Unit -1

Basic concepts and Automata Theory

Theory of computation

Theory of computation deals with whether and how efficiently problems can be solved on a model of computation using an algorithm.

Types of Theory of computation (TOC)

① **Computability Theory** : This theory is given by Godel, Church, Turing, Kleene and Post. This theory is used to determine which problems are computable (solvable) and which are not computable (unsolvable). It is also called recursive theory.

② **Computational complexity theory** : This theory deals with determining the complexity of the problem during the computation. If the problem is efficiently solvable, then it is called 'computationally easy' and if it cannot be solved efficiently, then it is called 'computationally hard'.

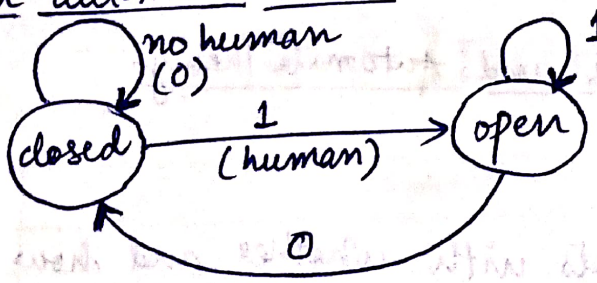
③ **Automata Theory** : Automata Theory deals with designing a self-propelled computing device (Automaton) that follows a predetermined sequence of operations automatically.

What is Automaton?

An automaton is a mathematical model of any computing device/machine that performs computations of an input by moving through a series of states.

- At each state of ~~the~~ computation, a transition function determines the next stage based on the present state and input.
- Once the computation reaches an accepting state, it accepts the input, else rejects it.

Eg An automatic door



Types of Automaton

1. Finite Automaton (FA)
2. Pushdown Automaton (PDA)
3. Linear Bounded Automaton (LBA)
4. Turing Machine (TM)

↓ Problem solving capability

Basic Terminologies

① **Symbol**: The symbol is the smallest building block in the theory of computation and can be any letter, number or even pictograms.

Eg a, b, A, Z, 0, 1, 

② **Alphabet**. An alphabet is a finite, non empty set of symbols. It is represented by Σ.

Eg $\Sigma\{0,1\}$ is binary alphabet
$\Sigma\{A,B,C,\ldots,Z\}$ is upper case alphabet
$\Sigma\{0,1,2,3,\ldots,9\}$ is decimal alphabet

③ **String**: It is the finite sequence of symbols from the alphabet.

Eg 101101 is a binary string,
25681 is a decimal string.
abcabbd is a lower case letter string,

Eg a, b, 1, 2, ab, aa, stu, a12b, etc.

* If there is no symbol in the string, it is called empty string. It is denoted by 'ϵ' (Epsilon) or 'λ'.

④ Length of the string : The length of the string is the number of symbols in that string. The standard notation for the length of string w is $|w|$

Eg $w = abcalo$

$|w| = 5$ = length of string.

- $|\lambda| = |\epsilon| = 0$.

⑤ Powers of Σ Let $\Sigma = \{0, 1\}$.

Σ^0 = set of all strings of length 0 : $\Sigma^0 = \{\epsilon\}$.

Σ^1 = set of all strings of length 1 : $\Sigma^1 = \{0, 1\}$.

Σ^2 = set of all strings of length 2 : $\Sigma^2 = \{00, 01, 10, 11\}$

Σ^3 = set of all strings of length 3 : $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

$\vdots$

Σ^n = set of all strings of length n.

⑥ Cardinality : no. of elements in a set

Cardinality of $\Sigma^n = |\Sigma^n| = 2^n$. ~~for $|\Sigma| = 2$~~.

if alphabet is over 2 entities.

⑦ Kleene closure

i) Kleen star closure (Σ^*)

The set of all possible strings over an alphabet Σ including the null string.

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$

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

$\Sigma^* = \{\epsilon\} \cup \{0, 1\} \cup \{00, 01, 11, 10\} \cup \{000, 001, 010, 011, 100, 101, 110, 111\} \cup \dots$

= set of all possible strings of all lengths over $\{0, 1\}$ (infinite set).

ii) Kleene positive closure : It is similar to Kleene star closure excluding null string (ϵ)

$$\Sigma^+ = \Sigma^* - \{\epsilon\}$$

If $\Sigma = \{0,1\}$, so, $\Sigma^+ = \{0, 1, 00, 01, 11, 10, 000, 001, 010, 011, 100, 101, 110, 111, 0000, \ldots\ldots\}$.

⑧ Reverse of a string (ω^R) : It is obtained by writing the symbols of the string in reverse order.

Eg of $\omega = abc$, then $\omega^R = cba$.

⑨ Language (L) : A language is a subset of strings over an alphabet Σ following some specific set of rules. It is represented by L.

If Σ is an alphabet, then L is a subset of Σ^* following certain rule (S)

Eg

- $\Sigma = \{0,1\}$, L_1 = set of all strings of length 2
  $= \{00, 01, 10, 11\}$. (finite Language)

- $\Sigma = \{0,1\}$, L_2 = set of all strings that begin with 0
  $= \{0, 00, 01, 000, 001, 010, 011, 0000, \ldots\}$

- $\Sigma = \{0,1\}$, $L_3 = \{\omega \in \{0,1\}^* \mid \omega$ has equal no. of 0s & 1s$\}$
  $= \{\lambda, 01, 10, 0011, 1100, 0101, 0110, 1010, \ldots\}$

- $\Sigma = \{a,b,c\}$, $L_4 = \{\omega \in \{a,b,c\}^* \mid \omega$ has strings starting with a & ending with c$\}$
  $= \{ac, abc, aabc, aac, abcc, abbc, \ldots\}$.

Infinite Languages

- Σ^* is called mother of all languages or universal set.

(10) **Grammar** - It specifies the rules that are required to generate strings (words) of a language. It is represented by G.

Each Grammar has 4 tuples :- $G = \{N, T, S, P\}$.

- N → set of non-terminals (variables). Represented by capital letters, eg: A, B, E, - - -
- T → set of terminals (constants). Represented by small letters or numeric values.

  eg: a, b, c, d, 1, 2, 3 - - -
- S → start Non terminal (variable). Used to create strings of languages.
- P → Production rule

  $\alpha \rightarrow \beta$

  (α: Left side of rule) (β: Right side of rule).

(11) **Formal Languages**

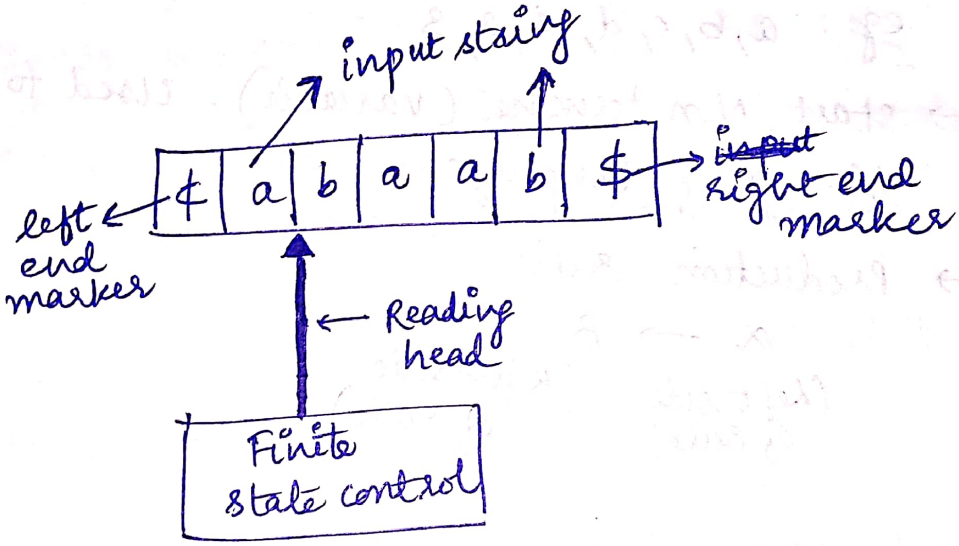
In the theory of computation, a formal language is a set of strings defined over some alphabet, where each string conforms to specific rules or patterns. These rules are usually described using mathematical models ~~such as~~ or automata. The concept of meaning is typically abstracted away, and the focus is on the syntactic structure of the language.

- Automata focuses on Formal languages.

Finite Automata (FA)

It is the most restricted mathematical model of computing devices (automatic machines). It is used in various applications such as string matching, lexical analysis phase of compiler, pattern matching, text editors and verification of hardware digital circuits.

- **Block diagram**

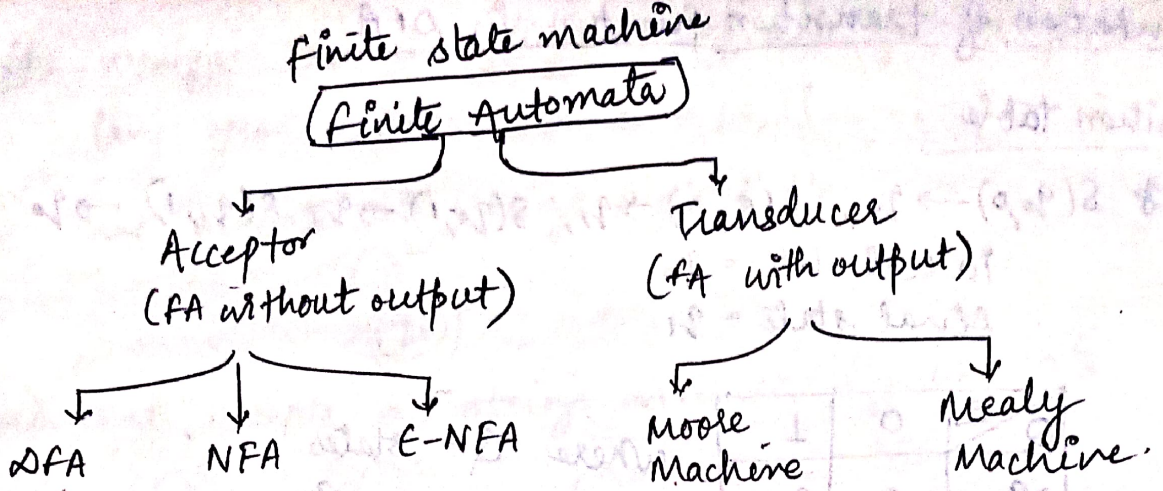


- **Components of finite automata**

Input Tape
- → divided into squares, each square consists of single symbol from the input alphabet.
- → There is left end marker (¢) and right end marker ($) on the tape.
- → Presence of end marker indicate that input tape is of finite length.

Reading Head
- → Reading Head is used to read symbol from the input tape.
- → Reading head reads one symbol at a time.
- → Normally reading head reads from left to right. However in specific FA (2-way FA). It moves in both direction.

Finite control
- → Finite control is responsible for controlling overall functioning of FA.
- → It decides which input symbol to read next and which next state to move on reading input.



DFA (Deterministic Finite Automata)

- It is represents a finite state machine that accepts or rejects a string
- 'Deterministic' specifies that the transition of DFA from one state to another state on a given input is unique i.e. there is only one next state.
- DFA are called finite because there are limited number of possible states that can be reached.
- It is necessary to define DFA for each input in each state.
- Null string (λ) is not allowed.
- DFA has 5 tuples representation.

$$M = \{Q, \Sigma, \delta, q_0, F\}.$$

where Q = ~~Total number of~~ finite, non empty set of states
Σ = finite, non empty set of input alphabets.
q_0 = Initial state
F = set of final states
δ = Transition function.

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

(Q: present state, Σ: input alphabet, Q': next stage)

~~Eg~~

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

- DFA is difficult to design
- DFA is can not be converted to NDFA.

• Representation of transition function in DFA

① Transition table :

eg $\delta(q_0,0) \to q_0$, $\delta(q_1,0) \to q_1$, $\delta(q_0,1) \to q_1$, $\delta(q_1,1) \to q_0$

initial state = q_0

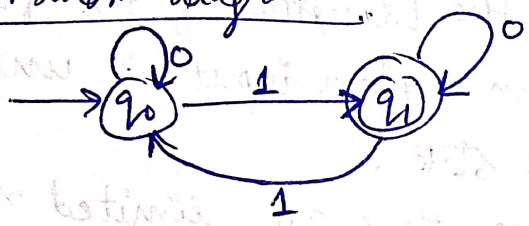
final state = q_1

Q \ Σ	0	1
→ q_0	q_0	q_1
✗ q_1	q_1	q_0

where Q = states

Σ = {0, 1}

② Transition diagram



Non final state (single circle)

final state (double circle)

• Acceptance of string by DFA

eg show that whether the given string "10110" is accepted by given DFA or not.

$\delta(q_0, 10110) \Rightarrow \delta(q_1, 0110)$

$\Rightarrow \delta(q_1, 110)$

$\Rightarrow \delta(q_0, 10)$

$\Rightarrow \delta(q_1, 0)$

$\Rightarrow \delta(q_1, \lambda)$

$\Rightarrow q_0$ (Non final state)

$\delta(q_0,1) \to q_1$

$\delta(q_1,0) \to q_1$

$\delta(q_1,1) \to q_0$

$\delta(q_0,1) \to q_1$

$\delta(q_1,0) \to q_1$

$\delta(q_1,\lambda) \to q_0$

So the string is not accepted.

i.e. the string does not belong to given DFA.

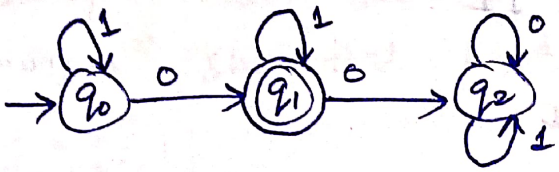
• DFA problems include

- string ending with a substring
- string starting with a substring
- string containing a substring
- Divisibility.

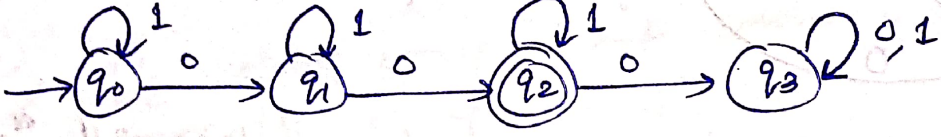
(8) Design transition diagram of DFA for the following languages defined over $\Sigma = \{0, 1\}$.

(a) That accepts all strings of 0's & 1's.

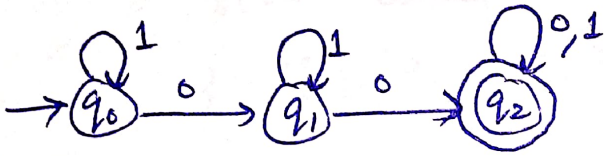
(b) That accepts all strings with exactly one 0's.



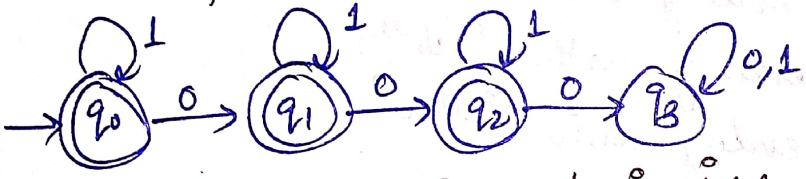
(c) That accepts all strings with exactly two 0's.



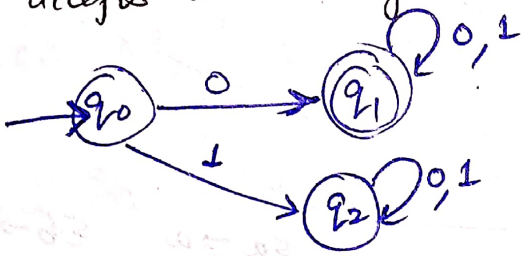
(d) That accepts all strings with atleast two 0s.



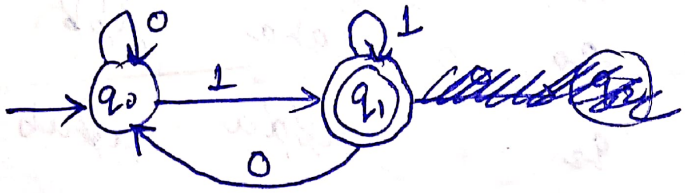
(e) That accepts all strings with atmost two 0s.



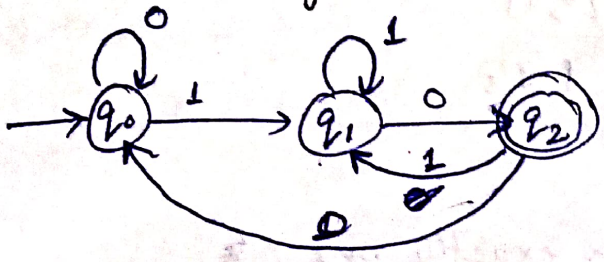
(f) That accepts all strings beginning with 0



(g) All strings ending with 1

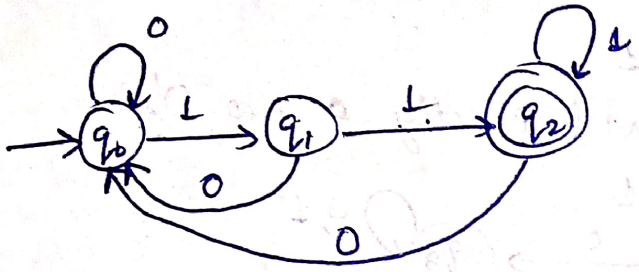


(h) All strings ending with 10



(i) All strings ending with 11.

1100101



Q2 Design a DFA that contain all strings ending with "aba"

(1) $\Sigma = \{a, b\}$

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

(3) $\rightarrow q_0$ - Initial state

q_1 - string ending with "a"

q_2 - string ending with "ab"

$*q_3$ - string ending with "aba"

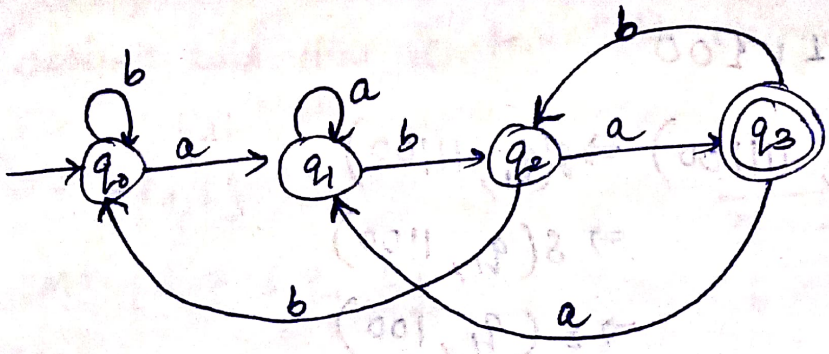
$\therefore Q = \{q_0, q_1, q_2, q_3\}$

(4) Transition table :-

		a	b
Σ	$\rightarrow q_0$	q_1	q_0
a	q_1	q_1	q_2
ab	q_2	q_3	q_0
aba	$*q_3$	q_1	q_2

$\Sigma a \rightarrow a$	$\Sigma b \rightarrow b$
$\not{a}a$	ab
aba	$\not{a}\not{b}\not{b}$
$\not{a}\not{b}\not{a}a$	$\not{a}\not{b}ab$

(5)



abaaba

Transition diagram.

