11/7/11

Page No.:
Date:

1. Introduction to theory of computation

⇒ Symbol

Symbol are the basic building blocks, which can be any character / token.

eg: A, B, C .... Z
0, 1, 2, 3

⇒ Alphabet

An alphabet is a finite non empty set of symbols. (every language has its own alphabet)

Here in TOC, we use symbol Σ for depicting alphabet.

eg: $\Sigma = \{0, 1\}$
$\Sigma = \{a, b, c, \ldots, z\}$
$\Sigma = \{a, b\}$

⇒ String

It is a finite sequence of symbols.

eg: $\Sigma = \{a, b\}$
String: aabb, aa, b, bab, and so on...

⇒ Language

A language is defined as a set of strings

Symbol → Alphabet → String → Language

Q) if $\Sigma = \{a, b\}$ then, find the following ?

$\Rightarrow \Sigma^0 = \{\epsilon\}$ $\Rightarrow$ Empty or length 0

$\Sigma^1 = \{a, b\}$

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

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

★ Σ^k is the set of all the strings from the alphabets Σ of length exactly k.

$\Sigma^k = \{w \mid |w| = k\}$
(using the symbols from the alphabet Σ)

Eg: Σ^{26} will contains strings with length 26

★ Kleene closure:
If Σ is a set of symbols, then we use Σ^* to denote the set of strings obtained by concatenating zero or more symbols from Σ of any length in general any string of any length which can have only symbols specified in Σ.

$$\Sigma^* = \bigcup_{i=0}^{i=\infty} \{w \mid |w| = i\}$$

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots\ldots \cup \Sigma^\infty$$

Page No.:
Date:

★ Positive Closure

If Σ is a set of symbols, then we use Σ^+ to denote the set of strings obtained by concatenating one or more symbols from Σ of any length in general any string of any length which can have only specific symbols specified. (except ϵ)

$$\Sigma^+ = \bigcup_{i=1}^{i=\infty} \{ w \mid |w| = i \}$$

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots\ldots \cup \Sigma^\infty$$

★ Finite automata

A finite automaton is a model that has a finite set of states and its control moves from one state to another state in response to external inputs

Finite automata can be broadly classified into two types:

1. Finite automata without output
   - (i) Deterministic finite automata
   - (ii) Non deterministic finite automata
   - (iii) Non deterministic finite automata with ϵ
2. Finite automata with output
   - (i) Mealy machine
   - (ii) Moore machine

Page No.:
Date:

✱ Deterministic Finite Automata

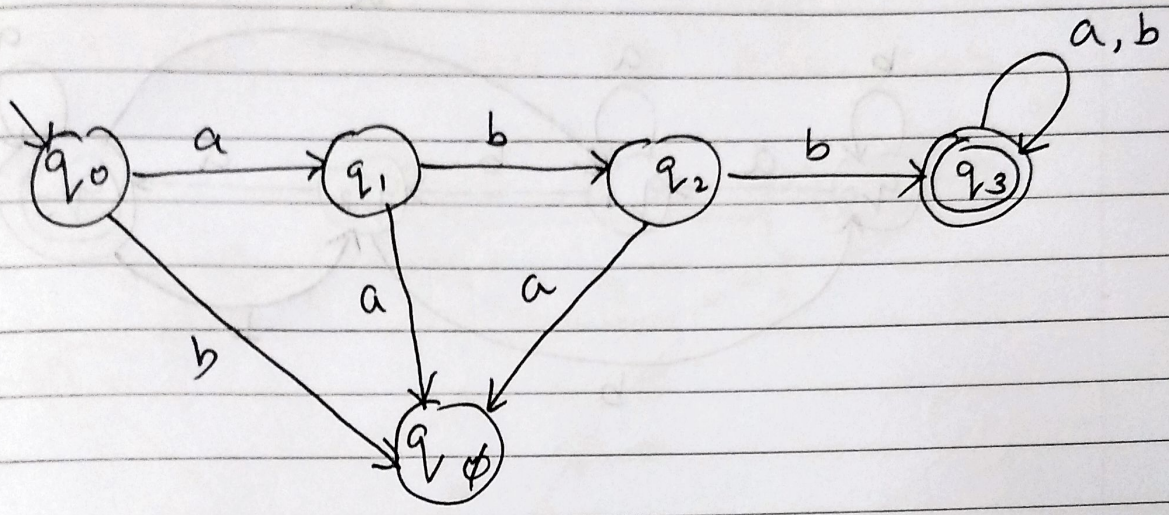
A deterministic finite automaton (DFA) is defined by 5-tuple $(Q, \Sigma, \delta, S, F)$ where:

- Q is a finite & non-empty set of states
- Σ is a finite non-empty set of finite input alphabet
- δ is a transition function $(\delta : Q \times \Sigma \rightarrow Q)$
- S / q_0 is initial state $(S \in Q)$
- F is a set of final states $(F \subseteq Q)$ $(0 <= |F| <= n$, where n is the number of states$)$

Page No.:
Date:

Q1) Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, where every accepted string 'w' starts with substring $s =$ 'abb'

=>

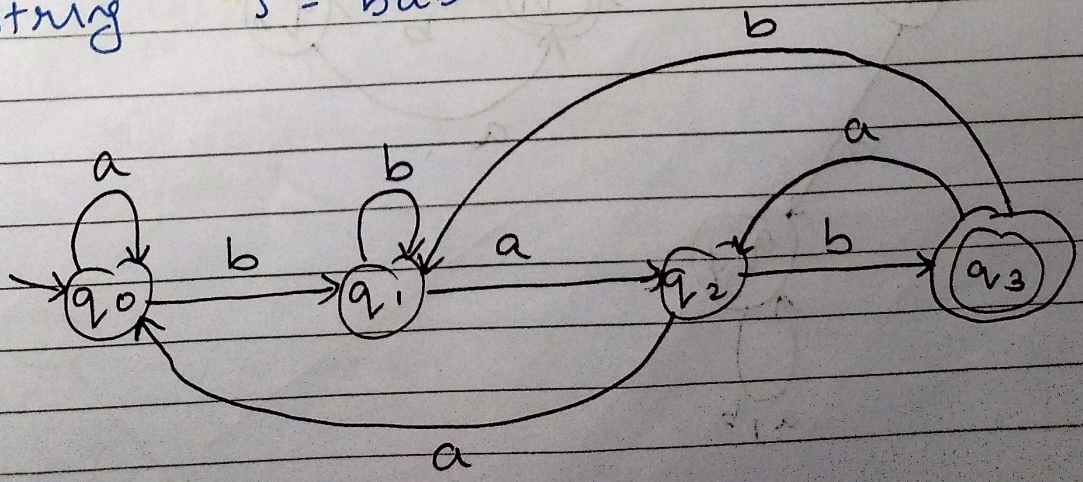


Steps :

