Introduction of theory of Computation 07 :-

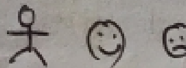
Automata theory (also known as Theory of Computation) is a theoretical branch of Computer Science & mathematics, which mainly deals with the logic of Computation with respect to simple machines, reffered to as automata.

Automata enables the scientists to understand how machines Compute the functions and solve problems. The main motivation behind developing automata theory was to develop methods to describe and analyse the dynamic behavior of discrete systems.

Automata is originated from the word "Automaton" which is closely related to "Automation".

The basic terminologies which is closely related to Automation or automata theory are as follows:

Symbol :- Symbol is the smallest building block, which can be any alphabet, letter or any picture.

eg :- a, b, c, 0, 1, - - - 

08

Alphabets (Σ) :- Alphabets are set of symbols, which are always finite.

eg:- $\Sigma = \{0, 1\}$ is an alphabet of binary digits.

$\Sigma = \{0, 1, \ldots 9\}$ is an alphabet of decimal digits.

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

$\Sigma = \{A, B, C, \ldots \ldots Z\}$

String :- String is a finite sequence of symbols from some alphabet. String is generally denoted as w and length of a string is denoted as $|w|$

Empty String is the string with zero occurence of symbols represented as ϵ (Epsilon)

eg:- The number of strings that can be generated over the alphabet $\{a, b\}$ of length 2 is $\{aa, ab, ba, bb\}$.

Note :- If the number of Σ's is represented by $|\Sigma|$, then number of strings of length 'n', possible over Σ is $|\Sigma|^n$

Language :- A language is a set of strings, chosen from some Σ^*. A language is a subset of Σ^*. A language which can be formed over Σ can be finite or infinite.

eg:- L_1 = { Set of strings of length 2 }
= { aa, ab, ba, bb } ⟶ Finite language.

L_2 : { Set of strings which starts with 'a' }
= { a, aa, ab, aba, aaa, abb, - - . . . } → which is infinite language.

Theory of Computation is normally divided in 3 parts:

i) Automata Theory and languages.
ii) Computability theory
iii) Complexity theory.

Automata theory :- It deals with the definitions & properties of mathematical models of Computation. The finite automata used in text processing, Compilers & hardware design. The Context free grammers are used in programming language & artificial intelligence.

Computability theory :- In the first half of the 20th Century, mathematicians such as Kurt Godel, Alan turing, and Alonzo Church discovered that certain basic problems

Cannot be solved by computers

mathematical statement is true or false.

Computability theory classify problems are solvabl
not solvable. Complexity theory classify problems as e
ones & hard ones.

<u>Complexity theory</u> :- it deals with easy problems (sort a
million items in a few seconds). Hard problems (schedule
a thousand classes in a hundered years). What makes
some problems hard and others easy computationally ?,
Complexity theory addresses this question.

<u>Why to study the theory of computation</u> :-

i). Determine what can and cannot be computed.

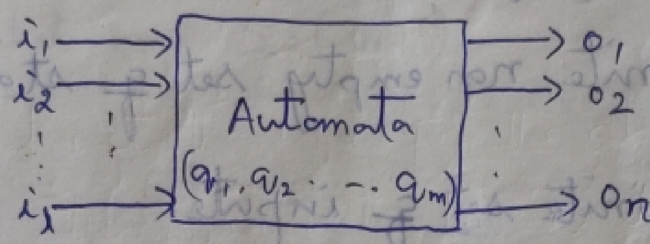
ii) How quickly & with how much memory.

iii) on what type of computational model.

Theory of Computation

Chapter - 1: Introduction.

Automata :- it is a system, used to perform computation without the participation of human. it is generally a mathematical model.



Inputs = $\Sigma = \{ i_1, i_2, \cdots i_l \}$

states = $Q = \{ q_1, q_2 \cdots q_m \}$

outputs = $\Delta = \{ o_1, o_2 \cdots o_n \}$

The output depends only one inputs, is called as Automata without a memory. (Independent of states)

eg :- $o_1 \rightarrow i_1$, $o_2 \rightarrow i_2 \cdots$

The output depends only on states, is called as Moore Machine. (Independent of inputs)

es :- $o_1 \rightarrow q_1$, $o_2 \rightarrow q_2 \cdots$

08 The output depends upon both states & inputs called as Mealy machine.

eg:- $o_1(\delta, i_1)$, $o_2(\delta, i_2)$

Finite Automata :- it is a 5 tuple system.

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

Q — Finite non empty set of states.

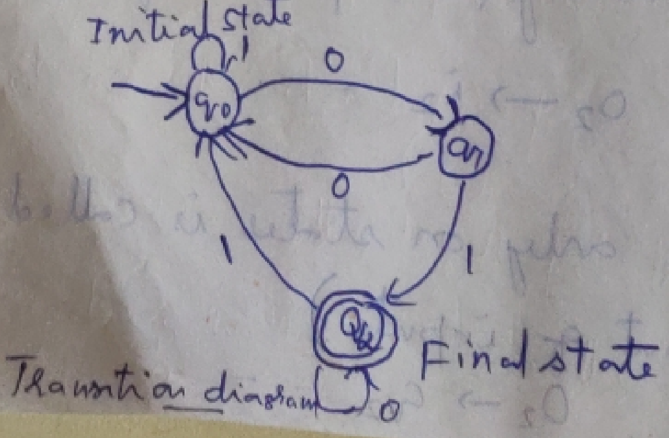
Σ — Finite set of inputs.

δ — transition function (change of state during transaction) $\delta : Q_1 \times \Sigma \rightarrow Q_2$

q_0 — initial state / Starting state. $q_0 \in q$.

F — Set of final states. $F \subseteq q$

Representation of finite automata by transition diagram.



Transition diagram

Transition Table

present state	Next state 0	Next state 1
$\rightarrow q_0$	q_1	q_0
q_1	q_0	q_2
q_2*	q_2	q_0

s 2 inputs

Representation by 5 tuples :-

$$M = (\underbrace{\{q_0, q_1, q_2\}}_{Q}, \underbrace{\{0, 1\}}_{\Sigma}, \delta, q_0, \{q_2\})$$

$\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_0$, $\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_2$, $\delta(q_2, 1) = q_0$

Input alphabet :- $\Sigma = (0, 1)$

$\Sigma^0 = |\epsilon| = 0$

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

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

$\Sigma^k = \{$Set of all strings over K$\}$

Construction of DFA

Type-01 :- For strings ending with a particular sub string

Step 1 :- Decide the minimum number of states required in the DFA and draw them.

Rule :- All strings ending with 'n' length, substring always require minimum $(n+1)$ states in this DFA.

Example :- Consider the regular expression -

$$(a*b)^* z$$

the string of above language will always end on 'z', so if $|z| = n$, then the DFA requires minimum $(n+1)$ states.

Step 2 :- Decide the strings for which you will construct the DFA.

Step 3 :- Construct the DFA for the above decided string.

Remember :- always prefer to go with the existing path. Create a new path only when you can't find a path to go with.

Step 4 :- After drawing the DFA for the above decided strings, send the left possible combinations to the starting state not over the dead configuration.

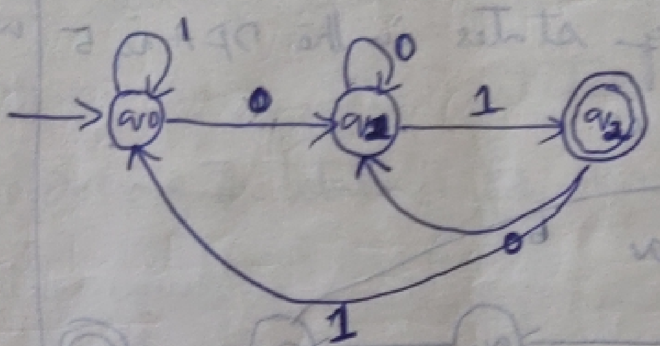
roblem 1 :- Draw the DFA for the language accepting strings ending with '01' over the input alphabet $\Sigma = \{0,1\}$

:- Regular expression for the given language is

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

minimum number of states in the DFA = 3

required DFA is



Strings we will Check

01 ✓

001 ✓

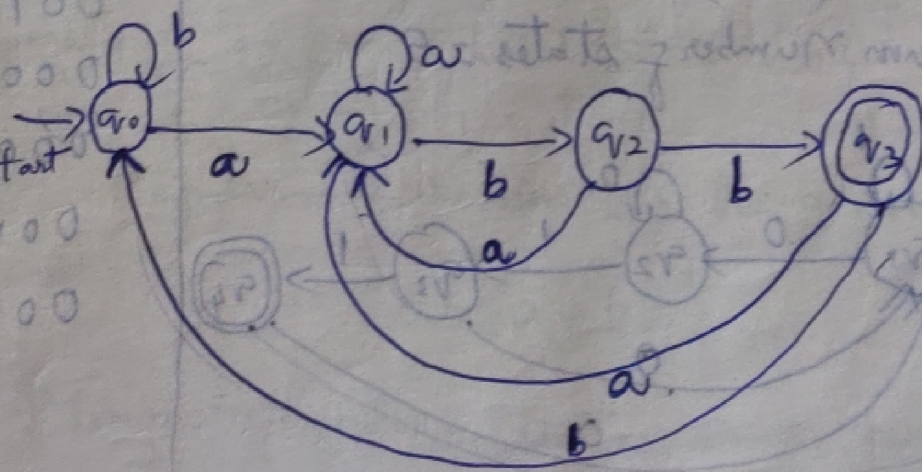
0101

2) Draw the DFA for the language accepting strings ending with abb over the input alphabet $\Sigma = \{a, b\}$

Sol^n :- Regular expression for the given language is

$$(a+b)^{*}abb$$

the minimum number of states in the DFA = 4



Strings we will Check

abb ✓

aabb ✓

ababb

abbabb

3) Draw the DFA for the language accepting strings ending with 'abba' over $\Sigma = \{a, b\}$

Sol :- Regular expression for the given language is

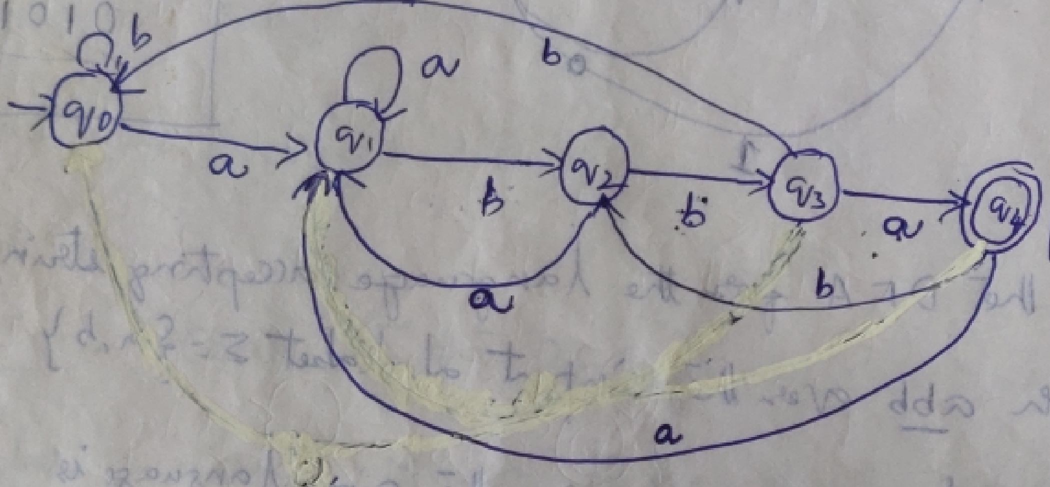
$$(a+b)^* abba$$

Minimum number of states in the DFA is 5

The required DFA is

Strings we will check

- abba
- aabba
- ababba
- abbaabba
- abbaabba

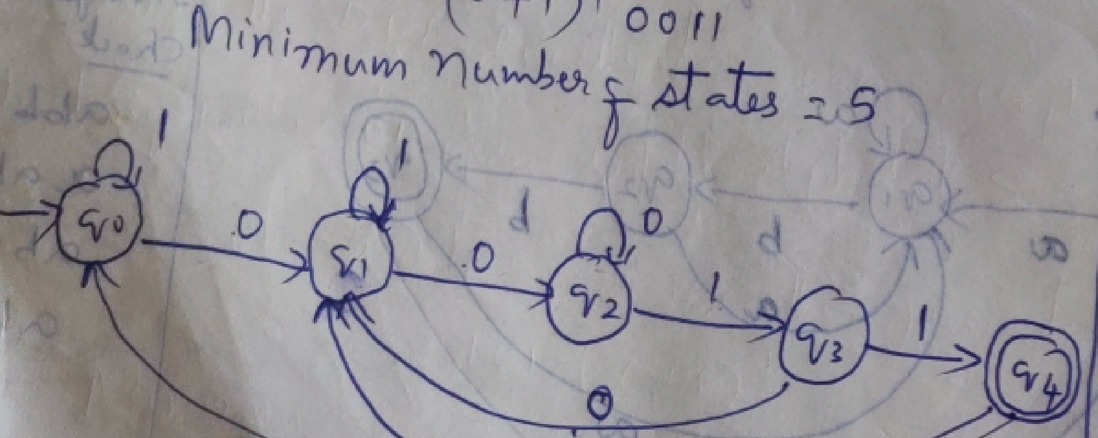


4) Draw the DFA for the language accepting strings ending with '0011' over $\Sigma = \{0, 1\}$

$$(0+1)^* 0011$$

Minimum number of states = 5

- 0011
- 00011
- 000011
- 0010011
- 00110011



For strings starting with a particular substring

Step 1 :- Decide the minimum number of states required in the DFA and draw them.

Rule :- All strings starting with 'n' length substring will require minimum of $(n+2)$ states in its DFA.

Step 2 :- Decide the strings for which you will construct the DFA.

Step 3 :- Construct the DFA for the above decided strings.

Remember :- Always go with the existing path. Create a new path only when you can't find a path to go with.

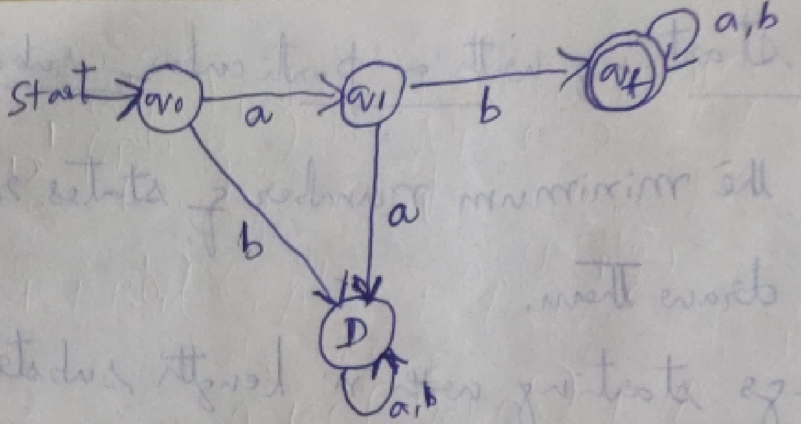
Step 4 :- After drawing the DFA for the above decided strings, send the left possible combinations to the dead state not over the starting state.

1) Draw the DFA for the language accepting strings starting with ab over input alphabet $\Sigma = \{a, b\}$

Sol^n :- Regular expression for the given language is

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

Minimum number of states in the DFA = 4



ab
aba
abab

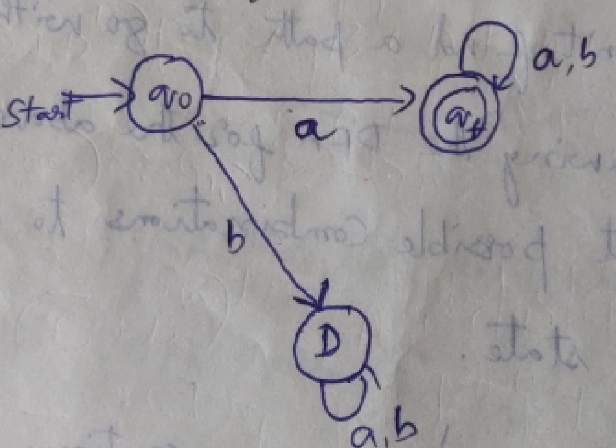
2) Draw the DFA for the language accepting strings starting with 'a' over input alphabet $\Sigma = \{a, b\}$

Sol^n :- Regular expression for the given language is

$a(a+b)^*$

Minimum number of states in the DFA = 3

a
aa

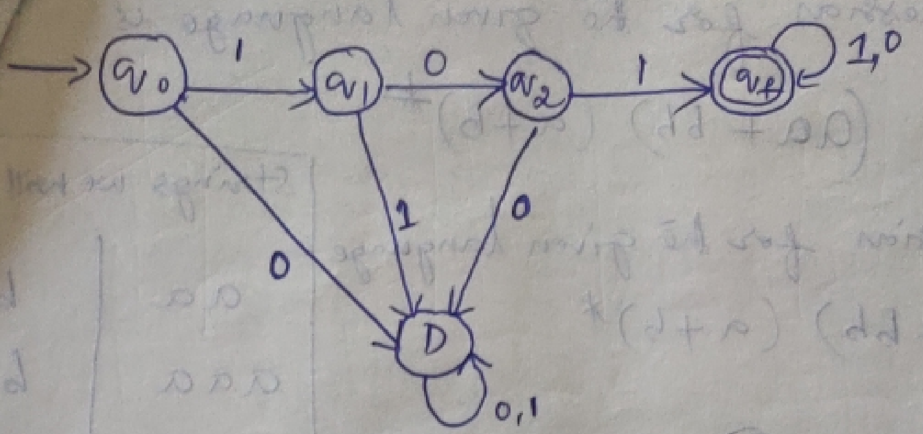


3) Draw the DFA for the language accepting strings starting with 101 over input alphabet $\Sigma = \{0, 1\}$

Sol^n :- Regular expression for the given language is

$101(0+1)^*$

Minimum number of states in the DFA = 5



101

1011

10110

101101

4) Construct a DFA that accepts a language L over $\Sigma = \{0,1\}$ such that L is the set of all strings starting with '00'