Q3 Design a DFA that contains all strings ending with "100". Show that 11100 is ~~not~~ accepted string.

step 1 (1) $\Sigma = \{0, 1\}$

step 2 (2) $M = \{Q, \Sigma, \delta, q_0, F\}$

step 3 (3) $\rightarrow q_0$: ~~string ending with "1"~~ initial state.

q_1 : string ending with "1"

q_2 : string ending with "10"

$*q_3$: string ending with "100"

$Q = \{q_0, q_1, q_2, q_3\}$.

(4) Transition table :-

		0	1
Σ	$\rightarrow q_0$	q_0	q_1
1	q_1	q_2	q_1
10	q_2	q_3	q_1
100	$*q_3$	q_0	q_1

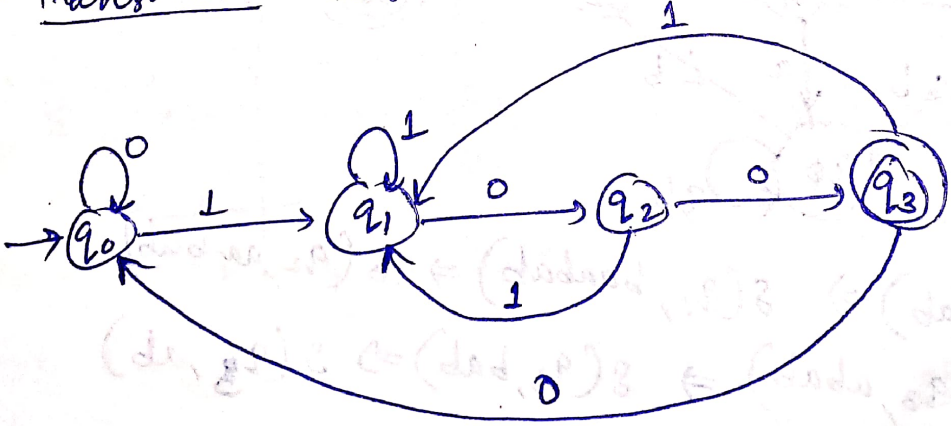
$\Sigma 0 \rightarrow 0$ $\quad \Sigma 1 \rightarrow 1$

10 $\quad$ ~~1~~

100 $\quad$ ~~10~~ 1

~~1000~~ $\quad$ ~~1001~~

step 4 (5) Transition diagram



string :- "111100"

$\delta(q_0, 111100) \Rightarrow \delta(q_1, 11100)$

$\Rightarrow \delta(q_1, 1100)$

$\Rightarrow \delta(q_1, 100)$

$\Rightarrow \delta(q_1, 00)$

$\Rightarrow \delta(q_2, 0)$

$\Rightarrow q_3$ (final state)

Hence "111100" is accepted and this string belongs to this DFA.

Q4 Design DFA containing all strings starting with "aba" by using rejection state q_ϕ

Check for "abaabab"

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

$M = \{Q, \Sigma, \delta, q_0, F\}$

$\rightarrow q_0 \Rightarrow$ initial state

$q_1 \Rightarrow$ string starts with "a"

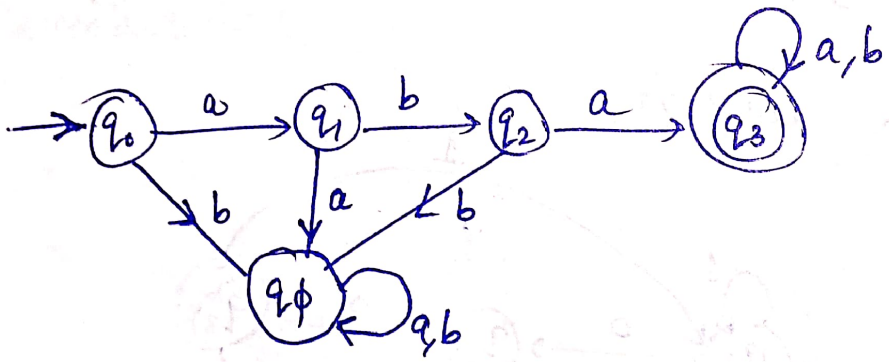
$q_2 \Rightarrow$ " " with "ab"

$* q_3 \Rightarrow$ " " " "aba"

$q_\phi \Rightarrow$ rejection state

Transition Table:

	a	b
$\Sigma \rightarrow q_0$	q_1	q_ϕ
$a \Leftarrow q_1$	q_ϕ	q_2
$ab \Leftarrow q_2$	q_3	q_ϕ
$aba * q_3$	q_3	q_3
q_ϕ	q_ϕ	q_ϕ



$\delta(q_0, abaabab) \Rightarrow \delta(q_1, baabab) \Rightarrow \delta(q_2, aabab)$

$\Rightarrow \delta(q_3, abab) \Rightarrow \delta(q_3, bab) \Rightarrow \delta(q_3, ab)$

$\rightarrow \delta(q_3, b) \Rightarrow q_3$.

(accepted) ✓

Q5 Design DFA that contains all strings having a substring as "bab"

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

$M = \{Q, \Sigma, \delta, q_0, F\}$

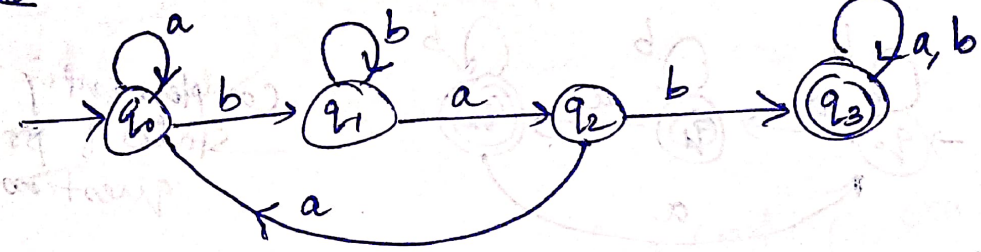
$\rightarrow q_0$: initial state

q_1 : string contains "b"

q_2 : string contains "ba"

$*q_3$: string contains "bab".

	a	b
$\Sigma \rightarrow q_0$	q_0	q_1
b q_1	q_2	q_1
ba q_2	q_0	q_3
bab $*q_3$	q_3	q_3



Q6 Construct DFA to check whether a given binary number is even ~~or odd~~.

When a binary number ends with 0, it is even

When it ends with 1, it is odd.

① $\Sigma = \{0, 1\}$

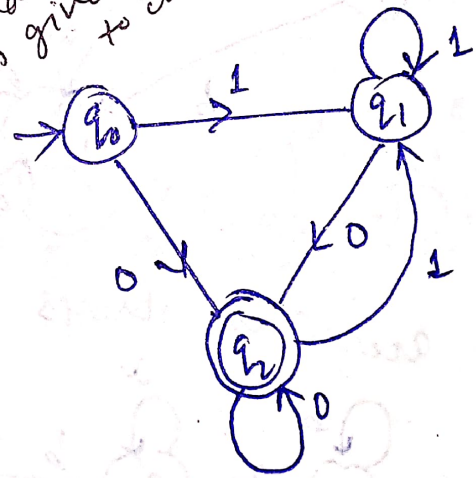
② $M = \{Q, \Sigma, \delta, q_0, F\}$

③ $\rightarrow q_0$ ⇒ initial state

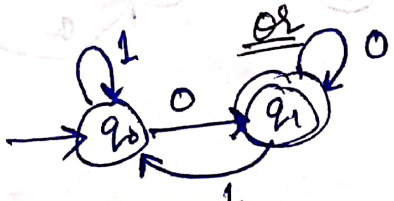
q_1 : ends with 1

$*q_2$ - ends with 0 → final → even (as given in questions to check).

	0	1
$\Sigma \rightarrow q_0$	q_2	q_1
1 q_1	q_2	q_1
0 $*q_2$	q_2	q_1

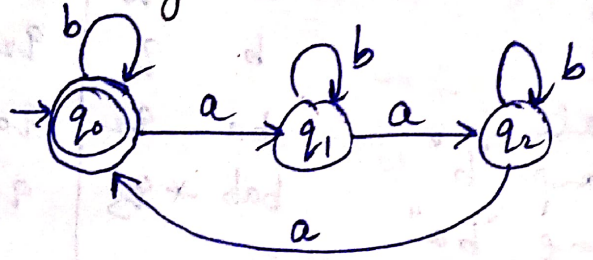


OR



Q7 Design the DFA for the following languages over $\Sigma = \{a, b\}$

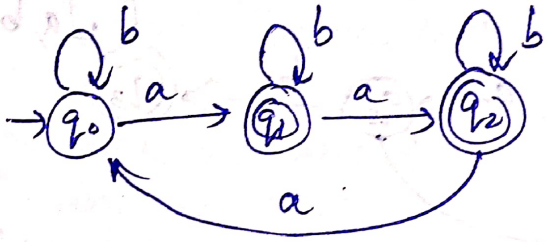
(a) accepts all strings in which no. of a's are multiple of 3.



~~0 is also a multiple of 3~~

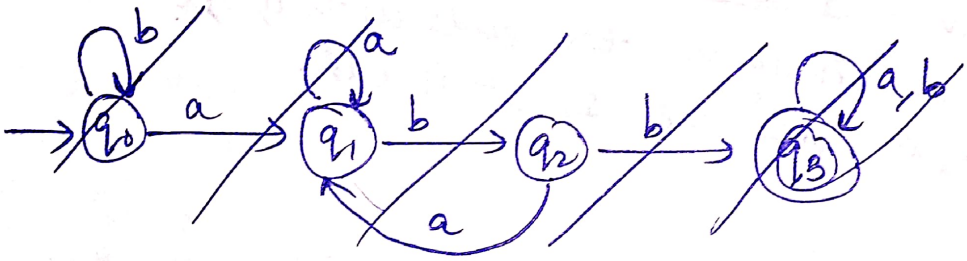
we can also consider 0 as a multiple of 3

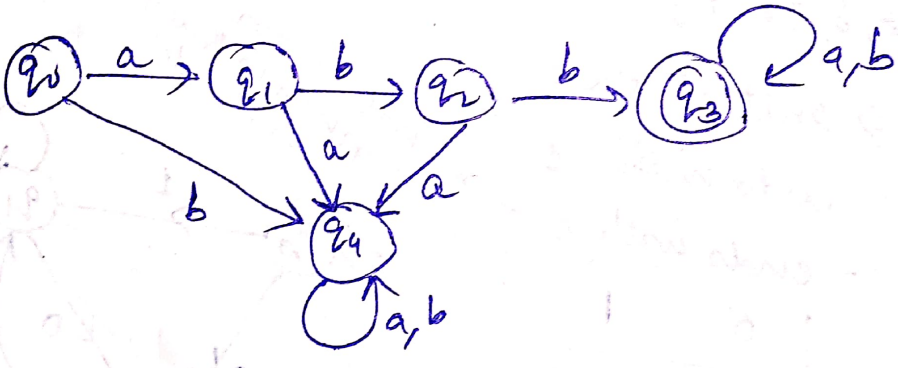
(b) accepts all strings in which no. of a's are not multiple of 3.



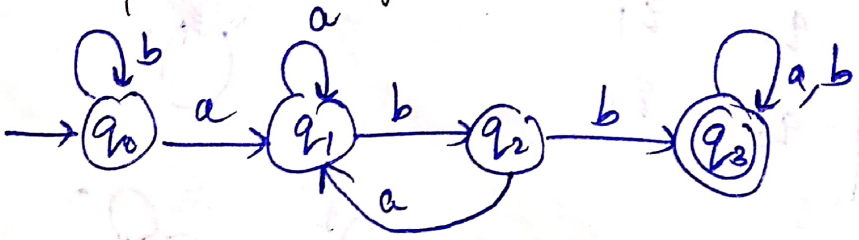
complementing the states of previous question.

(c) accepts all strings starting with "abb".

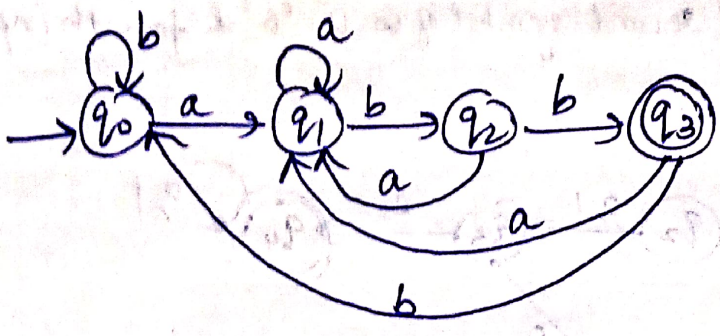




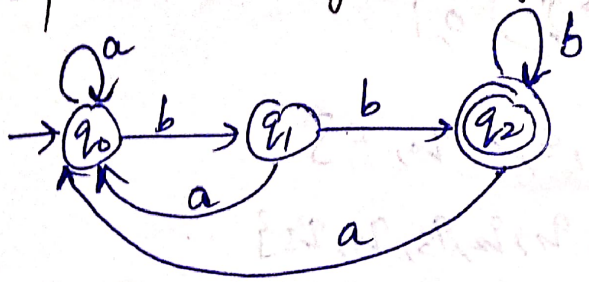
(d) accepts all strings with abb as substring



(e) accepts strings ending with "abb"



(f) accepts all strings ending with "bb"



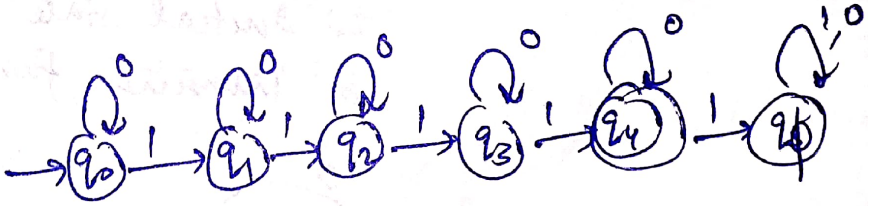
Q8. Design FA which accepts strings containing exactly four 1s.

① $\Sigma = \{0, 1\}$

② $M = \{Q, \delta, q_0, \Sigma, F\}$

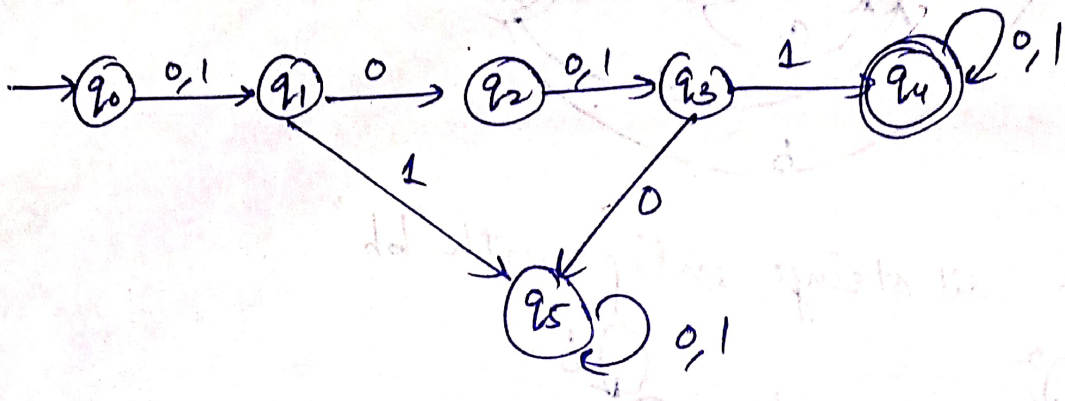
③ $\rightarrow q_0$: Initial state
q_1 : string contain exactly one 1
q_2 : " " " two 1s.
q_3 : " " " three 1s.
q_4 : " " " four 1s.

	0	1
$\rightarrow q_0$	q_0	q_1
..010.. q_1	q_1	q_2
..0110.. q_2	q_2	q_3
..0111.. q_3	q_3	q_4
..01110.. * q_4	q_4	reject (q_ϕ)
Rejection (q_ϕ) (dead state)	q_ϕ	q_ϕ



(Q9) Design a FA which accepts the Language

$L = \{ w \in (0,1)^* \mid$ second symbol of w is '0' & fourth input is 1$\}$



Let FA is $M = \{ Q, \Sigma, \delta, q_0, F \}$

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

$F = \{ q_4 \}$

Dead state $= q_5$

Eg 0001, 1011, 1011000 ....

Q10 Design FA for $L = \{ (01)^i 1^{2j} \mid i \geq 1, j \geq 1 \}$

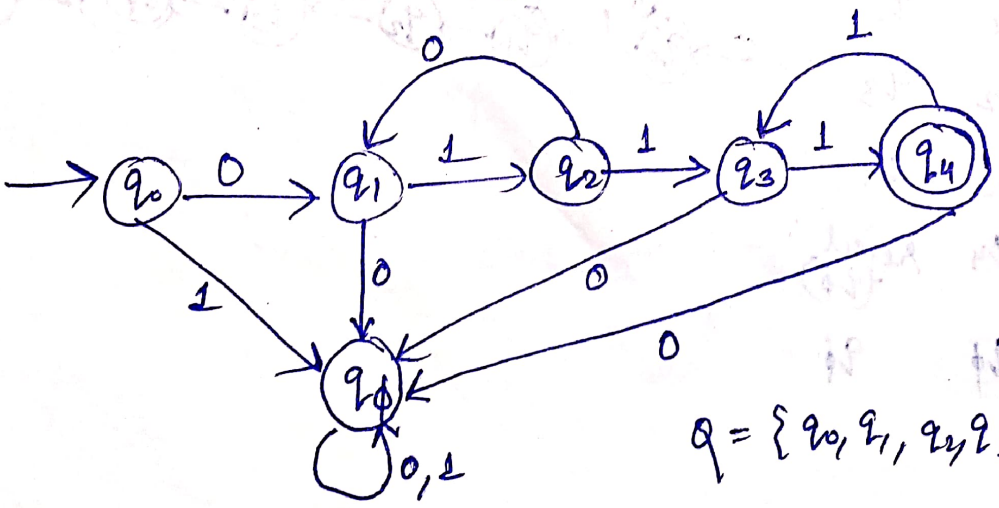
FA should accept all strings start with any number of "01" and end with even number of 1s

Sol Let FA be $M = \{ Q, \Sigma, \delta, q_0, F \}$

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

q_0: Initial state

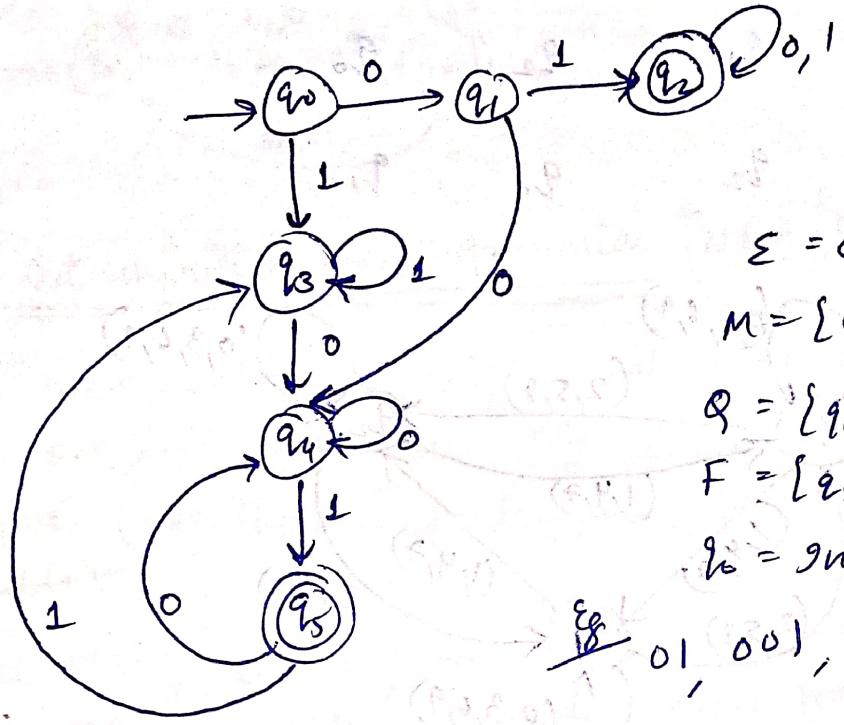
δ: Transition function



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

Q11. $\Sigma = \{0,1\}$ ~~starts with either~~

Either starts with 01 or end with 01.



