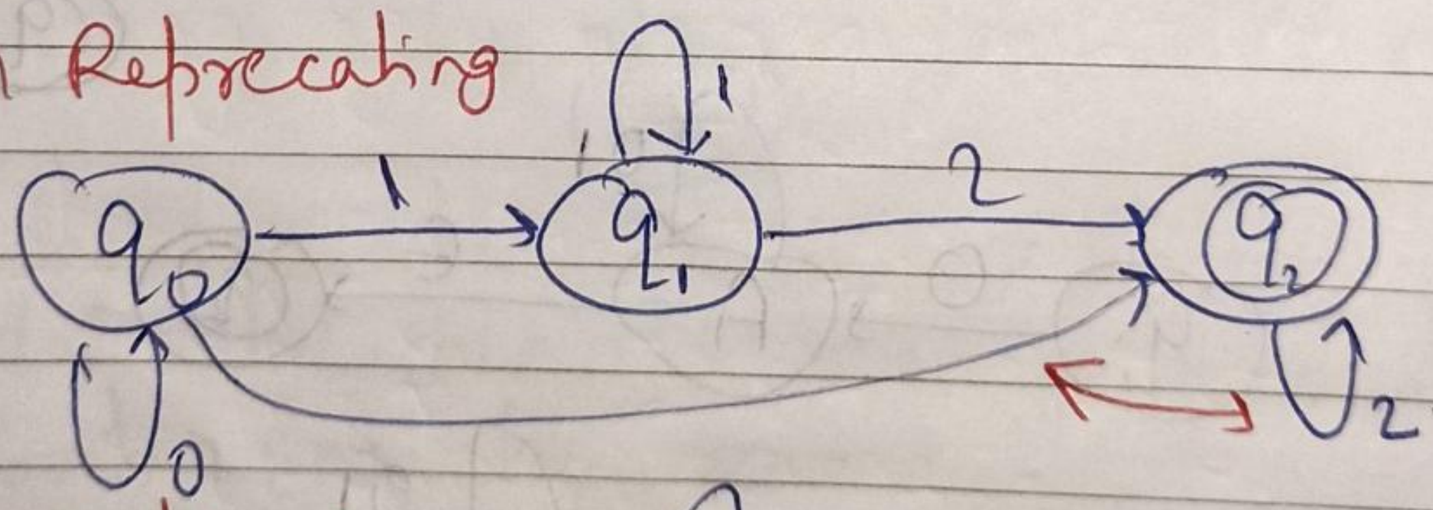
Sol →



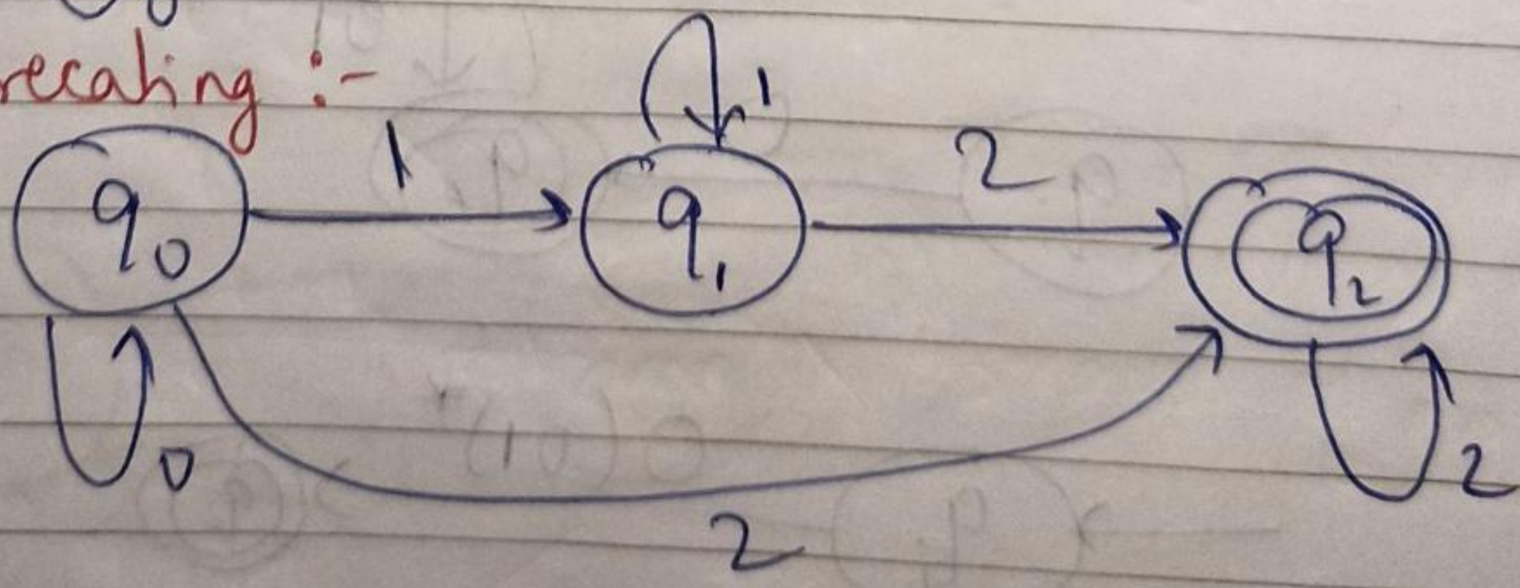
On Reprecating



On Reprecating



On reprecating :-



Regular Language

A language is regular if it can be expressed in term of regular expression.

finit	Infinit
→ String of length 0	→ String of having single b
→ String of length 1	→ String end with ab
→ String of length 2	→ String start with ba
→ Atmost / Atleast	→ begin & end with ab.

* R.E to R.L MCQ can be asked.

Closure Properties of Regular Language :-

1) Union :- If L1 and L2 are two regular language then their union is also regular.

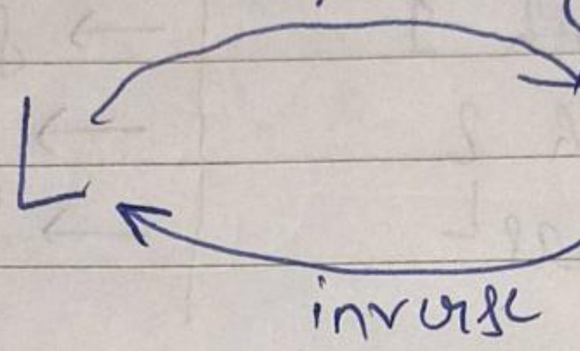
2) Intersection
3) Concatenation
4) Kleen Closure
5) Complement
6) ϕ, ϵ
7) Closed under reversal :- $L = ab$
   $L = ba$
8) Closure under Difference :- $A - B$
   $a^n - b^n$
9) Homomorphism :- It is a function that gives string for each symbol in that alphabet

E.g. $h(0) = ab$ $h(1) = c$

$L = \{01, 010\}$

$h(L) = \{abc, abcab\}$

10) Inverse homomorphism :- Let h be homomorphism and L is a language whose alphabets is output of h.



Point to remember to identify whether the language is regular :-

① Finite :- If any set is finite it is always regular

$a^n b^n$ ✓

a^{2^n} ✗ (Because there is GP here and for GP it is always not regular)

a^i ; i = prime ✗ (Because of no storage)

a^n $n \geq 2$ ✓

$a^n b^m$ $m = n$, $m \geq n$, $m < n$ ✗ (Because we have to compare & for comparison → Not regular)

$a^n b^n$ $n < 10^{10}$ → ∴

Concatenate :- If L_1 & L_2 both are regular then only $L_1 L_2$ is regular. Else NOT regular.

Mealy Machine :-

When output depend on the current input as well as state, then the FSM can be named to be mealy state Machine.

Mealy machine can be described by 6 tuples $(Q, q_0, \Sigma, O, \delta, \lambda')$.

$Q \rightarrow$ Set of states
$q_0 \rightarrow$ Initial state
$\Sigma \rightarrow$ finite set of input alphabet
$O \rightarrow$ output alphabet
$\delta \rightarrow$ Transition function $Q \times \Sigma \rightarrow Q$.
$\lambda' \rightarrow$ mapping function $Q \times \Sigma \rightarrow O$

* There is NO initial & final state in Mealy machine.