(1) Make the smallest possible string i.e. abb
(2) Check for the null combinations
(3) End

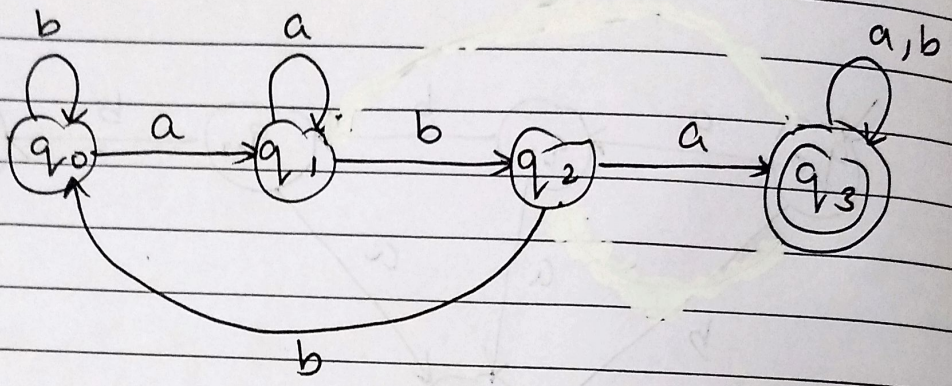
Q2) Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ where every accepted string 'w' ends with substring $s =$ 'bab'

=>



Page No.:
Date:

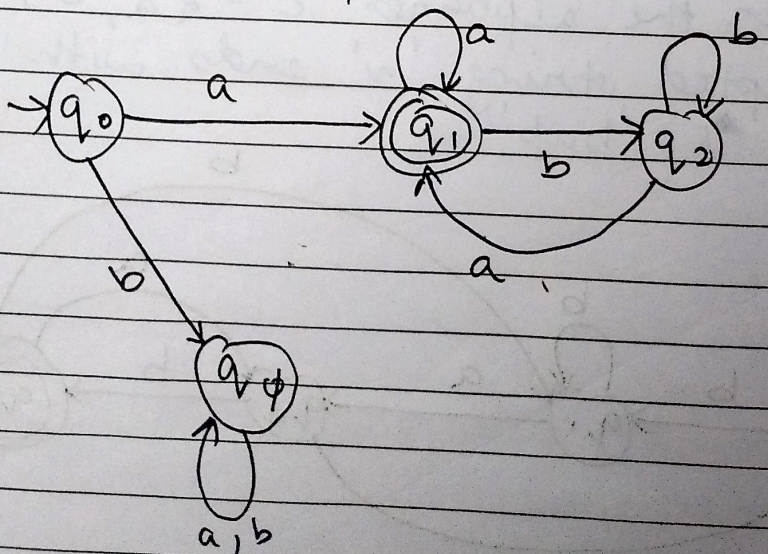
Q3) Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, where every accepted string 'w' contains substring $s =$ 'aba'



Q4) Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with 'a'.