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

$M = \{Q, \Sigma, \delta, q_0, f\}$

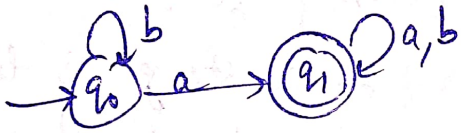
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$F = \{q_2, q_3\}$.

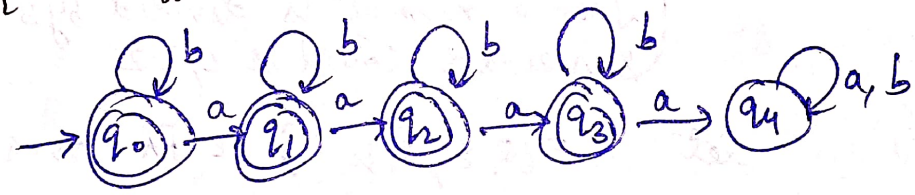
q_0 = initial state

Eg 01, 001, 011111, 0000101, ...

Q12 i) Design DFA $L = \{w : n_a(w) \geq 1, w \in (a,b)^*\}$ (atleast 1 a)



ii) $L = \{w : n_a(w) \leq 3, w \in (a,b)^*\}$ (atmost 3 'a')



Q13 Construct a DFA that contains all language which is divisible by 3.

In such questions, we take logic remainder from 0 to (n-1). There is no final state in such problem.

$q_0 \rightarrow$ 0 remainder

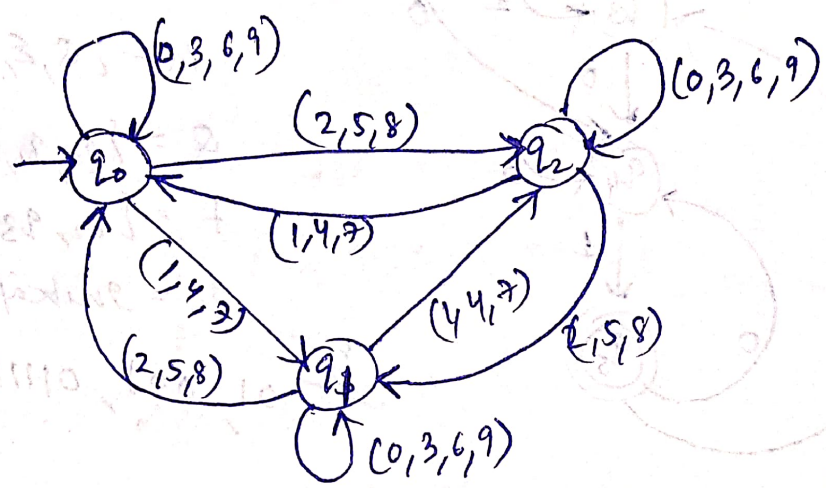
$q_1 \rightarrow$ 1 remainder

$q_2 \rightarrow$ 2 remainder

$\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$

	(0,3,6,9)	(1,4,7)	(2,5,8)
$0 - q_0$	q_0	q_1	q_2
$1 - q_1$	q_1	q_2	q_0
$2 - q_2$	q_2	q_0	q_1

DFA



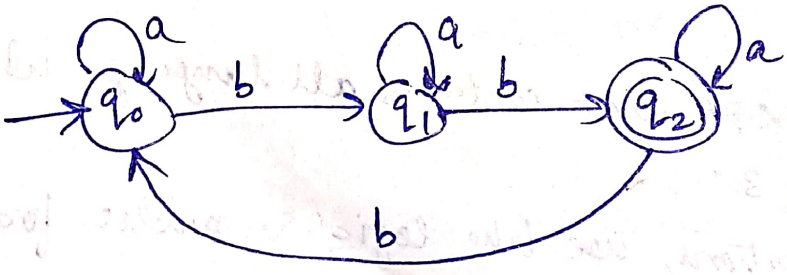
Q14. Design DFA for language

$$L = \{w \in (a,b)^* \mid n_b(w) \bmod 3 > 1\}$$

$n_b(w) \bmod 3$: number of b's in string is divided by 3 and remainder is the result.

$n_b(w) \bmod 3 > 1$: means that remainder is only 2, (when a number is divided by 3 then remainder could be 0, 1, 2)

let $M = \{Q, \Sigma, \delta, q_0, F\}$ be FA over $\Sigma = [a,b]$



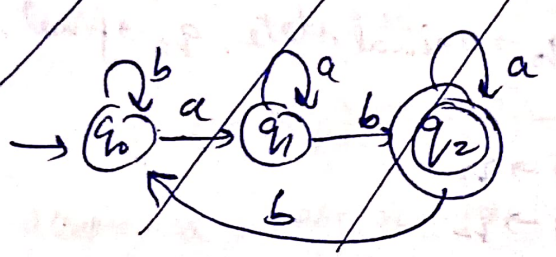
It will accept string having 2b's, 5b's, 8b's .....

so that remainder is always 2.

$$Q = [q_0, q_1, q_2]$$

$$F = [q_2]$$

Q $L = (ab \cup aba)^*$ accepts ab, aba / combination of both.



Non-Deterministic Finite Automata (NFA / NDFA)

- 'Non deterministic' specifies that transition of NDFA from one state to another state on a given input is not unique (i.e. there may be more than 1 next state possible)
- Null string (λ) is allowed.
- It is not necessary to define NDFA for each input in each state.
- NDFA also has five tuples.

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

where $Q \rightarrow$ finite, non empty set of states.
$\Sigma \rightarrow$ finite, non empty set of input alphabets.
$q_0 \rightarrow$ initial state
$F \rightarrow$ set of final states.
$\delta \rightarrow$ transition function

$$\boxed{Q \times (\Sigma \cup \lambda) \rightarrow 2^Q}$$

Present state, input alphabet, set of max. possible next states

eg $\delta(q_0, a) \rightarrow q_1$
$\delta(q_0, a) \rightarrow q_0$
$\delta(q_1, b) \rightarrow q_2$.

- NDFA is easy to design.
- NDFA can be converted to DFA

Representation of NFA

① using transition function

$Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $q_0 \rightarrow$ initial state, $q_2 \rightarrow$ final state

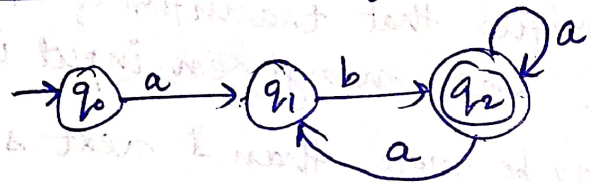
transition function

δ : $\delta(q_0, a) \rightarrow q_1$

$\delta(q_1, b) \rightarrow q_2$

$\delta(q_2, a) \rightarrow q_1, q_2$

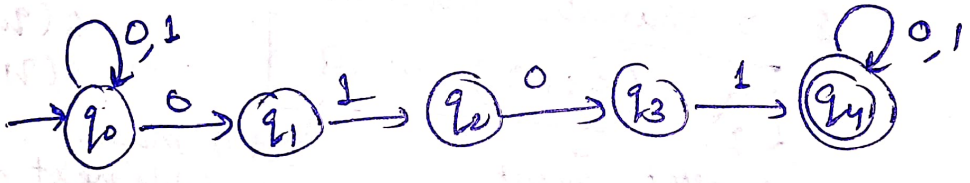
② using transition diagram



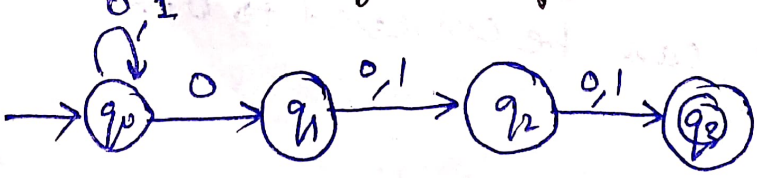
③ using Transition table

Q \ Σ	a	b
q_0	q_1	
q_1		q_2
q_2	q_1, q_2	

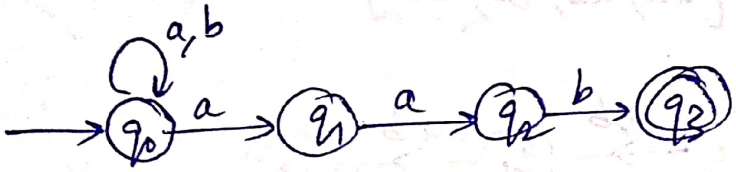
Q1.) i) Design NDFA to accept strings containing substring 0101 for $\Sigma = \{0, 1\}$



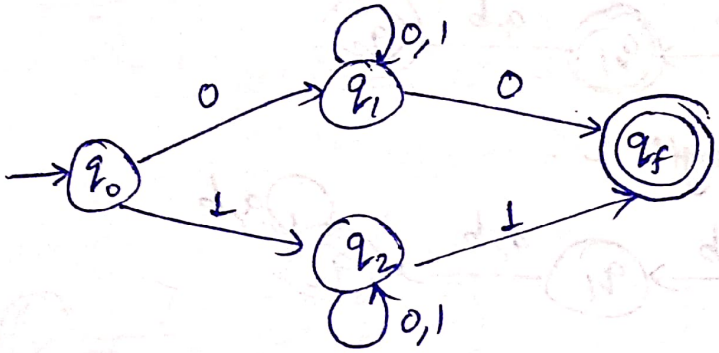
ii) Design NFA to accept strings over alphabet $\{0, 1\}$ such that the 3rd symbol from the right end is 0.



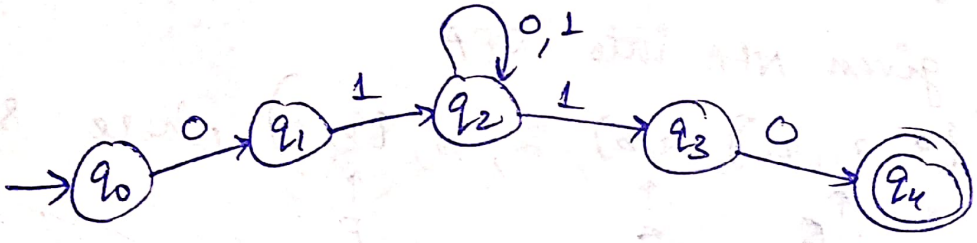
III) accepts the following Language
$L = \{x \in \{a,b\}^* \mid x \text{ ends with } aab\}$



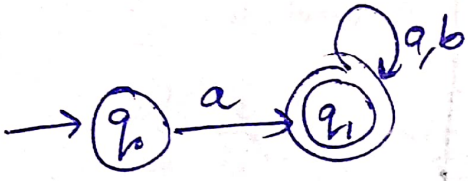
IV) accepts a binary no. having first & last bit same.
if start with 1, then should end with 1
" " " 0, " " " " 0



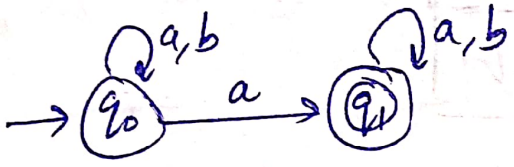
V) NFA that accepts all binary numbers that start with 01 & end with 10



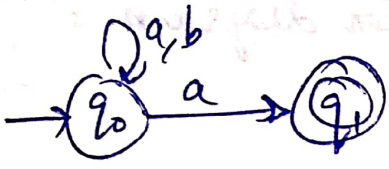
VI) starts with "a", $\Sigma = \{a,b\}$



VI) contains "a" as substring $\Sigma = \{a,b\}$.

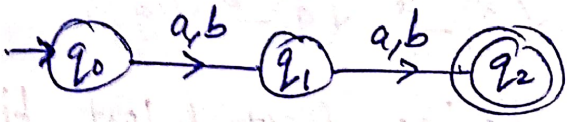


VII) ends with "a", $\Sigma = \{a,b\}$.



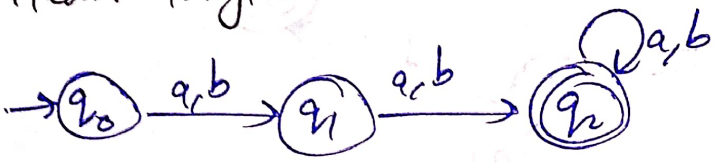
viii) accepts string of length "2" exactly. $\Sigma = \{a, b\}$

$L = \{aa, ab, ba, bb\}$



ix) ~~Atleast~~ Atmost length 2

x) Atleast length 2.



Conversion of NFA to DFA:

Q Convert the given NFA into DFA

$M = (\{q_0, q_1, q_2\}, (a,b), \delta, q_0, \{q_2\})$ where δ is

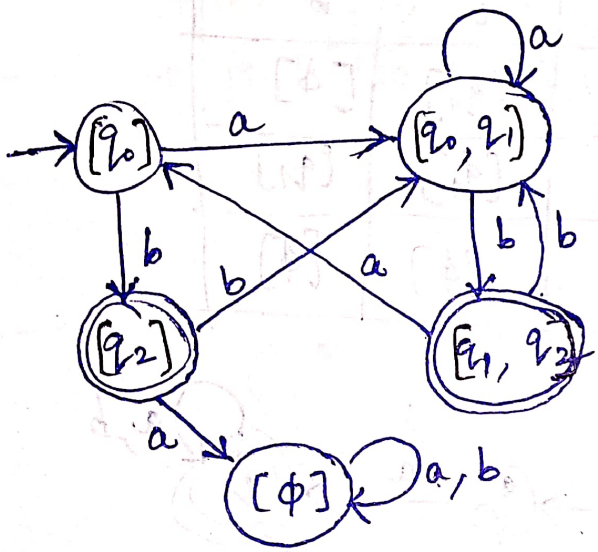
(Q, Σ, F)

given by ⊢

δ / Σ	a	b
→ q_0	q_0, q_1	q_2
q_1	q_0	q_1
* q_2		q_0, q_1

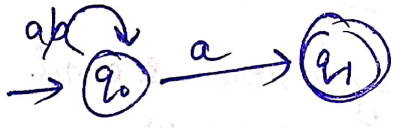
- first construct transition table of DFA and then transition diagram.

Q \ Σ	a	b
→ [q_0]	[q_0, q_1]	[q_2]
[q_0, q_1]	[q_0, q_1]	[q_1, q_2]
* [q_2]	[ϕ]	[q_0, q_1]
* [q_1, q_2]	[q_0]	[q_0, q_1]
[ϕ]	[ϕ]	[ϕ]



Q2 Construct NFA for accepting a string that ends with "a" over $\Sigma = \{a, b\}$ & then construct equivalent DFA

$L = \{a, ba, baa, bba, abba, \ldots\}$

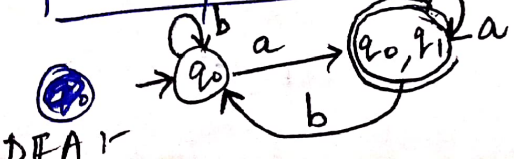


Tr. Tab. for NFA

Q \ Σ	a	b
→ q_0	{q_1, q_0}	{q_0}
* q_1	{ϕ}	{ϕ}

Tr. Table for DFA

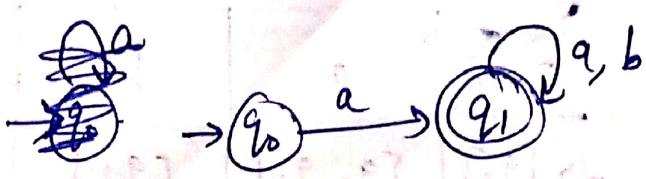
Q \ Σ	a	b
→ [q_0]	[q_0, q_1]	[q_0]
* [q_0, q_1]	[q_0, q_1]	[q_0]



DFA

Q3 starts with "a".

NFA Transitⁿ diagᵐ :-



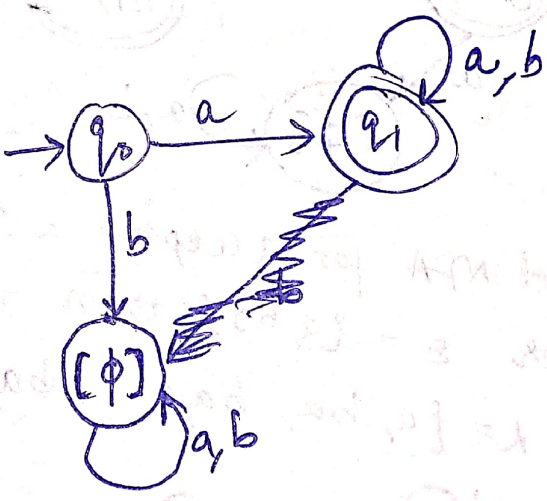
Transitⁿ Table :-

Q / Σ	a	b
→ q_0	q_1	
* q_1	q_1	q_1

DFA Transition Table

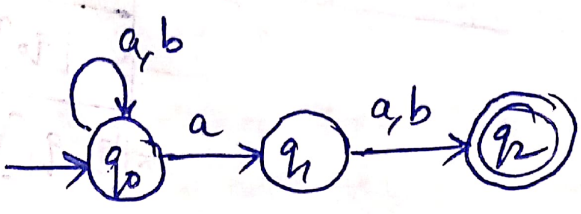
	a	b
→ [q_0]	[q_1]	[ϕ]
* [q_1]	[q_1]	[q_1]
[ϕ]	[ϕ]	[ϕ]

DFA Transition diagram



Q3 Construct a NFA for string where second symbol from RHS is "a" over $\Sigma = \{a, b\}$. Then construct equivalent DFA.

