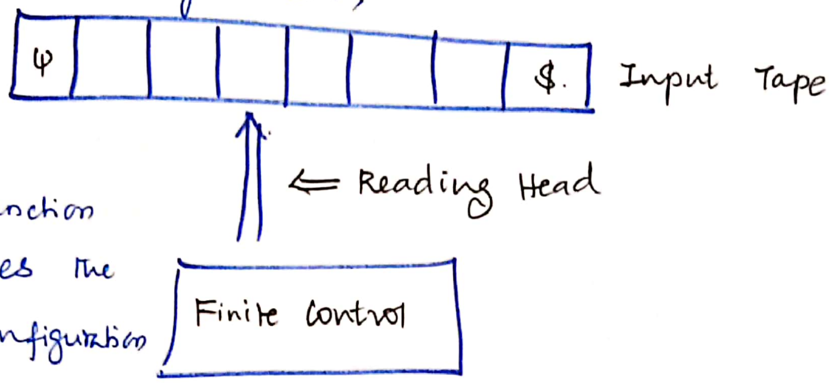
FORMAL LANGUAGES AND AUTOMATA THEORY

Introduction Finite Automata

A Finite Automata consists of a finite memory called Input tape.

A finite - Non empty set of states, an Input alphabet, a read only Head,



A Transition Function which defines the change of configuration

an Initial State, and a Model of Finite Automata finite - Non empty Set of Final States.

Input tape contains Cells and each cell contains one symbol from the Input alphabet.

w → leftmost cell

\$ – Right most cell.

* The Head reads one symbol on the Input Tape and finite control controls the next Configuration.

* The Head Can read left → right or Right → left.

One cell at a Time.

The Head can't write and can't move Backward.

So, FA can't remember its previous read symbols.

This is the major limitations of FA.

DFA

5 Tuples

$Q \rightarrow$ No of States

$\Sigma \rightarrow$ No of Input symbols

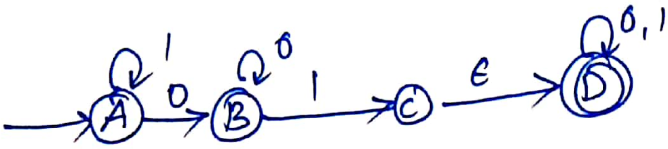
$F \rightarrow$ No of Final States

$q_0 \rightarrow$ Initial States

δ ~~δ~~ $\rightarrow$ Transition Function. — Delta — Δ. (or) δ

δ which maps the State one into another by using Input symbols.

δ



Types of Finite

Java source code is compiled into byte code when we use a Javac compiler.

* The bytecode gets saved on the Disk with the File Extension .class

* When the program is to be run, the byte code is converted, using the Just in Time (JIT) compiler.

* The Result is machine code which is then fed to the memory and is Executed.

Finite Automata :

The simplest machine to recognize patterns.

formal specification is,

$\{F, \delta, q_0, \Sigma, Q\}$ 5 Tuples.

→ Pattern matching → long string → short string.

Alphabets :

Alphabets denoted by Σ, is a Finite and nonempty set of symbols.

Ex:

1. $\Sigma \rightarrow$ containing all 26 characters used in English language, Then Σ is finite and Non empty set, and $\Sigma = \{a \ldots z\}$

2. $X = \{0, 1\}$

3. $Y = \{1, 2, 3 \ldots\}$ is not an alphabet because it is infinite (endless)

4. $Z = \{\ \}$ is not an alphabet because it is empty.

$\Sigma \rightarrow$ input symbol.

STRING:

A Finite sequence of symbols from some alphabet.

xyz

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

$\Sigma = \{\epsilon\}$.

$w = abcd.$ — $|w| = 4$

$n = 010$ $|n| = 3$

ϵ = empty string. $|n| = 0$

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

$\Sigma^2 = \{ab, aa, bb, ba\}$

$\Sigma^3 = \{aaa, aba, aab, abb, bab, baa, bba, bbb\}$

$\Sigma = \{0, 1, 2\}$

$\Sigma^2 = \{00, 01, 02, 10, 11, 12, 20), 21, 22\}$

$= 9.$

* Language:

L is Language over Σ, is a Subset of Σ^*.

* Collections of Strings over the Alphabet Σ.

* ϕ, ϵ are languages.

FA:

* can have one Initial state.
* more than one Final States.
* ϵ- null Input Symbol.
* State can perform the Transition one (or) are more then one by using Input Symbol.
* State should start with Initial state and end with

the Final states.

* operations (or) properties.

1) Union

2) Concatenation

3) Kleen closure.

Union.

L : L_1, L_2 Two languages.

$L_1 = \{ab, ba\}$

$L_2 = \{aa, bb\}$

$= \{ab, ba, aa, bb, abaa, abbb, baaa, babb\}$

$L_1 + L_2$, L_1 or L_2

L_1, L_2

Concatenation :

$L :> L_1, L_2$ Two languages.

