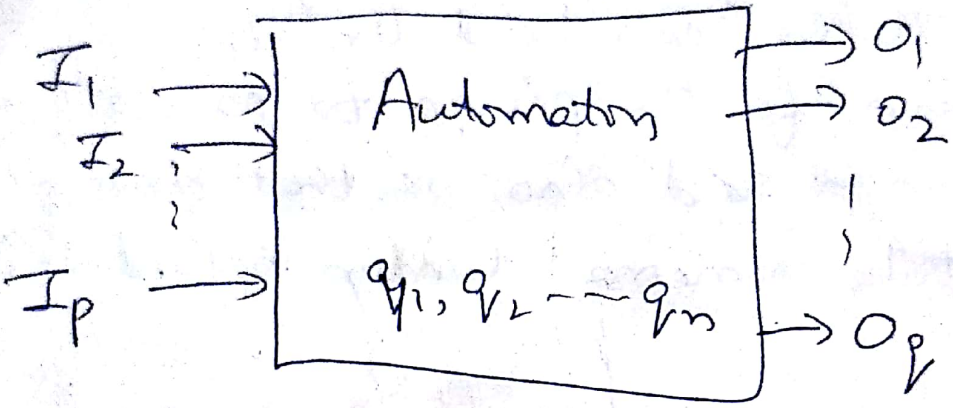
Automata

An automaton is defined as a system where energy, materials and information are transformed, transmitted and used for performing some functions without direct participation of man.

eg. automatic packing machines, automatic machine tools.



The characteristics of automaton are:-

(1) Input: At each of the discrete instants of time $t_1, t_2 \dots t_m$, the input values $I_1, I_2 \dots I_p$, each of which can take a finite no. of fixed values from input alphabet Σ.

(2) Output: $O_1, O_2 \dots O_{oq}$ are outputs of the model, each of which can take a finite no. of fixed values from output O.

(3) States: At any instant of time, the automaton can be in one of states $q_1, q_2 \dots q_n$.

(4) State relation:- The next state of an automaton at any instant of time is determined by the present state & present input.

Basic concepts

Alphabet:- It is a finite non empty set Σ of symbols.

Strip:- It is a finite sequence of symbols from the alphabet.

eg. if $\Sigma = \{a, b\}$

then a, b, aa, ab, bb, abb etc. are strips on Σ.

A **prefix** of a strip is any no. of leading symbols of that strip.

A **suffix** :- is any no. of trailing symbols.

eg. ω = abbab

Prefix = λ, a, ab, abb, abba, abbab

Suffix = λ, b, ab, bab, bbab, abbab

where empty strip is represented by λ, or ϵ.

Proper **suffix or proper prefix** :- A prefix or suffix of a strip, other than strip itself.

Concatenation of two strings $w + v$:- is the string obtained by appending the symbols of v to the right end of w,

ie if $w = a_1 a_2 \cdots a_n$;
$v = b_1 b_2 \cdots b_m$;

then $wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m$

eg. concatenation of ab and bab is

abbab

Reverse of string:- is obtained by writing the symbols in reverse order ie

if $w = a_1 a_2 \cdots a_n$ then $w^R = a_n \cdots a_2 a_1$

Length of string w:- denoted by $|w|$ is the no. of symbols in the string

eg. $|abab| = 4$

Note: (1) $|\lambda| = 0$

(2) $\lambda w = w\lambda = w$ for all strings w on Σ

(3) $|a| = 1$ for all $a \in \Sigma$

(4) $|wa| = |w| + 1$ for all strings w on Σ

(5) $a^R = a$ for all $a \in \Sigma$

(6) $(wa)^R = aw^R$ for all $a \in \Sigma$, w on Σ

If w is a string, w^n stands for string obtained by repeating w, n times for all w,

eg. if $w = abb$

then $w^0 = \lambda$

$w^1 = abb$

$w^2 = abbabb$

$w^3 = abbabbabb$

If Σ is an alphabet, then Σ^* denotes the set of string obtained by concatenating zero or more symbols from Σ.

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

$\Sigma^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, 001$

Σ^* is countable infinite set.

$\Sigma^+ = \Sigma^* - \{\lambda\}$ (4)

$\Sigma^+ = \{0, 1, 00, 01, 10 \ldots\}$

~~If Σ is an alphabet, then Σ^* denotes~~

Language :- is a subset of Σ^*

eg. $L_1 = \{00, 01, 10, 11\}$ is a finite language on Σ.

A string in a language L is called sentence of L.