Regular expression for the given language is

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

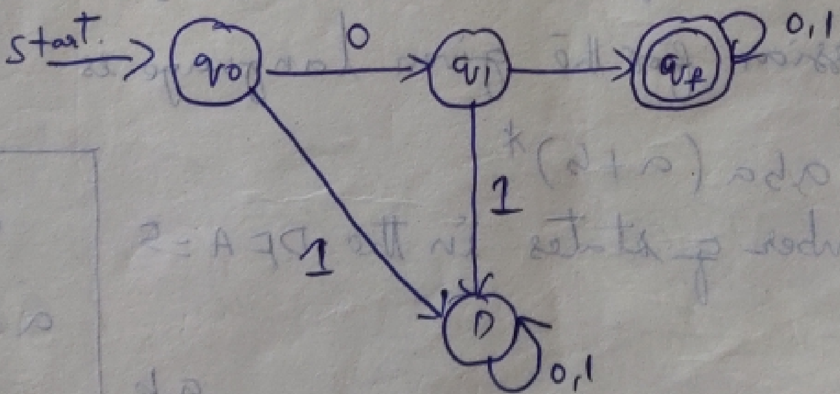
Minimum number of states in the DFA = 4

Strings we will check

00

000

0000



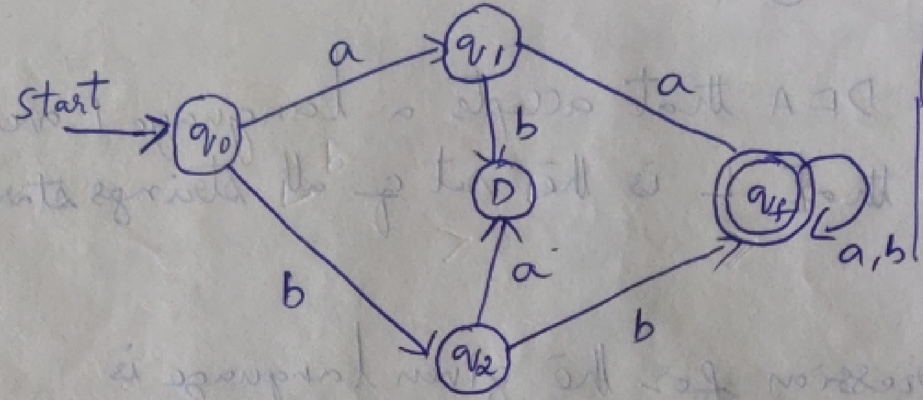
5) Construct a DFA that accepts a language L over $\Sigma = \{a,b\}$ such that L is the set of all strings starting with 'aa' or 'bb'

Regular expression for the given language is

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

Regular expression for the given language is $(aa+bb)(a+b)^*$

Strings we will ch	
aa	bb
aaa	bbb
aaaa	bbbb



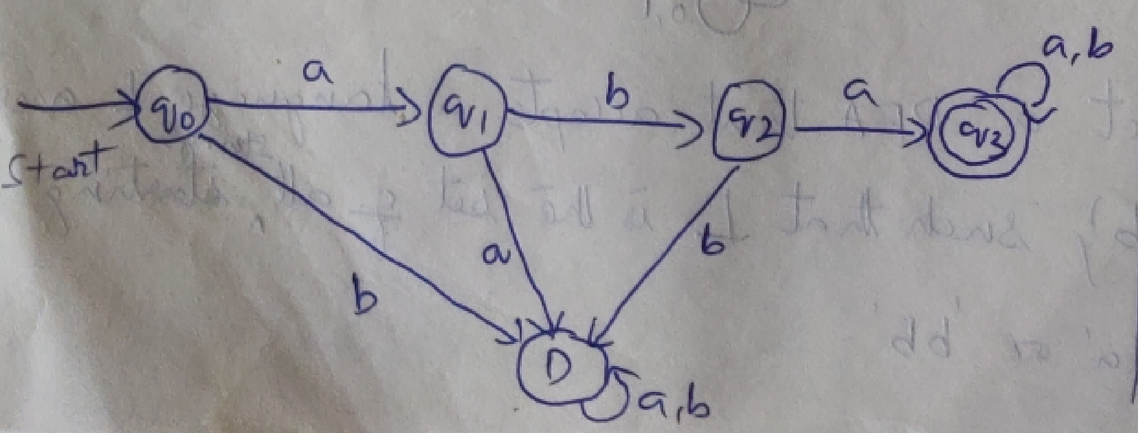
6) Draw the DFA that accepts a language L over $\Sigma = \{a,b\}$ such that L is the set of all strings starting with 'aba'

Regular expression for the given language is

$aba(a+b)^*$

Minimum number of states in the DFA = 5

aba
abaa
abaab
abaaba



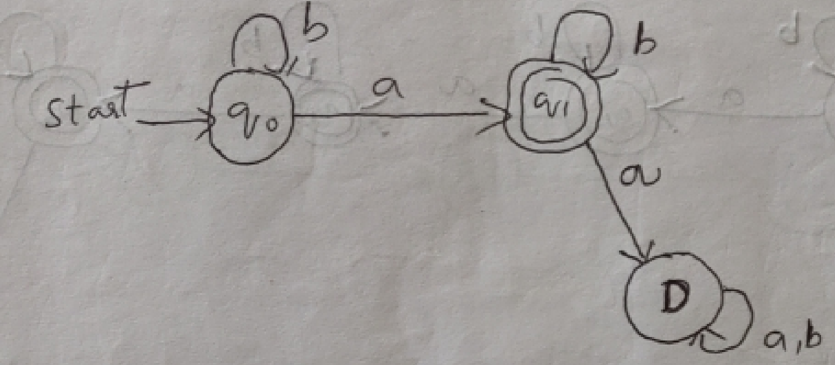
Design the DFA that accepts a language L over

= {a, b} having

i) exactly one 'a'

ii) Atleast one 'a'

iii) Atmost one 'a'

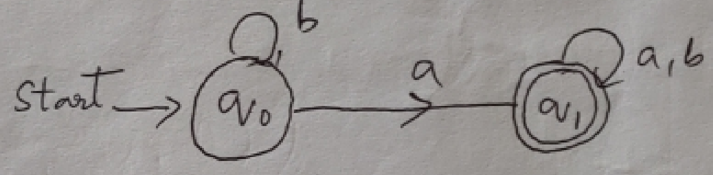
i) 

Strings

bab ✓

babb ✓

baa ✗

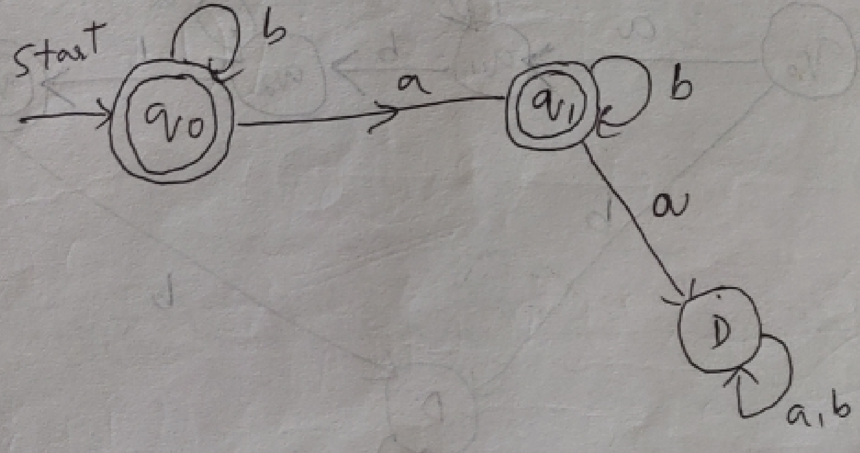
ii) 

Strings

bba ✓

baba ✓

bbb ✗

iii) 

Strings

b ✓

bab ✓

babab ✗

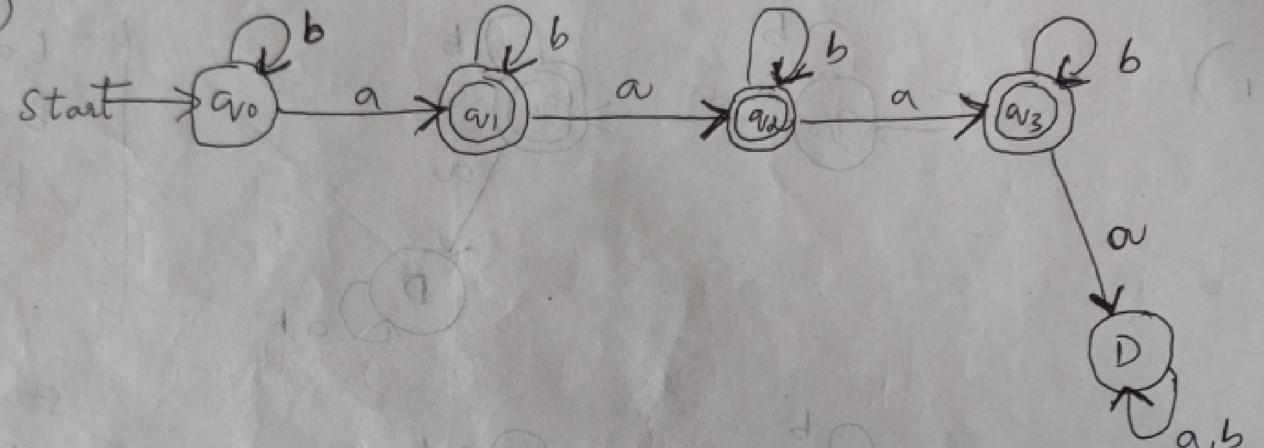
8) 14 Design the DFA that accepts a language L

$\Sigma = \{a, b\}$ such that L is the set of strings

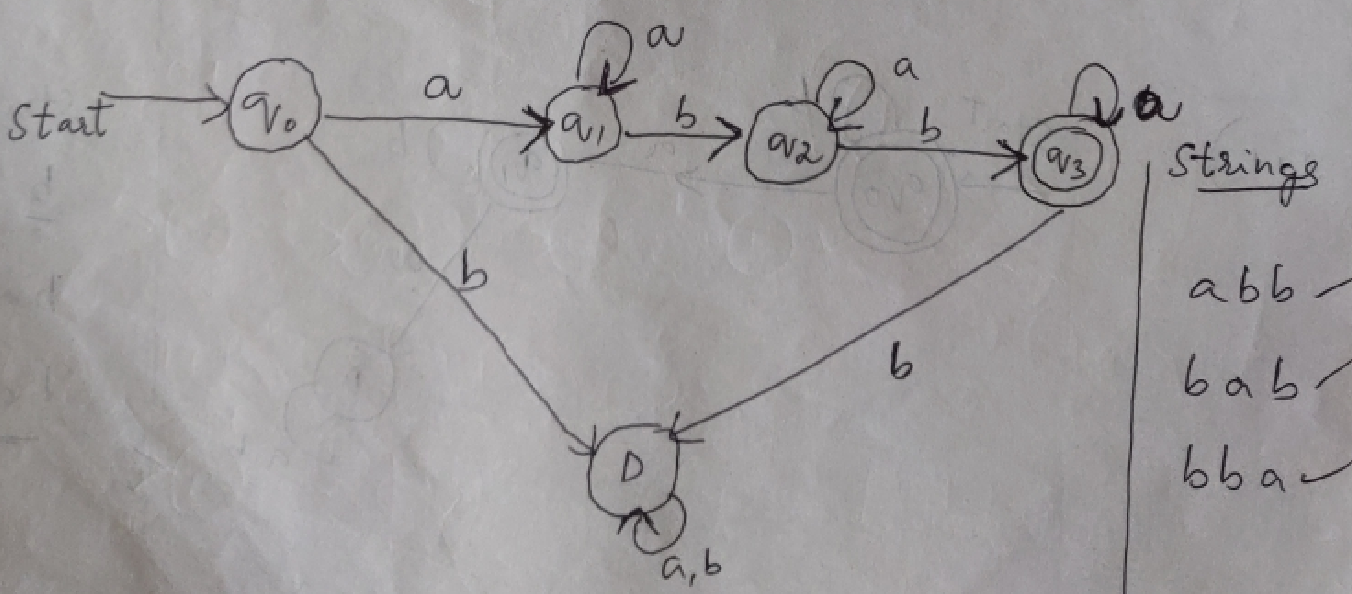
i) No more than '3a' or atmost '3a'

ii) with atleast 'a' & exactly two b's

i)



ii)



Strings

abb

bab

bba

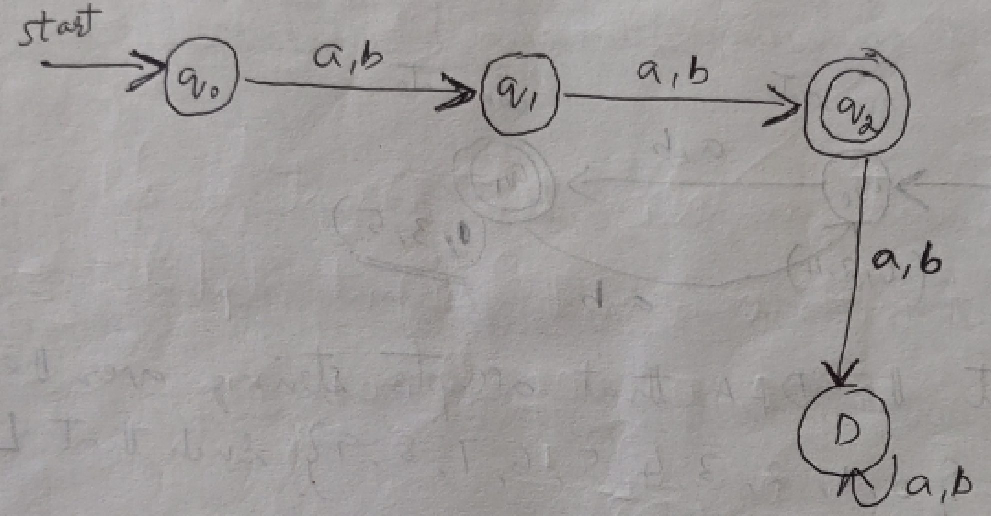
Construct a DFA that accepts a language L over $\Sigma = \{a, b\}$ such that L is the set of strings

i) The length of the string is exactly 'two'

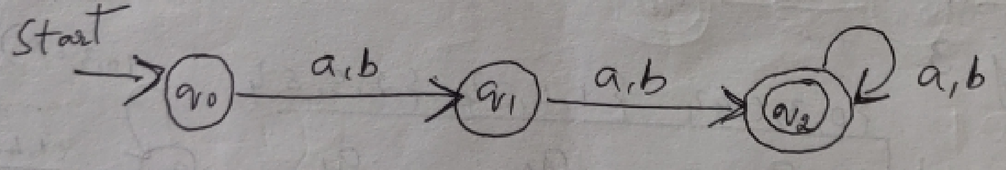
ii) length of the string is atleast 'two'

iii) length of the string atmost 2

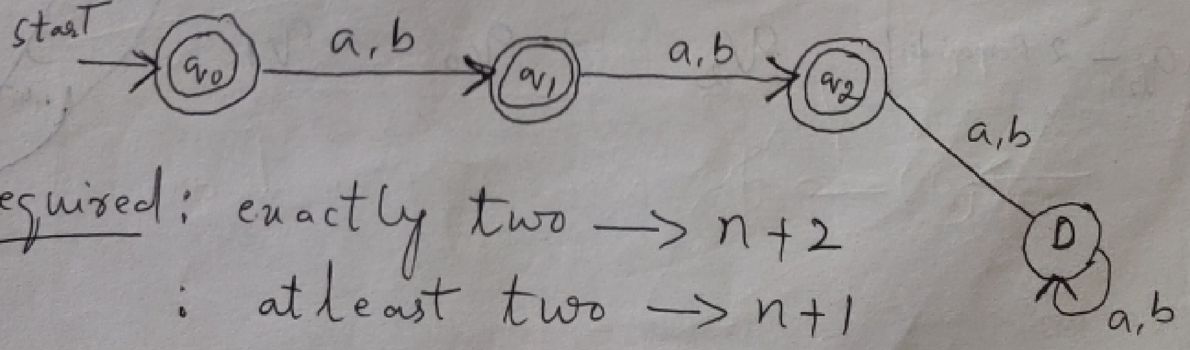
i)



ii)



iii)



states required: exactly two $\longrightarrow n+2$

: atleast two $\longrightarrow n+1$

atmost two $\longrightarrow n+2$

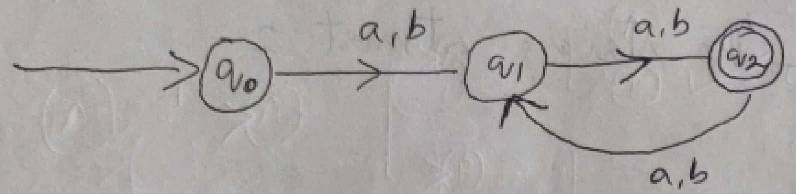
16

10) Construct the DFA that accepts a strings over input alphabet $\Sigma = \{a, b\}$ such that i) $|W| \bmod 2 = 0$

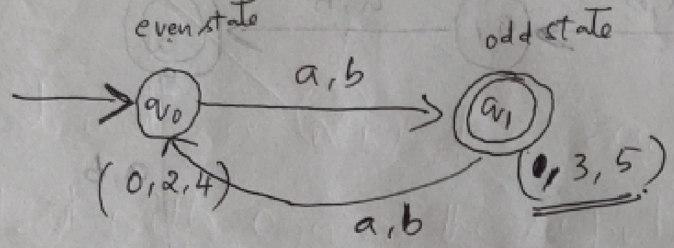
ii) $|W| \bmod 3 = 1$

$L = \{ \epsilon, aa, ab, ba, bb, aaaa, bbbb \ldots \}$

i)

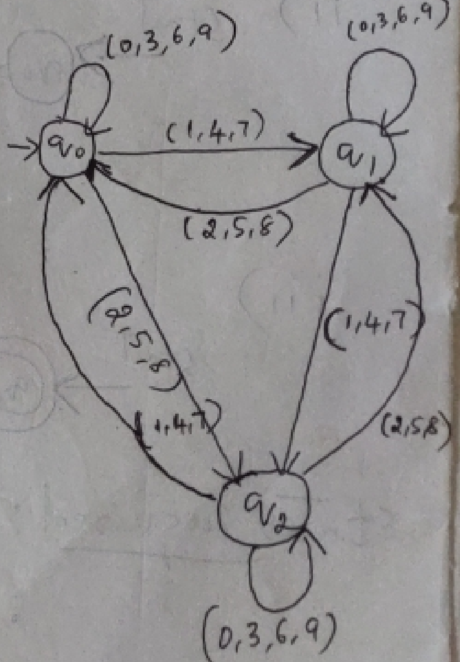


ii)



ii) Construct the DFA that accepts string over the input alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ such that L is divisible by 3.

	{0,3,6,9}	{1,4,7}	{2,5,8}
q_0 – 0 Remainder	q_0	q_1	q_2
q_1 – 1 Remainder	q_1	q_2	q_0
q_2 – 2 Remainder	q_2	q_0	q_1



-Deterministic Finite Automata :- NFA is similar to DFA except following additional features:

i) Null (or ϵ) move is allowed i.e., it can move forward without reading symbols.

ii) Ability to transmit to any number of states for a particular input.

NFA has a different transition function, rest is same as DFA

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

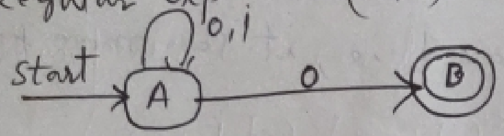
one important thing to note is, In NFA, if any path for an input string leads to a final state, then the input string is accepted.