$L = \{a, aa, aba, aaa, \ldots\ldots\}$

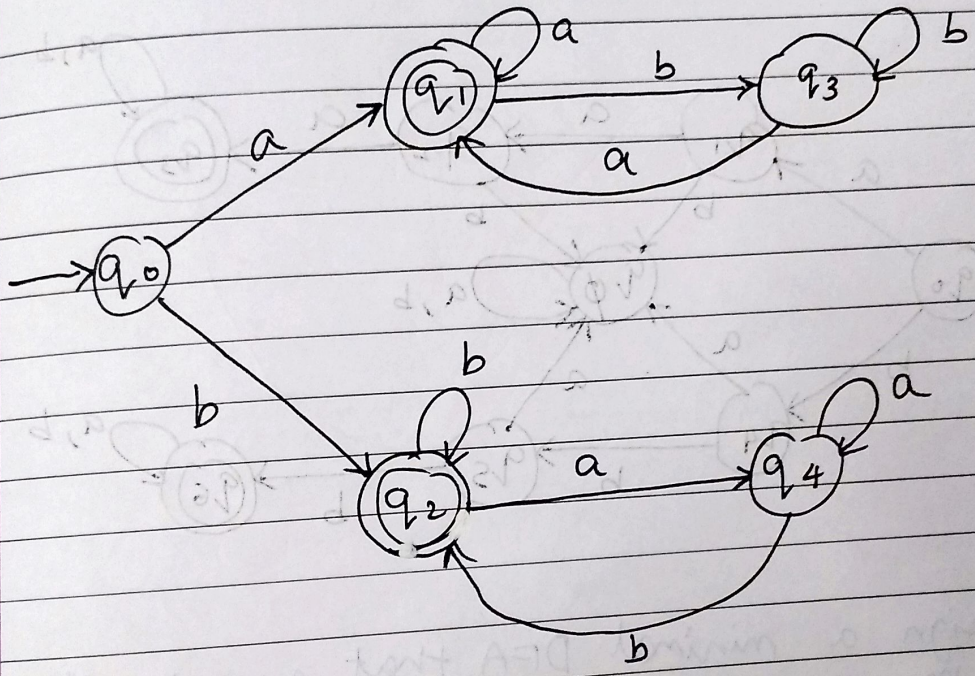
smallest possible



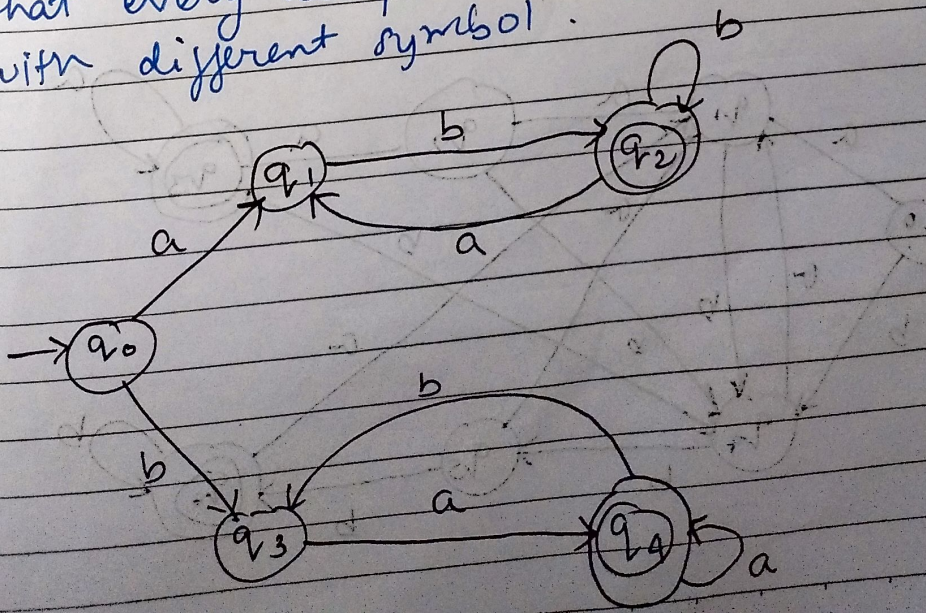
Page No.:
Date:

Q5) Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with same symbol

$L = \{a, b, aaa, aa, bb, aba, bab, \ldots\}$



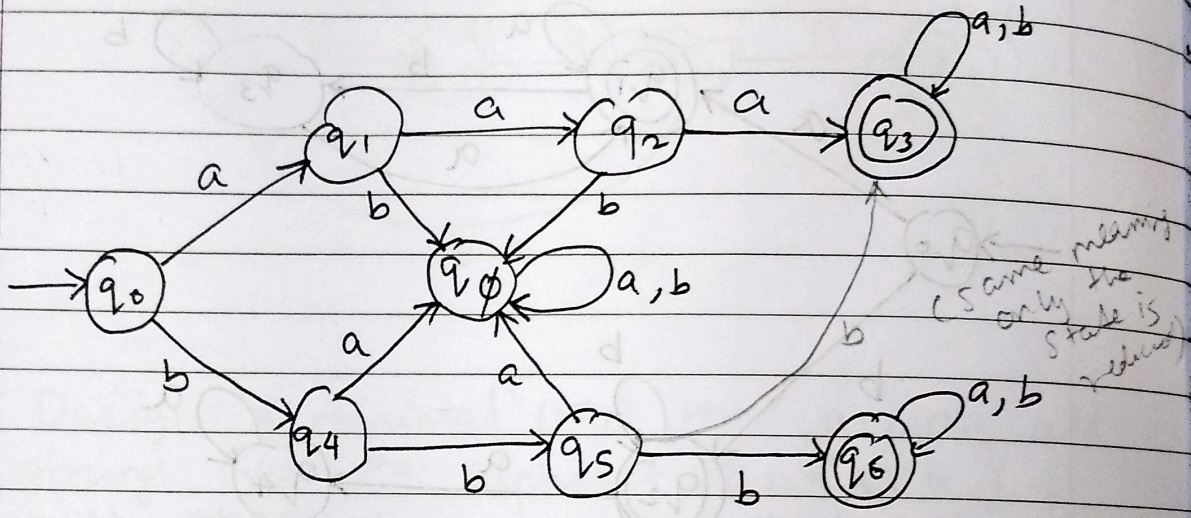
Q6) Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with different symbol.



Page No.:
Date:

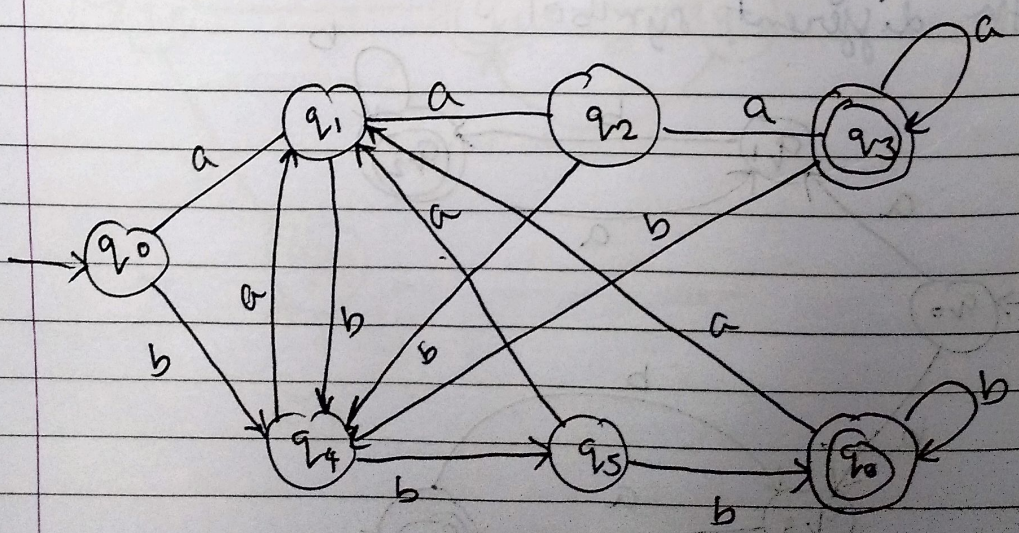
Q7) Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w, is like $w = SX$. $S = aaa / bbb$

start



Q8) Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w, is like $w = XS$. $S = aaa / bbb$