$L_2 = \{0^n 1^n : n \geq 0\}$ is infinite language on Σ.

$L_2 = \{\lambda, 01, 0011, 000111 \ldots\}$

Operations on languages

Complement of a language :- is defined as :- $\overline{L} = \Sigma^* - L$

Reverse of a language : is set of all string reversals ie

$L^R = \{w^R : w \in L\}$

Concatenation of 2 languages :- L_1 and L_2 :- is set of all strings obtained by concatenating any element of L_1 with any element of L_2 ie :

$L_1 L_2 = \{xy : x \in L_1 \text{ and } y \in L_2\}$

We define :-

L^n as L concatenated with itself n times, with the special case :-

$$L^0 = \{\lambda\}$$

and $L^1 = L$ for every language L

Star closure of a language is defined as,

$$L^* = L^0 \cup L^1 \cup L^2 - - -$$

Positive closure of a language is defined as

$$L^+ = L^1 \cup L^2 \cup L^3 - - -$$

Union, intersection and difference of 2 languages are same as that of 2 sets.

A finite automaton can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

(1) Q is a finite nonempty set of states.

(2) Σ is finite nonempty set of inputs called input alphabet.

(3) δ is a function which maps $Q \times \Sigma$ into Q and called direct transition function.

(4) $q_0 \in Q$ is initial state.

(5) $F \subseteq Q$ is set of final states.

Note; The transition function which maps $Q \times \Sigma^*$ into Q ie maps a state and strip of input symbols into a state is called indirect transition function.

Transition system | Transition graph

A transition graph of a transition system is a finite directed labelled graph in which each vertex (or node) represents a state and the directed edges indicate the transition of a state and edges are labelled with input or output. A transition system accepts a string w in Σ^* if:-

(1) there exists a path which originates from some initial state, goes along arrows + terminates at some final state, (2) Path value obtained by concatenation of all edge labels of path is equal to w.

Properties of Transition Functions

(1) $\delta(q, 1) = q$ is a finite automaton. This means that state of the system can be changed only by an input symbol.

(2) For all strings w and input symbols a,

$$\delta(q, aw) = \delta(\delta(q, a), w)$$
$$\delta(q, wa) = \delta(\delta(q, w), a)$$

Acceptability of a string by a finite automaton

A string is accepted by a finite automaton

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

if $\delta(q_0, x) = q$ for some $q \in F$

A final state is also called an accepting state

eg. Consider the finite state machine whose transition function δ is given below. Here $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$

Give the entire sequence of states for input string 1101.

Transition Function δ

State	Input 0	Input 1
→ q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

$$\begin{aligned}\delta(q_0, 1101) &= \delta(q_1, 101) \\ &= \delta(q_0, 01) \\ &= \delta(q_2, 1) \\ &= \delta(q_3, 1) \\ &= \delta(q_1, \wedge) \\ &= \delta(q_0, \wedge) = q_0\end{aligned}$$

Non deterministic Finite state Machine (NDFA)

The automaton we have discussed till now is Deterministic Finite automaton (DFA). But at each point, automata may have several posible moves ie several choices may exist for next state at any point. Such automata is called NDFA (Non deterministic finite automaton).

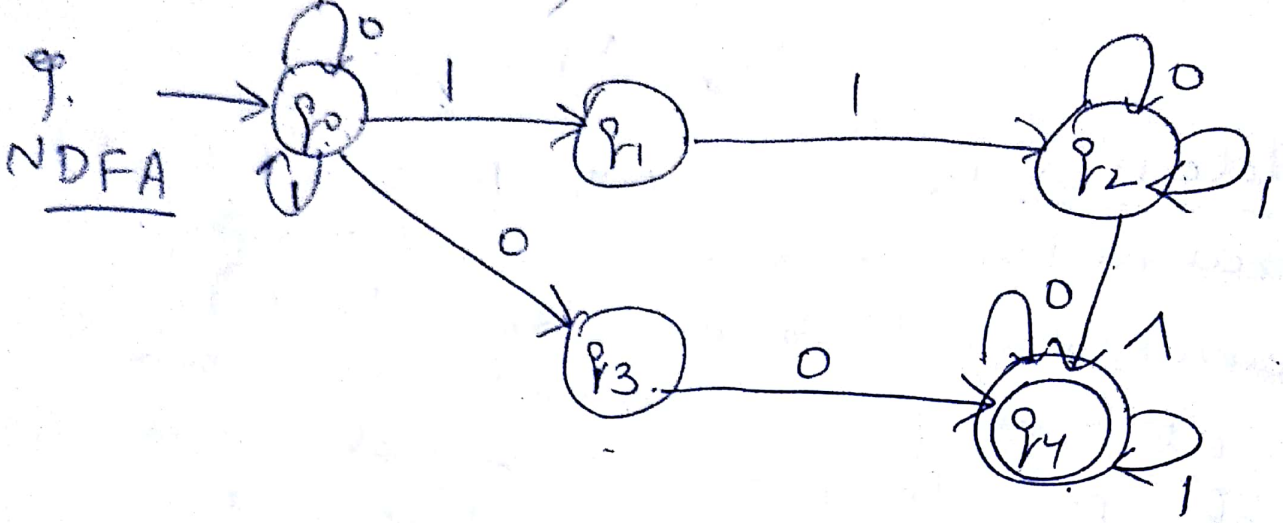
A NDFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where :-