Some important points:

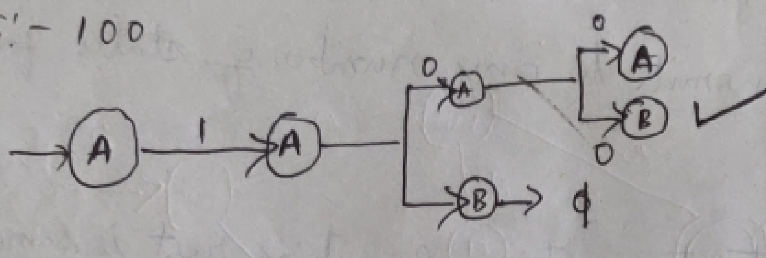
1. Every DFA is NFA but not vice versa
2. Both NFA and DFA have same power and each NFA can be translated into a DFA.
3. There are multiple final states in both DFA & NFA
4. NFA is more of a theoretical concept.
5. DFA is used in lexical analysis in compiler.

1) Constenct a NFA that accepts a sets of all strings that end with 0 over the input alphabet $\Sigma = \{0,1\}$

Ans:- Regular exp:- $(0+1)^*0$



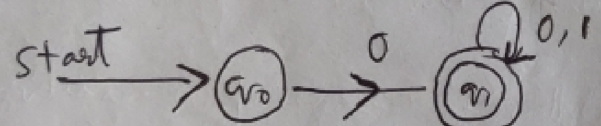
eg:- 100



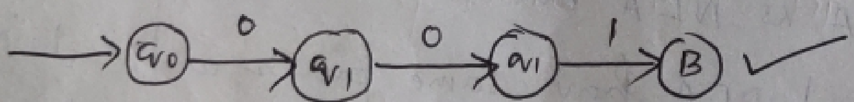
If there is any way to run the machine that ends in any set of states out of which atleast one state is a final state, then the NFA accepts.

2) Construct a NFA that accepts a set of all strings that start with 0 over the input alphabet $\Sigma = \{0,1\}$

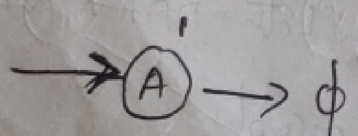
$L = \{0, 00, 01, 000, 001, 010 \text{———} \ldots\}$



eg:- 001



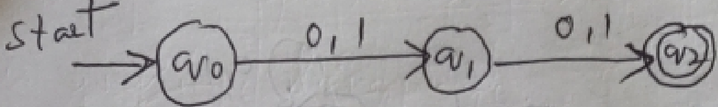
eg:- 101



Dead configuration.

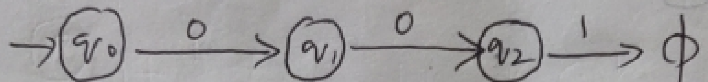
Construct a NFA that accepts sets of all strings over {0,1} of length 2.

$L_1 = \{00, 01, 10, 11\}$



001

eg:-



4) Construct a NFA for the following cases:

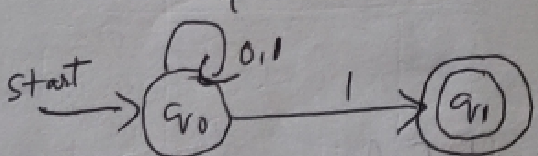
i) L1 = { set of all strings that ends with '1' }

ii) L2 = { set of all strings that contain '0' }

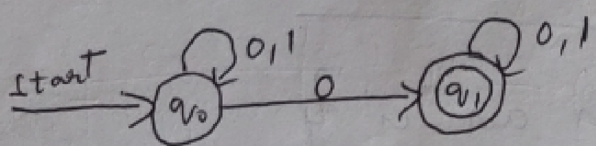
iii) L3 = { set of all strings that starts with '10' }

iv) L4 = { set of all strings that contain '01' }

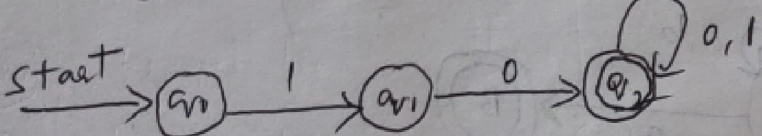
i) L1 = { 1, 01, 001, 101, 111, 0101 - - . }

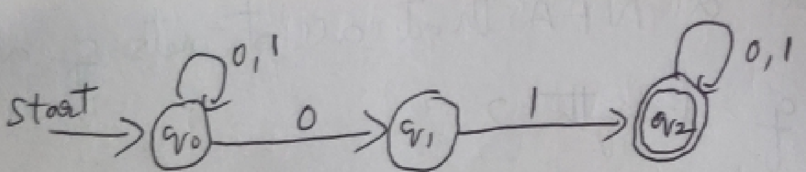


ii)

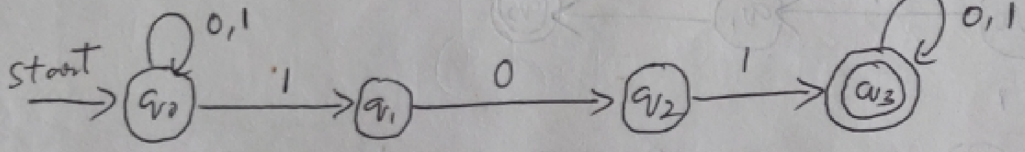


iii)



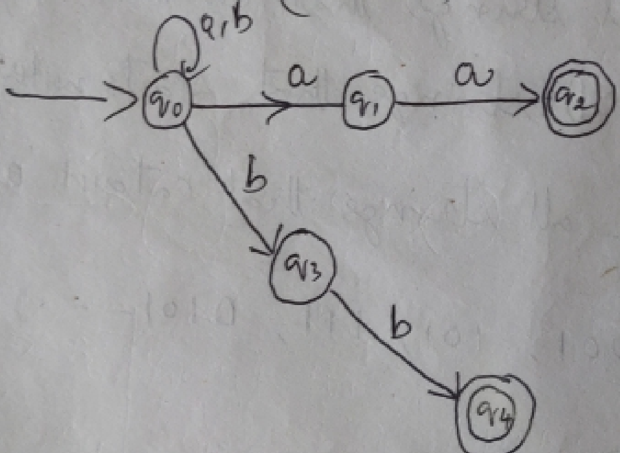
12;v) 

5) Construct a NFA that accepts a set of all strings over {0, 1} containing '101' as a substring.



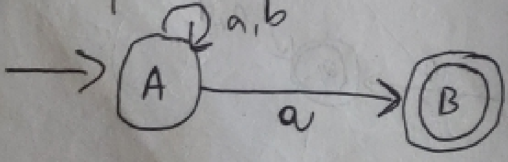
6) Construct a NFA that accepts set of all strings over {a, b} ending with aa or bb

$(aa+bb)^*$



Conversion from NFA to DFA

i) L_1 = { ends with an 'a' }

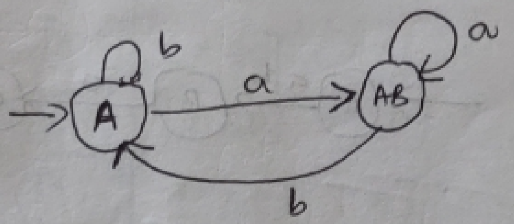


	a	b
→ A	{A,B}	{A}
B	{φ}	{φ}

Representation of state transition table of NFA.

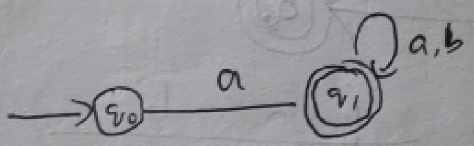
	a	b
→ A	[AB]	[A]
[AB]	[AB]	[A]
~~B~~	~~D~~	~~D~~
~~D~~	~~D~~	~~D~~

Representation of State transition table of DFA.



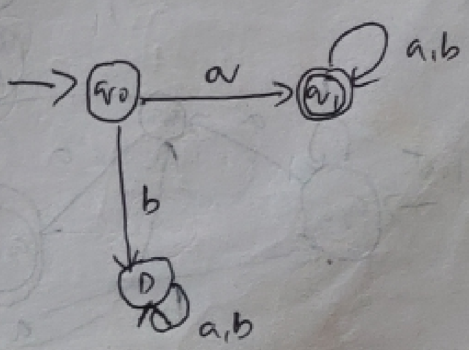
Equivalent of DFA

2) L_1 = { Starts with 'a' } over $\Sigma = \{a, b\}$



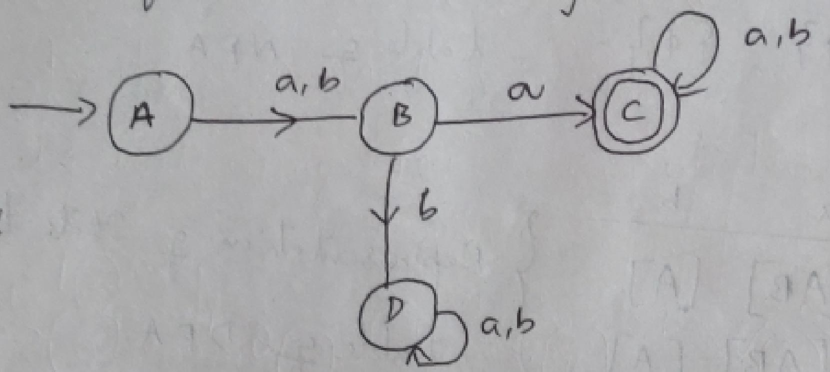
	a	b
→ q_0	q_1	φ
q_1	B	B

	a	b
→ q_0	q_1	D
q_1	q_1	q_1
D	D	D

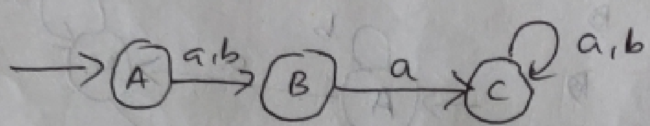


3) 14 L_1 = { set of all strings such that second symbol from LHS is 'a' }. Represent DFA & NFA

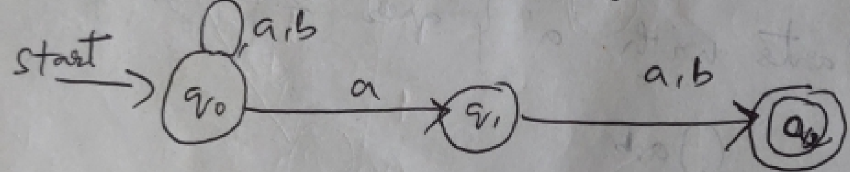
DFA :



NFA :



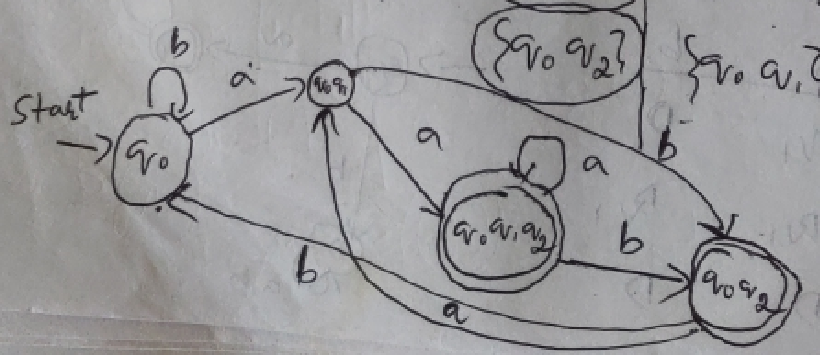
4) L = { set of strings such that second symbol from RHS is 'a' }.. Convert the NFA to Equivalent DFA.



NFA

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

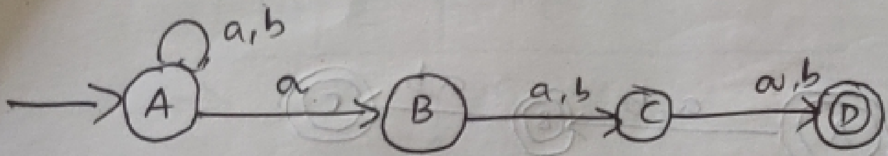
	a	b
→ q_0	$\{q_0\, q_1\}$	q_0
$\{q_0\, q_1\}$	$\{q_0\, q_1, q_2\}$	$\{q_0\, q_2\}$
$\{q_0\, q_1, q_2\}$	$\{q_0\, q_1, q_2\}$	$\{q_0\, q_2\}$
$\{q_0\, q_2\}$	$\{q_0\, q_1\}$	q_0



NFA

L : { all strings in which third symbol from RHS is 'a' }

L_1 : { aaa, aab, aba, abb - - - }

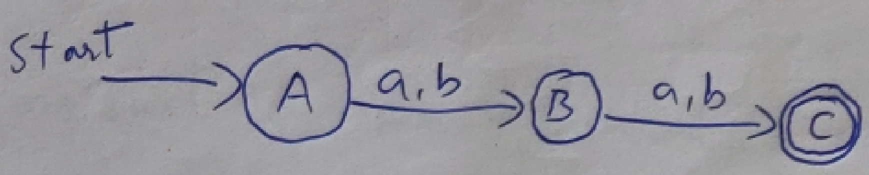


	a	b
→ A	{A,B}	{A}
B	{C}	{C}
C	{D}	{D}
* D	{φ}	{φ}

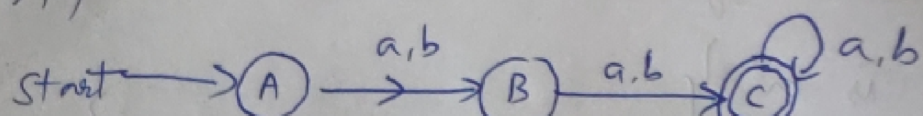
	a	b
[A]	[AB]	[A]
[AB]	[ABC]	[AC]
[AC]	[ABD]	[AD]
[AD]	[AB]	[A]
[ABC]	[ABCD]	[ACD]
[ABD]	[ABC]	[AC]
[ACD]	[ABD]	[AD]
[ABCD]	[ABCD]	[ACD]

6) NFA for strings of length i) exactly 2 ii) atleast 2 iii) atmost 2. over the language Σ = (a, b)

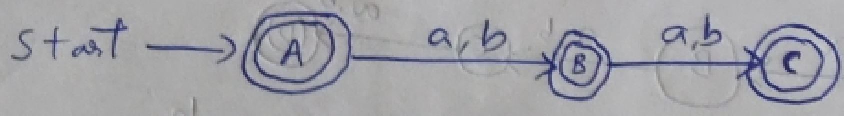
i) L_1 = { ab, aa, ba, bb }



12 ii)



iii)



onstruct a DFA Equivalent to $M = (\{q_1, q_2, q_3, q_4\},$
$\{0, 1\}, \delta, q_1, \{q_4\})$ where δ is given below:

states/Σ	0	1
$\rightarrow q_1$	q_1, q_2	q_1
q_2	q_3	q_2
q_3	q_4	q_4
q_4	—	q_3

Ans:-

states/Σ	0	1
$\rightarrow [q_1]$	$[q_1 q_2]$	$[q_1]$
$[q_1, q_2]$	$[q_1 q_2 q_3]$	$[q_1 q_2]$
$[q_1, q_2 q_3]$	$[q_1 q_2 q_3 q_4]$	$[q_1 q_2 q_4]$
$[q_1 q_2 q_3 q_4]$	$[q_1 q_2 q_3 q_4]$	$[q_1 q_2 q_3 q_4]$
$[q_1 q_2 q_4]$	$[q_1, q_2, q_3]$	$[q_1, q_2 q_3]$

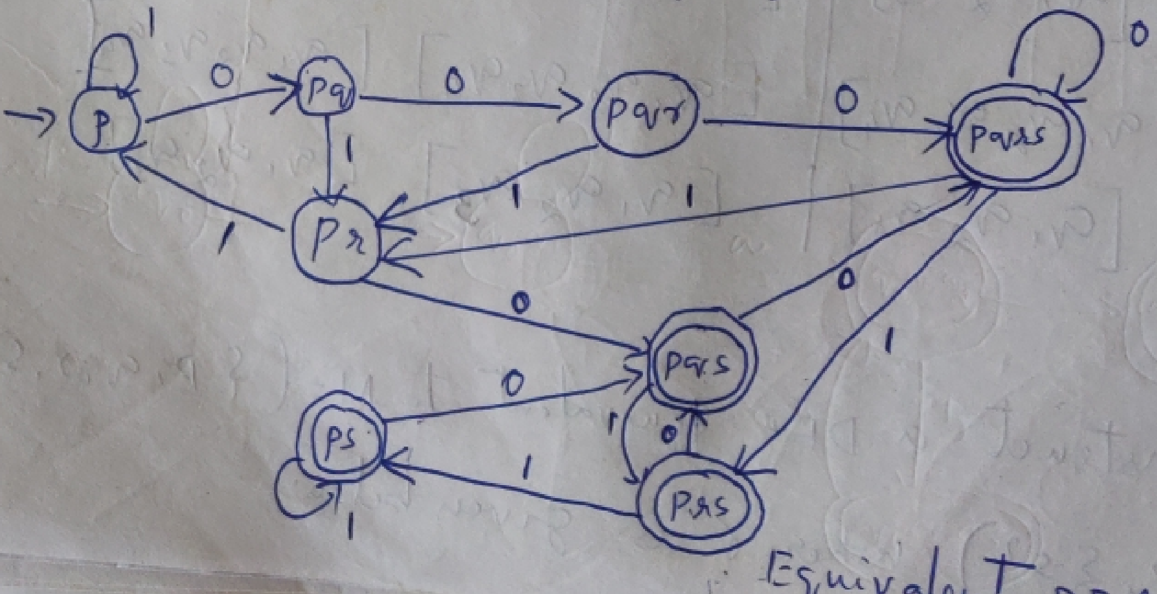
2) Construct a DFA Equivalent to $M = (\{P, q, r, s\}, \{0, 1\}$
$\delta, P, \{s\})$ where δ is given below:

10

states	0	1
→ p	{p, q}	{p}
q	{r}	{r}
r	{s}	φ
(s)	{s}	{s}

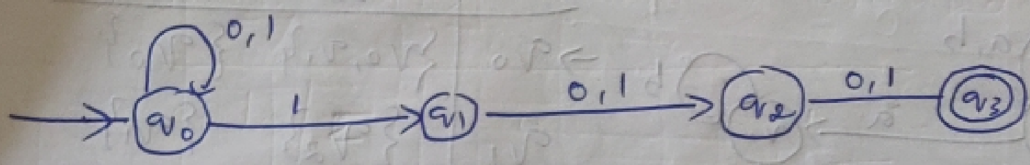
Ans :-

states	0	1
[p]	[pq]	[p]
[pq]	[pqr]	[pr]
[pqr]	[pqrs]	[pr]
[pr]	[pqs]	[p]
[pqrs]	[pqrs]	[prs]
[prs]	[pqs]	[pqs]
[pqs]	[pqrs]	[prs]
[ps]	[pqs]	[ps]