T.T of Mealy :-

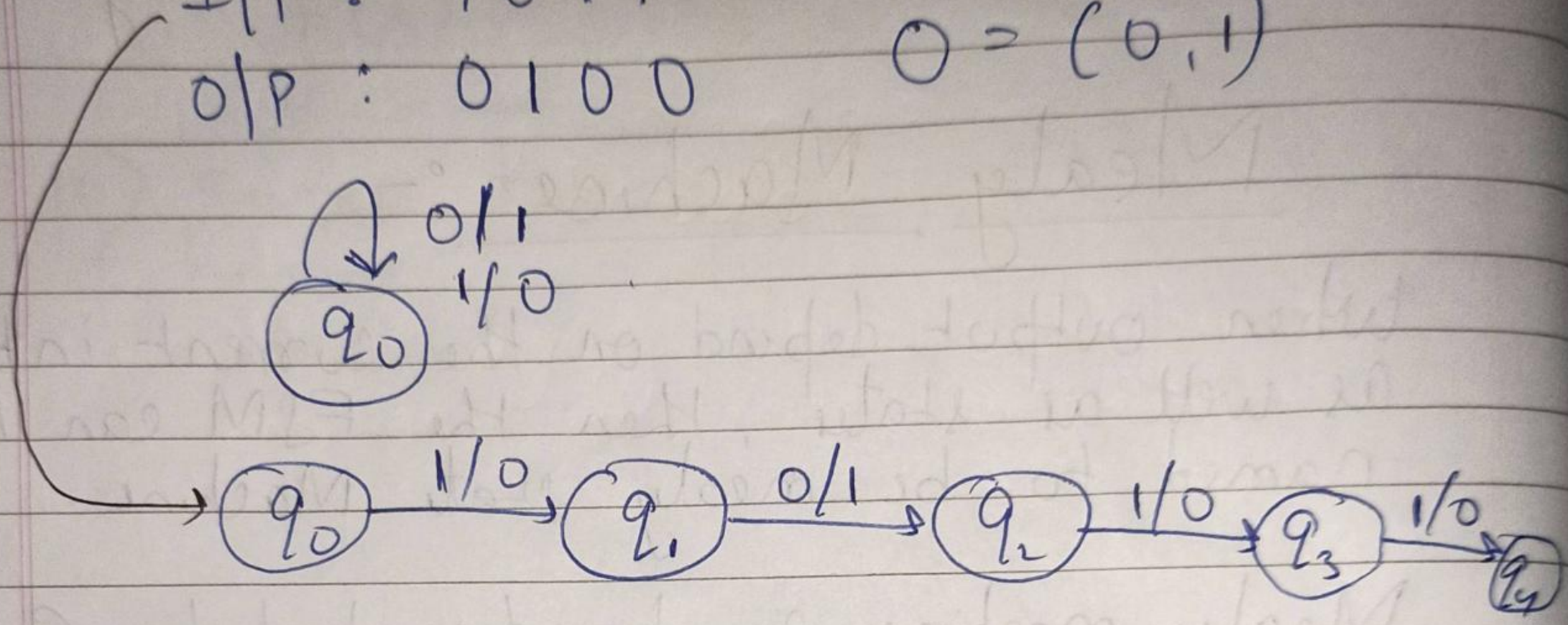
	a		b	
	State	O/P	State	O/P
q_0	q_1	b	q_2	b
q_1	q_2	b	q_1	a
q_2	q_2	b	q_0	a

Read → (q_0, a) input pe q_1 state pe jayega but output b hoga.

Q Design mealy m/c for 1's Complement.