$L_1 = \{aa, bb\}$

$L_2 = \{ab, ba\}$

$L_1 \cdot L_2$, $L_1 \odot L_2$, $L_1 L_2$, L_1 into L_2 is,

$\Rightarrow \{aaab, aaba, bbab, bbba\}$

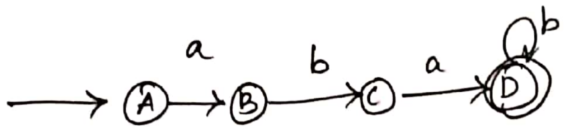
Kleen closure. L^*

$* \rightarrow n$ no of

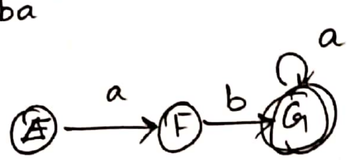
① Obtain an NFA to accept the Following language

$L = \{ w \mid w \in abab^n \text{ or } aba^n \text{ where } n \geq 0 \}$

$abab^n$ $n = 1 \ldots\ldots$

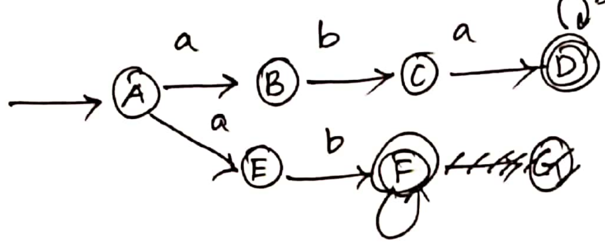


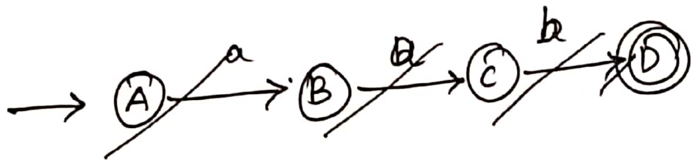
aba^n

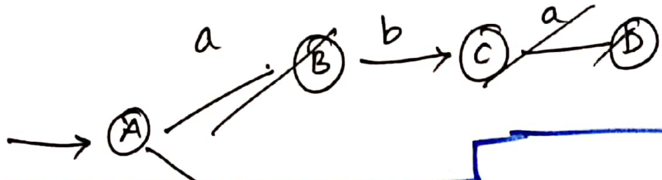


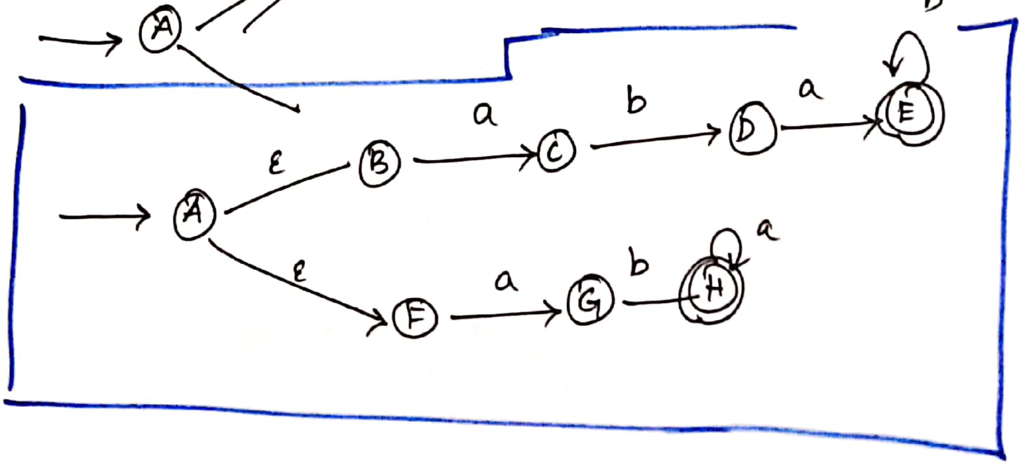
(DFA)

without 'e'

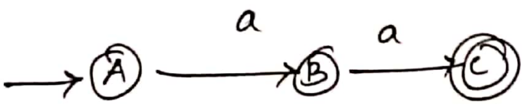


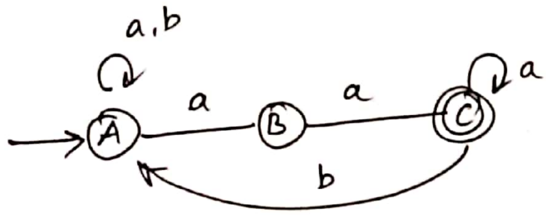




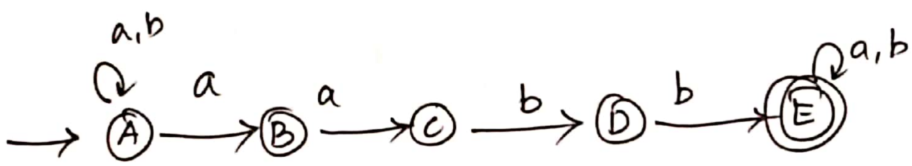


② Design NFA to accept strings with a's and b's such that the string end with 'aa'



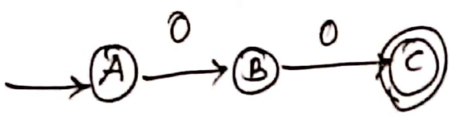


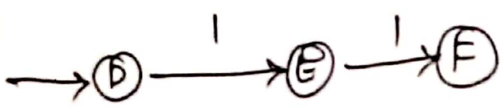
③ Design an NFA to accept of all string with double 'a' followed by double 'b'.