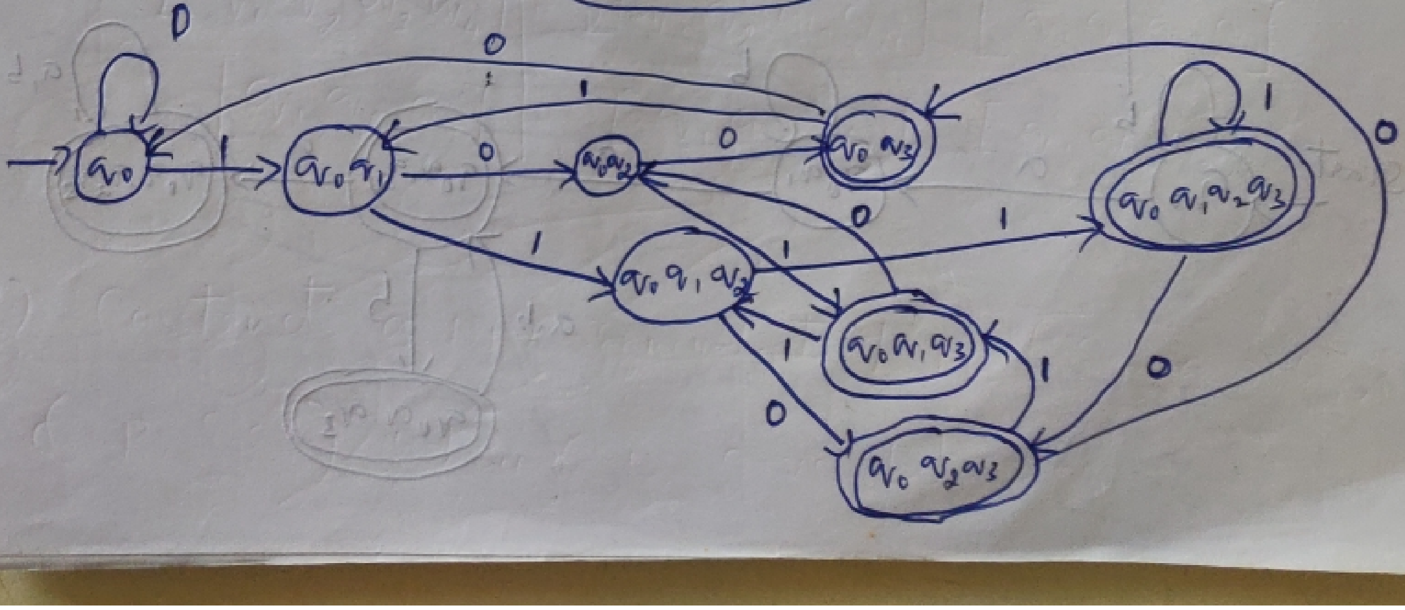
Equivalent DFA

Convert NFA to Equivalent DFA for the transition diagram, using subset construction method.

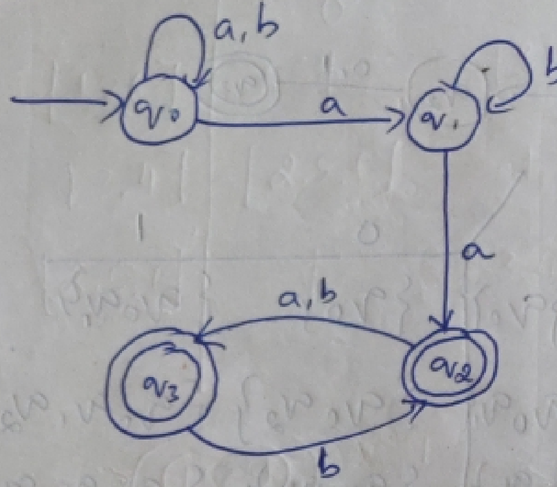


	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$
$*q_3$	$\{\phi\}$	$\{\phi\}$

	0	1
$\rightarrow\{q_0\}$	$\{q_0\}$	$\{q_0 q_1\}$
$\{q_0 q_1\}$	$\{q_0 q_2\}$	$\{q_0 q_1 q_2\}$
$\{q_0 q_2\}$	$\{q_0 q_3\}$	$\{q_0 q_1 q_3\}$
$\{q_0 q_3\}$	$\{q_0\}$	$\{q_0 q_1\}$
~~$\{q_0, q_1\}$~~	~~$\{q_0 q_2\}$~~	~~$\{q_0 q_1 q_2\}$~~
$\{q_0 q_1 q_2\}$	$\{q_0 q_2 q_3\}$	$\{q_0 q_1 q_2 q_3\}$
$\{q_0 q_1 q_3\}$	$\{q_0 q_2\}$	$\{q_0 q_1 q_2\}$
$\{q_0 q_2 q_3\}$	$\{q_0 q_3\}$	$\{q_0, q_1 q_3\}$
$\{q_0 q_1 q_2 q_3\}$	$\{q_0 q_2 q_3\}$	$\{q_0 q_1 q_2 q_3\}$



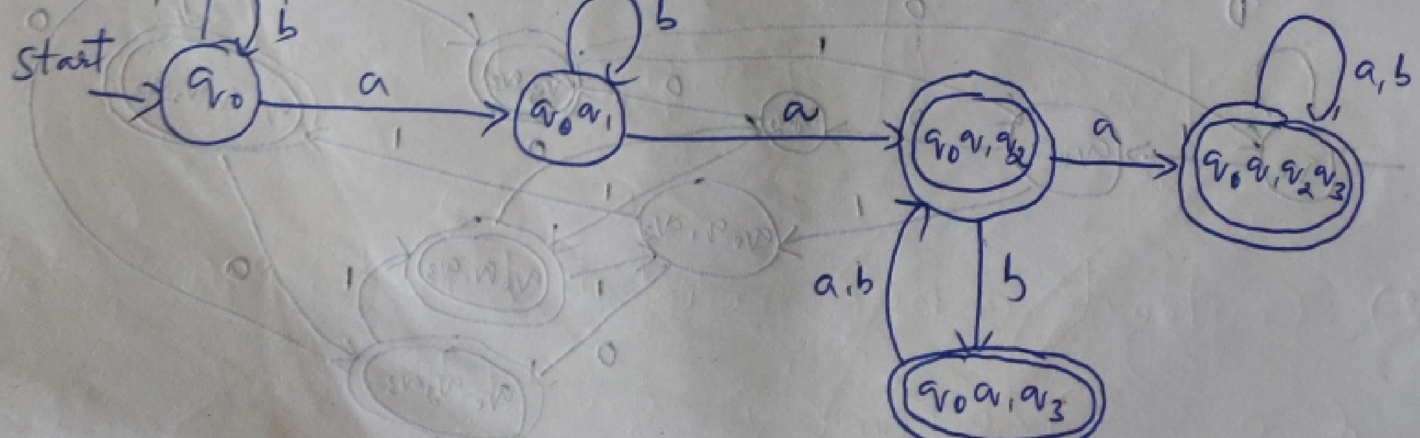
4) 12 Construct a Equivalent DFA for the NFA sh... below:



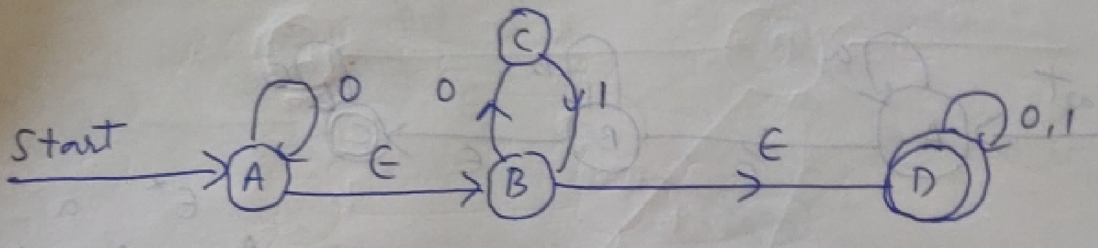
	a	b
$\to q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_1\}$
q_2^*	$\{q_3\}$	$\{q_3\}$
q_3^*	ϕ	$\{q_2\}$

Ans:-

P.S	N.S a	b
$\to q_0$	$\{q_0 q_1\}$	$\{q_0\}$
$\{q_0 q_1\}$	$\{q_0 q_1 q_2\}$	$\{q_0 q_1\}$
$\{q_0 q_1 q_2\}$	$\{q_0 q_1 q_2 q_3\}$	$\{q_0 q_1 q_3\}$
$\{q_0 q_1 q_2 q_3\}$	$\{q_0 q_1 q_2 q_3\}$	$\{q_0 q_1 q_2 q_3\}$
$\{q_0 q_1 q_3\}$	$\{q_0 q_1 q_2\}$	$\{q_0 q_1 q_2\}$



Conversion from ϵ-NFA to NFA



ϵ-closure (A) = {A, B, D}

	0	1
A	{ABCD}	{D}
B	{CD}	{D}
C	{ϕ}	{BD}
D	{D}	{D}

A ϵ* 0 ϵ*

A — A — A — A, B, D
A — B — C — C, D
A — D — D — D

A ϵ* 1 ϵ*

A — A — ϕ
A — B — ϕ
A — D — D → D

ϵ* 0 ϵ*

B — B — C — C
B — D — D — D

ϵ* 1 ϵ*

B — B — ϕ
B — D — D → D

ϵ* 0 ϵ*

C — C — C → ϕ

ϵ* 1 ϵ*

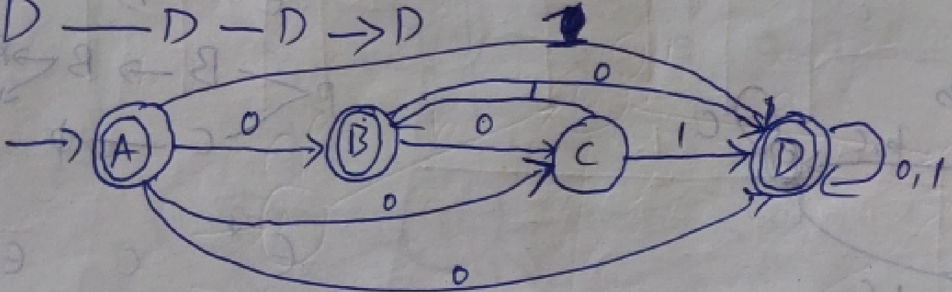
C — C — B — BD

ϵ* 0 ϵ*

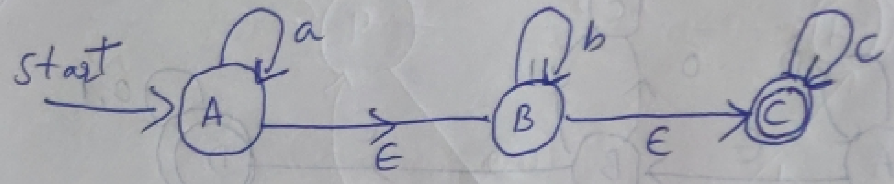
D — D — D → D

ϵ* 1 ϵ*

D — D — D → D



2) ε - NFA to NFA



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

ε* a ε*

A — A — A — A, B, C

A — B — φ

A — C — φ

ε* b ε*

A — A — φ

A — B — B — B, C

A — C — φ

ε* c ε*

A — A — φ

A — B — φ

A — C — C → C

ε* a ε*

C — C — φ

ε* b ε*

C → C → φ

ε* c ε*

C — C — C — C

ε* a ε*

B — B — φ

B — C — φ

ε* b ε*

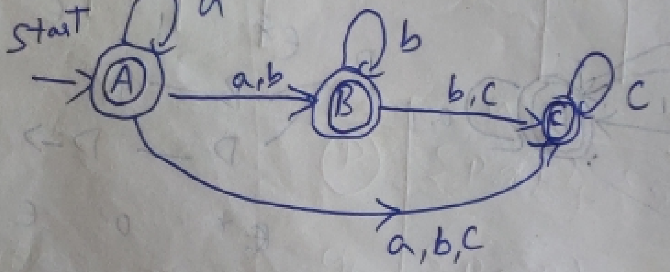
B — B → B → B, C

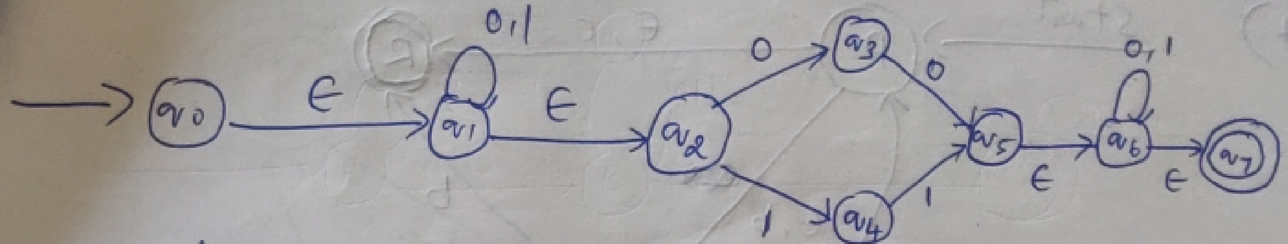
B — C → φ

ε* c ε*

B — B — φ

B — C — C → C

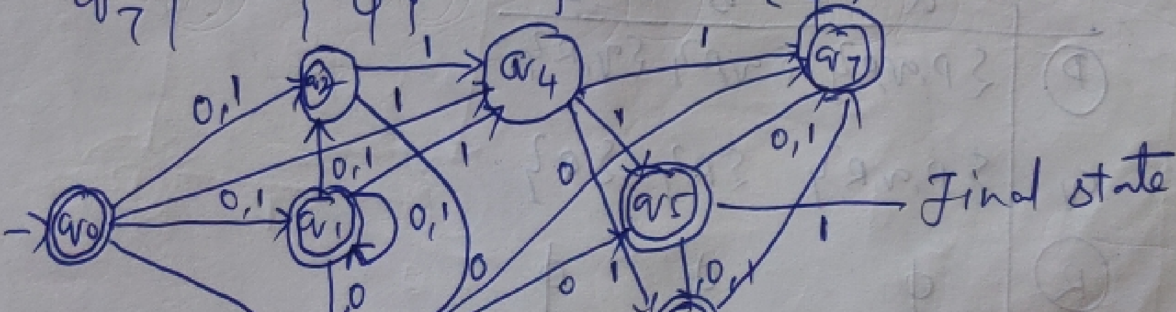




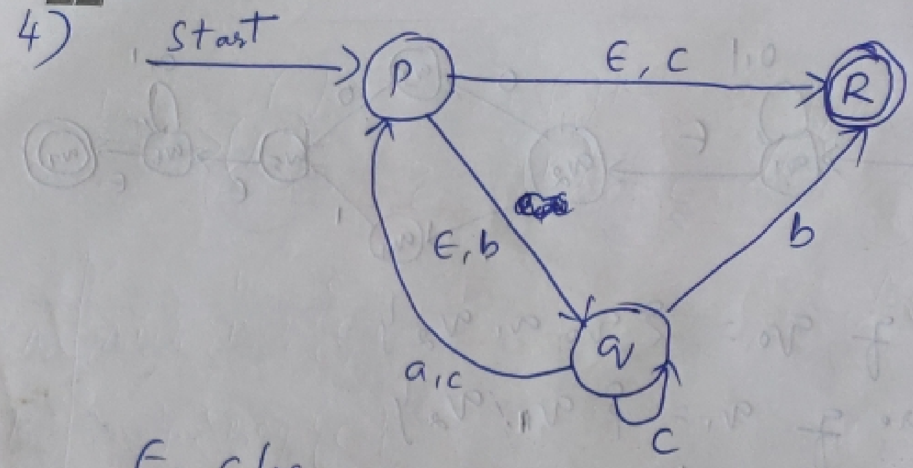
ϵ- closure of q_0 = $\{q_0\ q_1, q_2\}$

ϵ- closure of q_1 = $\{q_1, q_2\}$

	0	1
q_0	$\{q_1, q_2\ q_3\}$	$\{q_1, q_2\ q_4\}$
q_1	$\{q_1, q_2\ q_3\}$	$\{q_1\ q_2\ q_4\}$
q_2	$\{q_3\}$	$\{q_4\}$
q_3	$\{q_5\ q_6\ q_7\}$	$\{\phi\}$
q_4	$\{\phi\}$	$\{q_5\ q_6\ q_7\}$
q_5	$\{q_6\ q_7\}$	$\{q_6\ q_7\}$
q_6	$\{q_6\ q_7\}$	$\{q_6\ q_7\}$
q_7	$\{\phi\}$	$\{\phi\}$



4)



ε- Closure (P) = {P, q, R}

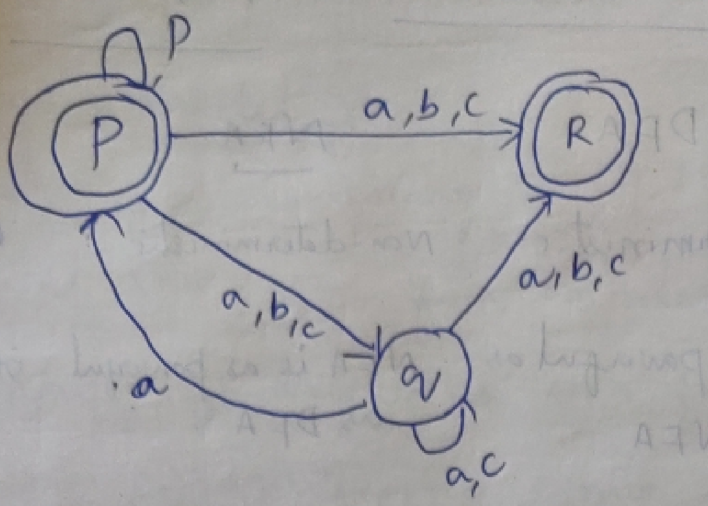
ε- Closure (q) = {q}

	ε*	a	ε*
P	P	φ	φ
	q	P	P
	R	φ	q
			R
q	q	P	P, q, r
R	R	φ	–

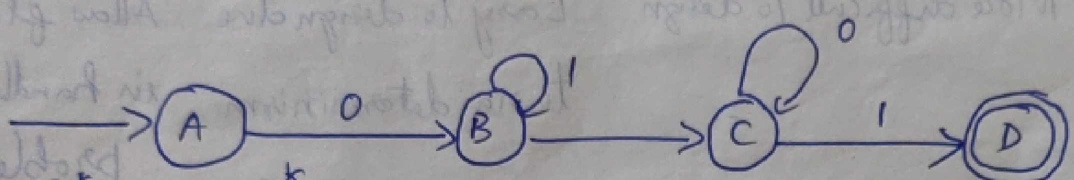
	ε*	b	ε*
P	P	q	q
	q	R	R
	R	φ	
q	q	R	R
R	R	φ	–