1. Q is a finite nonempty set of states.
2. Σ is a finite nonempty set of inputs.
3. δ is a transition function mapping from $Q \times \Sigma$ into 2^Q which is power set of Q i.e set of all subsets of Q.
4. $q_0 \in Q$ is intial state
5. $F \subseteq Q$ is set of final states

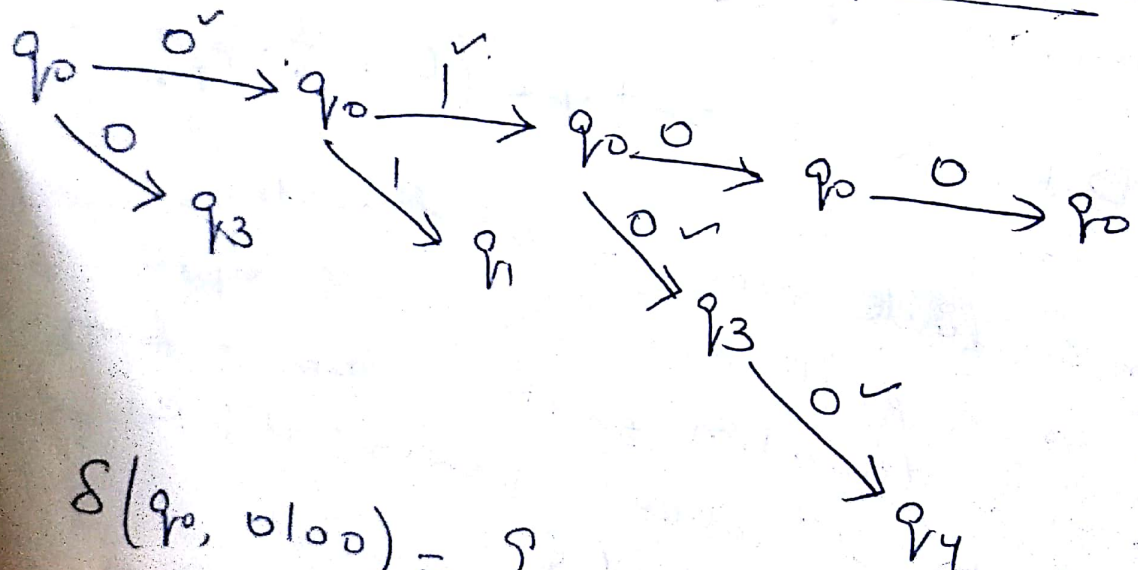
The difference between DFA and NDFA is that in DFA, the outcome of δ is a single state ie an element of Q, but for NDFA, the outcome of δ is subset of Q (ie it can be set of many states)

q.