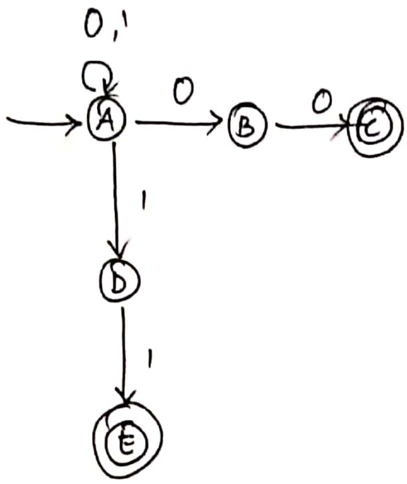


NFA.

④ Design an NFA to accept strings with 0's and 1's such that string contains two consecutive 0's or Two consecutive 1's.







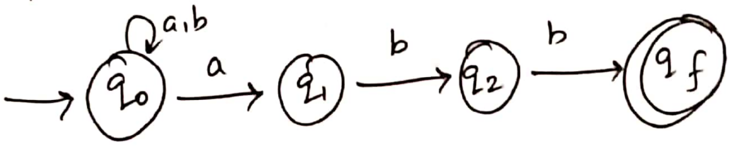
NFA TO DFA CONVERSION

Step1: perform Transition from Initial States.

Step2: If new states are there, then apply Transition and perform till the states get old.

Step3 : go to the Next state.

Ex:



Step 4: If the Two states reaching to the Same state by using the Input Symbol, then merge the state.

$\delta(q_0, a) = \{q_0, q_1\}$ — New state

$\delta(q_0, b) = \{q_0\} \rightarrow$ old state

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

$= \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\} \rightarrow$ old state

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

$= (q_0) \cup (q_2)$

$= \{q_0, q_2\} \rightarrow$ new state

$\delta((q_0, q_2), a) = \delta(q_0, a) \cup \delta(q_2, a)$

$= (q_0, q_1) \cup \phi$

$= (q_0, q_1) \rightarrow$ old state

$\delta((q_0, q_2), b)) = \delta(q_0, b) \cup \delta(q_2, b)$

$= (q_0) \cup (q_f)$

$= \{q_0, q_f\}$ — new state.

$\delta((q_0, q_f), a) = \delta(q_0, a) \cup \delta(q_f, a)$

$= (q_0, q_1) \cup \phi$

$= (q_0, q_1) \rightarrow$ old state.

$\delta((q_0, q_f), b) = \delta(q_0, b) \cup \delta(q_f, b)$

$= (q_0) \cup \phi$

$= q_0 \rightarrow$ old state.

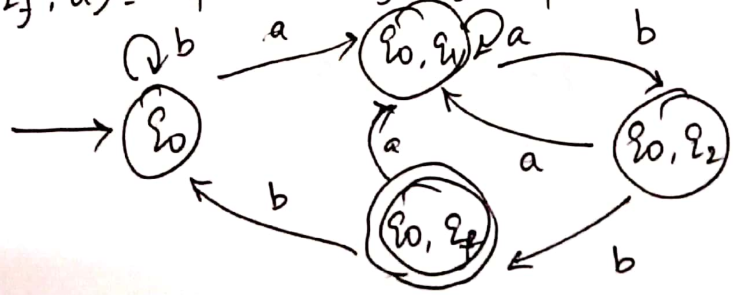
$\delta(q_1, a) = \phi$

$\delta(q_1, b) = \{q_2\} \rightarrow$ new state

$\delta(q_2, a) = \phi$

$\delta(q_2, b) = q_f \rightarrow$ new state.

$\delta(q_f, a) = \phi$ $\quad \delta(q_f, b) = \phi$.



$\rightarrow$ DFA.

1. Construct eq DFA for

NFA $M = (\{p, q, r, s\}, \{0,1\}, \delta, p, \{q, s\})$

where δ is given below,

22-1-19.