NFA



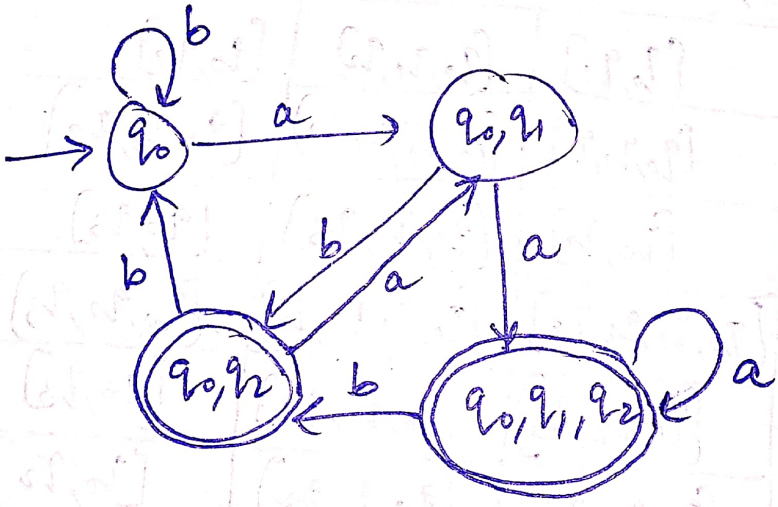
NFA δ:

	Σ / Q	a	b
→	q_0	q_0, q_1	q_0
	q_1	q_2	q_2
*	q_2	∅	

DFA δ

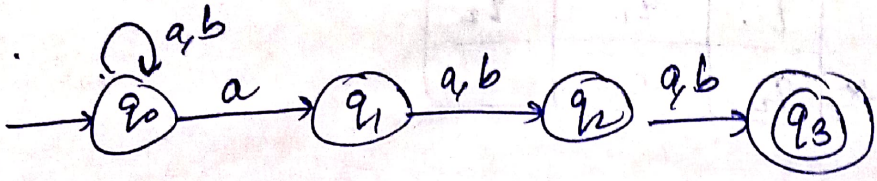
	Σ / Q	a	b
→	$[q_0]$	$[q_0, q_1]$	$[q_0]$
	$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_2]$
*	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_0, q_2]$
*	$[q_0, q_2]$	$[q_0, q_1]$	$[q_0]$

DFA



Q Construct NFA for a string where third symbol from RHS is "a" over $\Sigma = \{a, b\}$

NFA



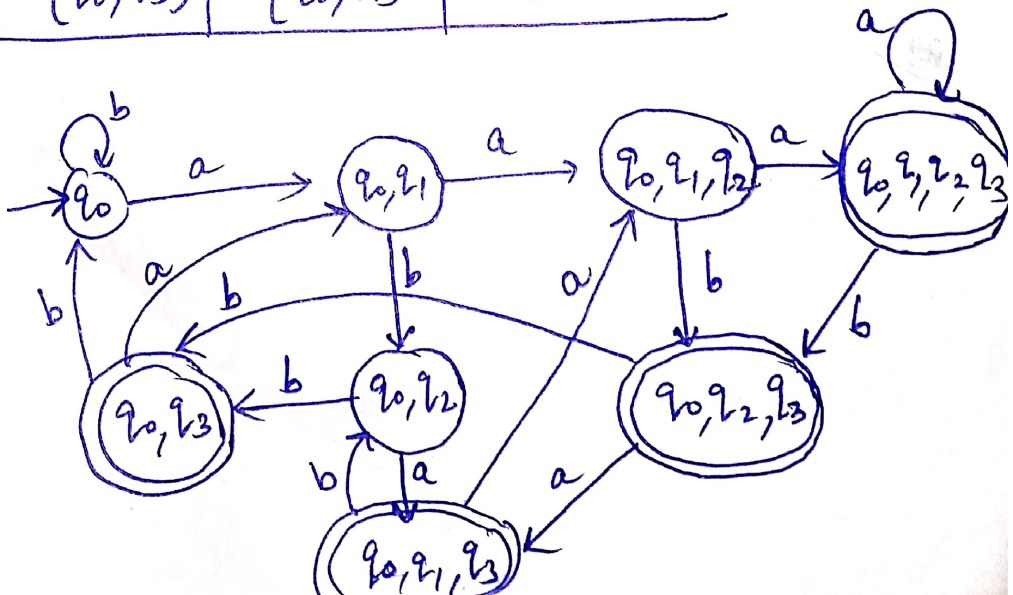
NFA Table

	Σ / Q	a	b
→	q_0	q_0, q_1	q_0
	q_1	q_2	q_2
	q_2	q_3	q_3
*	q_3	ϕ	

DFA Table

	Σ / Q	a	b
→	$[q_0]$	$[q_0, q_1]$	$[q_0]$
	$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_2]$
	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_2, q_3]$
	$[q_0, q_2]$	$[q_0, q_1, q_3]$	$[q_0, q_3]$
*	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_2, q_3]$
*	$[q_0, q_2, q_3]$	$[q_0, q_1, q_3]$	$[q_0, q_3]$
*	$[q_0, q_1, q_3]$	$[q_0, q_1, q_2]$	$[q_0, q_2]$
	$[q_0, q_3]$	$[q_0, q_1]$	$[q_0]$

DFA



Q8 Convert the following NFA into DFA

$M = \{ \{p, q, r, s\}, \{0, 1\}, \delta, p, \{q, s\} \}$.

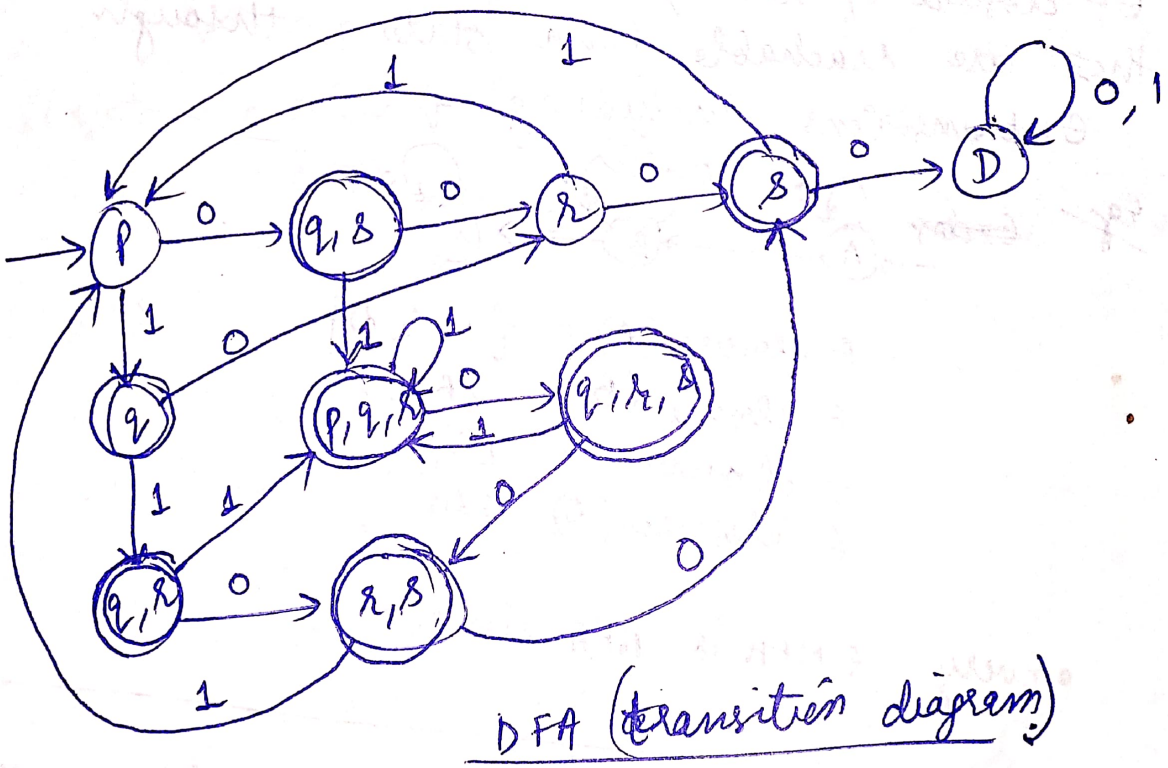
$Q \quad \Sigma \quad \delta \quad q_0 \quad F$

Q \ Σ	0	1
$\rightarrow p$	q, s	q
$* q$	r	q, r
r	s	p
$* s$	ϕ	p

Sol. Transition Table for DFA

Q \ Σ	0	1
$\rightarrow [p]$	$[q, s]$	$[q]$
$* [q, s]$	$[r]$	$[p, q, r]$
$* [q]$	$[r]$	$[q, r]$
$[r]$	$[s]$	$[p]$
$* [p, q, r]$	$[q, r, s]$	$[p, q, r]$
$* [q, r]$	$[r, s]$	$[p, q, r]$
$* [s]$	$[\phi]$	$[p]$
$* [q, r, s]$	$[r, s]$	$[p, q, r]$
$* [r, s]$	$[s]$	$[p]$
$[\phi]$	$[\phi]$	$[\phi]$

let $[\phi]$ be D denoting dead state.

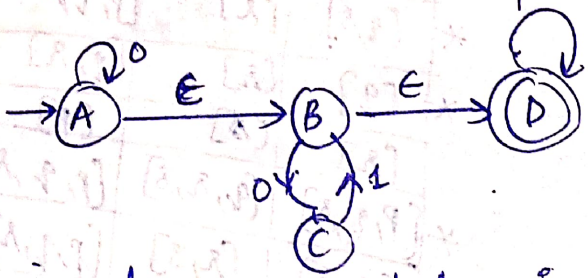


DFA (transition diagram)

ϵ-NFA (NFA with ϵ-transition(s))

- ϵ-NFA is a representation that allows an automaton to change its state without input
- ϵ- means "empty string"
- It does not only translate its state after getting input from alphabet set but also without any input symbol. This transition without input is called "null move"

eg



- It is also represented using five tuples:-

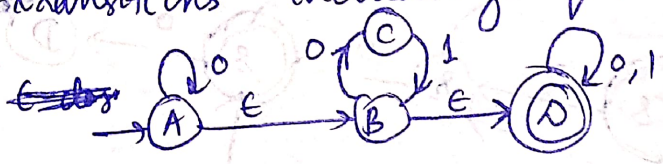
$$M = \{Q, \Sigma, \delta, q_0, F\}$$

where $\delta: Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$

ϵ-closure(q)

ϵ-closure of state q is defined as the set of states that are reachable from state q through ϵ-transitions including q.

eg



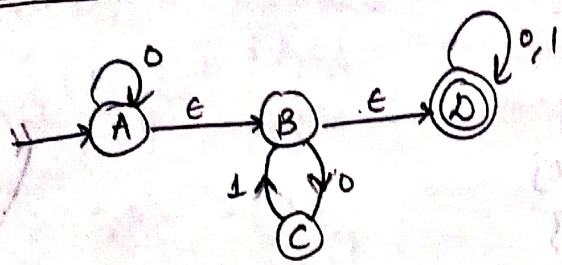
ϵ closure (A) = {A, B, C, D}
ϵ closure (B) = {B, C, D}
ϵ closure (C) = {C}
ϵ closure (D) = {D}

- Every ϵNFA is NFA also.

Note :- The questions based on ϵ-NFA to NFA conversions solved from the next page uses a formula representation which is a little bit wrong. But overall answer is correct, so Refer videos for this topic.

ε-NFA to NFA Conversion

Q



convert into NFA

Sol

ε closure (A) = {A, B, D}
ε closure (B) = {B, D}
ε closure (C) = {C}
ε closure (D) = {D}

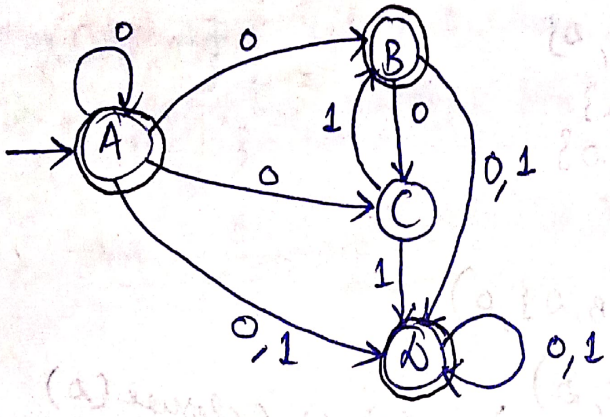
- $\delta(A,0)$ = ε-closure ({A, B, D}, 0)
  = ε-closure (A, C, D)
  = ε-closure(A) ∪ ε-closure(C) ∪ ε-closure(D)
  = {A, B, C, D}
- ~~$\delta(B,0)$ = ε-closure ({B, D},~~
- $\delta(A,1)$ = ε-closure ({A, B, D}, 1)
  = ε-closure (D)
  = {D}
- $\delta(B,0)$ = ε-closure ({B, D}, 0)
  = ε-closure (C, D)
  = {C, D}
- $\delta(B,1)$ = ε-closure ({B, D}, 1)
  = {D}.
- $\delta(C,0)$ = ε-closure ({C}, 0) = ϕ
- $\delta(C,1)$ = ε-closure ({C}, 1) ~~= {D}~~
  ~~{B, D} = {D}~~. = {B, D}
- $\delta(D,0)$ = ε-closure ({D}, 0) = {D}.
- $\delta(D,1)$ = ε-closure ({D}, 1) = {D}.

$\delta(A,0)$
= ε-closure [δ {ε-cl(A), 0}]
= ε-cl [δ {(A, B, D), 0}]
= ε-cl [A, C, D]

{A, B, C, D}, ε, D
A δ D

Note: $\delta'(A,0)$ = ε-closure [δ {ε-closure (A), 0}]
correct way = ε-closure [δ {(A, B, D), 0}]
= ε-closure [δ(A,0), δ(B,0), δ(D,0)] = ε-closure (A, C, D) = {A, B, C, D}

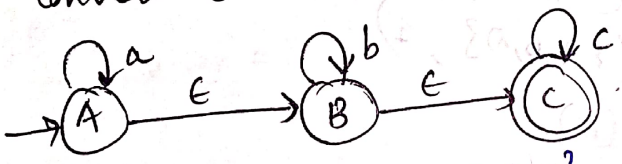
	0	1
→ * A	{A,B,C,D}	{D}
* B	{C,D}	{D}
C	{ϕ}	{B,D}
* D	{D}	{D}



To find the final state, we check from which state we can reach the final state by ϵ-transitions. Those states would be the final state

Here, final states are A, B, D

Q Convert ϵ NFA to NFA



- ϵ-closure (A) = {A, B, C}
- ϵ-closure (B) = {B, C}
- ϵ-closure (C) = {C}

$\delta(A,a)$ = ϵ-closure ({A, B, C}, a)
= ϵ-closure (A)
= {A, B, C}

$\delta(A,b)$ = ϵ-closure ({A, B, C}, b)
= ϵ-closure (B)
= {B, C}

$\delta(A,c)$ = ϵ-closure ({A, B, C}, c)
= ϵ-closure (C)
= {C}

$\delta(A,a)$
= ϵ-closure (δ(ϵ-closure(A), a))
= ϵ-closure (δ({A, B, C}, a))
= ϵ-closure (A)
= A, B, C

Similarly:

$\delta(B,a) = \{\phi\}$

$\delta(B,b) = \{B,C\}$

$\delta(B,c) = \{C\}$

$\delta(C,a) = \{\phi\}$

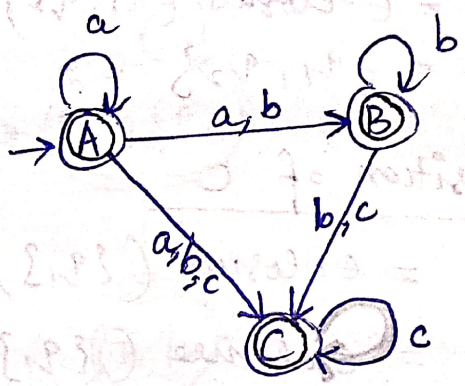
$\delta(C,b) = \{\phi\}$

$\delta(C,c) = \{C\}$

Transition Table

	a	b	c
→ * A	{A,B,C}	{B,C}	{C}
* B	{φ}	{B,C}	{C}
* C	{φ}	{φ}	{C}

Diagram



Note :- ϵ-NFA, NFA & DFA are equivalent to each other.

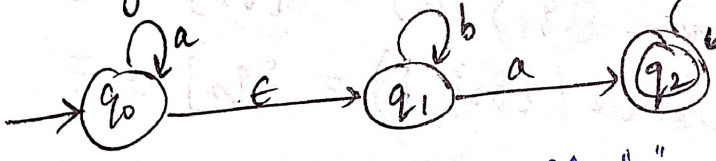
~~Conversion: ϵ-NFA → NFA → DFA~~

~~To convert ϵ-NFA to DFA, there are two ways~~

~~1. Convert ϵ-NFA to NFA, then NFA to DFA~~

~~2. Direct conversion ϵ-NFA to DFA~~

Q) Directly convert ϵ-NFA to DFA



Calculate initial state of DFA "A" using initial state of NFA "q_0"

So, A = initial state = ϵ-closure(q_0) = $\{q_0, q_1\}$

- for transition of A

$\delta(A,a)$ = ϵ-closure(A, a) = ϵ-closure($\{q_0,q_1\}$, a)

= ϵclosure(q_0, q_2) = $\{q_0, q_1, q_2\}$ = B

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

$= \{q_1\} = C$

- for transition of B

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

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

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

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

$= \{q_1,q_2\} = D$

- for transition of C

$\delta(C,a) = \epsilon\text{closure}(\{q_1\},a) = \epsilon\text{closure}(q_2) = \{q_2\} = E$

$\delta(C,b) = \epsilon\text{closure}(\{q_1\},b) = \epsilon\text{closure}(q_1) = \{q_1\} = C$

- for transition of D

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

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

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

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

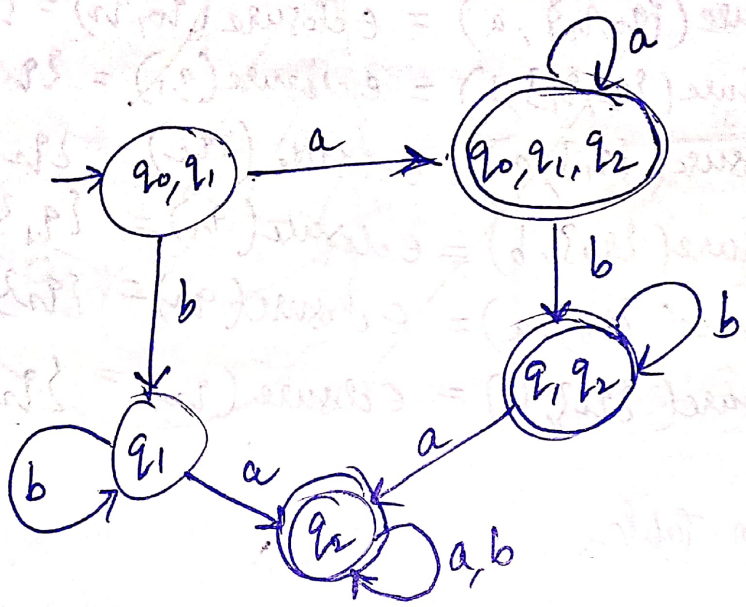
- for transition of E

$\delta(E,a) = \epsilon\text{-closure}(\{q_2\},a) = \{q_2\} = E$

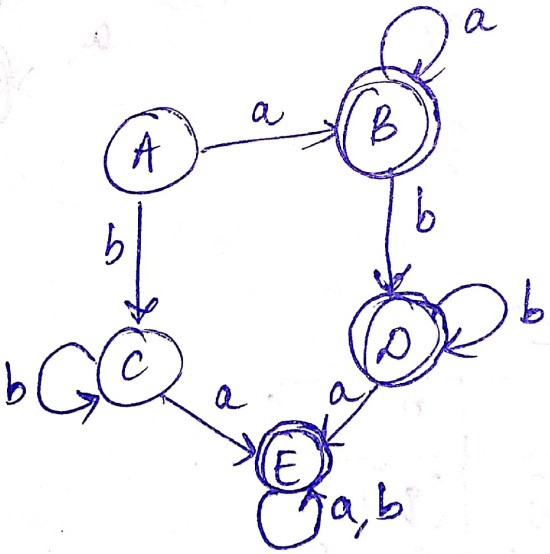
$\delta(E,b) = \epsilon\text{-closure}(\{q_2\},b) = \{q_2\} = E$

Q \ Σ	a	b
→ A	B	C
* B	B	D
C	E	C
* D	E	D
* E	E	E

Q \ Σ	a	b
→ $\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1\}$
* $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
$\{q_1\}$	$\{q_2\}$	$\{q_1\}$
* $\{q_1, q_2\}$	$\{q_2\}$	$\{q_1, q_2\}$
* $\{q_2\}$	$\{q_2\}$	$\{q_2\}$



or



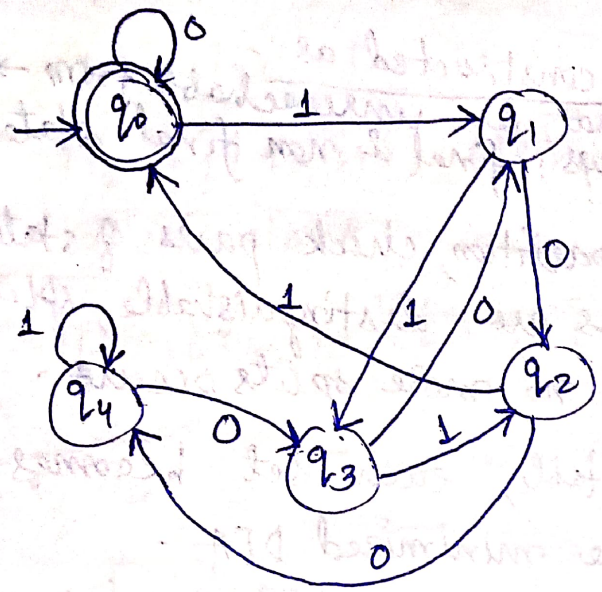
Ex Q

Design a DFA that accepts all the binary numbers that are divisible by 5.