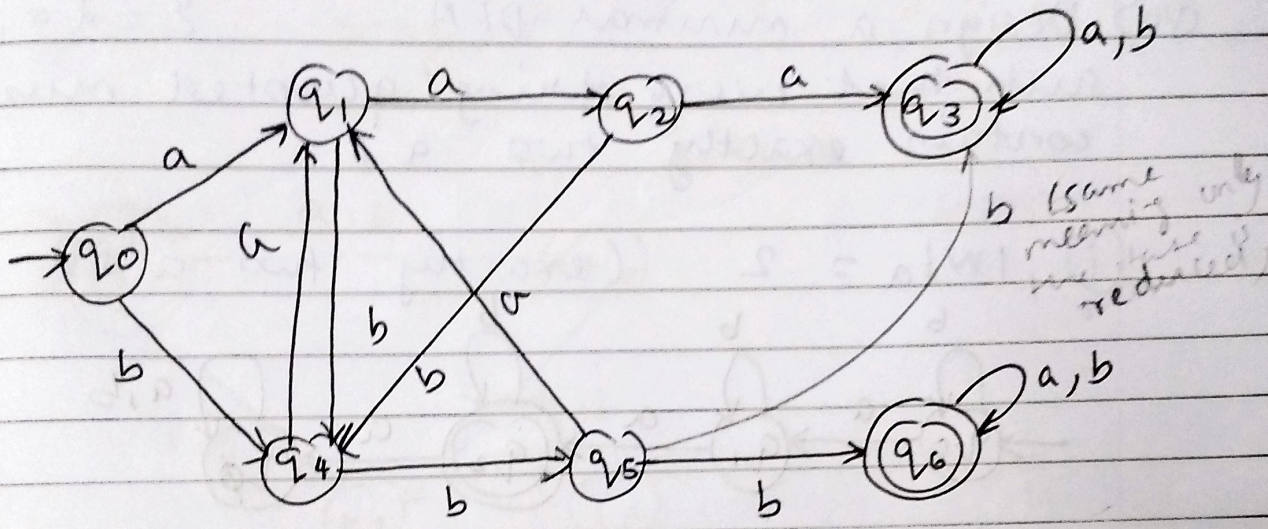
end



Q9) Design a minimal DFA ..... $\Sigma = \{a, b\}$ ..... $w = xSx$, $S = aaa/bbb$

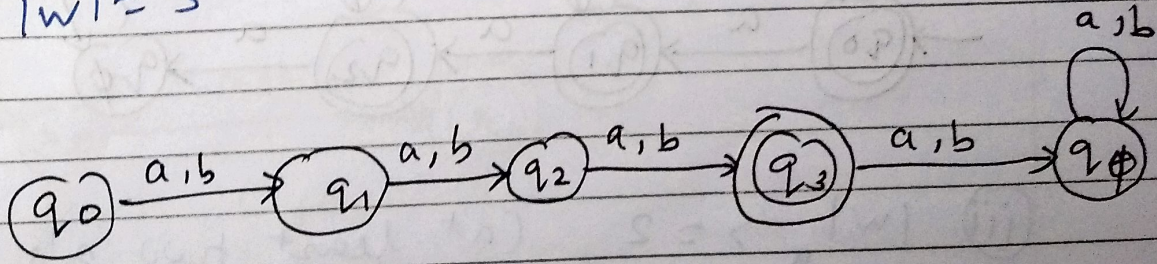
in middle or substring

=>

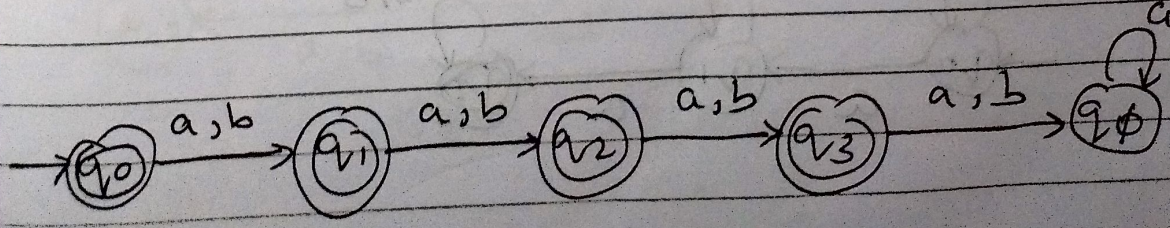


Q10) Design a minimal DFA ..... $\Sigma = \{a, b\}$, such that string 'w' must be accepted like