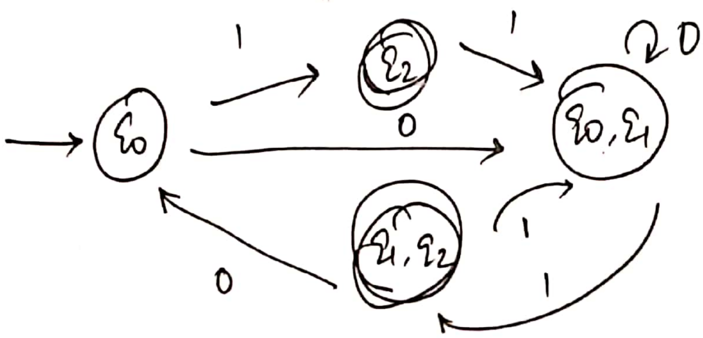
17/ 3, 4, 5, 28, 36
51, 56, 8, 18, 24

	0	1
p	(q, s)	(q)
q	(r)	(q, r)
r	(s)	(q, r)
s	-	(p)

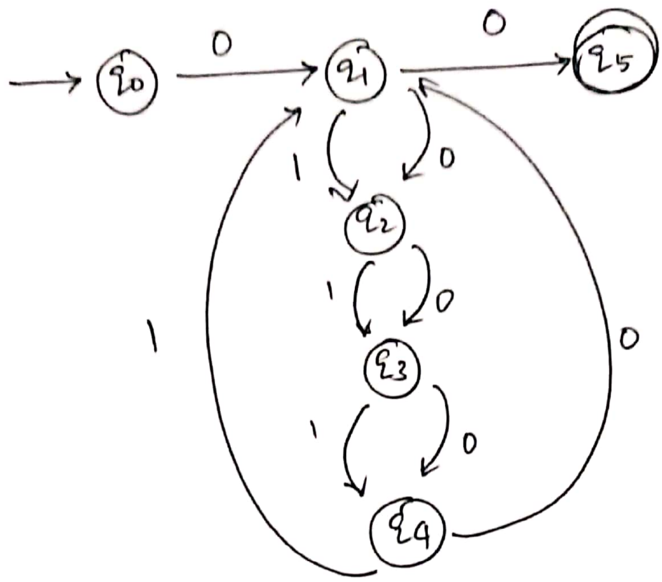
2. Find a DFA eq to NFA $M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_2\})$,

where δ is defined as follows,

	0	1
q_0	(q_0, q_1)	(q_2)
q_1	(q_0)	(q_1)
q_2	-	(q_0, q_1)



A NFA which accepts set of strings over {0,1} such that some two zero's are separated by a string over {0,1} whose length is 4n (n≥0) is shown in below Figure.



NDFA.

Design NFA which accepts set of all strings containing 1100 or 1010 as substrings.

DESIGN NFA

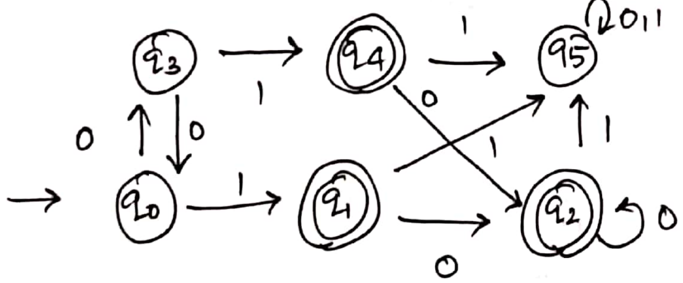
Minimization of DFA.

* more then one DFA will accept same language.
* minimum possible states Has less time Complexity.
* Doesn't affect the languages accepted by Automata.

Step 1: Divide the states into Two Groups

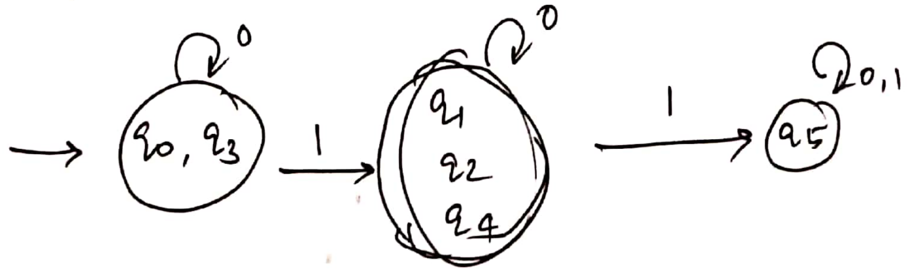
$G_1 \rightarrow$ Final states

$G_2 \rightarrow$ Set of non-Final States.