Sol $L = \{ w \mid w \bmod 5 = 0 \}$

$L = \{ 000, 101, 1010, 1111, \ldots \}$.

	rem	state
0 → 000	0	q_0
1 → 001	1	q_1
2 → 010	2	q_2
3 → 011	3	q_3
4 → 100	4	q_4
5 → 101	0	q_0
6 → 110	1	q_1
7 → 111	2	q_2
8 → 1000	3	q_3
9 → 1001	4	q_4
10 → 1010	0	q_0



remainder = 0, 1, 2, 3, 4, (5 states)

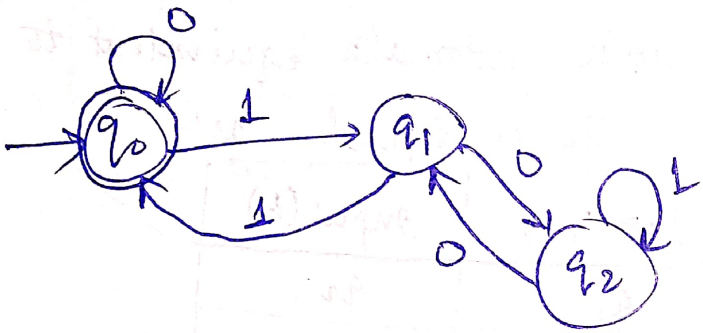
	0	1
→ * q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

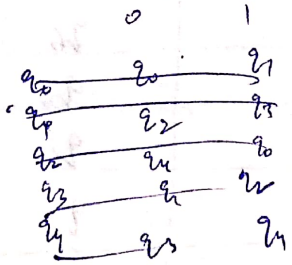
shortcut.

Q Design a DFA that accepts all the binary numbers that are divisible by 3

$L = \{ w \mid w \bmod 3 = 0 \}$

	rem	state
0 → 000	0	q_0
1 → 001	1	q_1
2 → 010	2	q_2
3 → 011	0	q_0
4 → 100	1	q_1
5 → 101	2	q_2
6 → 110	0	q_0
7 → 111	1	q_1
8 → 1000	2	q_2
9 → 1001	3	q_0
10 → 1010	0	q_1





shortcut

	0	1
→ * q_0	q_0	q_1
q_1	q_2	q_0
* q_2	q_1	q_2

Minimization of DFA

- DFA Minimization stands for converting a given DFA to its equivalent DFA with minimum number of states. It is also called DFA optimization
- It uses partioning Algorithm.
- The minimized DFA can be constructed as :-
  1. Remove all the states that are unreachable from q_0
  2. split states into two groups : final & non final states
  3. start with an initial partition, check pairs of states in each set. If pairs are distinguishable, split the set. Repeat until no more splits occurs.
  4. Once partitioning is stable, each set becomes a single state in the minimized DFA.
- Note : Two states (q_i, q_j) are distinguishable in partition P_k if for any input symbol a, $\delta(q_i, a)$ and $\delta(q_j, a)$ are in different sets in partition P_{k-1}

Q. Construct a minimum state automata equivalent to a DFA whose transition table is defined as :-

Q(state)	Input (a)	Input (b)
→ q_0	q_1	q_2
q_1	q_4	q_3
q_2	q_4	q_3
* q_3	q_5	q_6
* q_4	q_7	q_6
q_5	q_3	q_6
q_6	q_6	q_6
q_7	q_4	q_6

0-equivalent

$\pi_0 = \{\{q_3, q_4\}, \{q_0, q_1, q_2, q_5, q_6, q_7\}\}$

1-equivalent

$\pi_1 = \{\{q_3, q_4\}, \{q_0, q_6\}, \{q_1, q_2\}, \{q_5, q_7\}\}$

2-equivalent

$\pi_2 = \{\{q_3, q_4\}, \{q_0\}, \{q_6\}, \{q_1, q_2\}, \{q_5, q_7\}\}$

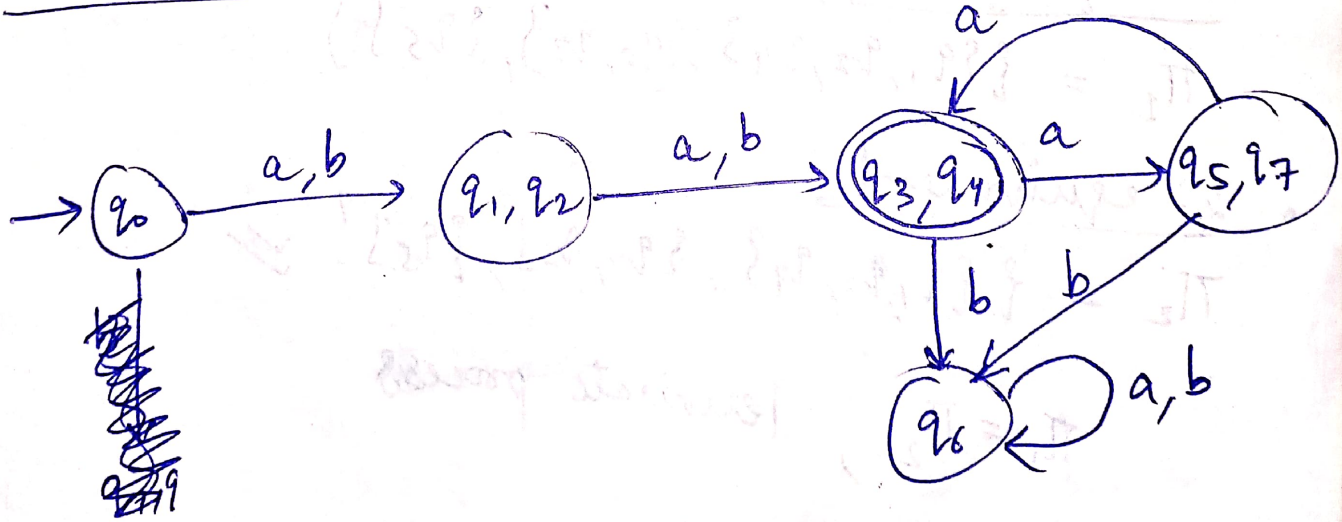
3-equivalent

$\pi_3 = \{\{q_3, q_4\}, \{q_0\}, \{q_6\}, \{q_1, q_2\}, \{q_5, q_7\}\}$

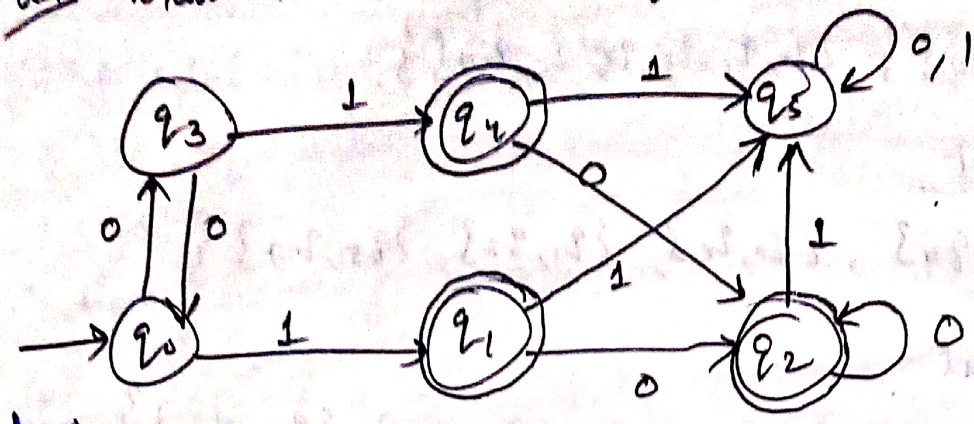
- $\pi_2 = \pi_3$ (Terminate process ✓)

state (Q)	a (Input)	b (Input)
→ * $[q_0]$	$[q_1, q_2]$	$[q_1, q_2]$
$[q_6]$	$[q_6]$	$[q_6]$
* $[q_3, q_4]$	$[q_5, q_7]$	$[q_6]$
$[q_1, q_2]$	$[q_3, q_4]$	$[q_3, q_4]$
$[q_5, q_7]$	$[q_3, q_4]$	$[q_6]$

Minimised DFA :-



Q2 Draw minimised DFA for the following DFA :-



Sol)

- Transition Table

Q/Σ	0	1
→ q_0	q_3	q_1
* q_1	q_2	q_5
* q_2	q_2	q_5
q_3	q_0	q_4
* q_4	q_2	q_5
q_5	q_5	q_5

~~There is one unreachable state from initial state~~

- 0 Equivalent

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

- 1 - equivalent

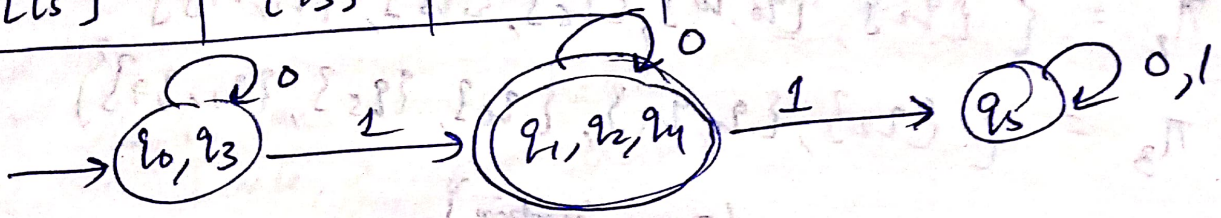
$\pi_1 = \{\{q_1, q_2, q_4\}, \{q_0, q_3\}, \{q_5\}\}$

- 2 - equivalent

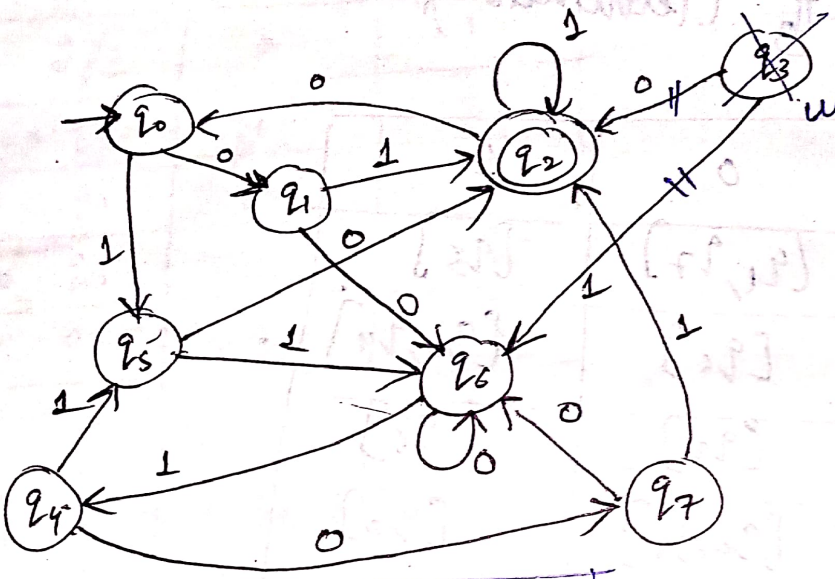
$\pi_2 = \{\{q_1, q_2, q_4\}, \{q_0, q_3\}, \{q_5\}\}$ ✓

$\pi_1 = \pi_2$, Terminate process

Q | Σ	0	1
→ [q0, q3]	[q0, q3]	[q1, q2, q4]
* [q1, q2, q4]	[q1, q2, q4]	[q5]
* [q5]	[q5]	[q5]



Q3



Sol

TT

Q | Σ	0	1
→ q0	q1	q5
q1	q1	q2
* q2	q0	q2
~~q3~~	~~q2~~	q6
q4	q7	q5
q5	q2	q6
q6	q6	q4
q7	q6	q2

remove unreachable state i.e q3

$\pi_0 = \{\{q_2\}, \{q_0, q_1, q_4, q_5, q_6, q_7\}\}$

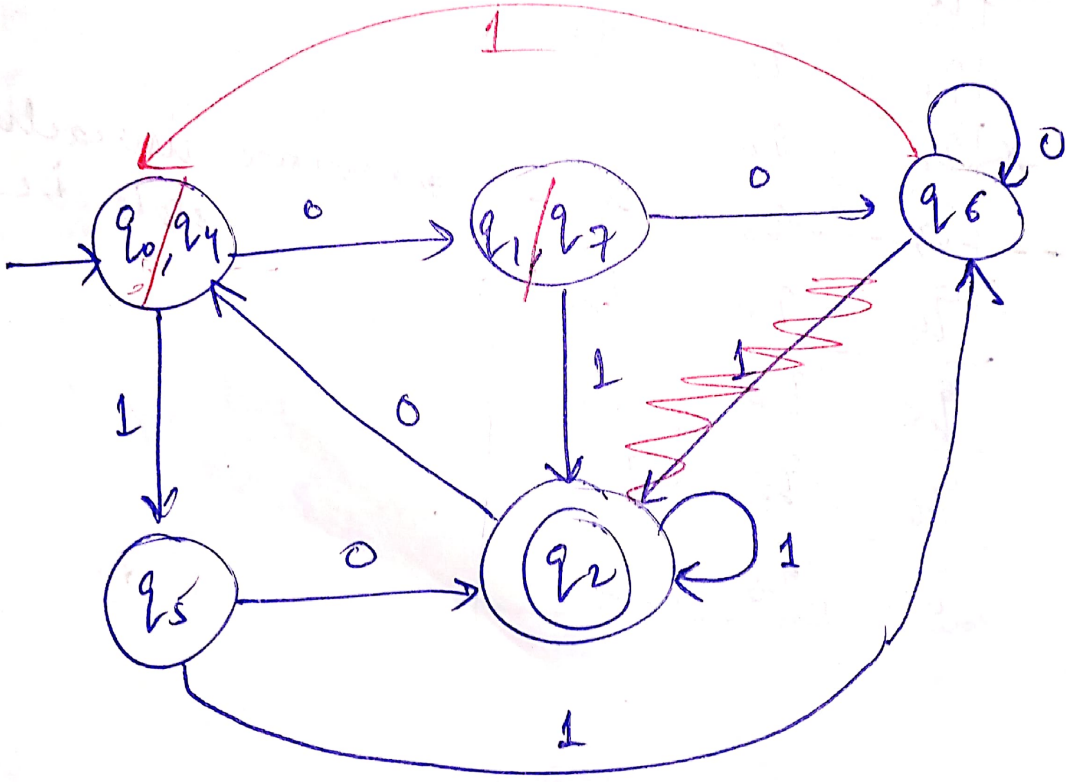
$\pi_1 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_5\}\}$

$\pi_2 = \{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_5\}, \{q_1, q_7\}\}$

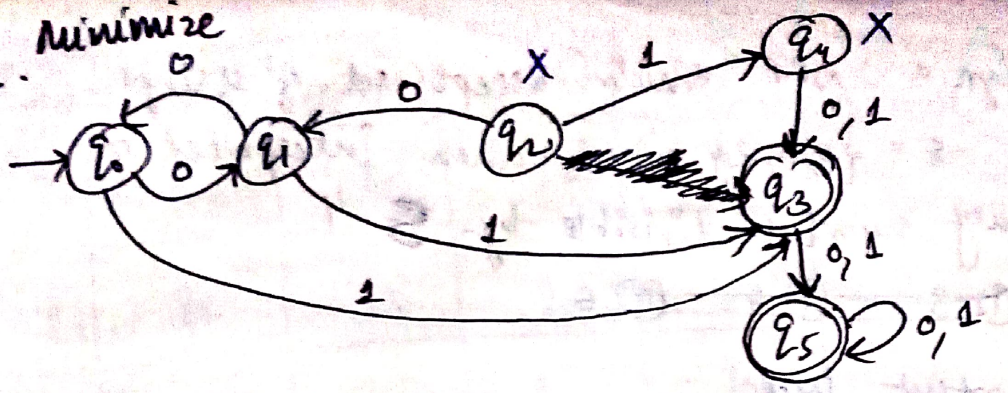
$\pi_3 = \{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_5\}, \{q_1, q_7\}\}$

$\pi_2 = \pi_3$ (Termination)

Q/Σ	0	1
→ $[q_0, q_4]$ ✓	$[q_1, q_7]$	$[q_5]$
$[q_6]$ ✓	$[q_6]$	$[q_0, q_4]$
$[q_5]$ ✓	$[q_2]$	$[q_6]$
* $[q_2]$ ✓	$[q_0, q_4]$	$[q_2]$
$[q_1, q_7]$ ✓	$[q_6]$	$[q_2]$



Q4. Minimize



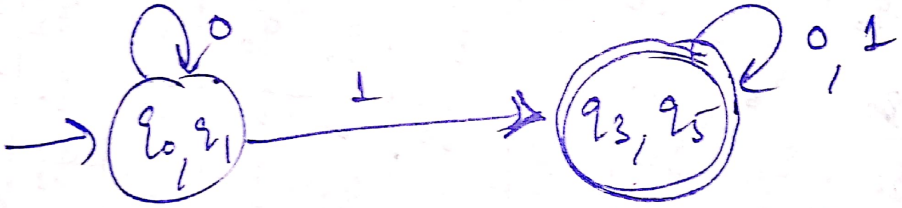
q_2, q_4 are unreachable hence, remove them.