NDFA



States reached while processing 0100 :-

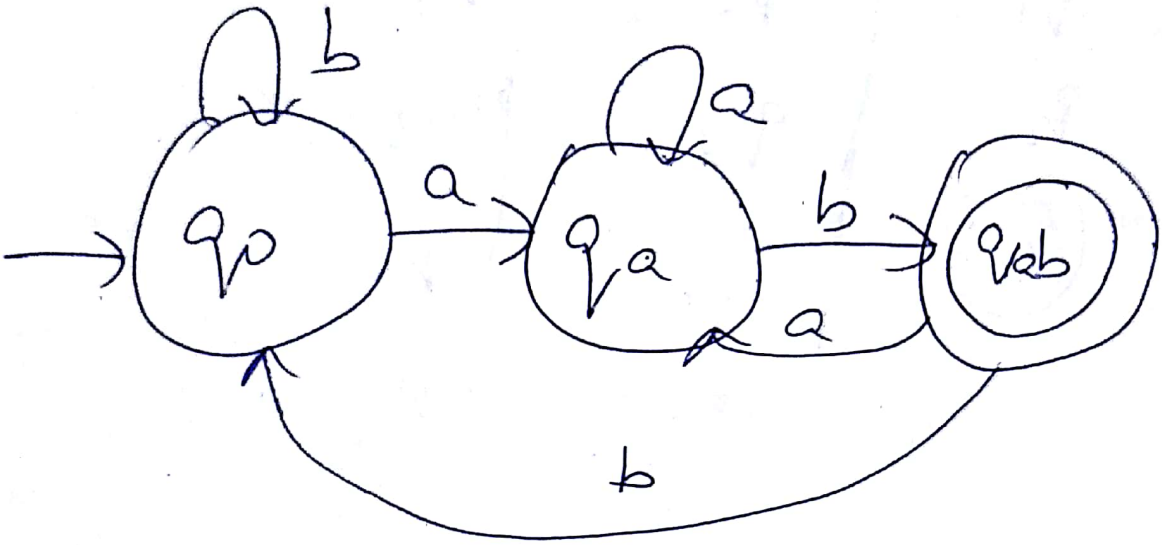


$\delta(q_0, 0100) = \{q_0, q_3, q_4\}$

Q: Design a DFA that accepts all strings with suffix ab. (7)

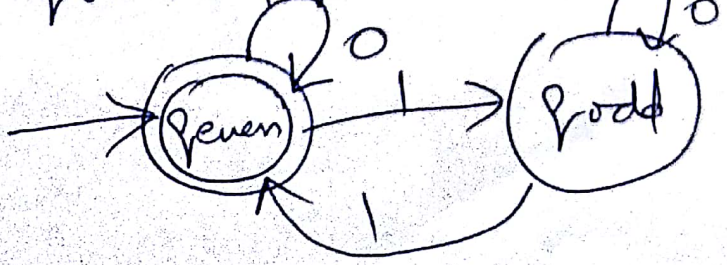
A:

State/Σ	a	b
→q_0	q_a	q_0
q_a	q_a	q_{ab}
(q_{ab})	q_a	q_0

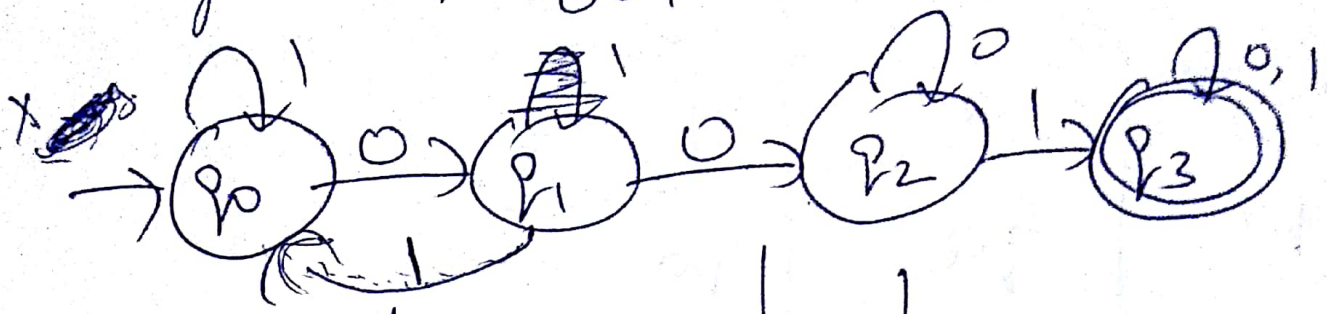


Q: Design a DFA that accepts all strings with even no. of 1's

State/Σ	0	1
→q_{even}	q_{even}	q_{odd}
q_{odd}	q_{odd}	q_{even}



Q: Design a dfa that accepts all strings with 001 as substring. ★?



state / Σ	0	1
→ q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_2	q_3
(q_3)	q_3	q_3

Equivalence of NDFA and DFA

- A DFA can simulate the behaviour of NDFA by increasing the no. of states.

- A NDFA is more general machine without being more powerful.

Theorem:
For every NDFA, there exists a DFA which simulates the behaviour of NDFA, ie if L is accepted by NDFA, then there exists a DFA which also accepts L.

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be NDFA accepting L. We construct a DFA M' as

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

where:

1) $Q' = 2^Q$ (any state in Q' is denoted by $[q_1, q_2 \dots q_j]$ where $q_1, q_2 \dots q_j \in Q$.