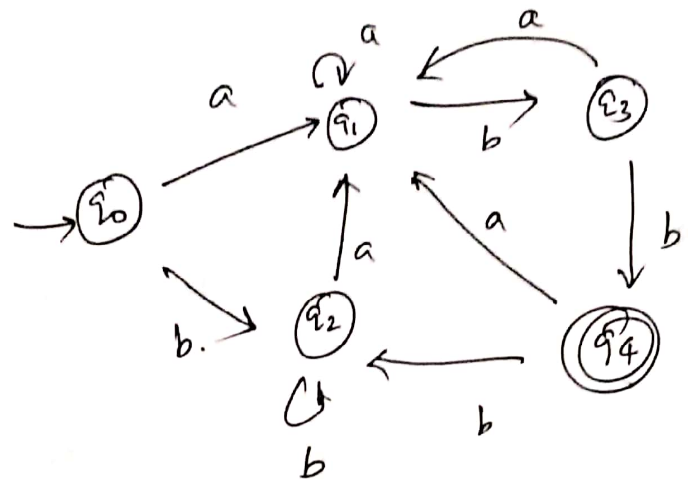


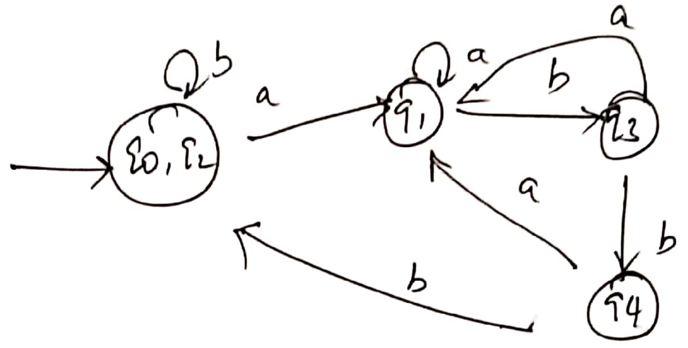
Step 2: If states are reaching to same Group then merge the states, otherwise partition not Possible.

Step 3: Stop the process.



1



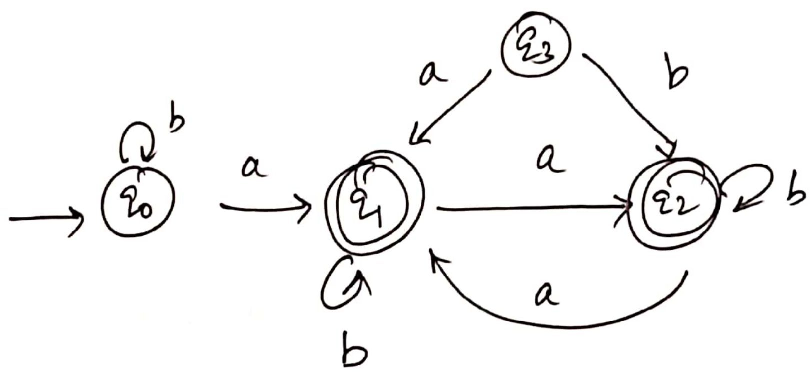


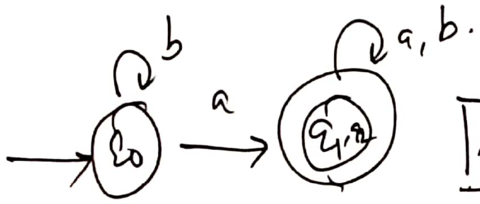
2, 11, 19, 21, 22, 24, 30, 31, 32, 33, 36, 37, 40, 41, 46, 47, 49, 50, 53, 59