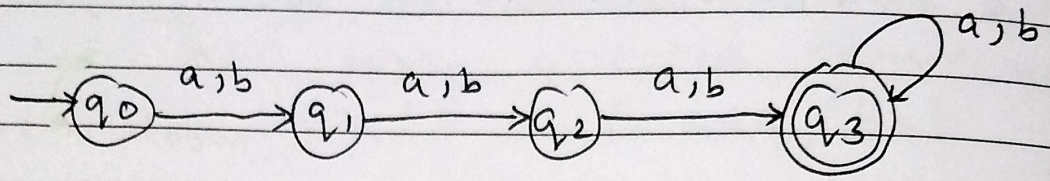
(i) $|w| = 3$



(ii) $|w| <= 3$

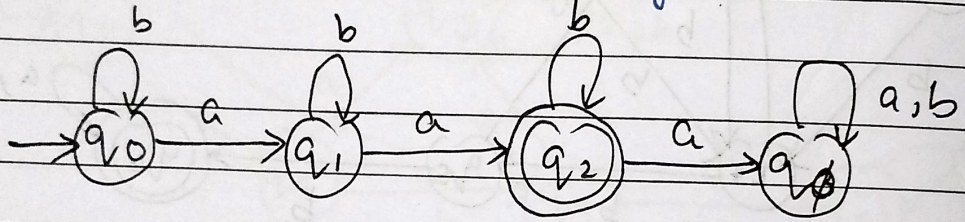


(iii) $|w| >= 3$

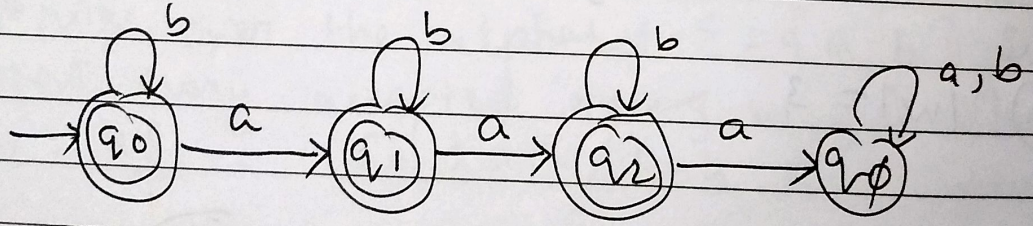


Q11) Design a minimal DFA ..... $\Sigma = \{a, b\}$, such that every string accepted must contain ..... two a's

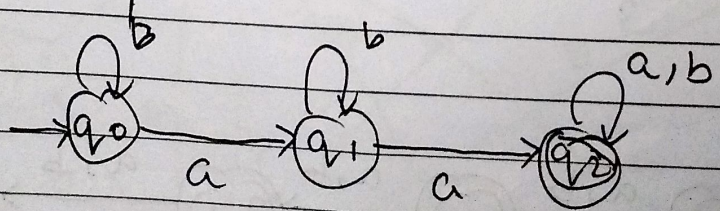
(i) $|w|_a = 2$ (exactly two a's)



(ii) $|w|_a <= 2$ (at most two a's)



(iii) $|w|_a >= 2$ (at least two a's)

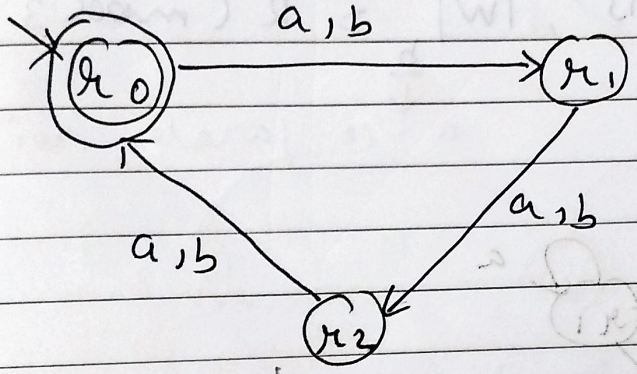


Page No.:
Date:

Q12) Design a minimal DFA ..... $\Sigma = \{a, b\}$ ..... 'w' must be accepted like

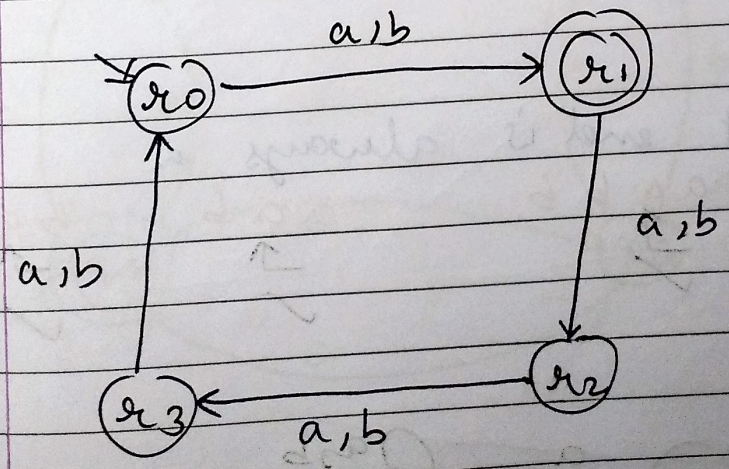
(i) $|w| = 0 \pmod 3$ (remainder)

It tells we have to get the answer as 0 by dividing it with 3 : 0, 3, 6, 9, 12, ...



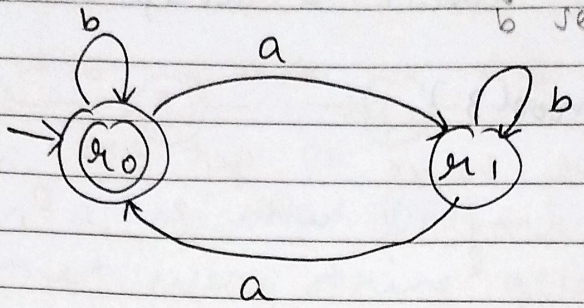
(ii) $|w| = 1 \pmod 4$ ⟹ and their GP

shortcut formula : 1, 5, 9, 13, 17, 21, ...



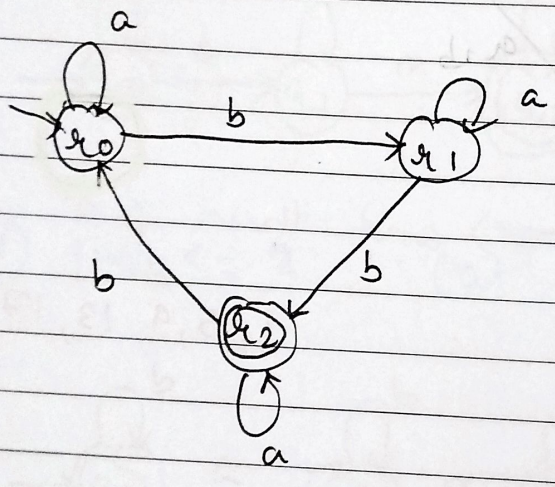
(iii) number of a is, $|w|_a = 0 \pmod 2$