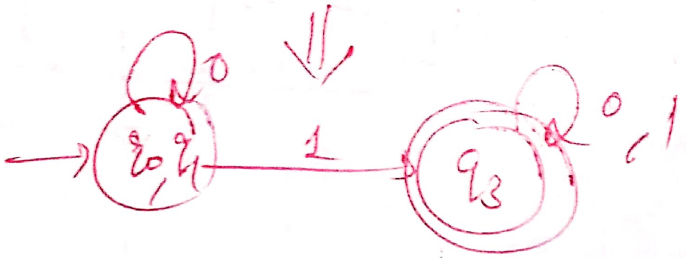
Q/Σ	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3
* q_3	q_5	q_5
* q_5	q_5	q_5

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

$\pi_1 = \{\{q_0, q_1\}, \{q_3, q_5\}\}$ $\pi_0 = \pi_1$ (Terminate)

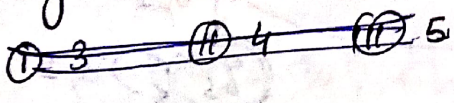
Σ/Q	0	1
→ $[q_0, q_1]$	$[q_0, q_1]$	$[q_3, q_5]$
* $[q_3, q_5]$	$[q_3, q_5]$	$[q_3, q_5]$





$\therefore q_3 = q_5$

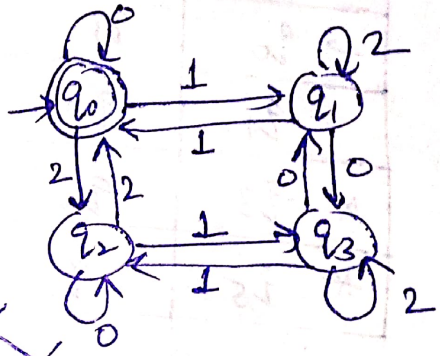
Q Design a DFA which accept set of strings over $\Sigma = \{0, 1, 2\}$ which when interpreted as ternary number divisible by 4

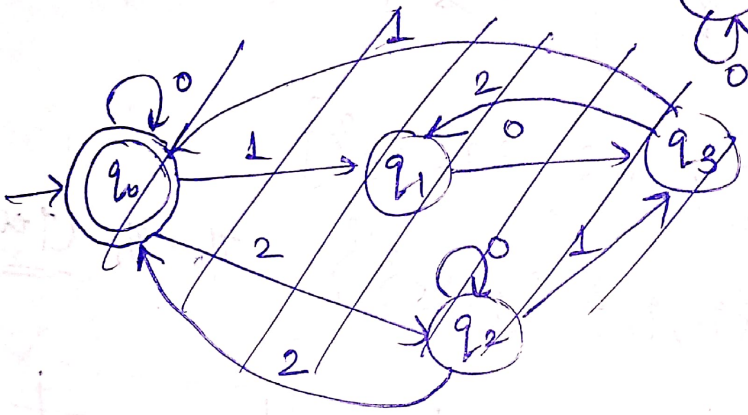


Sol By shortcut trick

remainders :- 0, 1, 2, 3 (four states possible)

Q/Σ	0	1	2
→* q_0	q_0	q_1	q_2
q_1	q_3	q_0	q_1
q_2	q_2	q_3	q_0
q_3	q_1	q_2	q_3





Q. DFA for decimal numbers divisible by 4

4 states :- q_0, q_1, q_2, q_3

Σ/Q	0	1	2	3	4	5	6	7	8	9
q_0	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1
q_1	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3
q_2	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1
q_3	q_2	q_3	q_0	q_1	q_2	q_3	q_0	q_1	q_2	q_3

Σ \ Q	(0,4,8)	(1,5,9)	(2,6)	(3,7)
q_0	q_0	q_1	q_2	q_3
q_1	q_2	q_3	q_0	q_1
q_2	q_0	q_1	q_2	q_3
q_3	q_2	q_3	q_0	q_1

Finite automata with output

- In this, they will take input as well as keep on writing output. It consist of six tuples -

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

$Q \rightarrow$ finite set of states.
$\Sigma \rightarrow$ input alphabet
$\delta \rightarrow Q \times \Sigma \rightarrow Q$ (Transition function)
$q_0 \rightarrow$ initial state
$\Delta \rightarrow$ Output Alphabet (symbols which are supposed to be an output)
$\lambda \rightarrow$ output function → determines what will be the o/p.

- It is similar to FA except for the ability to produce o/p.

Mealy Machine

Mealy Machines are finite state machine (FSM) with output value and its output depends on present state and current input symbol.

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

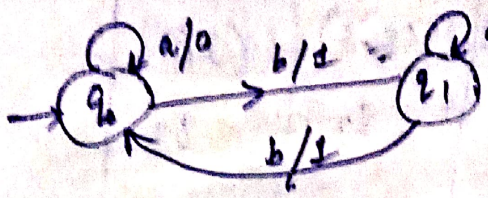
By Tuples :- $(Q, \Sigma, \Delta, q_0, \delta, \lambda)$

Eg

Random Transition Table

Present State (PS)	Next state (NS)			
	Input a = 0		Input b = 1	
	State	output	State	output
→ q_0	q_2	0	q_2	0
q_1	q_1	1	q_1	0
q_2	q_0	1	q_1	1

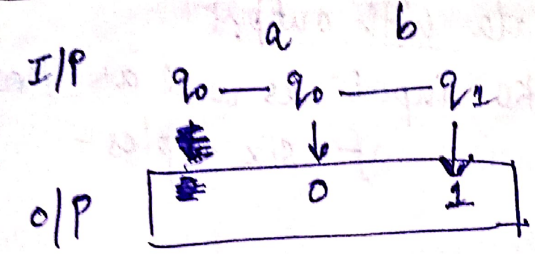
Eg
Random diagram



- output associated with the input.

where $a, b - \Sigma$
$0, 1$ - output, Δ

$\lambda(q_0, a) \to 0$ $\quad \delta(q_0, a) \to q_0$
$\lambda(q_0, b) \to 1$ $\quad \delta(q_0, b) \to q_1$
$\lambda(q_1, a) \to 0$ $\quad \delta(q_1, a) \to q_1$
$\lambda(q_1, b) \to 1$ $\quad \delta(q_1, b) \to q_0$



- on getting input of length n, the output is of length n, too

Moore Machine

- Moore machine are finite state machine (FSM) with output value, and its output depends only on present state
- output is associated with the state here.
- It contains six tuples:-

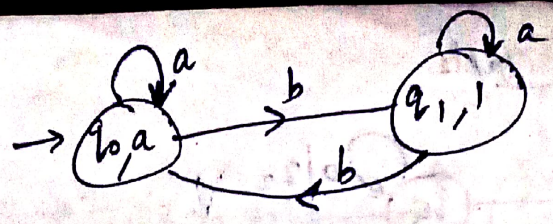
$(Q, \Sigma, \Delta, q_0, \delta, \lambda)$

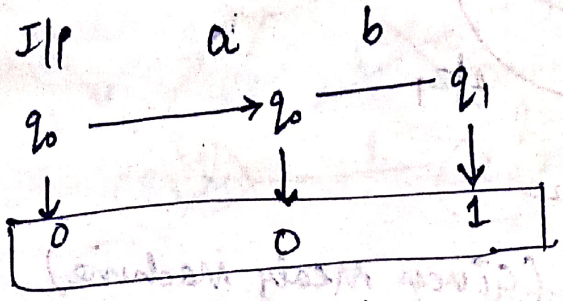
$\delta : Q \times \Sigma \to Q$
$\lambda : Q \to \Delta$

Eg
Random Transition Table

Present state (PS)	Next State		output
	i/p a=0	i/p b=1	
$\to q_1$	q_3	q_2	1
q_2	q_1	q_4	0
q_3	q_2	q_1	1
q_4	q_4	q_3	1

Eg Random diagram.





* on getting i/p of length n, the output is of length (n+1)

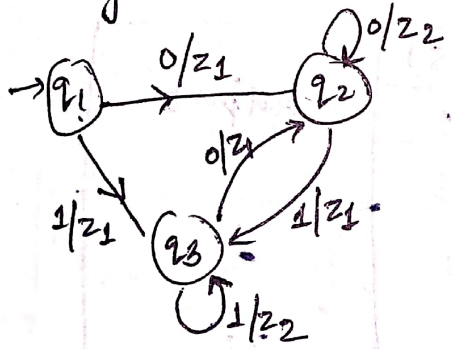
Equivalence of Moore & Mealy Machine.

Moore $\equiv$ Mealy (convertible / equivalent)

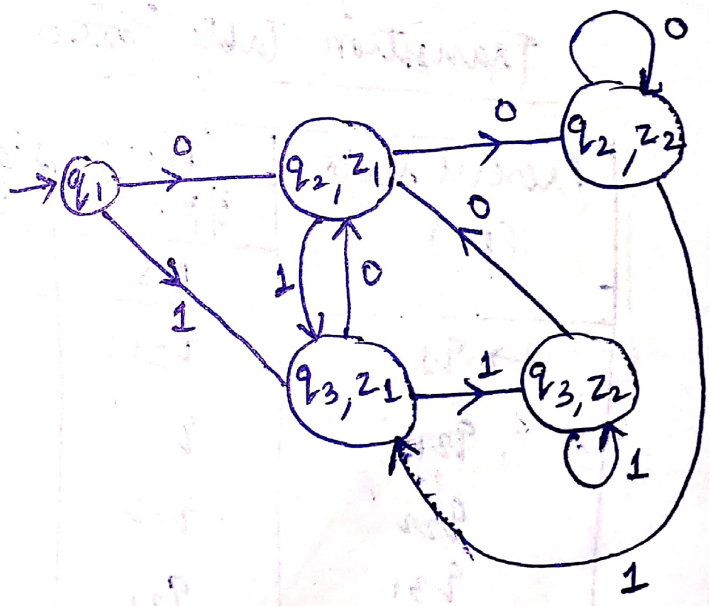
- **Conversion of Mealy Machine to Moore Machine**

Steps:
- → starts with initial state (always)
- → check i/p $\{0,1\}$ one by one.
- → what is next state & move the o/p from transition to that state.
- → continue this process for next state one by one.

Q Mealy → Moore

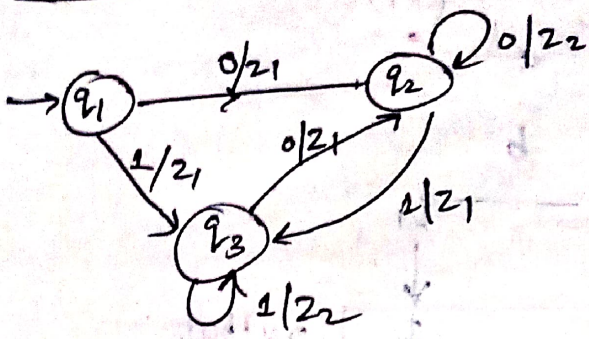


(directly from diagram)



from transition Table

Q



Transition Table : (Given Mealy Machine)

Present state (PS)	Next state (NS)			
	input = 0		input = 1	
	NS	output	NS	output
→ q_1	q_2	z_1	q_3	z_1
q_2	q_2	z_2	q_3	z_1
q_3	q_2	z_1	q_3	z_2

$q_2 \Rightarrow z_1, z_2$

$q_3 \Rightarrow z_1, z_2$

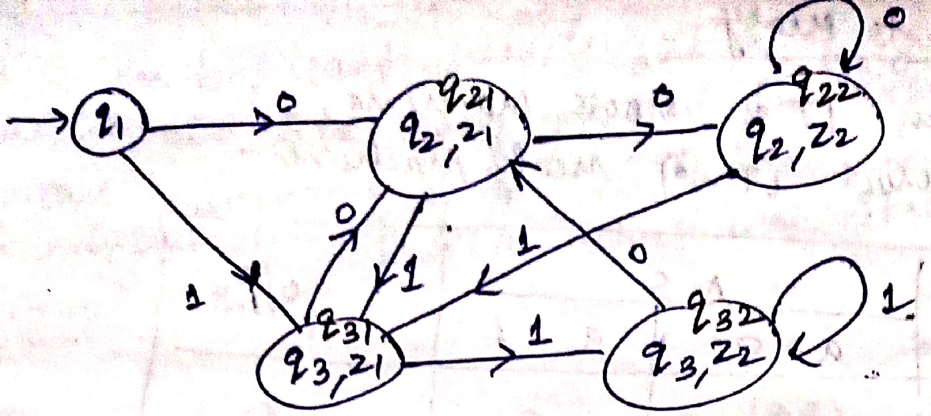
$q_1 \Rightarrow$ No output

$q_2 \rightarrow q_{21}, q_{22}$

$q_3 \rightarrow q_{31}, q_{32}$

Transition Table (Moore Machine)

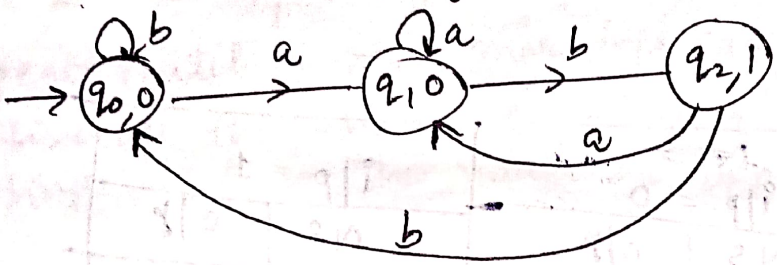
Present state (PS)	Next state (NS)		output
	i/p = 0	i/p = 1	
	NS	NS	
→ q_1	q_{21}	q_{31}	
q_{21}	q_{22}	q_{31}	z_1
q_{22}	q_{22}	q_{31}	z_2
q_{31}	q_{21}	q_{32}	z_1
q_{32}	q_{21}	q_{32}	z_2

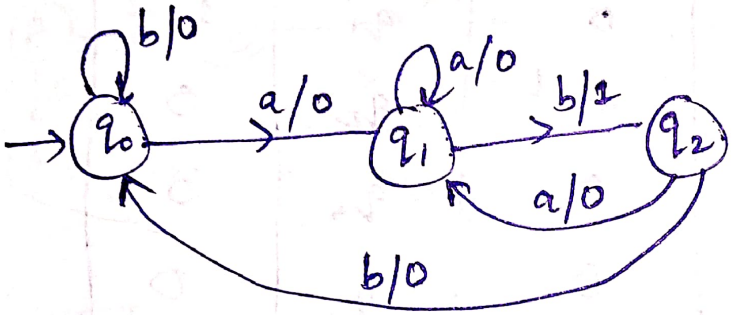


- **Conversion of Moore machine to Mealy machine**

Q ~~directly diagram to diagram~~

Moore → Mealy





Moore TT

PS	a	b	Δ
q_0	$[q_1, 0]$	$[q_0, 0]$	0
q_1	$[q_1, 0]$	$[q_2, 1]$	0
q_2	$[q_1, 0]$	$[q_0, 0]$	1

⟹

Mealy TT

PS	a		b	
	NS	O/P	N/S	O/P
q_0	q_1	0	q_0	0
q_1	q_1	0	q_2	1
q_2	q_1	0	q_0	0

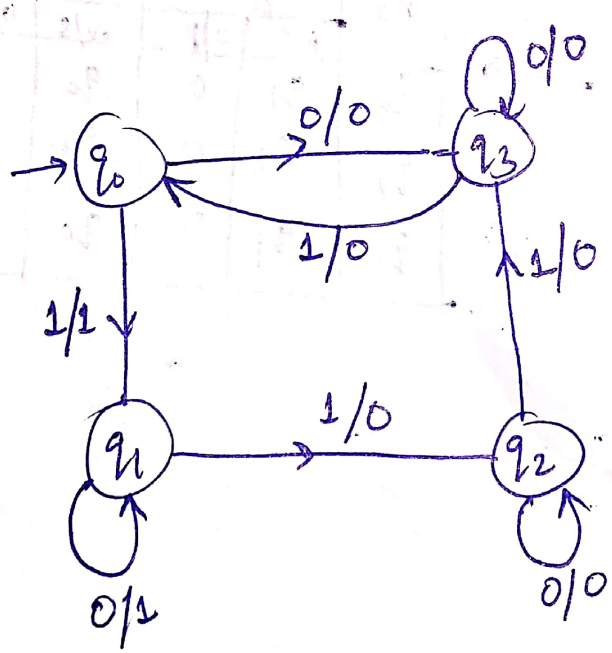
Q Moore to Mealy

Given TT of moore machine, Construct TT of mealy machine -

PS	NS		o/p
	a=0	a=1	
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

Mealy :-

PS	NS i/p = 0		NS i/p = 1	
	NS	o/p	NS	o/p
→ q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0



Myhill Nerode Theorem (Table Filling Method)

→ It is an important characterization of regular language and it also has many practical implications

→ Another consequence of its this theorem is an algorithm for minimization of DFA which is a vital step in automata theory

Algo :

① Draw a table for all pairs of states (P, Q) → no duplicate pair, no (P, P) pairing

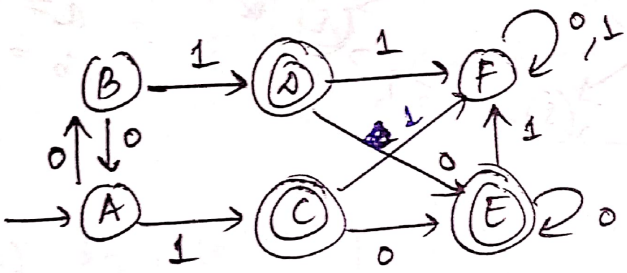
② Mark all pairs where $P \in F$ & $Q \notin F$

③ If there are any unmarked pairs (P, Q) such that $[\delta(P, x), \delta(Q, x)]$ is marked, then mark (P, Q) where x is an input symbol

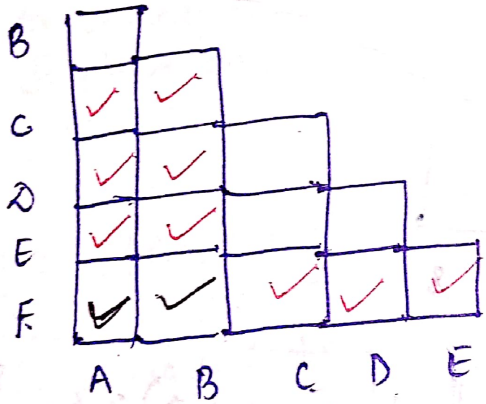
④ Repeat until no more markings can be made

⑤ Combine all the unmarked pairs and make them a single state in the minimized DFA.

Q1



Sol



(B, A): $\delta(B,0) = A$, $\delta(B,1) = D$; $\delta(A,0) = B$, $\delta(A,1) = E$ } No mark.