~~16~~

18 → 2, 4

2



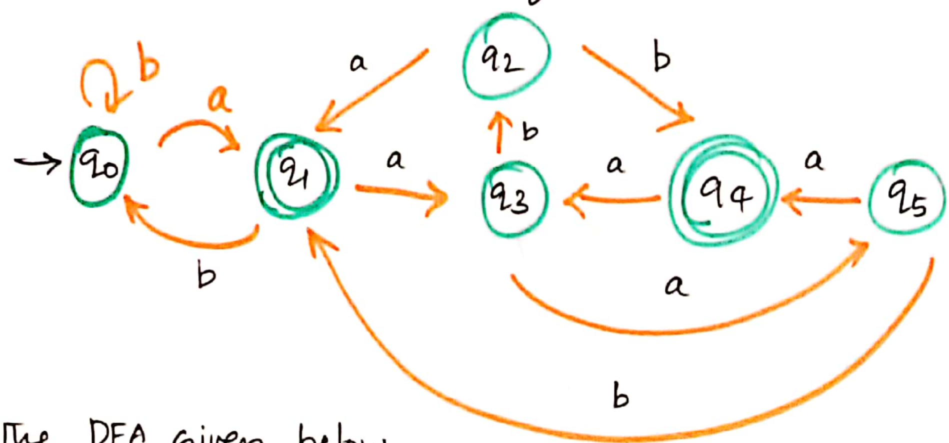


$\boxed{\text{Answers.}}$

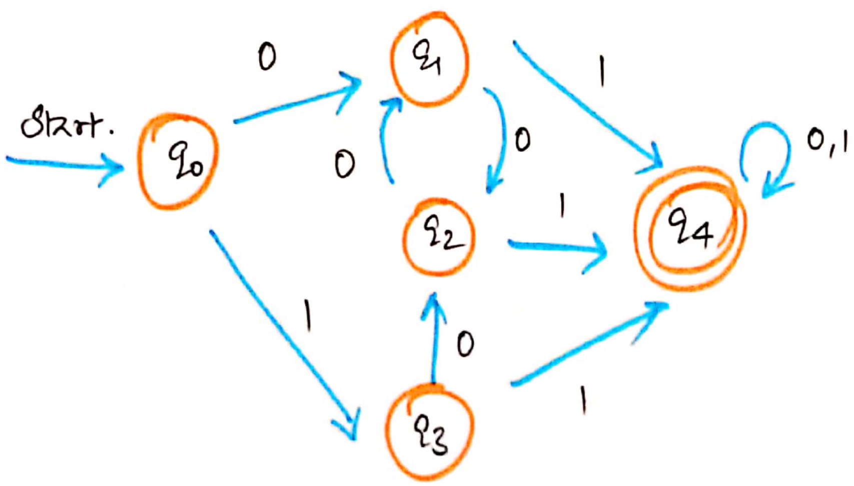
Consider the Following DFA and minimize it using Equivalence theorem,

Q / Σ	0	1
a	b	c
b	a	d
c	e	f
d	e	f
e	e	f
f	f	f

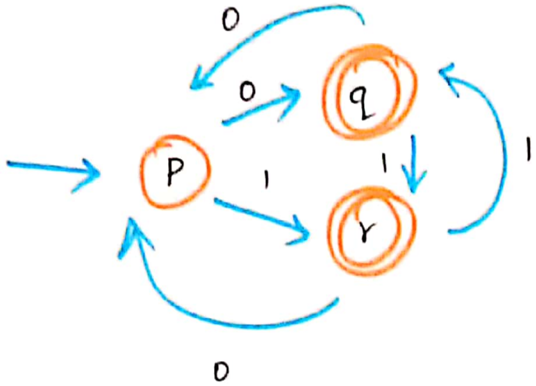
✓ Minimize the FA given below and show both given and reduced are equivalent.



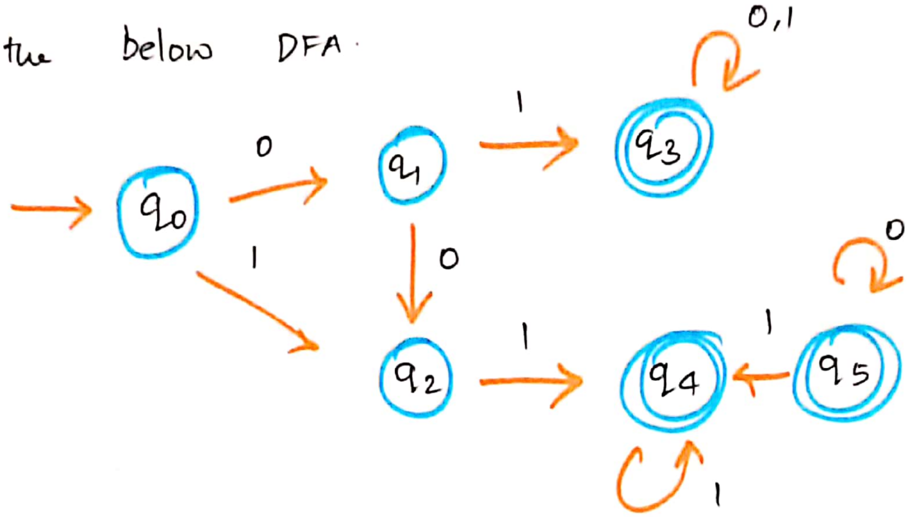
✓ Reduce the DFA given below.



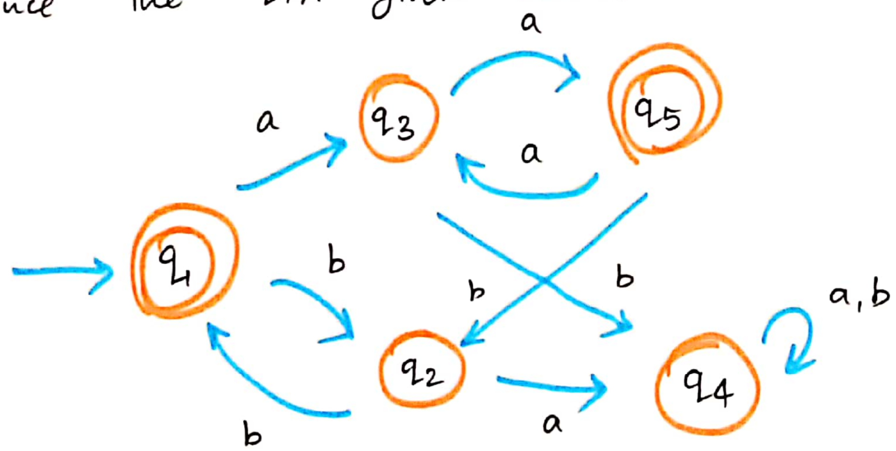
Reduce the DFA given below



✓ minimize the below DFA.