b se farak nai padta



(iv) number of a is, $|w|_b = 2 \pmod 3$

a se farak nai padta



(v) 2nd from left end is always a

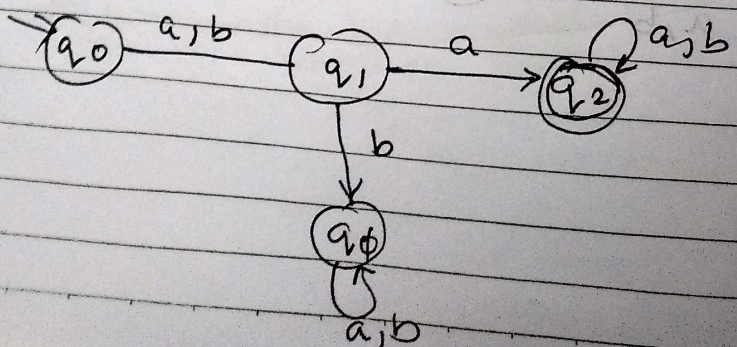
bba ✗

baba ✓

aaab ✓

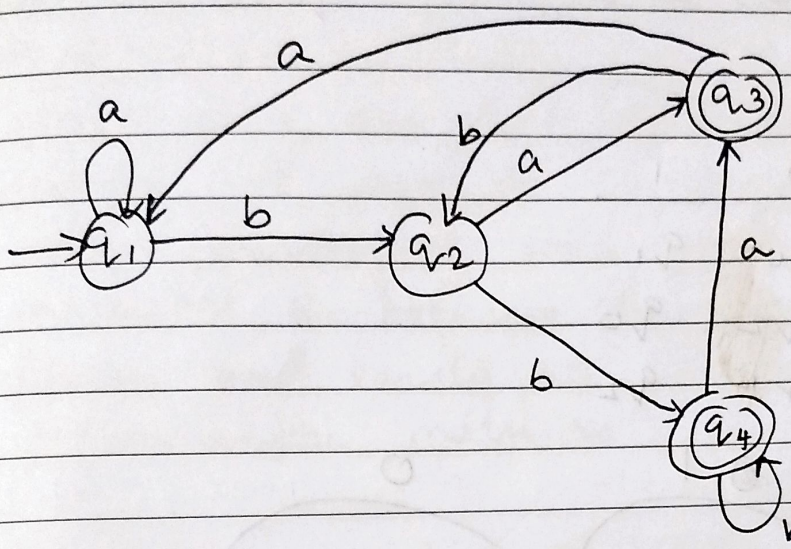
aab ✓

bab ✓

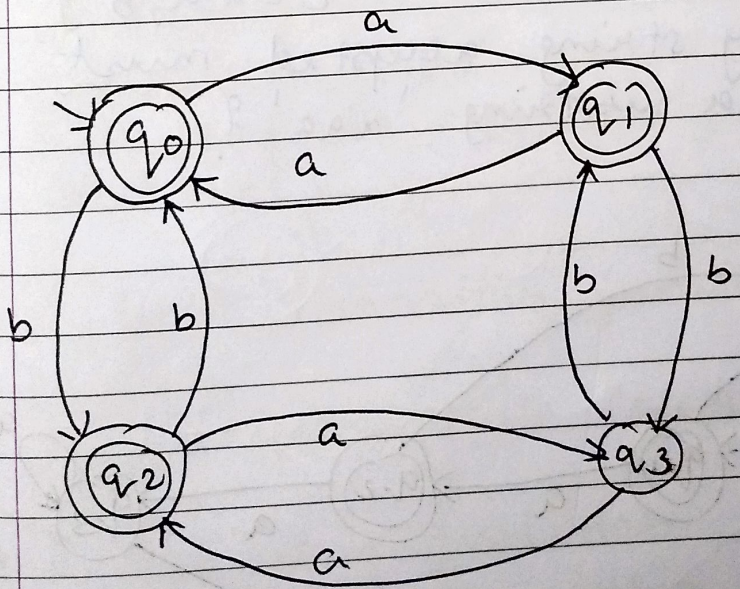


(vi) 2nd from right end is always b

2nd last alphabet must be a



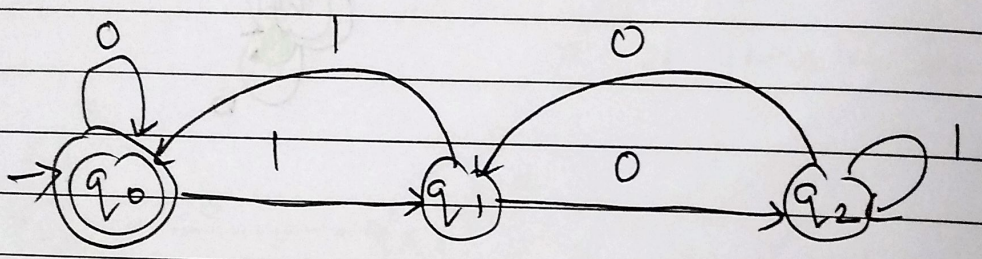
(vii) no of a = 0 (mod 2) || no of b = 0 (mod 2)



Page No.:
Date:

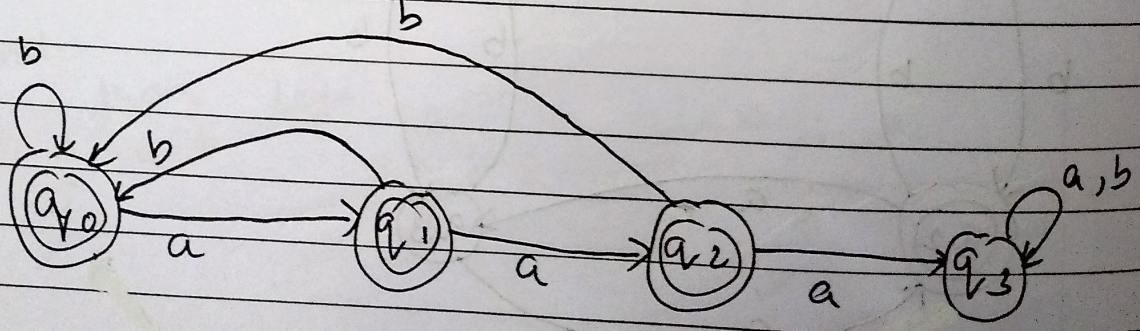
Q13) Design a minimal DFA ..... $\Sigma = \{0, 1\}$, such that every string 'w' which is accepted has a decimal equivalent $0 \pmod 3$

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



⇒ Compliment

Q14) Design a minimal DFA ..... $\Sigma = \{a, b\}$, such that every string accepted must not contain a substring 'aaa'?