2) $q_0' = [q_0]$

3) F' is set of all subsets of Q containing element of F.

4) $\delta'([q_1, q_2 \dots q_i], a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \dots \cup \delta(q_i, a)$

- when M has n states, corresponding automa has 2^n states, but we need not consi δ for all these 2^n states, but only fo those that are reachable from $[q_0]$. So, u start construction of δ for $[q_0]$ and continue b considering only the states appearing earlier under the input columns and construct δ for such states. We halt when no more new states appear.

Q: Construct a DFA equivalent to:-

$M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_0\})$

where δ is:-

State/Σ	0	1
→(q0)	q0	q1
q1	q1	q0, q1

For DFA

(1) Q' states are 2^Q ie $2^2 = 4$
They are ϕ, $[q_0]$, $[q_0, q_1]$, $[q_1]$

(2) $[q_0]$ is initial state.

(3) $[q_0]$ and $[q_0, q_1]$ are final states as these are only states containing q_0.

(9)

State / Σ	0	1
q_0	q_0	q_1
q_1	q_1	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	(q_0, q_1)

We start with initial state q_0, and make δ for next state encountered in the second row, next state is $[q_0, q_1]$, so we make δ for $[q_0, q_1]$ in third row, since we don't get any new state, we stop.

$$\text{for } \delta([q_0, q_1], 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$$
$$= q_0 \cup q_1$$
$$= [q_0, q_1]$$

$$\text{for } \delta([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$$
$$= q_1 \cup [q_0, q_1]$$
$$= [q_0, q_1]$$

Q: Find DFA equivalent to:-

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

where δ is given as:-

State/Σ	a	b
$\rightarrow q_0$	q_0, q_1	q_2
q_1	q_0	q_1
q_2	ϕ	q_0, q_1

(1) $Q' = 2^Q = 2^3 = 8$ ie ϕ, $[q_0]$, $[q_1]$, $[q_2]$, $[q_0, q_1]$, $[q_0, q_2]$, $[q_1, q_2]$, $[q_0, q_1, q_2]$

(2) Initial state $q_0' = [q_0]$

(3) Final state, F' = States containing final state q_2 ie $[q_2]$, $[q_0, q_2]$, $[q_1, q_2]$ and $[q_0, q_1, q_2]$

δ'

State/Σ	a	b
$[q_0]$	$[q_0, q_1]$	$[q_2]$
$[q_2]$	ϕ	ϕ
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
$[q_0, q_2]$	$[q_0, q_1]$	$[q_0, q_1]$
$[q_1, q_2]$	$[q_0]$	$[q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1]$	$[q_0, q_1]$

δ'

State / Σ	a	b
q_0	$[q_0, q_1]$	q_2
q_2	ϕ	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_1, q_2]$
$[q_1, q_2]$	q_0	$[q_0, q_1]$

(10)

For $\delta([q_0, q_1], a) = \delta(q_0, a) \cup \delta(q_1, a)$

$= [q_0, q_1] \cup [q_0]$

$= [q_0, q_1]$

For $\delta([q_0, q_1], b) = \delta(q_0, b) \cup \delta(q_1, b)$

$= q_2 \cup q_1$

$= [q_1, q_2]$

For $\delta([q_1, q_2], a) = \delta(q_1, a) \cup \delta(q_2, a)$

$= q_0 \cup \phi$

$= q_0$

For $\delta([q_1, q_2], b) = \delta(q_1, b) \cup \delta(q_2, b)$

$= q_1 \cup [q_0, q_1]$

$= [q_0, q_1]$

~~Construction of Minimum automata~~

Minimization of finite automata

We construct an automaton with minimum no. of states equivalent to a given automaton M. As our interest lies only in strings accepted by M, what really matters is whether a state is final state or not.

Two states q_1 and q_2 are equivalent (ie $q_1 \equiv q_2$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final states or both of them are nonfinal states for $x \in \Sigma^*$.

But it is difficult to construct $\delta(q_1, x)$ and $\delta(q_2, x)$ for all $x \in \Sigma^*$ (as there are infinite no. of strings in Σ^*), we give following definition:-

2 states q_1 and q_2 are k-equivalent ($k > 0$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final states or both nonfinal states for all strings x of length k or less.