✓ Reduce the DFA given below,



Scanned with CamScanner

$$\delta'(q_0,a) = \epsilon\text{-closure}(\delta(\delta'(q_0,\epsilon),a))$$

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

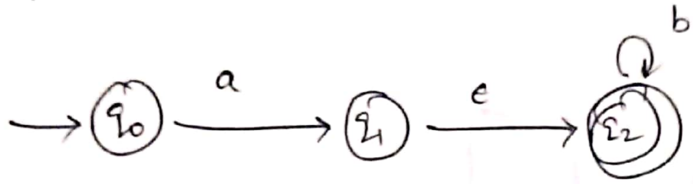
$$= \epsilon\text{-closure}(\delta(q_0),a))$$

$$\boxed{\delta(q_0,\epsilon) = \epsilon\text{-closure}(q_0)}$$

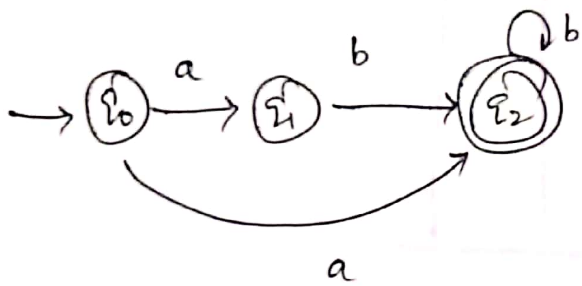
$=$

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

$$\boxed{\delta'(q_0,a) = \{q_1,q_2\}}$$



after NFA – without epsilon.

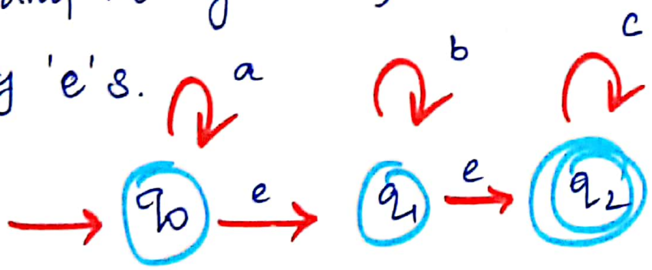


NFA with ε- moves

* The term 'ε - moves' refers that Transition Takes Place without reading any symbols in the Input.

* Any NFA is also an NFA with ε-moves.

any no of a's followed by b's and followed by any no of 'c's.



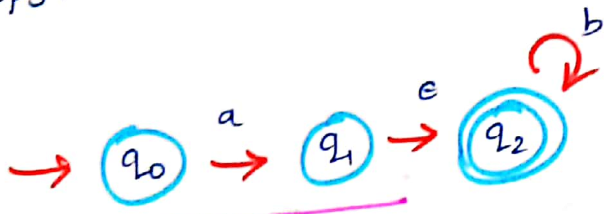
NFA-ε to without NFA.

Step 1: Find out ε-closure for all the states

Step 2: Find out Transition for remaining I/P Symbols to all the States.

Step 3: Convert NFA to DFA.

Ex:



	a	b	ε
→ q_0	q_1	ϕ	ϕ
q_1	ϕ	ϕ	q_2
q_2	ϕ	q_2	ϕ

ϵ-closure (q_0) =

$$\delta'(q_0, \epsilon) = \epsilon\text{-closure}(q_0)$$
$$= \{q_0\}$$

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

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

$\delta'(q_0,a) = \epsilon\text{-closure}(\delta(\underline{\delta'(q_0,\epsilon)},a))$

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

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

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

$\boxed{\delta'(q_0,a) = \{q_1,q_2\}}$

$\delta'(q_0,b) = \epsilon\text{-closure}(\delta(\underline{\delta'(q_0,\epsilon)},b))$

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

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

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

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

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

$\delta'(q_0,b) = \{\cancel{q_2}\}$

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

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

$\boxed{\delta'(q_0,b) = \{\phi\}}$

$\delta'(q_1,a) = \epsilon\text{-closure}(\delta(\underline{\delta'(q_1,\epsilon)},a))$

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

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

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

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

$\boxed{\delta'(q_1,a) = \epsilon\text{-closure}(\phi) = \phi}$

$\delta'(q_1, b) = \epsilon\text{-closure}(\delta(\underline{\delta'(q_1, \epsilon)}, b))$