	ε*	c	ε*
P	P	R	R
	q	q	q
	R	φ	–
q	q	q	q
R	R	φ	

	a	b	c
P	{P, q, R}	{q, R}	{q, R}
q	{P, q, R}	{R}	{q}
R	φ	φ	φ



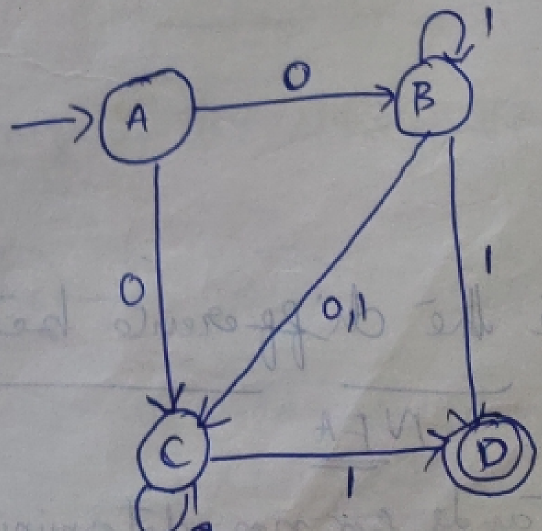
5)



	ϵ^*	0	ϵ^*
A	A	B	B, C
B	B, C	ϕ, C	ϕ, C
C	C	C	C
D	D	ϕ	ϕ

	ϵ^*	1	ϵ^*
A	A	ϕ	ϕ
B	B, C	B, D	B, C, D
C	C	D	D
D	D	ϕ	ϕ

	0	1
$\rightarrow$ A	{B, C}	ϕ
B	{C}	{B, C, D}
C	{C}	{D}
D	ϕ	ϕ



write the difference between DFA, NFA and ε-NFA

Parameter	DFA	NFA	ε-NFA
Transition	Deterministic	Non-deterministic	Non-Determin
power	FA is powerful as an NFA	NFA is as powerful as DFA	It's powerful as FA
Design	More difficult to design	Easy to design due to non determinism	Allow flexibili in handling N problem.
Acceptance	It is easy to find whether W∈L as transition are deterministic	It is difficult to find whether W∈L as there are several paths. Back tracking is required to explore several parallel paths.	A transitio is useful i constructing a composite FA with respect to union, concatenation & star.

write the difference between DFA & NFA

NFA	DFA
NFA stands for non-deterministic finite automata	DFA stands for deterministic finite automata.

plications of Finite Automata :- Finite automata are used as a model for

i) Software for designing digital circuits.

ii) lexical analyzer of a Compiler.

iii) Searching for keywords in a file or on the web.

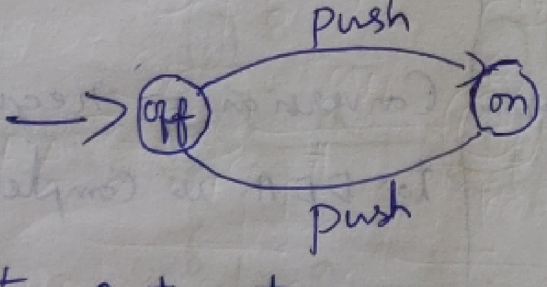
iv) Software for verifying finite state systems such as Communication protocols.

v) Computer security.

vi) Computer graphics & fractal Compression.

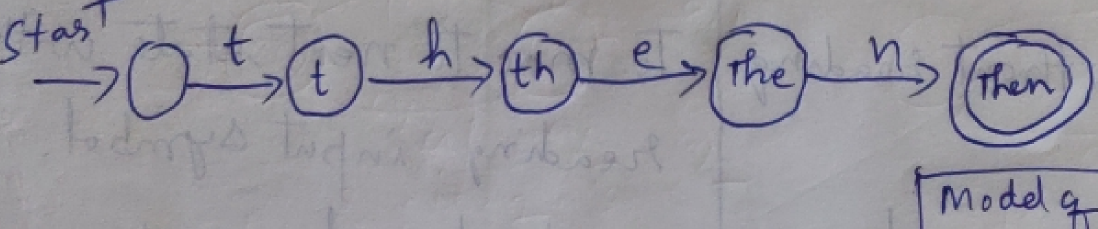
vii) Virtual Currency.

i) Finite Automation modelling an on/off digital switch.



Model of Computation:
A program.

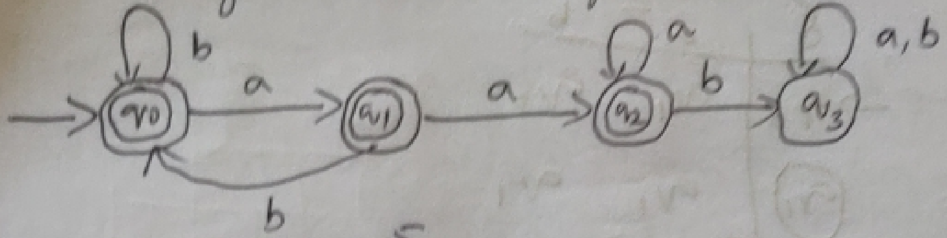
ii) Finite Automaton recognizing the string, then



Model of description:
Specification

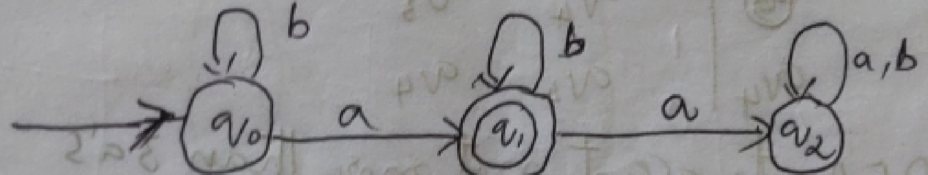
Backtracking is not allowed in NFA	Backtracking is allowe DFA.
Easy to design due to non-deterministic transition.	More difficult to design du deterministic transition
It permits empty string transition	It cannot permit empty string transition.
It requires less memory space	It requires more memory space.
Every NFA is not a DFA	Every DFA is a NFA
String is accepted by NFA, if one of the possible transition ends in final state	String accepted by DFA, if it is transition into final state.
Conversion of regular expression to NFA is easy	Conversion of regular expression to DFA is complex.
In The NFA, we can move into next state without reading input symbol.	In the DFA, it is not possible To move to next state without reading input symbol.
Sometimes practical implementation is not feasible	practical implementation is feasible.

Obtain a DFA to accept strings of a's and b's except those containing the substring abb.



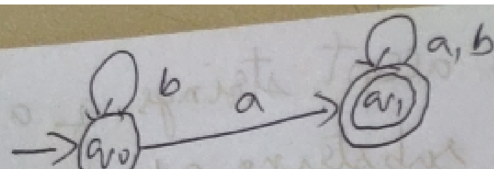
δ	a	b
Initial → q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_3
q_3	q_3	q_3

2) obtain a DFA to accept strings of a's & b's having exactly one a, atleast one a, not more than three a's



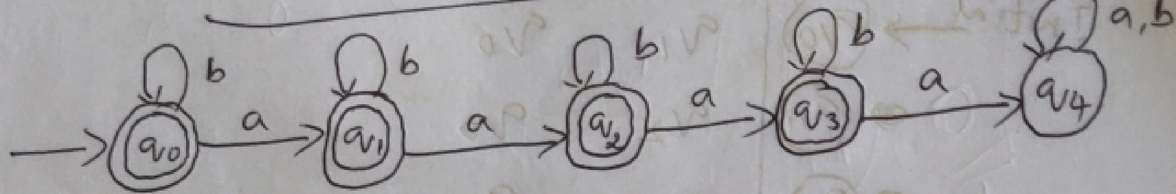
δ	a	b
→ q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_2	q_2

DFA to accept exactly one a.



δ	a	b
→q_0	q_1	q_0
(q_1)	q_1	q_1

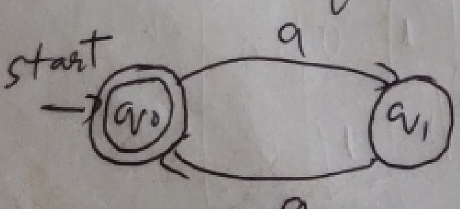
DFA to atleast one a



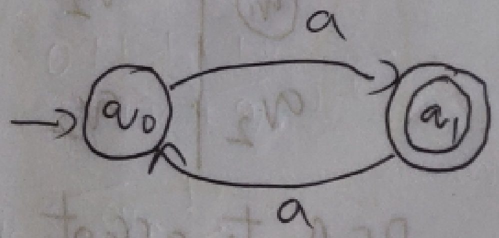
δ	a	b
→(q_0)	q_1	q_0
(q_1)	q_2	q_1
(q_2)	q_3	q_2
(q_3)	q_4	q_3
q_4	q_4	q_4

DFA to accept not more than 3 a's

3) obtain a DFA to accept even number of a's and odd number of a's



even number of a's



odd number of a's

Obtain a DFA to accept strings of a having

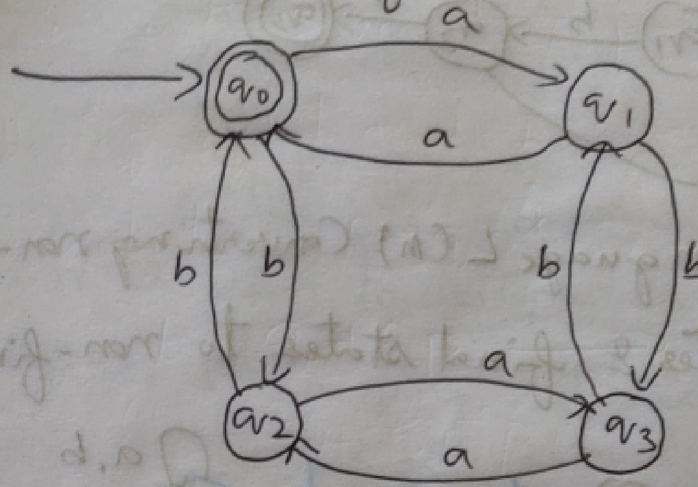
i) even number of a's and b's

ii) Even number of a's & odd number of b's

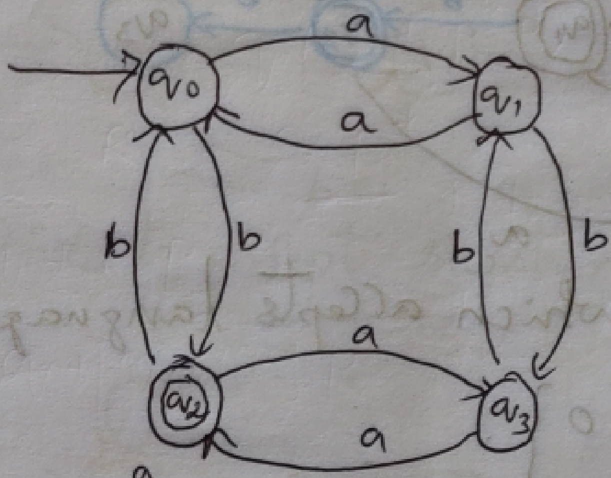
iii) odd number of a's & Even number of b's

iv) odd number of a's & odd number of b's

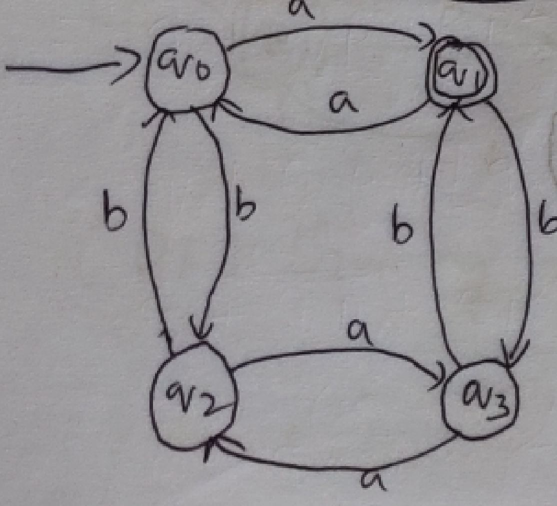
i)



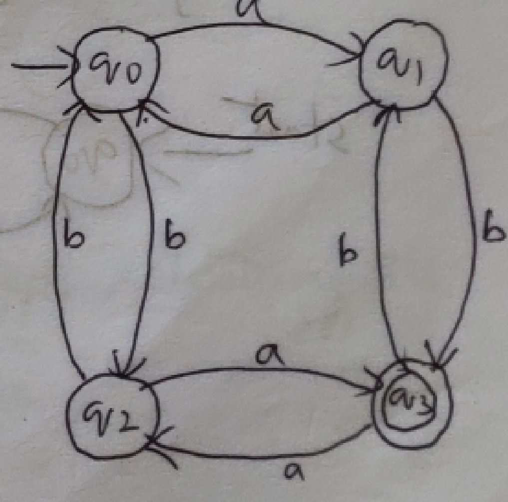
ii)



iii)

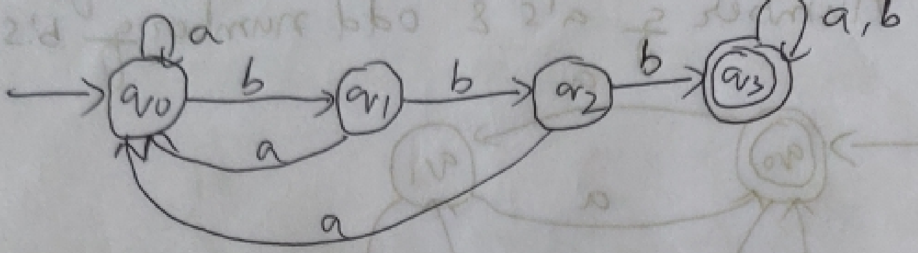


iv)

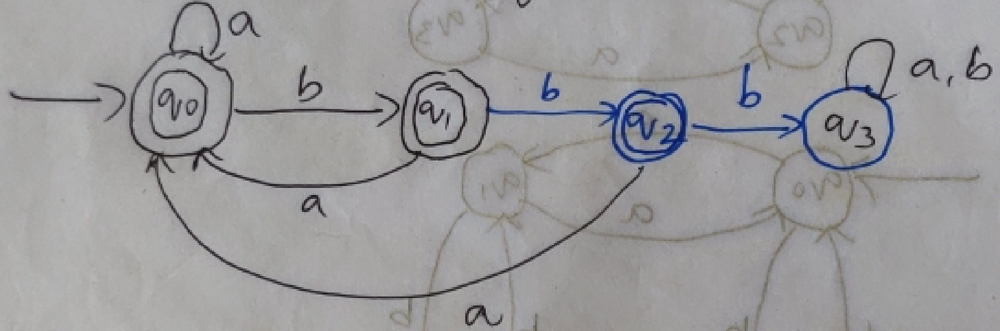


5) obtain a DFA that accepts the language L = { w | w ∈ {a, b}* } and w does not contain 3 consecuti

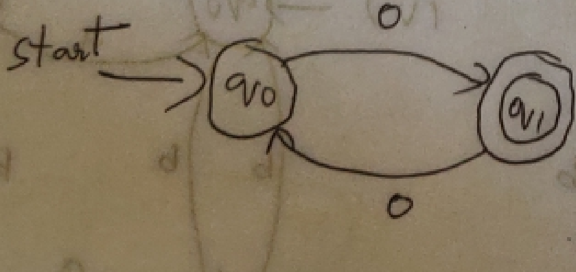
We first consider a language L(M) = { w | w ∈ {a, b}* and w contains 3 consecutive b's



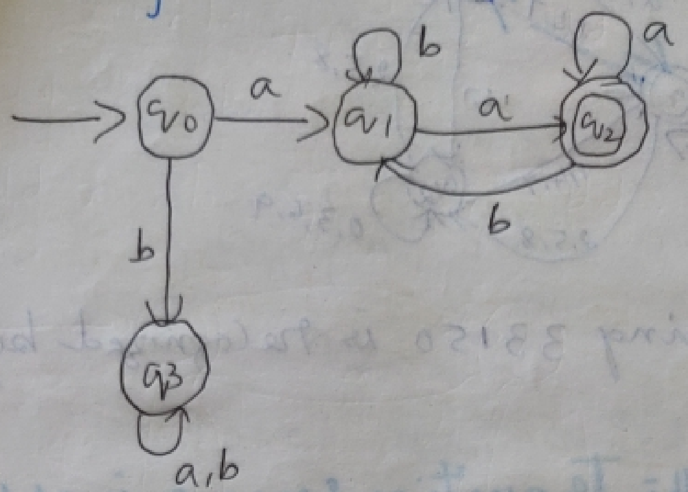
Now, we can get language L(M) converting non-final states to final states & final states to non-final states.



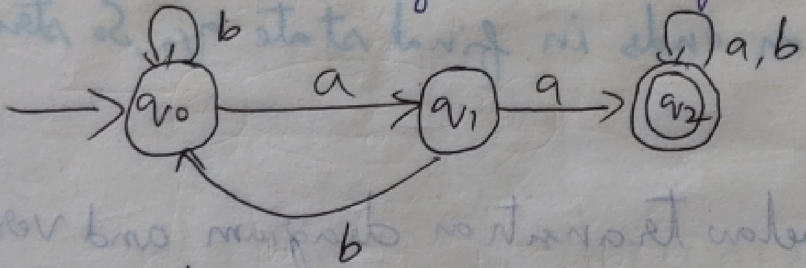
6) Design a DFA which accepts language L = { 0, 000, 00000 ---- } over { 0 }



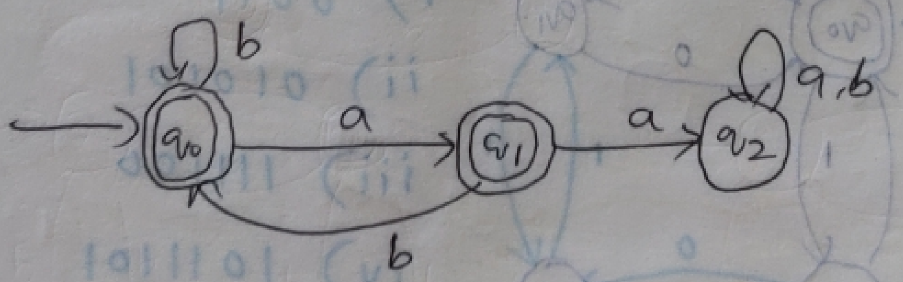
(O)btain a DFA to accept the language $L = \{awa \mid w \in (a+b)^*\}$.



8) obtain a DFA to accept strings of a's and b's except those containing substring aa.