Sol → I/P : 1011 $\Sigma = (0, 1)$

O/P : 0100 $O = (0, 1)$

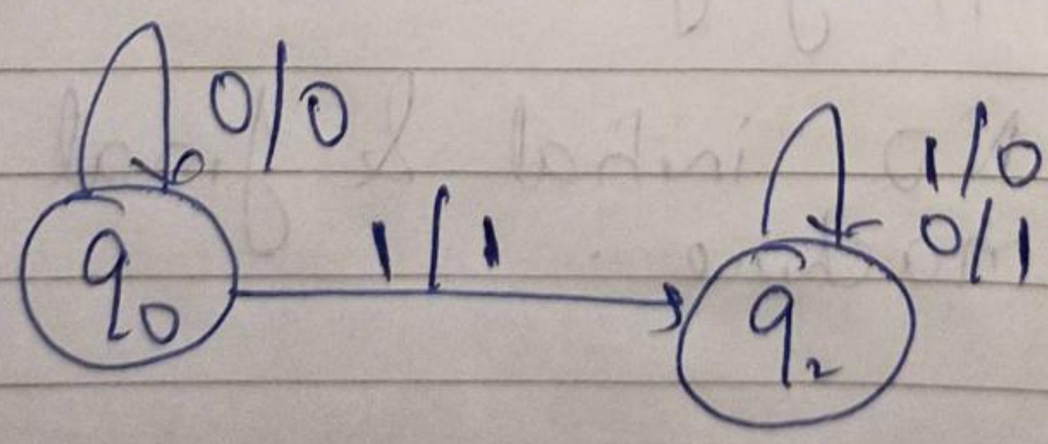


Q 2's Complement :-

I/P :- 11001 | 1 I/P : 11001 | 000

O/P :- 00110 | 1 O/P : 00110 | 000

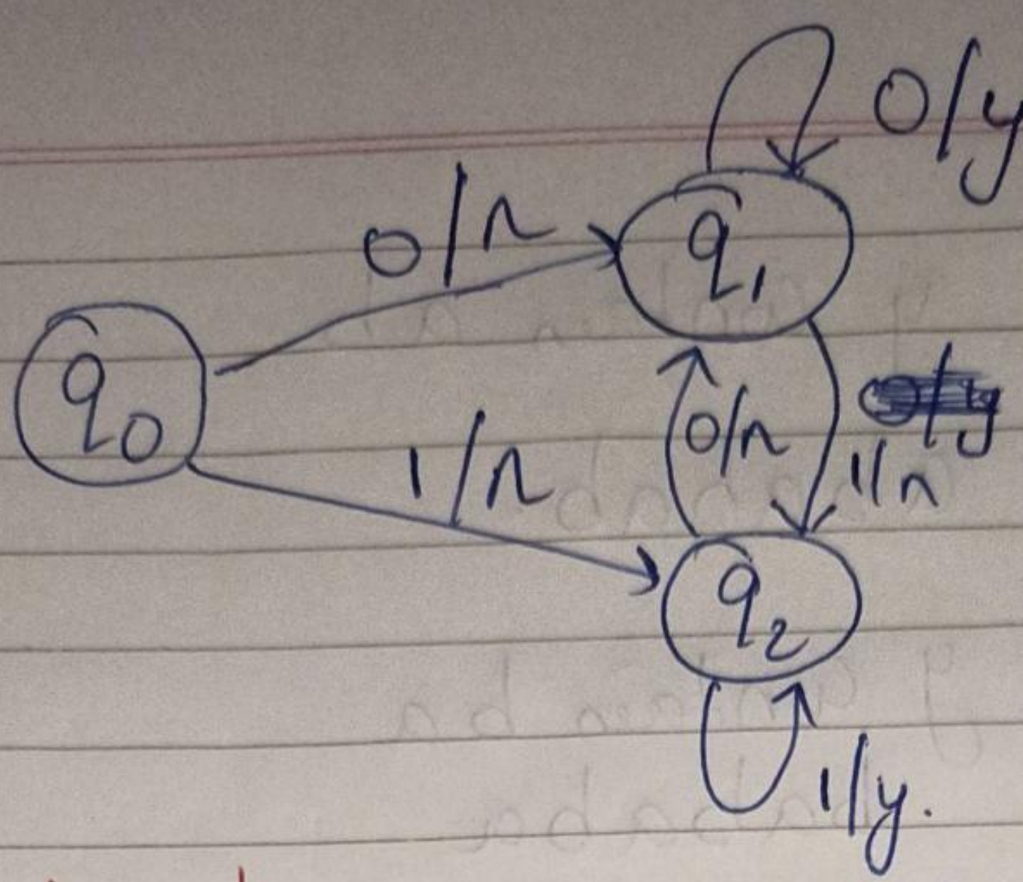
⇒ Isme jab 1st 1 aayega uske bad se inversion hoga else baki me same.



Q. $(0+1)^*(00+11)$

→ Ending with 00 or 11.

Sol →



→ Output is coming in this

Pumping Lemma :-

It is used to prove that language is not regular.

Statement → If A is a language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into 3 pieces s=xyz, satisfying following conditions :-

For every $i \geq 1$

(i) $y \neq \epsilon$

(ii) $xy \leq p$.

(iii) $xy^iz \in A$.

Q $w = a^1ba^2a \Rightarrow w = xyz$.

Sol → Case I : y contains b

$\Rightarrow$ b bbb

Case II :- y contain ab.

ababab.

Case III :- y contain ba

bababa

Since for any y it is not satisfying Condition. So. it is not Regular.