(F, A): $\delta(F,0) = F$, $\delta(F,1) = F$; $\delta(A,0) = B$, $\delta(A,1) = C$ } (F, A) marked.

(D, C): $\delta(D,0) = E$, $\delta(D,1) = F$; $\delta(C,0) = E$, $\delta(C,1) = F$ } No mark.

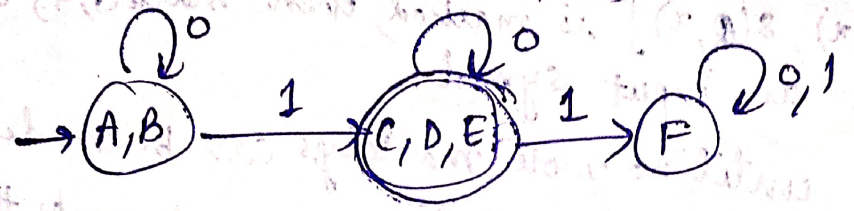
(F,B): $\delta(F,0) = F$, $\delta(A,0) = A$ } $\delta(F,1) = F$, $\delta(B,1) = D$ } (F,B) mark

(E,C): $\delta(E,0) = E$, $\delta(C,0) = E$ } $\delta(E,1) = F$, $\delta(C,1) = F$ } No mark

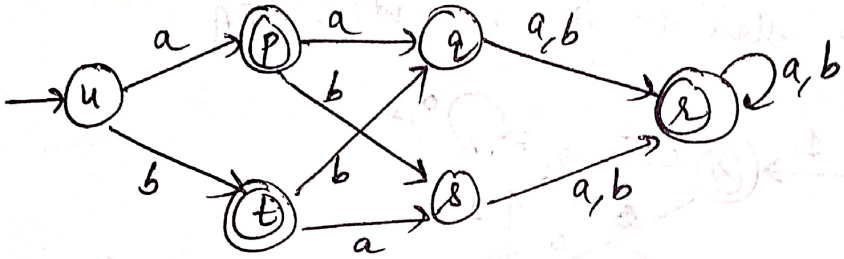
(E,D): $\delta(E,0) = E$, $\delta(D,0) = E$ } $\delta(E,1) = F$, $\delta(D,1) = F$ } No mark

unmarked pairs: (B,A), (D,C), (E,C), (E,D)

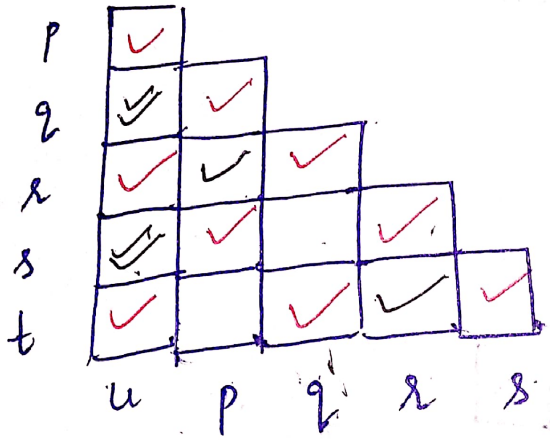
(D,C,E)



Q2



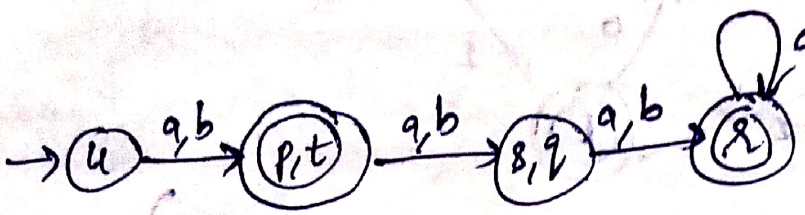
a = 0
b = 1



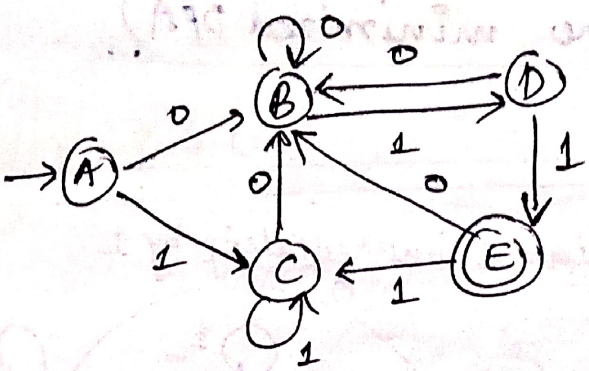
unmarked

(q,u): $\delta(q,0) = r$, $\delta(u,0) = p$; $\delta(q,1) = r$, $\delta(u,1) = t$	(r,p): $\delta(r,0) = r$, $\delta(p,0) = q$; $\delta(r,1) = r$, $\delta(p,1) = s$
(s,u): $\delta(s,0) = r$, $\delta(u,0) = p$; $\delta(s,1) = r$, $\delta(u,1) = t$	(t,p): $\delta(t,0) = s$, $\delta(p,0) = q$; $\delta(t,1) = q$, $\delta(p,1) = s$
(s,q): $\delta(s,0) = r$, $\delta(q,0) = r$; $\delta(s,1) = r$, $\delta(q,1) = r$	(t,s): $\delta(t,0) = s$, $\delta(s,0) = r$; $\delta(t,1) = q$, $\delta(s,1) = r$

unmarked (t,p), (s,q),



Q3



- (B,A)
  $\delta(B,0) = B$] $\delta(B,1) = D$]
  $\delta(A,0) = B$] $\delta(A,1) = C$]
  ① unmarked
  ② mark (B,A)

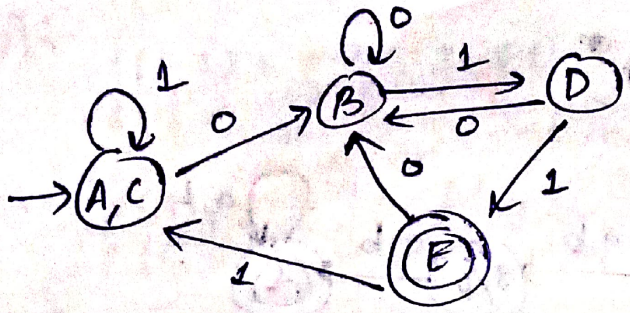
- (C,A)
  $\delta(C,0) = B$] $\delta(C,1) = C$]
  $\delta(A,0) = B$] $\delta(A,1) = C$]
  ① unmarked (always)

- (D,A)
  $\delta(D,0) = B$] $\delta(D,1) = E$]
  $\delta(A,0) = B$] $\delta(A,1) = C$]
  ① mark (D,A)

- (C,B)
  $\delta(C,0) = B$] $\delta(C,1) = C$]
  $\delta(B,0) = B$] $\delta(B,1) = D$]
  ① unmarked
  ② mark (C,B)

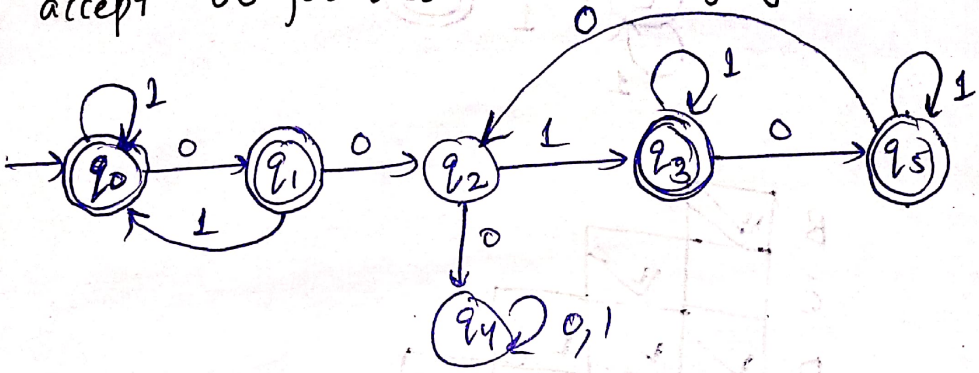
- (D,B)
  $\delta(D,0) = B$] $\delta(D,1) = E$]
  $\delta(B,0) = B$] $\delta(B,1) = D$]
  ① mark (D,B)

- (D,C)
  $\delta(D,0) = B$] $\delta(D,1) = E$]
  $\delta(C,0) = B$] $\delta(C,1) = C$]
  ① mark (D,C)

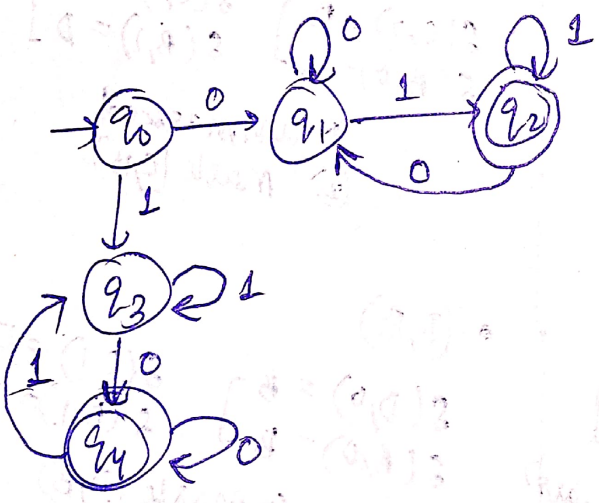


This is the minimized DFA.

Q DFA accept 00 followed immediately by 1



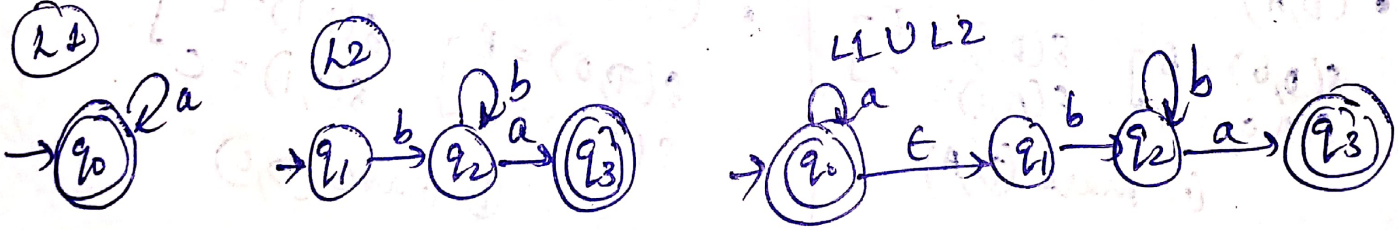
Q DFA accept : left most symbol differ from right most symbol



Q Design NFA with four states for

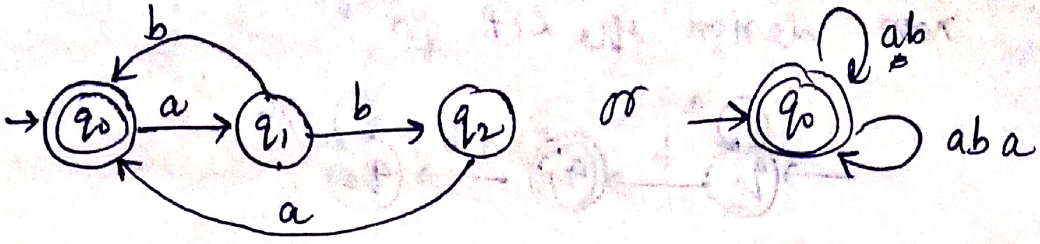
$L = \{(a^n : n \geq 0) \cup (b^n a : n \geq 1)\}$

$L1$ $L2$



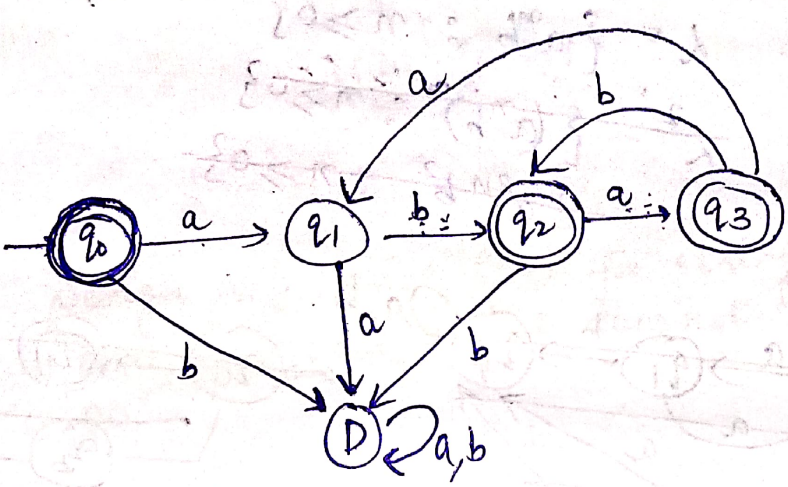
Q NFA

$L = (ab \cup aba)^*$ Null string also accepted

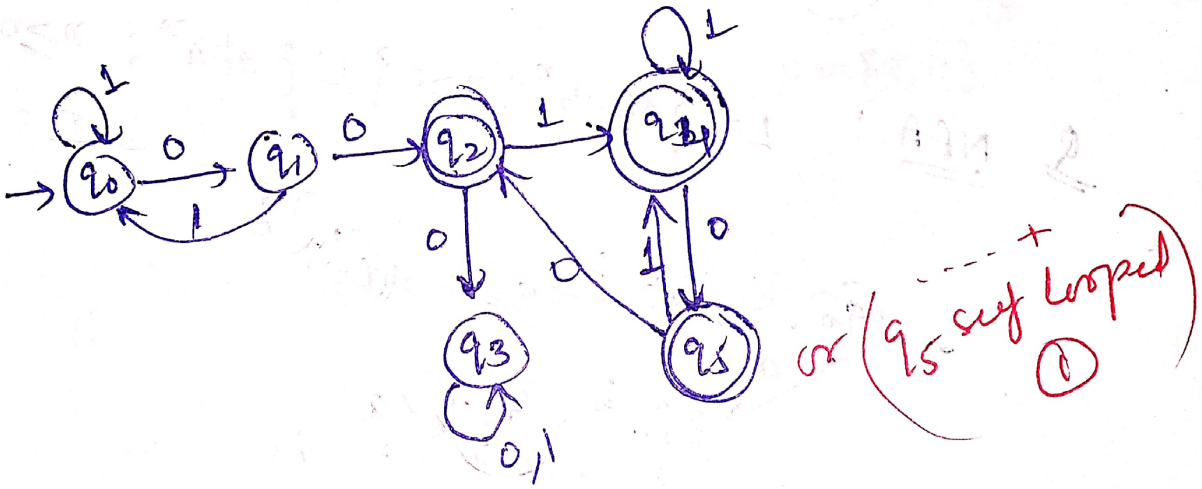


Q DFA

$L = (ab \cup aba)^*$

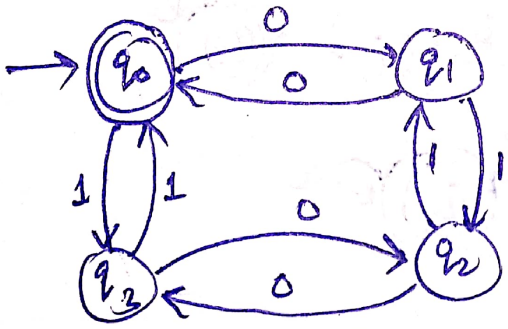


Q DFA accepts "00" as substring but not "000"

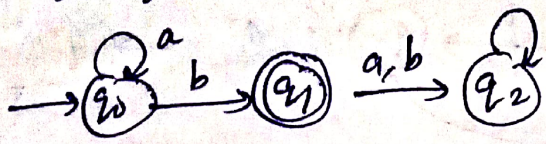


Q ~~DFA:- last two symbols same.~~

Q even number of 0s & 1s. should be accepted



Q Here is given a below DFA for the language L now design the DFA for L^2



Sol.

By this DFA, language can be stated as →

$(01)^2$

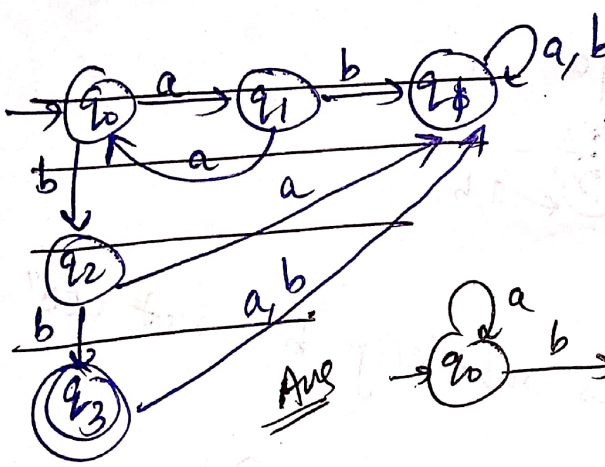
0101

$$L = \{a^n b : n \geq 0\}$$

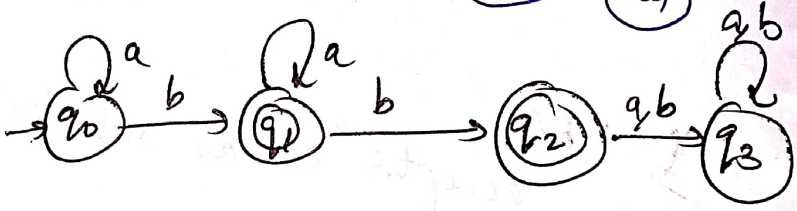
abab

$$L^2 = \{(a^n b)^2 : n \geq 0\}$$

$$= \{a^{2n} b^2 : n \geq 0\}$$

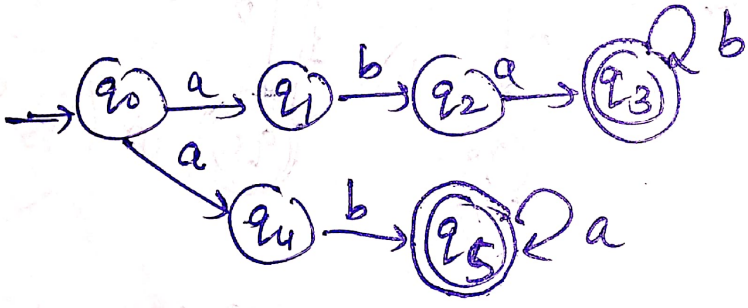


Ans

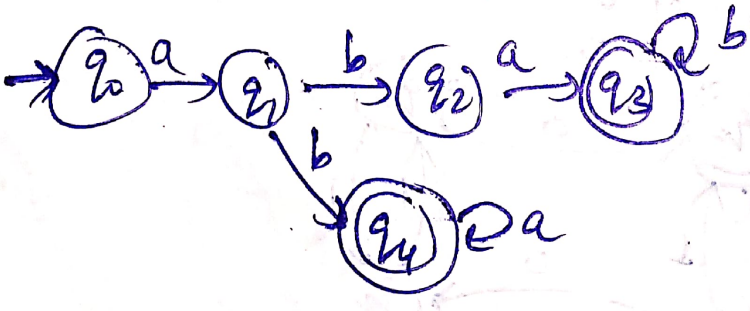


Q NFA : $L = \{abab^n : n \geq 0\} \cup \{aba^n : n \geq 0\}$.