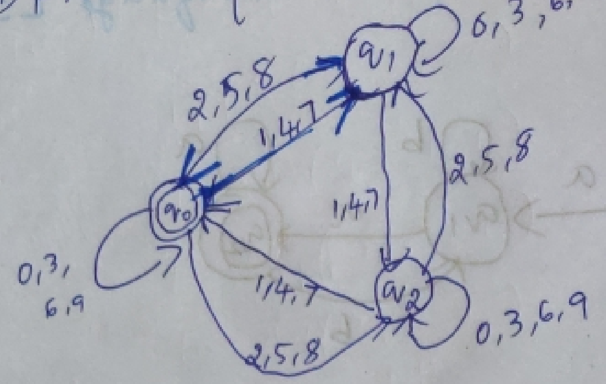


DFA to accept substring aa



DFA showing not to accept substring aa

9) Let DFA $M = \{Q, \Sigma, \delta, q_0, F\}$ is shown.



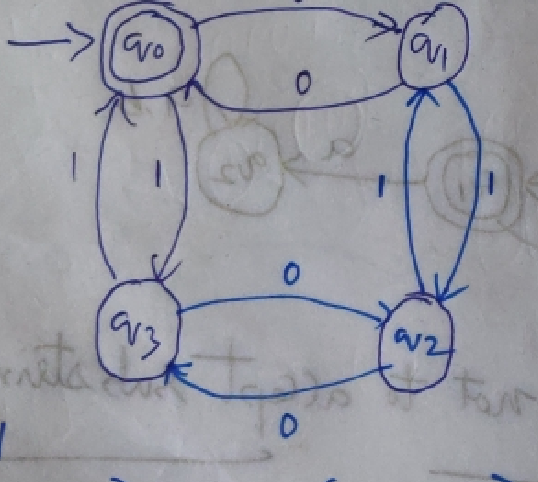
Check that string 33150 is recognized by above DFA or not?

For the string 33150 the transition sequence is as follows:

$$\rightarrow (q_0) \xrightarrow{3} (q_0) \xrightarrow{3} (q_0) \xrightarrow{1} (q_1) \xrightarrow{5} (q_0) \xrightarrow{0} ((q_0))$$

Since, transition ends in final state q_0, so string is recognized.

10) Consider below transition diagram and verify whether the following strings will be accepted or not?



i) 0011
ii) 010101
iii) 111100
iv) 1011101

i) 0011

$\delta(q_0, 0011) \vdash \delta(q_1, 011)$

$\vdash \delta(q_0, 11) \vdash (q_3, 1) \vdash q_0$

$\therefore$ 0011 is accepted.

ii) 010101

$(q_0, 010101) \vdash \delta(q_1, 10101)$
$\vdash \delta(q_2, 0101)$
$\vdash \delta(q_3, 101)$
$\vdash \delta(q_0, 01)$
$\vdash \delta(q_1, 1)$
$\vdash q_2$

$\therefore$ 010101 is not accepted.

iii) 111100

$\delta(q_0, 111100) \vdash \delta(q_3, 11100)$
$\vdash \delta(q_0, 1100)$
$\vdash \delta(q_3, 100)$
$\vdash \delta(q_0, 00)$
$\vdash \delta(q_1, 0)$
$\vdash q_0$

$\therefore$ 111100 is accepted.

iv) 1011101

$\delta(q_0, 1011101) \vdash \delta(q_3, 011101)$
$\vdash \delta(q_2, 11101)$
$\vdash \delta(q_1, 1101)$
$\vdash \delta(q_2, 101)$
$\vdash \delta(q_1, 01)$
$\vdash \delta(q_0, 1)$
$\vdash q_3$ $\therefore$ 1011101 is not accepted.

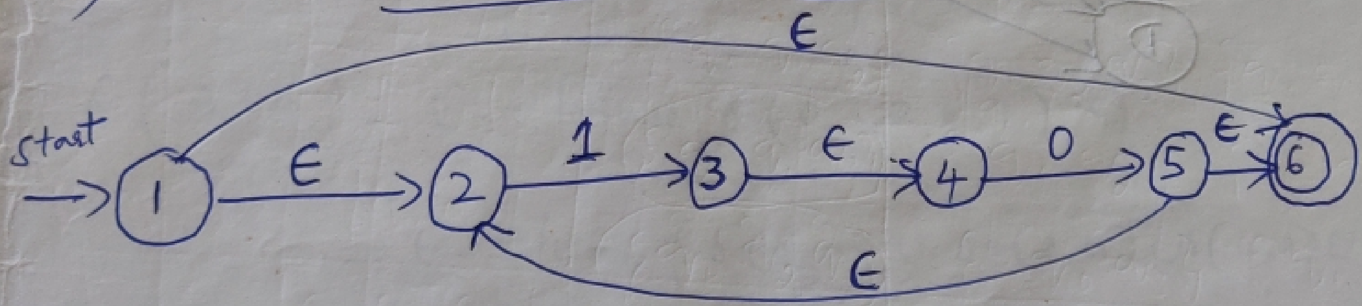
version from ε-NFA to DFA :- the steps for the conversion is as follows:

i) Find the ε-closure of all states from the NFA

ii) Draw a transition table.

iii) start computing the DFA table from the first state and take the resulting states as the next state in each step.

i) Convert ε-NFA to equivalent DFA.



ε-closure(1) = {1, 2, 6}

ε-closure(2) = {2}

ε-closure(3) = {3, 4}

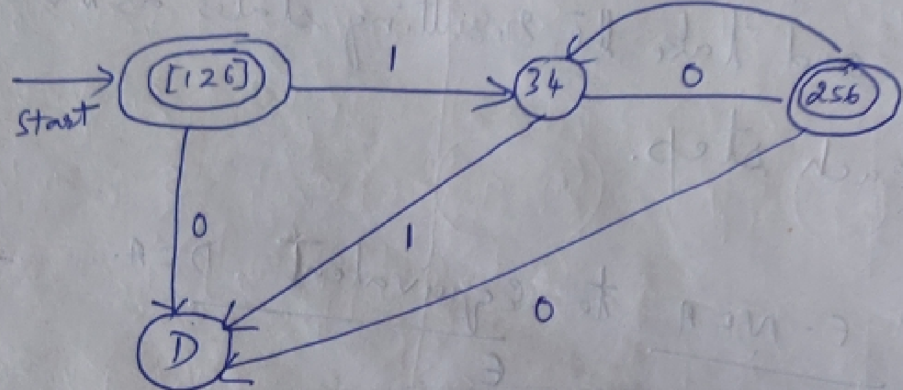
ε-closure(4) = {4}

ε-closure(5) = {2, 5, 6}

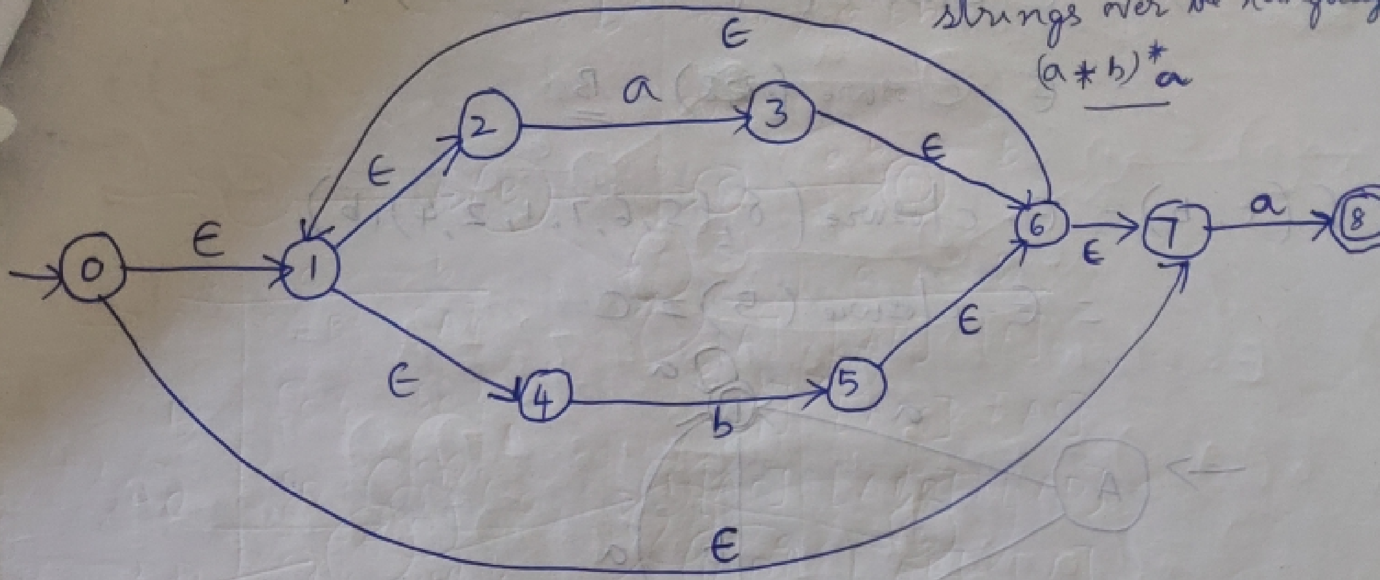
ε-closure(6) = {6}

2) 14 Draw the transition table

	0	1
[1,2,6]*	ϕ	[3 4]
[3,4]	[2,5,6]	ϕ
[2,5,6]*	ϕ	[3,4]



Convert ϵ-NFA to DFA for the state diagram over the input alphabet $\Sigma = (a, b)$ that should accepts the strings over the language $(a*b)^* a$ **13**



ϵ-closure(0) = $\{0, 1, 2, 4, 7\}$ = A (New state in DFA)

$\delta(A, a) = \epsilon\text{-closure}(\delta(0,1,2,4,7), a)$

$= \epsilon\text{-closure}(\delta(0,a) \cup \delta(2,a) \cup \delta(4,a) \cup \delta(7,a))$

$= \epsilon\text{-closure}(3,8) = \{3,6,1,7,2,4,8\} = B$

$\delta(A, b) = \epsilon\text{-closure}(\delta(0,1,2,4,7), b)$

$= \epsilon\text{-closure}(5) = \{5,6,7,1,2,4\} = C$

$\delta(B, a) = \epsilon\text{-closure}(\delta(3,6,1,7,2,4,8), a)$

$= \epsilon\text{-closure}(3,8) = B$

$\delta(B, b) = \epsilon\text{-closure}(\delta(3,6,1,7,2,4,8), b)$

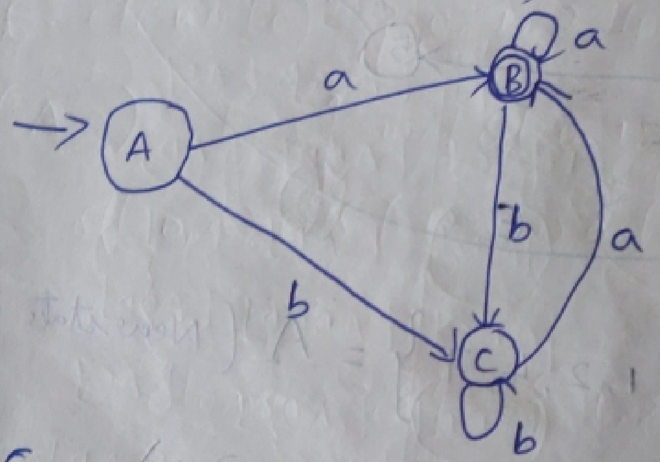
$= \epsilon\text{-closure}(5) = C$

$\delta(C, a) = \epsilon\text{-closure}(\delta(5,6,7,1,2,4), a)$

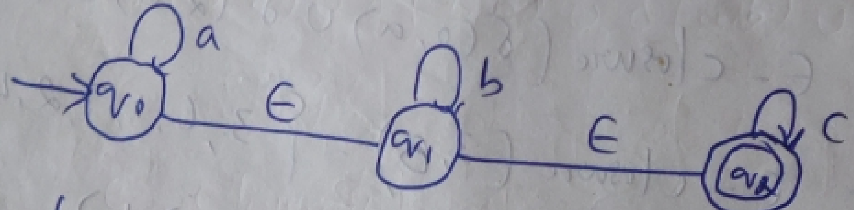
$= \epsilon\text{-closure}(3,8) = \underline{\underline{B}}$

$\delta(C, b) = \epsilon\text{-closure}(\delta,(5,6,7,1,2,4), b)$

$= \epsilon\text{-closure}(5) = C$



2) ϵ-NFA to DFA



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

$C = \{q_2\}$

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

$= \epsilon\text{-closure}(q_0) = A$

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

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

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

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

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

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

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

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

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

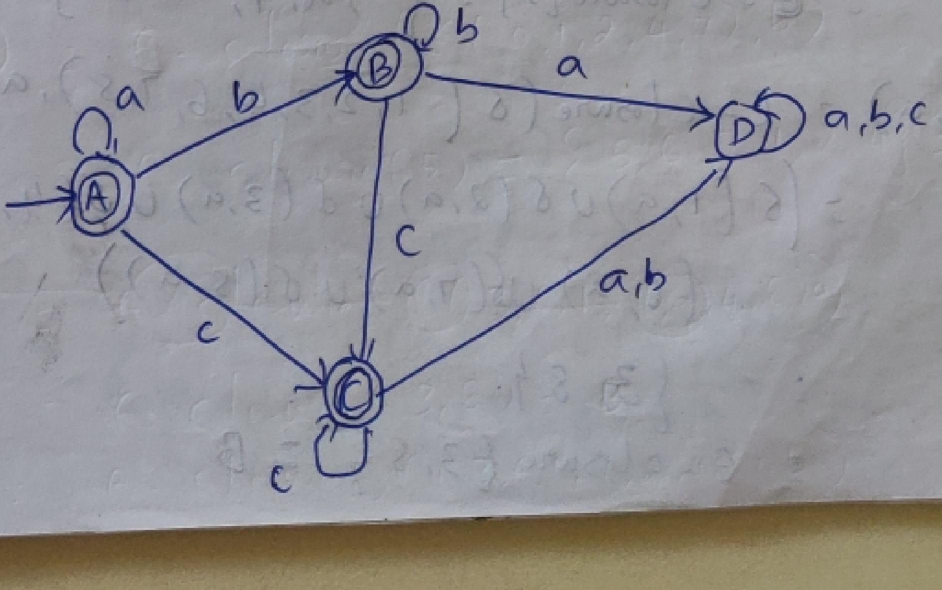
$= \epsilon\text{-closure}(q_2) = C$

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

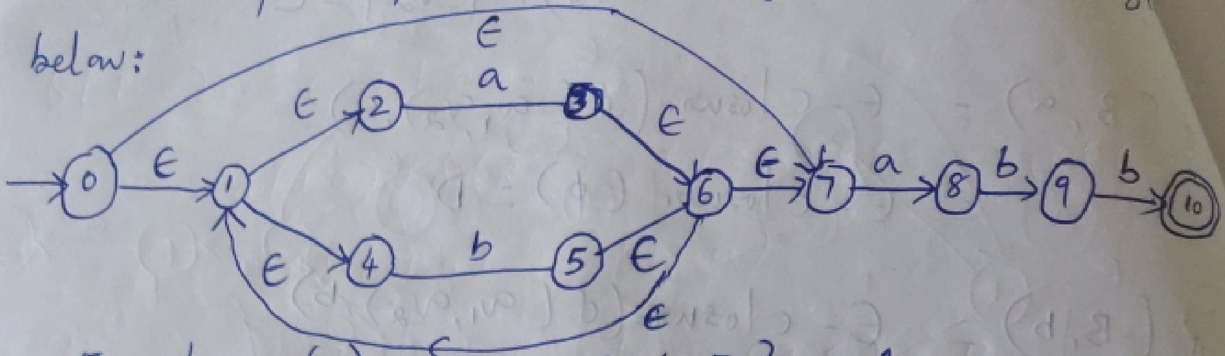
$\delta(C, a) = \epsilon\text{-closure}(\delta(q_2, a)) = \phi = D$

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

$\delta(C, c) = \epsilon\text{-closure}(\delta(q_2, c) = \phi = D$



3) Convert ε-NFA to DFA for the language (a ... over the input alphabet Σ = {a, b} & ε-NFA is g... below:



ε-closure(0) = {0, 1, 2, 4, 7} = A

δ(A, a) = ε-closure(δ(0, 1, 2, 4, 7), a)

= δ((0, a) ∪ δ(1, a) ∪ δ(2, a) ∪ δ(4, a) ∪ δ(7, a))

= {3, 8}

= ε-closure {3, 8} = {1, 2, 3, 4, 6, 7, 8} = B

δ(A, b) = ε-closure(δ(0, 1, 2, 4, 7), b)

= δ(0, b) ∪ δ(1, b) ∪ δ(2, b) ∪ δ(4, b) ∪ δ(7, b)

= {5}

= ε-closure {5} = {1, 2, 4, 5, 6, 7} = C

δ(B, a) = ε-closure(δ(1, 2, 3, 4, 6, 7, 8), a)

= (δ(1, a) ∪ δ(2, a) ∪ δ(3, a) ∪ δ(4, a) ∪ δ(6, a) ∪ δ(7, a) ∪ δ(8, a))

= {3, 8}

= ε-closure {3, 8} = B

$\delta(B, b) = \epsilon\text{-closure}(\delta(1,2,3,4,6,7,8), b)$

$= \delta(1,b) \cup \delta(2,b) \cup \delta(3,b) \cup \delta(4,b) \cup \delta(6,b) \cup$
$\delta(7,b) \cup \delta(8,b)$

$= \epsilon\text{-closure}(5,9)$

$= \{1,2,4,5,6,7,9\} = D$

$\delta(C, a) = \epsilon\text{-closure}(\delta(1,2,4,5,6,7), a)$

$= \delta(1,a) \cup \delta(2,a) \cup \delta(4,a) \cup \delta(5,a) \cup \delta(6,a)$
$\cup \delta(7,a)$

$= \epsilon\text{-closure}(3,8) = \underline{\underline{B}}$

$\delta(C, b) = \epsilon\text{-closure}(\delta(1,2,4,5,6,7), b)$

$= \delta(1,b) \cup \delta(2,b) \cup \delta(4,b) \cup \delta(5,b) \cup \delta(6,b)$
$\cup \delta(7,b)$

$= \epsilon\text{-closure}(5) = \underline{\underline{C}}$

$\delta(D, a) = \epsilon\text{-closure}(\delta(1,2,4,5,6,7,9), a)$

$= \delta(1,a) \cup \delta(2,a) \cup \delta(4,a) \cup \delta(5,a) \cup$
$\delta(6,a) \cup \delta(7,a) \cup \delta(9,a)$