Page No.:
Date:

⇒ Non deterministic finite automata

Non deterministic machine are only theoretical machine, i.e. in the first place they are not implementable and neither we want to implement them, the only reason we study non-deterministic machines is because they are easy to design and easily be converted into deterministic machines

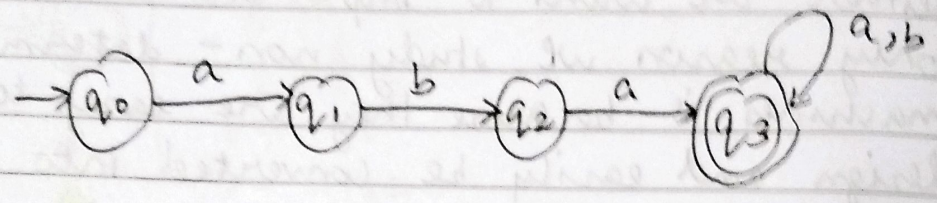
⇒ Formal description of NDFA

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

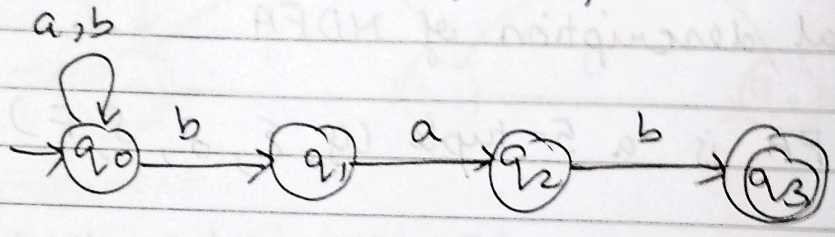
- Q is a finite and non-empty set of states
- Σ is a finite non-empty set of finite input alphabet.
- δ is a transition function $\delta : Q \times \Sigma \rightarrow 2^Q$
- q_0 is initial state (always one) $(q_0 \in Q)$
- F is a set of final states $(F \subseteq Q)$ $(0 <= |F| <= N)$ where N is the number of states

Q Design a NDFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, where every accepted string 'w'

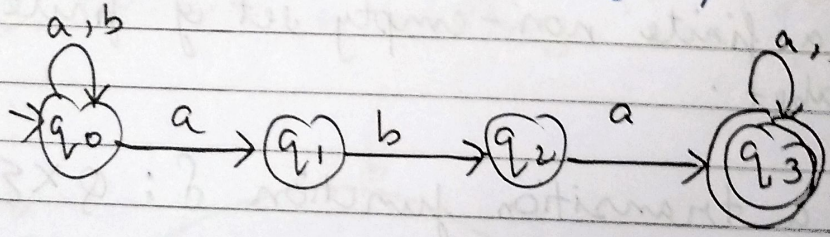
(i) Starts with substring 's', where s = 'aba'



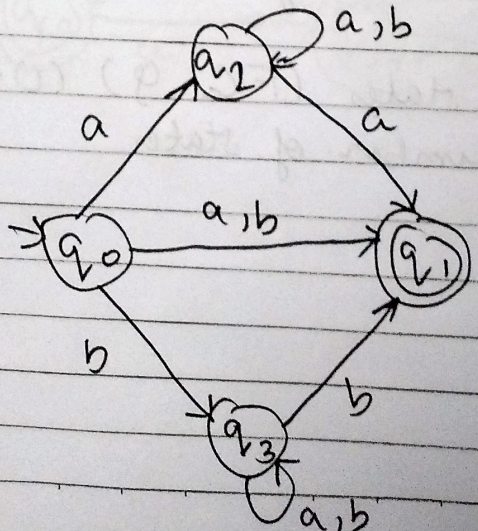
(ii) ends with substring 's', where s = 'bab'



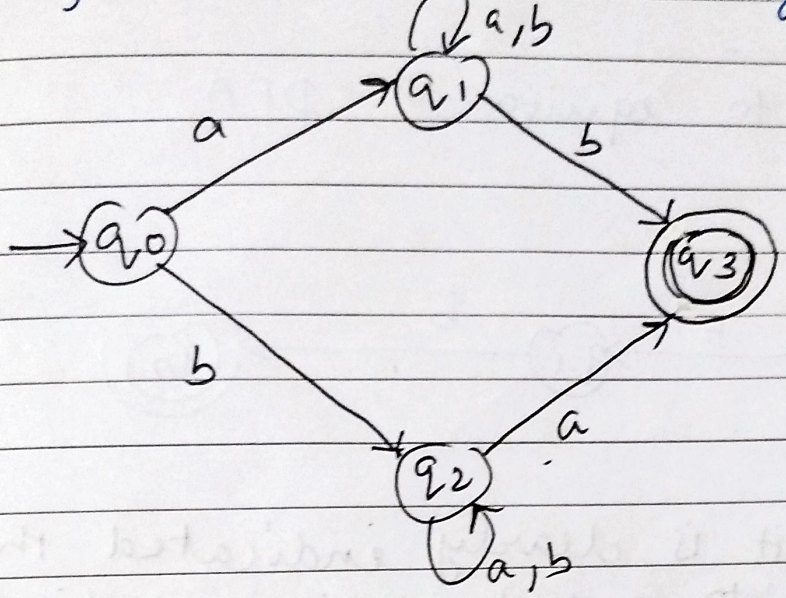
(iii) string 'w' contains substring 's', where s = 'aba'