Construction of Minimum Automaton

(1) Construction of π_0 :- For 0-equivalence:

$\pi_0 = \{q_1^0, q_2^0\}$

where $q_1^0 = F$ and $q_2^0 = Q - F$

(2) Construction of π_{k+1} from π_k :-

Let q_i^k be any subset in π_k. If q_1 and q_2 are in q_i^k, find out whether $\delta(q_1, a)$ and $\delta(q_2, a)$ are in same equivalence class in π_k for all $a \in \Sigma$. If so, q_1 and q_2 are $(k+1)$ equivalent. So, q_i^k is further divided into $(k+1)$ equivalence classes. Repeat this for every subset in π_k.

(3) Construct π_n for $n = 1, 2 \dots$ until $\pi_n = \pi_{n+1}$

(4) For the required minimum state automata the states are the equivalence classes obtained in step 3 i.e elements of π_n. State table is obtained by replacing a state q by corresponding equivalence class $[q]$

Q: Construct a minimum state automaton equivalent to finite automaton given by:-

State / Σ	0	1
→ q_0	q_1	q_5
q_1	q_6	q_2
(*q_2*)	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

$Q_1^0 = F = \{q_2\}$; $Q_2^0 = Q - Q_1^0$

$\pi_0 = \{\{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}\}$
$(0,2)\quad (1,5)(6,2)(2,6)(7,5)(2,6)(6,4)(6,2)$

Here $\{q_2\}$ cannot be partitioned further, for q_0 & q_1 :

$\delta(q_0, 0) = q_1$ and $\delta(q_1, 0) = q_6$

q_1 and q_6 are in same equivalence class in π_0

$\delta(q_0, 1) = q_5$ and $\delta(q_1, 1) = q_2$

But q_5 and q_2 are in different equivalence class in π_0, so q_0 and q_1 are in different subsets in π_1.

$\pi_1 = \{q_2\}\ \{q_0, q_4, q_6\}\ \{q_1, q_7\}\ \{q_3, q_5\}$
(0,2) (1,5) (4,5) (6,2) (6,2) (2,6) (3,6)

In $\{q_0, q_4, q_6\}$:-

$\delta(q_0, 0) = q_1$ and $\delta(q_4, 0) = q_7$ and $\delta(q_6, 0) = q_6$
where q_1, q_7 and q_6 are in same subset in π_0
Also, $\delta(q_0, 1) = q_5$ and $\delta(q_4, 1) = q_5$ and $\delta(q_6, 1) = q_4$
where q_5 and q_4 are in same subset in π_0

For $\{q_1, q_7\}$
$\delta(q_1, 0) = \delta(q_7, 0) = q_6$
$\delta(q_1, 1) = \delta(q_7, 1) = q_2$

For $\{q_3, q_5\}$
$\delta(q_3, 0) = \delta(q_5, 0) = q_2$
$\delta(q_3, 1) = \delta(q_5, 1) = q_6$

$\pi_2 = \{q_2\}\ \{q_0, q_4\}\ \{q_6\}\ \{q_1, q_7\}\ \{q_3, q_5\}$
(0,2) (1,5) (7,5) (6,4) (6,2) (6,2) (2,6) (3,6)

We split the set $\{q_0, q_4, q_6\}$ in π_1 into $\{q_0, q_4\}$ and $\{q_6\}$ in π_2 as:

$\delta(q_0, 0) = q_1$, $\delta(q_4, 0) = q_7$, $\delta(q_6, 0) = q_6$
$\delta(q_0, 1) = q_5$, $\delta(q_4, 1) = q_5$, $\delta(q_6, 1) = q_4$

q_1 and q_7 are in same subset in π_1, but q_6 and q_4 are in different subset. Also, q_5 and q_4 are in different subset.

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

At $\pi_2 = \pi_3$, we stop.

So minimum state automata M' as follows-

$$M' = (Q', \{0,1\}, \delta', q_0', F')$$

$$Q' = \{[q_2], [q_0, q_4], [q_6], [q_1, q_7], [q_3, q_5]\}$$

$$q_0' = [q_0, q_4], \quad F' = [q_2]$$

δ'

state / Σ	0	1
$[q_0, q_4]$	$[q_1, q_7]$	$[q_3, q_5]$
$[q_1, q_7]$	$[q_6]$	$[q_2]$
$[q_2]$	$[q_0, q_4]$	$[q_2]$
$[q_3, q_5]$	$[q_2]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_0, q_4]$

Q: Construct the minimum state automaton equivalent to finite automaton given by.

state	a	b
$\rightarrow q_0$	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_1	q_6	q_6
q_7	q_4	q_6