$= \epsilon\text{-closure}(3,8) = \underline{\underline{B}}$

$\delta(D, b) = \epsilon\text{-closure}(\delta(1,2,4,5,6,7,9), b)$

$= \delta(1,b) \cup \delta(2,b) \cup \delta(4,b) \cup \delta(5,b) \cup \delta$
$(6,b) \cup \delta(7,b) \cup \delta(9,b)$

ϵ- Closure$(5, 10) = E$

$= \{1, 2, 3, 5, 6, 7, 10) = E$

$\delta(E, a) = \epsilon\text{-closure}(\delta(1, 2, 3, 5, 6, 7, 10), a)$

$= \delta(1, a) \cup \delta(2, a) \cup \delta(3, a) \delta(5, a) \cup \delta(6, a)$

$\cup \delta(7, a) \cup \delta(10, a)$

$= (3, 8) = \epsilon\text{-closure}(3, 8) = \underline{\underline{B}}$

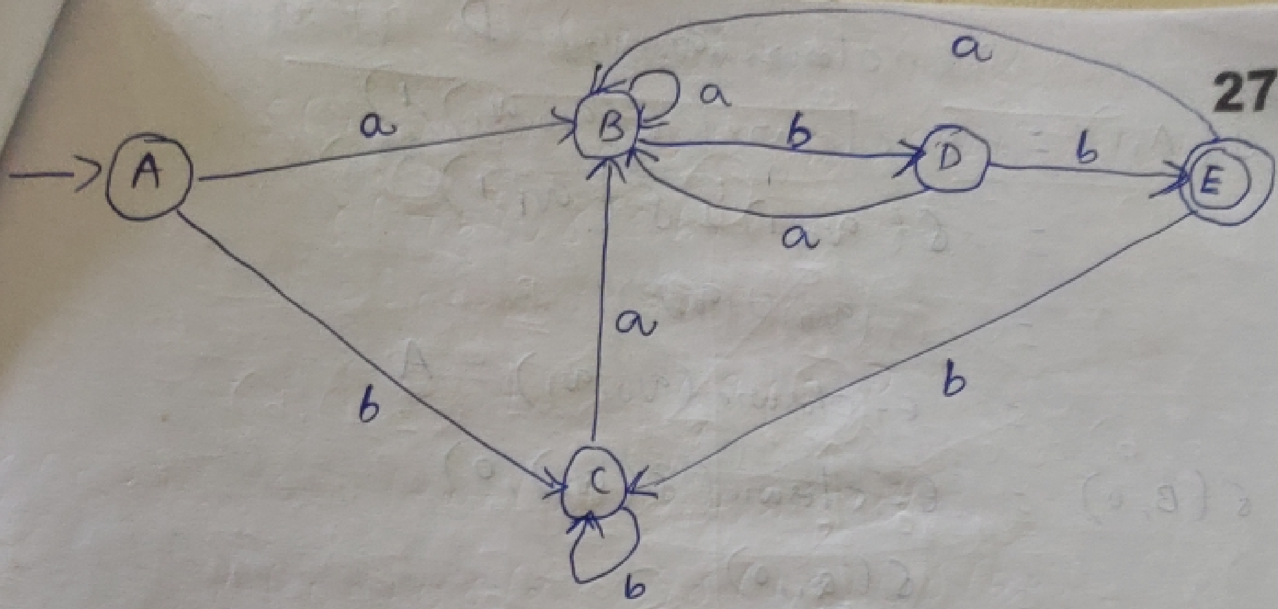
$\delta(E, b) = \epsilon\text{-closure}(\delta(1, 2, 3, 5, 6, 7, 10), b)$

$= \delta(1, b) \cup \delta(2, b) \cup \delta(3, b) \cup \delta(5, b) \cup \delta(6, b)$

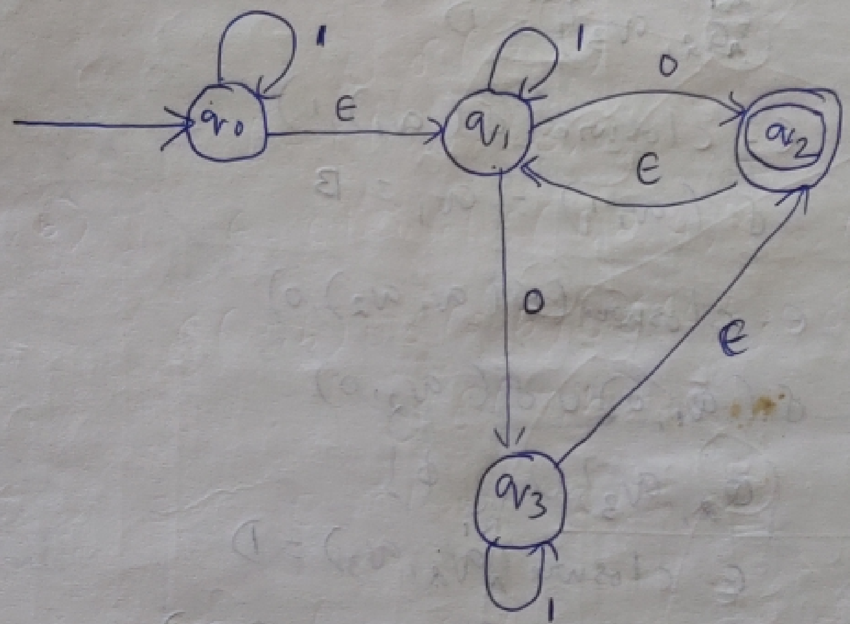
$\cup \delta(7, b) \cup \delta(10, b)$

$= \epsilon\text{-closure}(5) = C$

NFA state	DFA state	I/P a	b
{0, 1, 2, 4, 7}	A	B	C
{1, 2, 3, 4, 6, 7, 8}	B	B	D
{1, 2, 4, 5, 6, 7}	C	B	C
{1, 2, 4, 5, 6, 7, 9}	D	B	E
{1, 2, 3, 5, 6, 7, 10}	E	B	C



4)



ϵ- closure $(q_0) = \{q_0 q_1\} = A$

ϵ- closure $(q_1) = \{q_1\} = B$

ϵ- closure $(q_2) = \{q_1 q_2\} = C$

ϵ- closure $(q_3) = \{q_1 q_2 q_3\} = D$

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

$= \delta(q_0, 0) \cup \delta(q_1, 0)$

$= (\phi) \cup (q_2 q_3)$

ϵ - closure $(q_1, q_2, q_3) = D$

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

$= \delta(q_0, 1) \cup \delta(q_1, 1)$

$= (q_0) \cup (q_1)$

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

$\delta(B, 0) = \epsilon\text{-closure}(\delta(q_1), 0)$

$= \delta(q_1, 0)$

$= \{q_1, q_2, q_3\} = D$

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

$= \delta(q_1, 1) = q_1 = B$

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

$= \delta(q_1, 0) \cup \delta(q_2, 0)$

$= \{q_2, q_3\} \cup \{\phi\}$

$= \epsilon\text{-closure}(q_1, q_2, q_3) = D$

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

$= \delta(q_1, 1) \cup (q_2, 1)$

$= \{q_1\} \cup \{\phi\}$

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

$\delta(D, 0) = \epsilon\text{-Closure}(\delta(D), 0)$

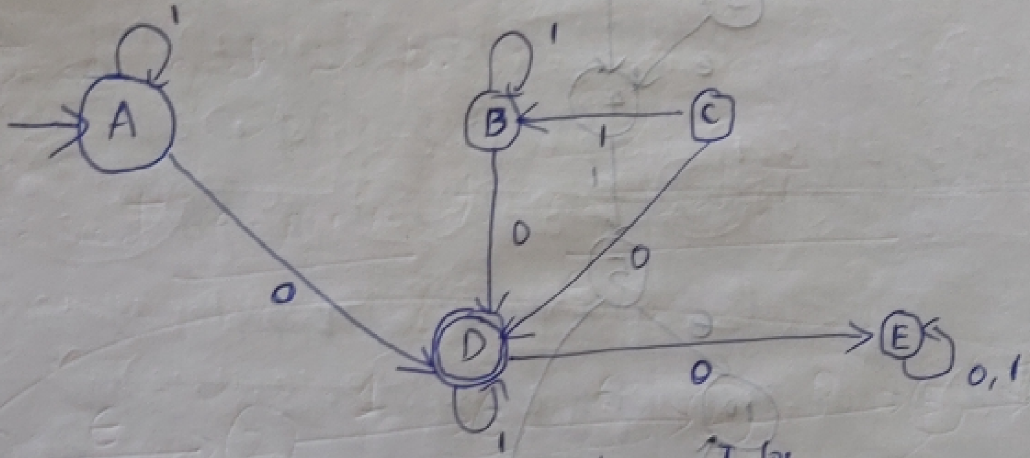
$= \delta(q_2, q_3), 0)$

$= \delta(q_2, 0) \cup \delta(q_3, 0)$

$= \{\phi\} \cup \{\phi\} = \phi = E$

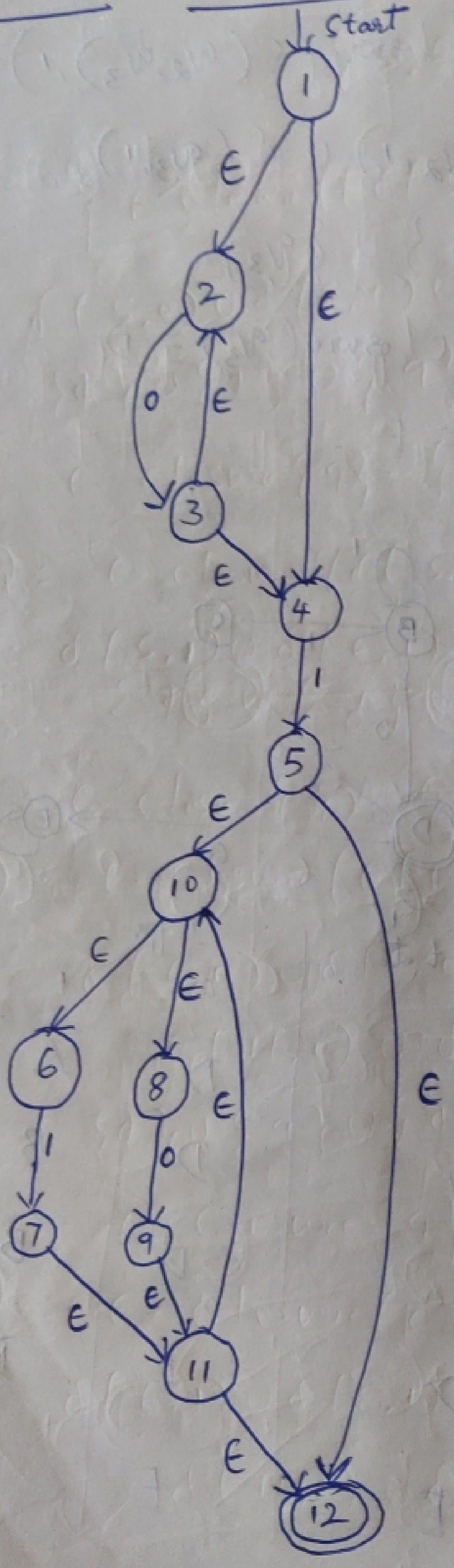
$$\delta(D,1) = \epsilon\text{-closure}(\delta(D),1)$$
$$= \epsilon\text{-closure}(\delta(q_2,q_3),1)$$
$$= \delta(q_2,1) \cup \delta(q_3,1)$$
$$= \{\phi\} \cup \{q_3\}$$
$$= \epsilon\text{-closure}(q_3) = D$$

$$\delta(E,0) = \delta(E,1) = E$$



NFA state	DFA state	I/ps 0	1
$A = \{q_0, q_1\}$	A	D	A
$B = \{q_1\}$	B	D	B
$C = \{q_1, q_2\}$	C	D	B
$D = \{q_1 q_2 q_3\}$	D	E	D
$E = \{q_4\}$	E	E	E

Convert ε-NFA to Equivalent DFA :-



$\epsilon\text{-closure}\{1\} = \{1, 2, 4\} = A$

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

$= \delta((1, 2, 4), 0))$

$= \delta((1, 0) \cup \delta(2, 0) \cup \delta(4, 0))$

$= \{\phi\} \cup \{3\} \cup \{\phi\}$

$= \epsilon\text{-closure}\{3\} = \{3, 2, 4\} = \{2, 3, 4\} = B$

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

$= \delta((1, 2, 4), 1))$

$= \delta((1, 1) \cup \delta(2, 1) \cup \delta(4, 1))$

$= \{\phi\} \cup \{\phi\} \cup \{5\}$

$= \epsilon\text{-closure}\{5\} = \{5, 10, 12, 6, 8\}$

$= \{5, 6, 8, 10, 12\} = C$

$\delta(B, 0) = \epsilon\text{-closure}((\delta(2, 3, 4), 0))$

$= \delta((2, 0) \cup \delta(3, 0) \cup \delta(4, 0))$

$= \{3\} \cup \{\phi\} \cup \{\phi\}$

$= \epsilon\text{-closure}\{3\} = \underline{\underline{B}}$

$\delta(B, 1) = \epsilon\text{-closure}((\delta(2, 3, 4), 1))$

$= \delta((2, 1) \cup \delta(3, 1) \cup \delta(4, 1))$

$= \{\phi\} \cup \{\phi\} \cup \{5\}$

ϵ- closure$\{5\} = \{5, 10, 12, 6, 8\}$
$= \{5, 6, 8, 10, 12\} = C$

$\delta(C, 0) = \{\epsilon\text{-closure}((\delta(5, 6, 8, 10, 12), 0))\}$
$= \delta(5, 0) \cup \delta(6, 0) \cup \delta(8, 0) \cup \delta(10, 0) \cup \delta(12$
$= \{\phi\} \cup \{\phi\} \cup \{9\} \cup \{\phi\} \cup \{\phi\}$
$= \epsilon\text{-closure}\{9\} = \{6, 8, 9, 10, 11, 12\} = D$

$\delta(C, 1) = \epsilon\text{-closure}((\delta(5, 6, 8, 10, 12), 1)$
$= \delta(5, 1) \cup \delta(6, 1) \cup \delta(8, 1) \cup \delta(10, 1) \cup \delta(12, 1)$
$= \{\phi\} \cup \{7\} \cup \{\phi\} \cup \{\phi\} \cup \{\phi\}$
$= \epsilon\text{-closure}(7) = \{6, 7, 8, 10, 11, 12\} = E$

$\delta(D, 0) = \epsilon\text{-closure}(\delta(6, 8, 9, 10, 11, 12), 0)$
$= \delta(6, 0) \cup \delta(8, 0) \cup \delta(9, 0) \cup \delta(10, 0) \cup$
$\delta(11, 0) \cup \delta(12, 0)$
$= \{\phi\} \cup \{9\} \cup \{\phi\} \cup \{\phi\} \cup \{\phi\} \cup \{\phi\}$
$= \epsilon\text{-closure}\{9\} = D$

$\delta(D, 1) = \epsilon\text{-closure}(\delta(6, 8, 9, 10, 11, 12), 1)$
$= \delta(6, 1) \cup \delta(8, 1) \cup \delta(9, 1) \cup \delta(10, 1) \cup$
$\delta(11, 1) \cup \delta(12, 1)$

$\{7\} \cup \{\phi\} \cup \{\phi\} \cup \{\phi\} \cup \{\phi\} \cup \{\phi\}$

$= \epsilon\text{-closure}\{7\} = E$

$\delta(E,0) = \epsilon\text{-closure}(\delta(6,7,8,10,11,12),0)$

$= \delta(6,0) \cup \delta(7,0) \cup \delta(8,0) \cup \delta(10,0) \cup \delta(11,0)$
$\cup\, \delta(12,0)$

$= \{\phi\} \cup \{\phi\} \cup \{9\} \cup \{\phi\} \cup \{\phi\} \cup \{\phi\}$

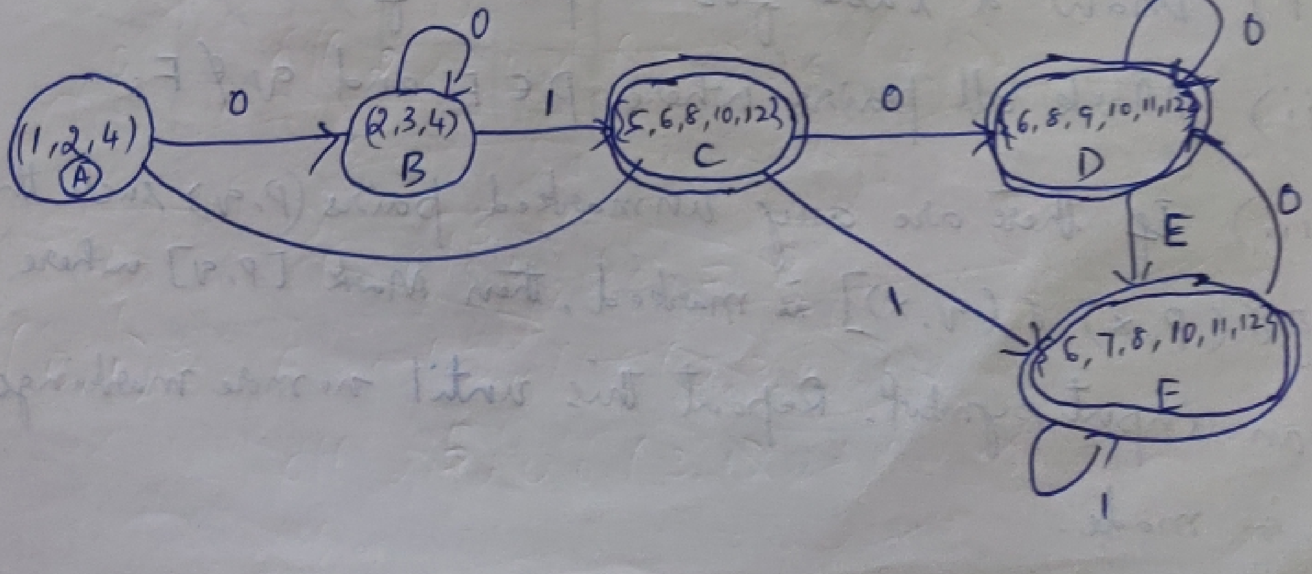
$= \epsilon\text{-closure}\{9\} = D$

$\delta(E,1) = \epsilon\text{-closure}(\delta(6,7,8,10,11,12),1)$

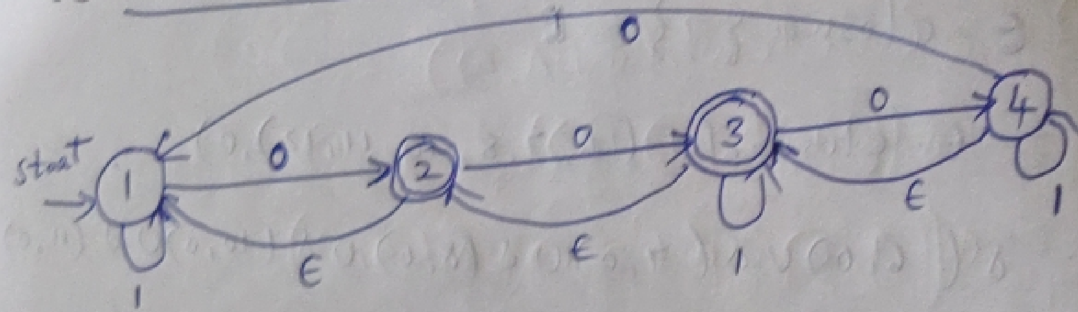
$= \delta(6,1) \cup \delta(7,1) \cup \delta(8,1) \cup \delta(10,1) \cup$
$\delta(11,1) \cup \delta(12,1)$