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

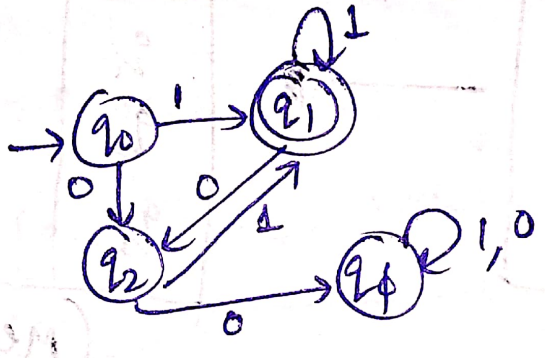


OR.

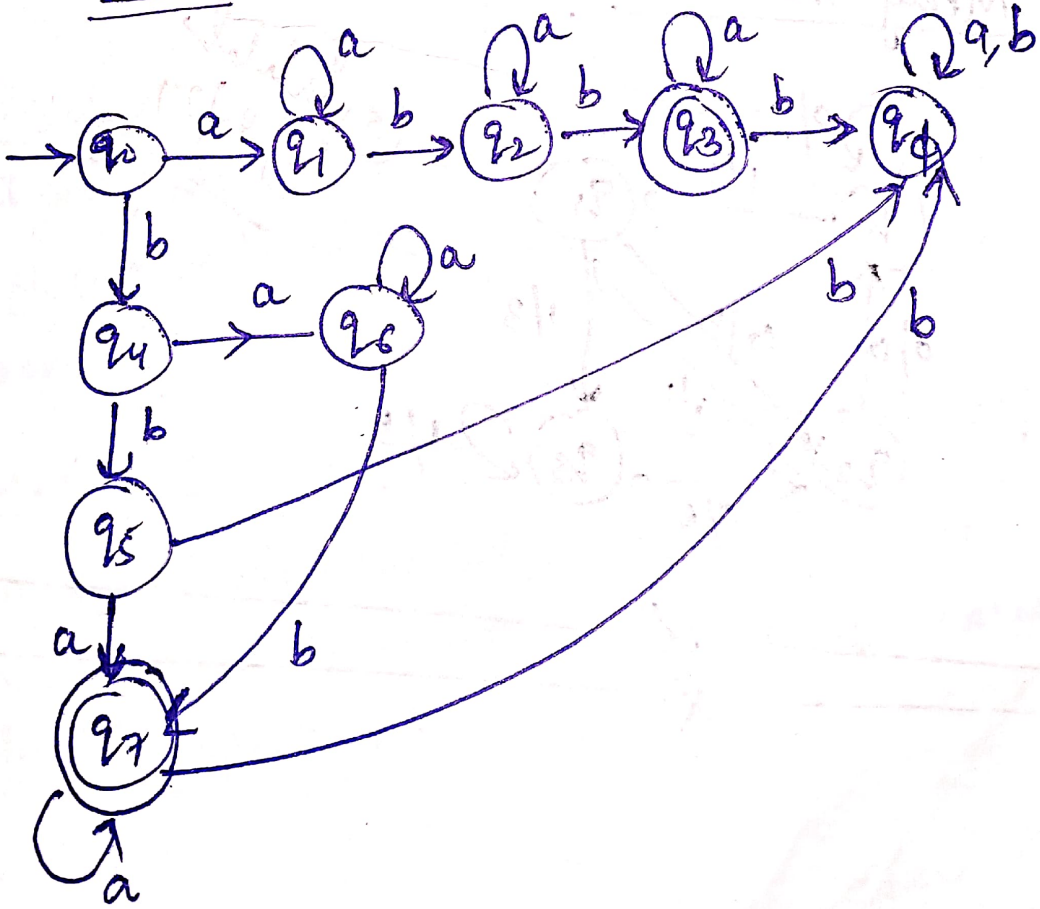


Q Design NFA over (0,1) for strings ~~accepting~~ that ends with "1" but does not contain substring "00"

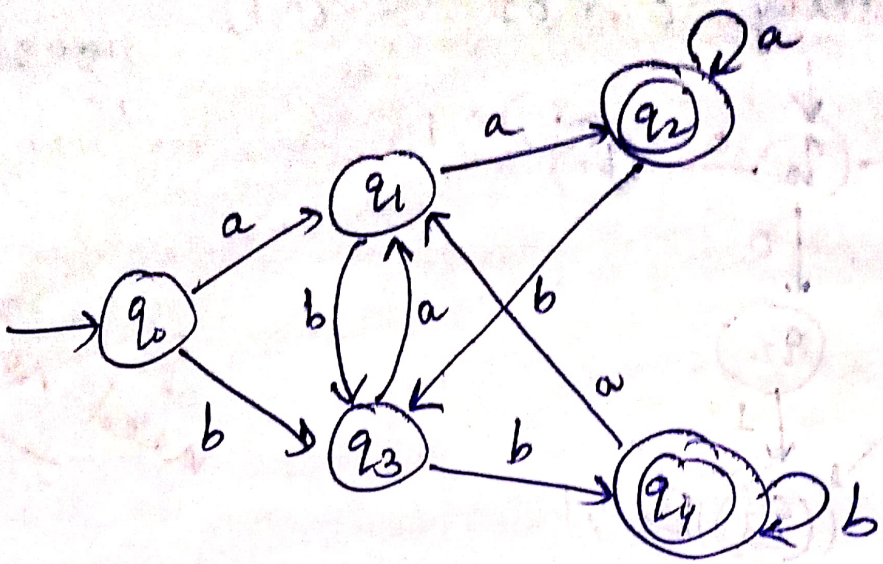
Sol.



Q DFA :- $n_a \geq 1$ & $n_b = 2$. $\Sigma = \{a, b\}$



Q DFA: Last two symbols are same



Q Construct a mealy machine for accepting binary strings divisible by 4.

Sol.

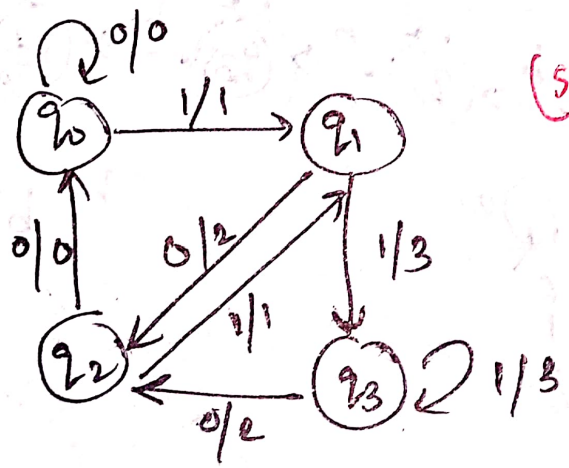
ε | Q	0	1	o/p
q_0	q_0	q_1	0
q_1	q_2	q_3	1
q_2	q_0	q_1	2
q_3	q_2	q_3	3

(Moore TT)

Q | Σ	0		1	
	NS	o/p	NS	o/p
q_0	q_0	0	q_1	1
q_1	q_2	2	q_3	3
q_2	q_0	0	q_1	1
q_3	q_2	2	q_3	3

(Mealy TT)

Mealy machine



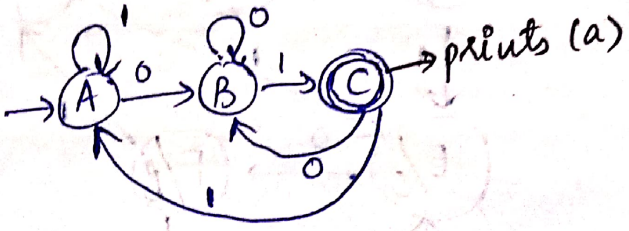
Moore Machine Constructions

Q1. prints 'a' whenever sequence is "01" encountered

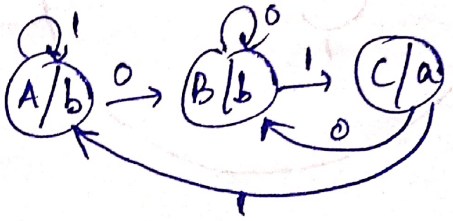
i/p $\Sigma = [0,1]$.

o/p $\Delta = [a,b]$

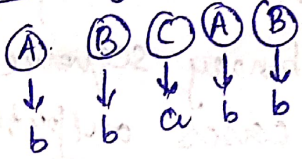
DFA:



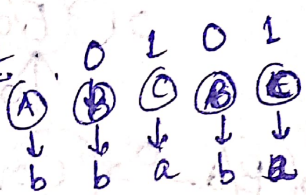
Moore machine:



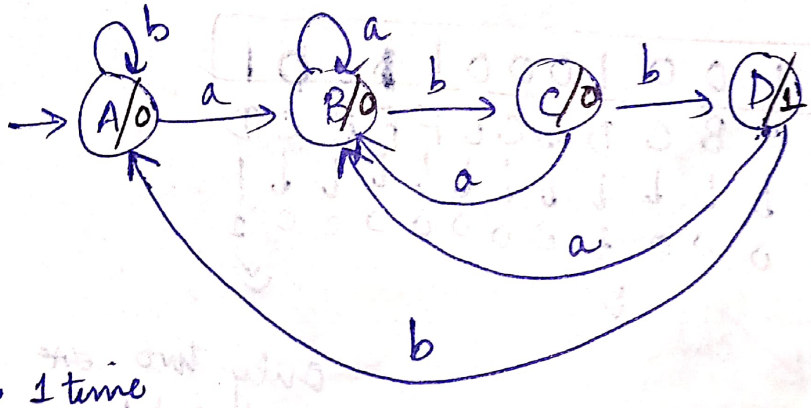
Few examples: i) input 0 1 1 0 ⇒ output bbabb



ii) input 0 1 0 1 ⇒ output bbaba.

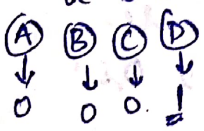


Q2 Construct Moore machine to count occurrences of 'abb' in any i/p string. $\Sigma = [a,b]$, $\Delta = [0,1]$



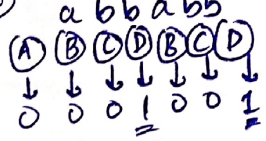
Examples

① a b b



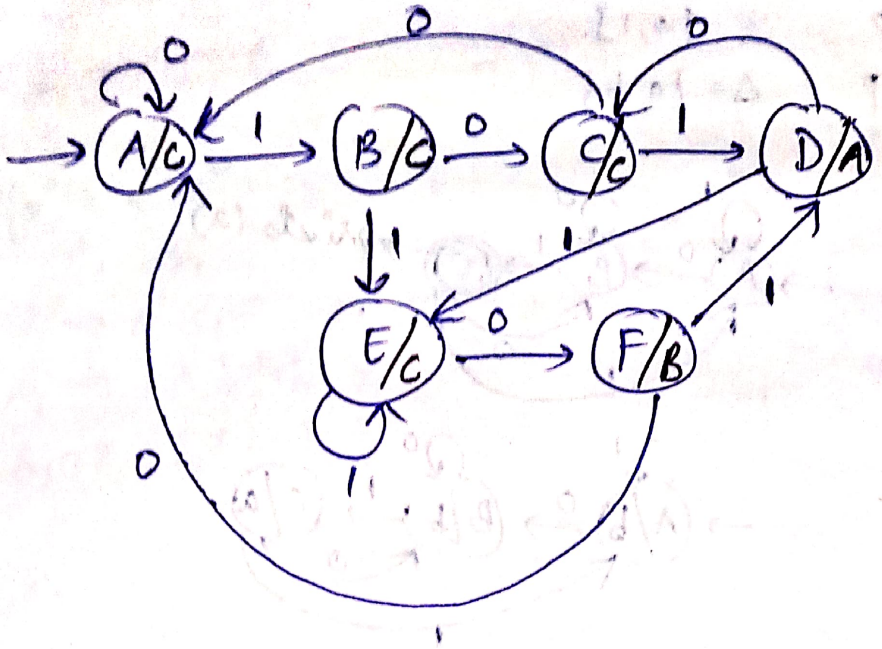
⇒ 1 time

② a b b a b b

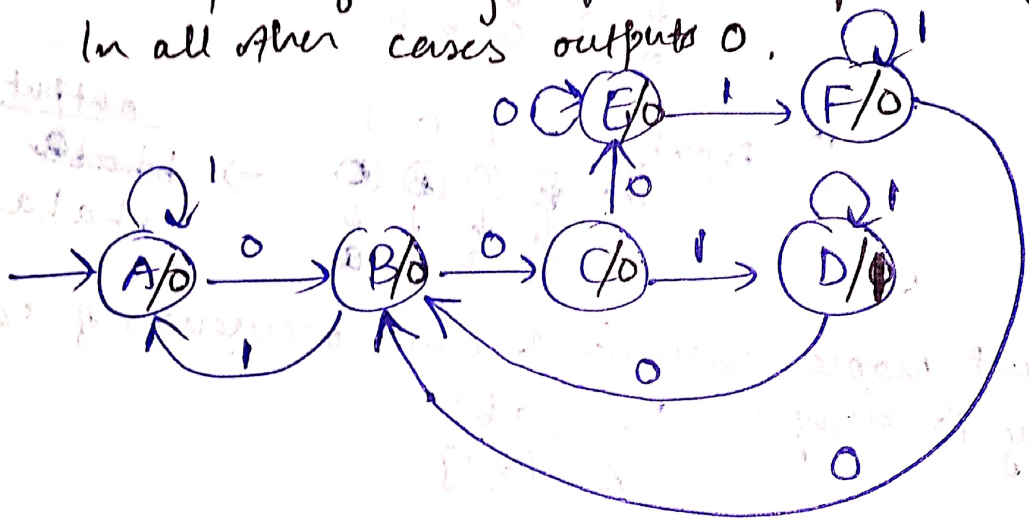


⇒ 2 times

Q3 for binary sequence if substring 101, outputs A, If input substring 110, outputs B, otherwise C.



Q4 Design Moore Machine for generating the o/p 1 if input of binary sequence '1' is preceded by exactly 2 0s In all other cases outputs 0.



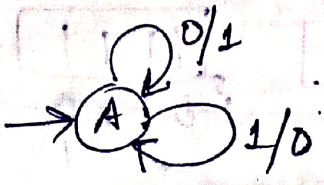
Eg

1	1	0	0	1	0	0	0	1	0	0	1
A	A	A	B	C	D	B	C	E	F	F	B	C	D
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0	0	0	0	0	1	0	0	0	0	0	0	0	1

Only two 1s are preceded by exactly 2 0s.

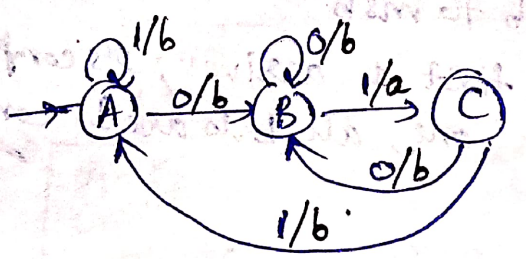
Mealy machine Construction

Q1 produces 1's complement of binary string

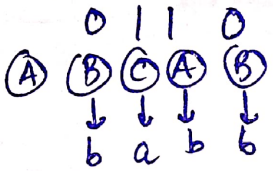


Q2 prints 'a' when sequence "01" is encountered

$\Sigma = [0, 1]$ $\Delta = [a, b]$

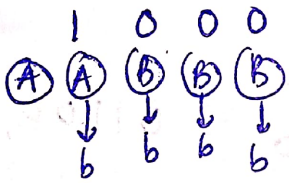


Eg



⇒ baabb

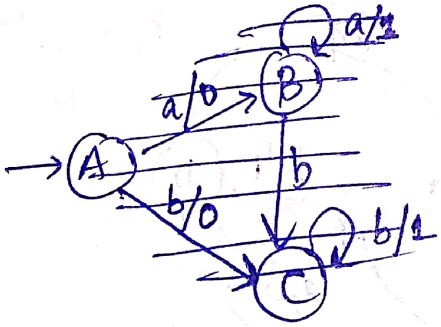
Eg

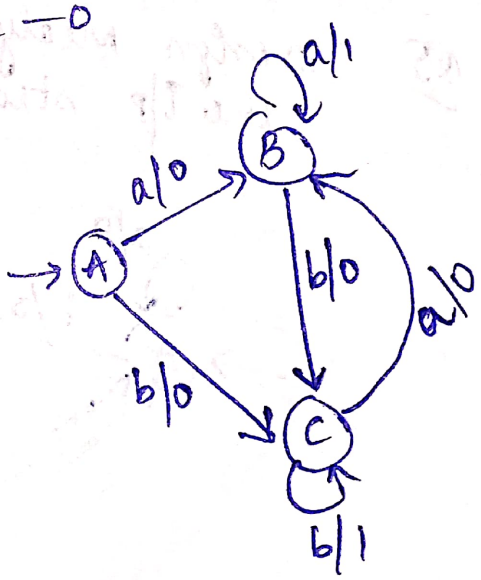


⇒ bbbb (no "01" encountered)

Q3 strings should end with "aa" or "bb"

aa – 1
bb – 1
else – 0





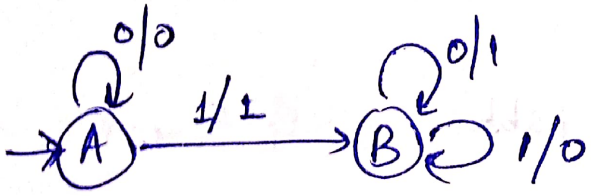
Q4 2's complement of binary number
"last carry bit neglected"

eg
10100	11100	1111
1^s 01011	1^s 00011	1^s 0000
+ 1	+ 1	+ 1
01100	00100	0001

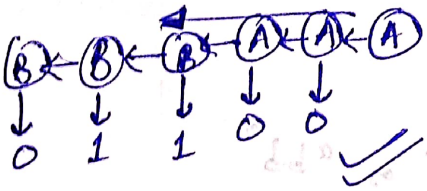
~~Logic until first one~~

Logic from lsb to msb
- until first 1 arrives, 2's comp is same
- after first one, all bits are 1's complemented

lsb to msb

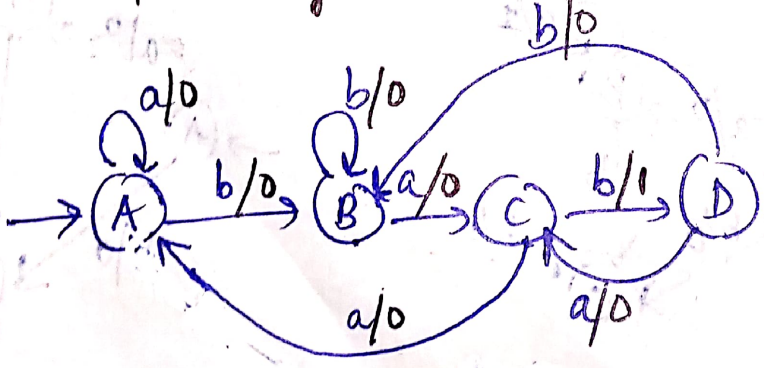


eg $\overset{M}{1}0100\overset{L}{}$



$10100 \Rightarrow 01100$.

Q5 Design Mealy machine that gives o/p 1 if the i/p strings ends in "bab"; otherwise it outputs 0.



baa