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

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

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

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

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

$\boxed{\delta'(q_1, b) = \{q_2\}}$

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

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

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

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

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

$\boxed{\delta'(q_2, a) = \phi}$

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

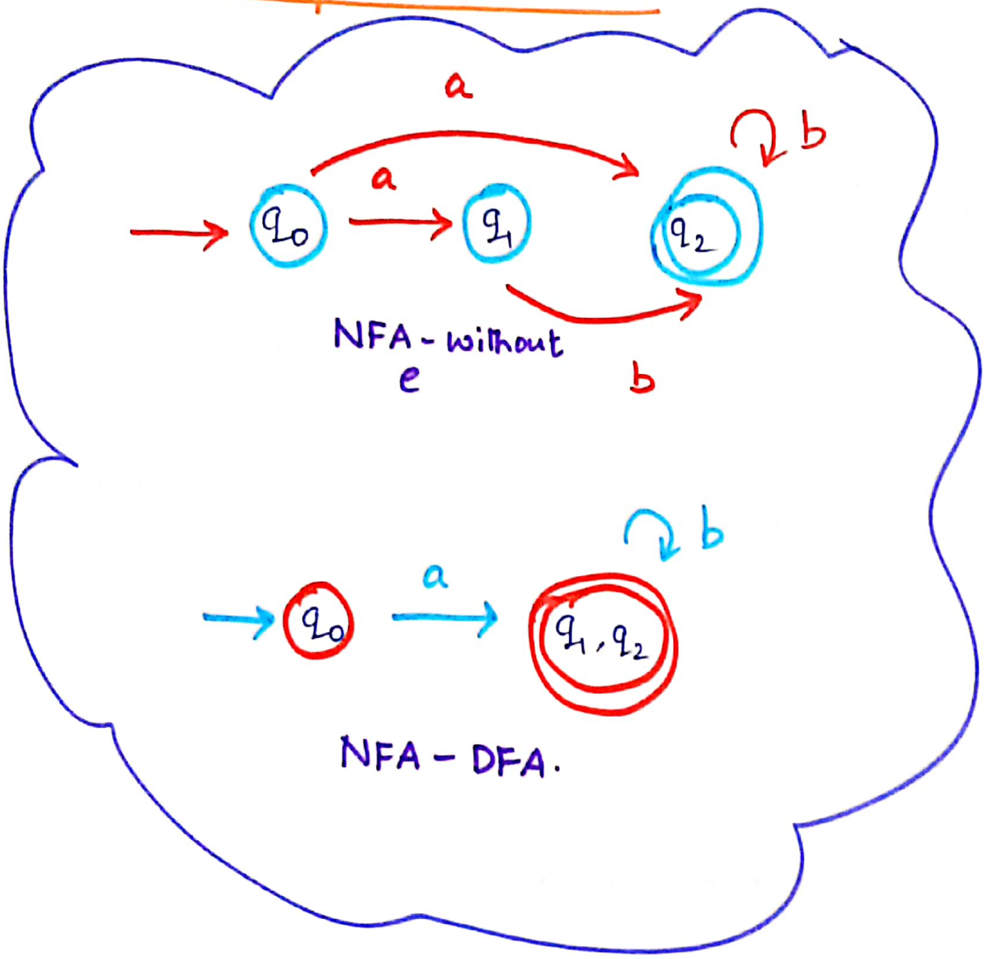
$= \epsilon\text{-closure}(\delta(\underline{\epsilon\text{-closure}(q_2)}, b))$

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

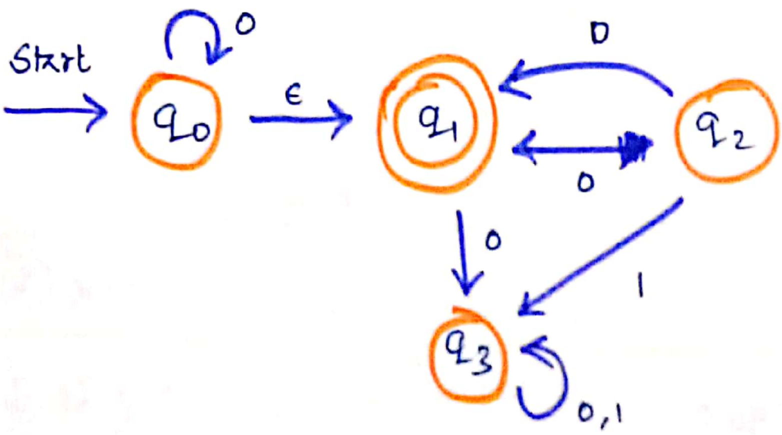
$= \epsilon\text{-closure}(q_2)$

$\boxed{\delta'(q_2, b) = q_2}$

	a	b
q_0	q_1, q_2	ϕ
q_1	ϕ	q_2
q_2	ϕ	q_2



Convert The Following NFA with ϵ - moves into equivalent NFA without ϵ - moves.



	0	1	ϵ
$\rightarrow q_0$	q_0	ϕ	q_1
q_1	q_3	q_2	ϕ
q_2	q_1	q_3	ϕ
q_3	q_3	q_3	ϕ

Step1:

ϵ - Transition for all states.

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

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

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

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

Step2:

Extended Transition,

$$\delta'(q_0, 0) = \epsilon\text{-closure}(\delta(\underline{\delta'(q_0, \epsilon)}, 0))$$
$$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), 0))$$
$$= \epsilon\text{-closure}(\delta(q_0, q_1), 0))$$
$$= \epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0))$$
$$= \epsilon\text{-closure}(q_0 \cup q_3\}$$
$$= \epsilon\text{-closure}(q_0, q_3)$$
$$= \epsilon\text{-closure}(q_0) \cup \epsilon\text{-closure}(q_3)$$

$$\boxed{\delta'(q_0, 0) = \{q_0, q_1\} \cup \{q_3\} = \{q_0, q_1, q_3\}}$$