$= \{7\} \cup \{\phi\} \cup \{9\} \cup \{\phi\} \cup \{\phi\} \cup \{\phi\}$

$= \epsilon\text{-closure}\{7\} = E$

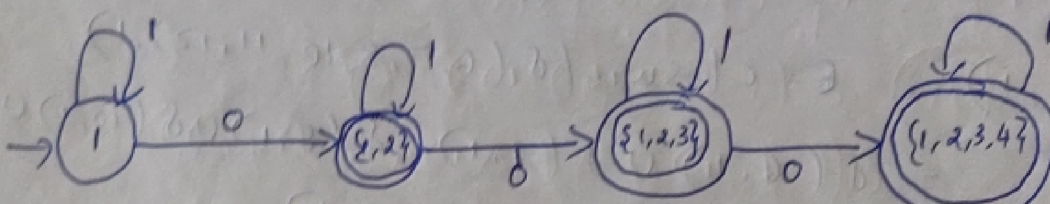


Convert E-NFA to DFA Equivalent :-



ϵ-closure$\{1\}$ = $\{1\}$ ϵ-closure$\{3\}$ = $\{1, 2, 3\}$

ϵ-closure$\{2\}$ = $\{1, 2\}$ ϵ-closure$\{4\}$ = $\{1, 2, 3, 4\}$



Minimization of DFA by Table Filling Method (Myhill - Nerode Theorem)

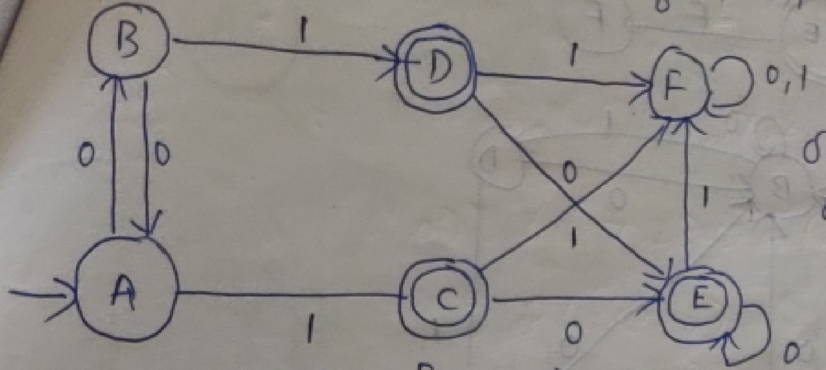
The steps of the algorithm is as follows:

i) Draw a table for all pairs of states (P, q)

ii) Mark all pairs where $P \in F$ and $q \notin F$.

iii) If there are any unmarked pairs (P, q) such that [δ(P, x), δ(q, x)] is marked, then Mark [P, q] where 'x' is an input symbol. Repeat this until no more markings can be made.

Combine all the unmarked pairs and make them a
ngle state in the minimized DFA



	A	B	C	D	E	F
A	X	X	X	X	X	X
B		X	X	X	X	X
C			X	X	X	X
D				X	X	X
E					X	X
F						X

	A	B	C	D	E
B					
C	X	X			
D	X	X			
E	X	X			
F	X	X	X	X	X

(B, A)

$\left.\begin{array}{l}\delta(B,0) = A \\ \delta(A,0) = B\end{array}\right\}$ $\left.\begin{array}{l}\delta(B,1) = D \\ \delta(A,1) = C\end{array}\right\}$

(D, C)

$\left.\begin{array}{l}\delta(D,0) = E \\ \delta(C,0) = E\end{array}\right\}$ $\left.\begin{array}{l}\delta(D,1) = F \\ \delta(C,1) = F\end{array}\right\}$

(E, C)

$\left.\begin{array}{l}\delta(E,0) = E \\ \delta(C,0) = E\end{array}\right\}$ $\left.\begin{array}{l}\delta(E,1) = F \\ \delta(C,1) = F\end{array}\right\}$

(E, D)

$\left.\begin{array}{l}\delta(E,0) = E \\ \delta(D,0) = E\end{array}\right\}$ $\left.\begin{array}{l}\delta(E,1) = F \\ \delta(D,1) = F\end{array}\right\}$

(F A)

$\delta(F,0) = F$ $\quad \delta(F,1) = F$

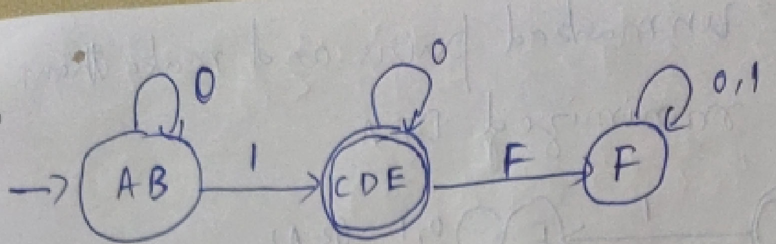
$\delta(A,0) = B$ $\quad \delta(A,1) = C$

(F, B)

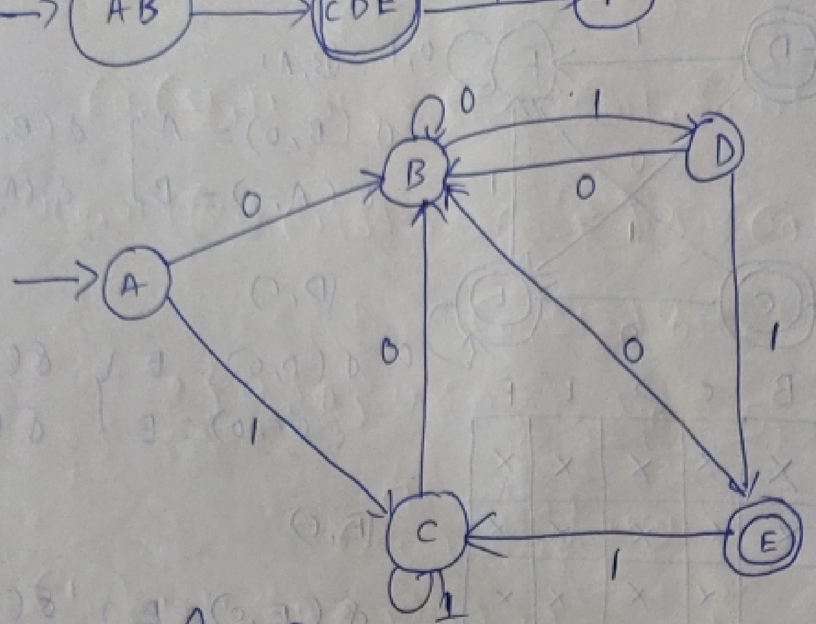
$\delta(F,0) = F$ $\quad \delta(F,1) = F$

$\delta(B,0) = A$ $\quad \delta(A,1) = C$

(A B) (C, D) (E, C) (E, D)



2)



	A	B	C	D
B	X			
C		X		
D	X	X	X	
E	X	X	X	X

$\delta(D, B)$

$\left.\begin{array}{l}\delta(D,0) = B \\ \delta(B,0) = B\end{array}\right\}$ $\begin{array}{l}\delta(D,1) = E \\ \delta(B,1) = D\end{array}$

$\delta(D, C)$

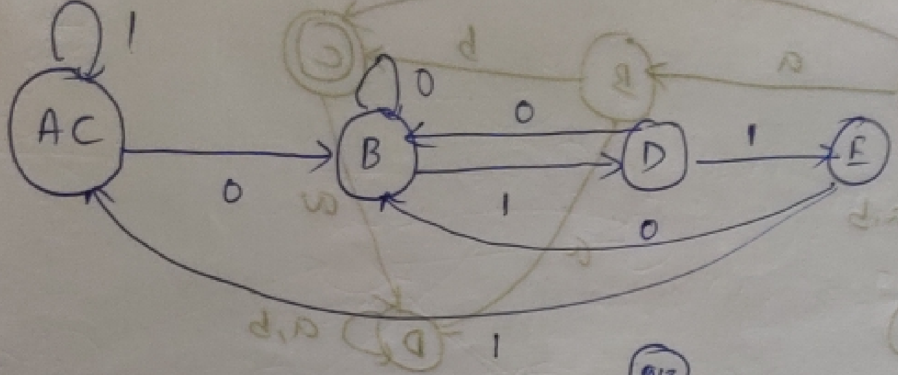
$\left.\begin{array}{l}\delta(D,0) = B \\ \delta(C,0) = B\end{array}\right\}$ $\begin{array}{l}\delta(D,1) = E \\ \delta(C,1) = C\end{array}$

$(B, A) = \left.\begin{array}{l}\delta(B,1) = D \\ \delta(A,1) = C\end{array}\right\}$ $\left.\begin{array}{l}\delta(B,0) = B \\ \delta(A,0) = B\end{array}\right\}$

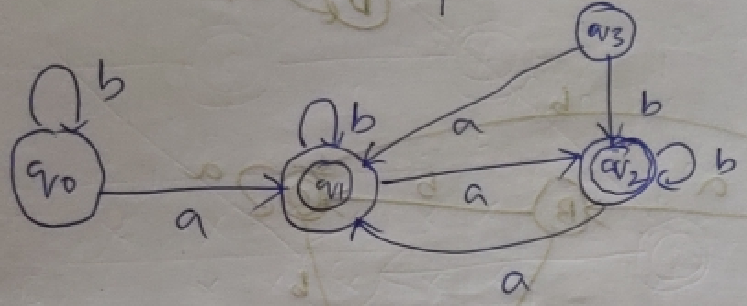
$(C, A) = \left.\begin{array}{l}\delta(C,0) = B \\ \delta(A,0) = B\end{array}\right\}$ $\left.\begin{array}{l}\delta(C,1) = C \\ \delta(A,1) = C\end{array}\right\}$

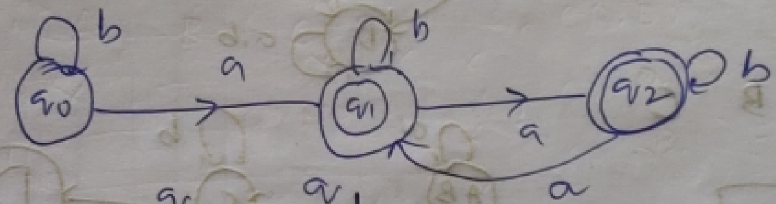
$(C, B) = \left.\begin{array}{l}\delta(C,0) = B \\ \delta(B,0) = B\end{array}\right\}$ $\left.\begin{array}{l}\delta(C,1) = C \\ \delta(B,1) = D\end{array}\right\}$

$(D, A) = \left.\begin{array}{l}\delta(D,0) = B \\ \delta(A,0) = B\end{array}\right\}$ $\begin{array}{l}\delta(D,1) = E \\ \delta(A,1) = C\end{array}$

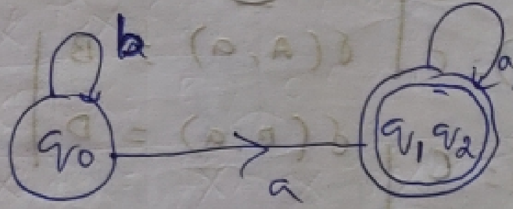

1
AC
0
B
0
D
1
E
0
1
0
1

3)

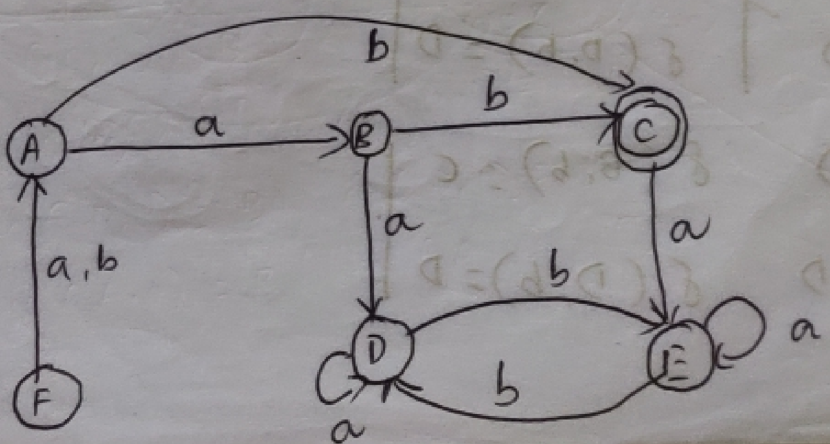

b
q0
a
b
q1
a
q3
b
q2
b
a
a


b
q0
a
b
q1
q2
b
a
a

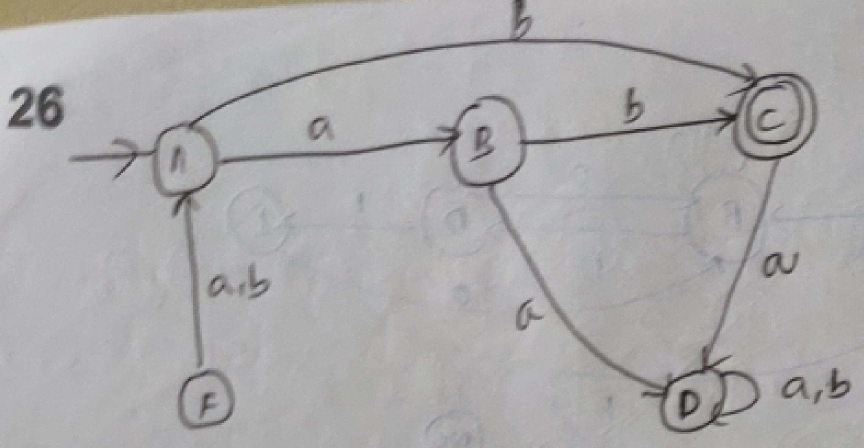
q_1	X	
q_2	X	

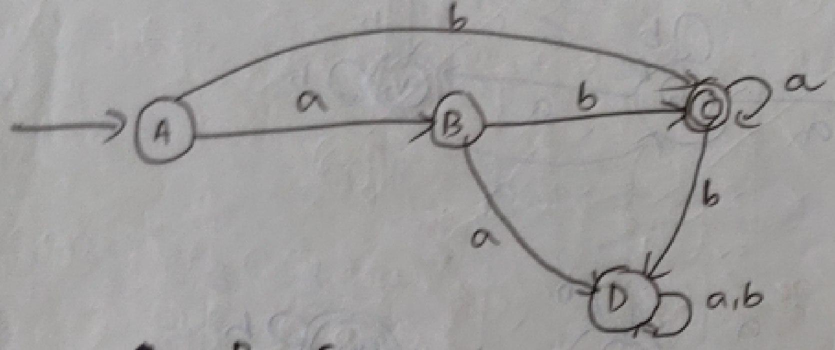

b
q0
a
q1, q2
a,b

4)

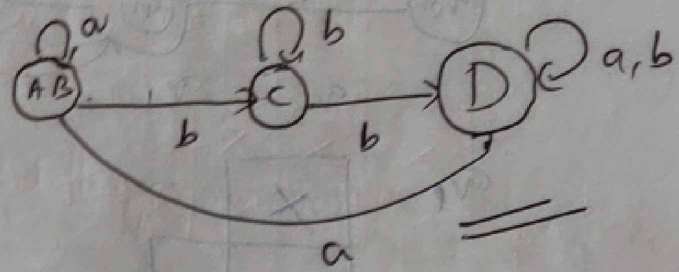

b
A
a
B
b
C
a
a
a,b
b
D
E
a
F
b
a

26





	A	B	C
B			
C	X	X	
D	X	X	X



$\delta(A, a) = B$ | $\delta(A, b) = C$

$\delta(B, a) = D$ | $\delta(B, b) = C$

AB

$\delta(A, a) = B$ | $\delta(A, b) = C$

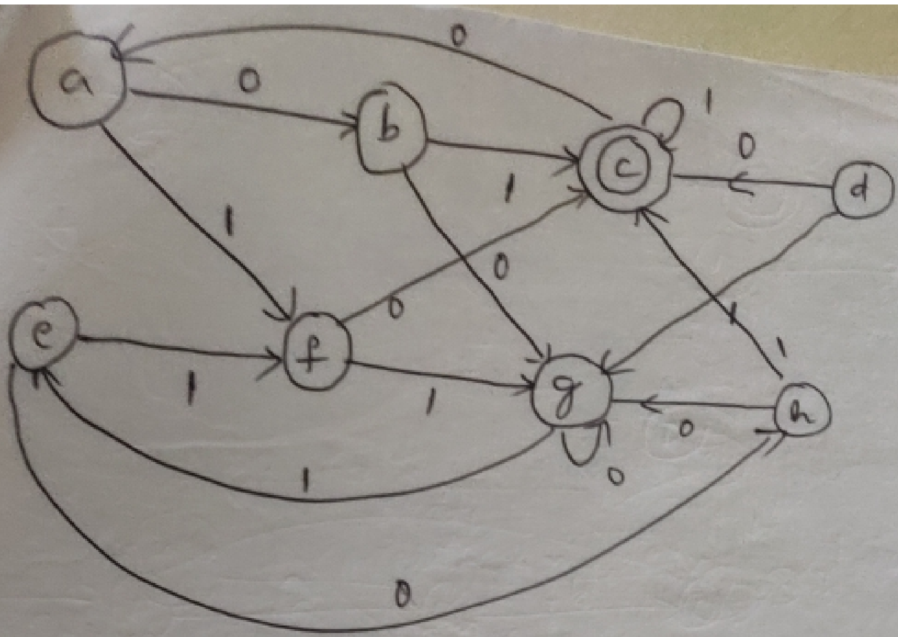
$\delta(B, a) = D$ | $\delta(B, b) = C$

$\delta(A, a) = B$ | $\delta(A, b) = C$

$\delta(D, a) = D$ | $\delta(D, b) = D$

$\delta(B, a) = D$ | $\delta(B, b) = C$

$\delta(D, a) = D$ | $\delta(D, b) = D$



	a	b	c	d	e	f	g
b	X						
c	X	X					
d	X	X	X				
e		X	X	X			
f	X	X	X		X		
g	X	X	X	X	X	X	
h	X		X	X	X	X	X