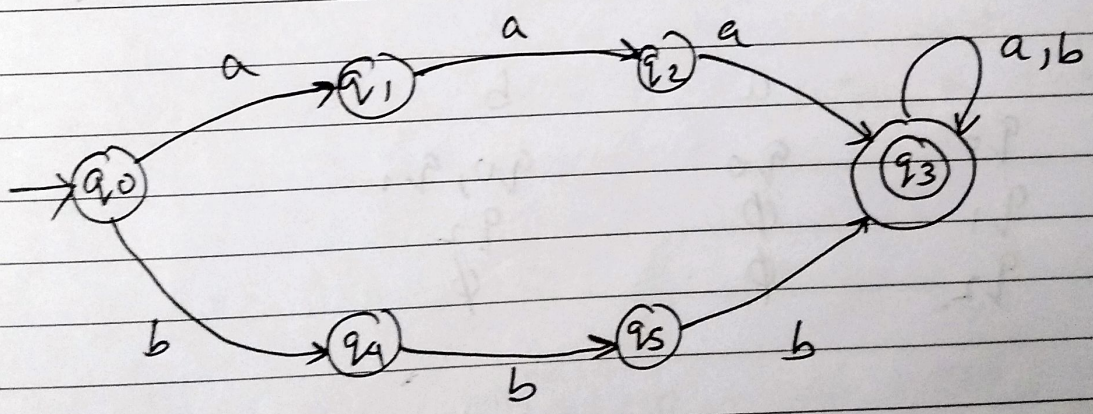
(iv) start and end with same symbol



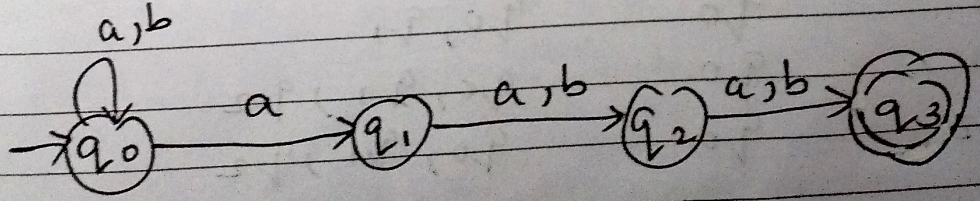
(v) string start and with different symbol



(vi) string w, is like $w = sx$, where $s = aaa/bbb$



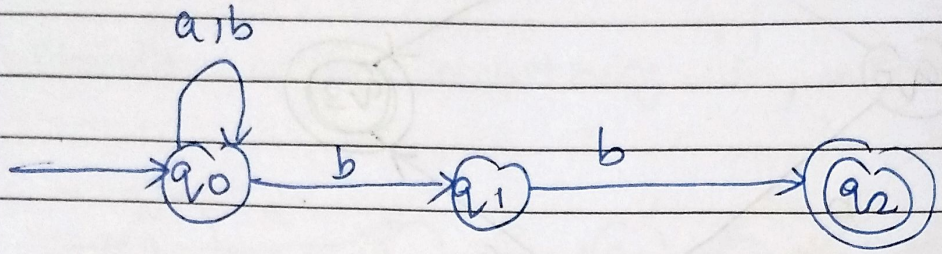
(vii) string . 3rd from right end is always a ?



⇒ NFA to DFA

Q) Convert to equivalent DFA

(i)



⇒ In this, it is clearly indicated that we have to accept strings ending with 'bb' over $\Sigma = \{a, b\}$

Step 1: Convert the NFA into a table

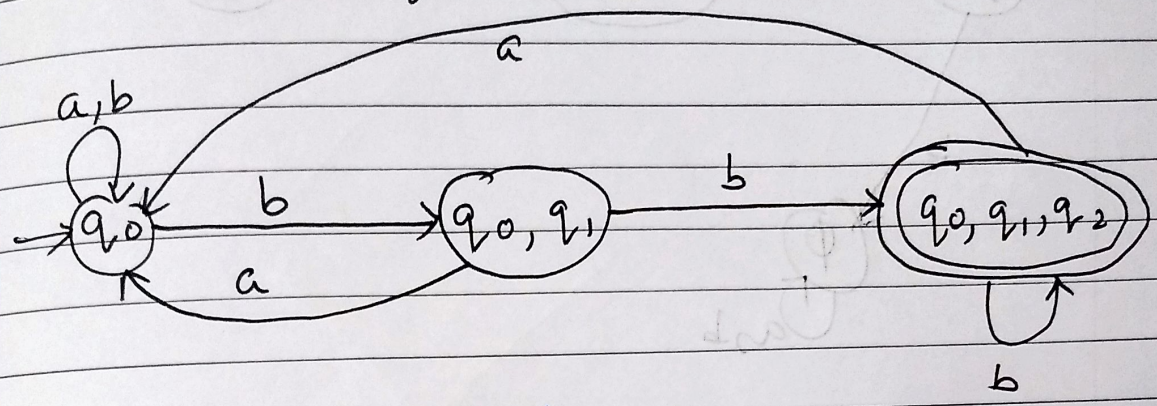
		a	b
→	q_0	q_0	q_0, q_1
	q_1	ϕ	q_2
	q_2	ϕ	ϕ

Step 2: Convert the table into DFA table

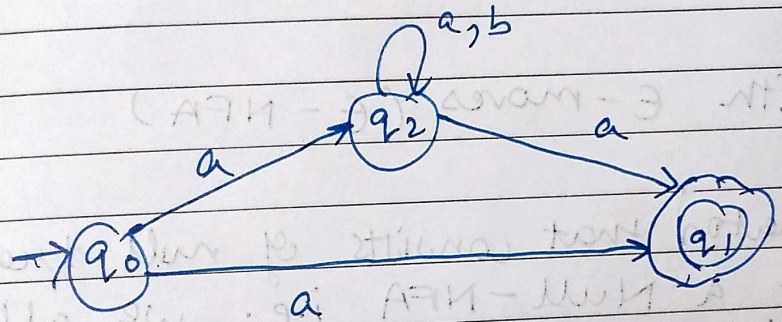
	a	b
q_0	q_0	q_0, q_1
q_0, q_1	q_0	q_0, q_1, q_2
q_0, q_1, q_2	q_0	q_0, q_1, q_2

Page No.:
Date:

Step 3: Convert the DFA table into diagram



(ii)

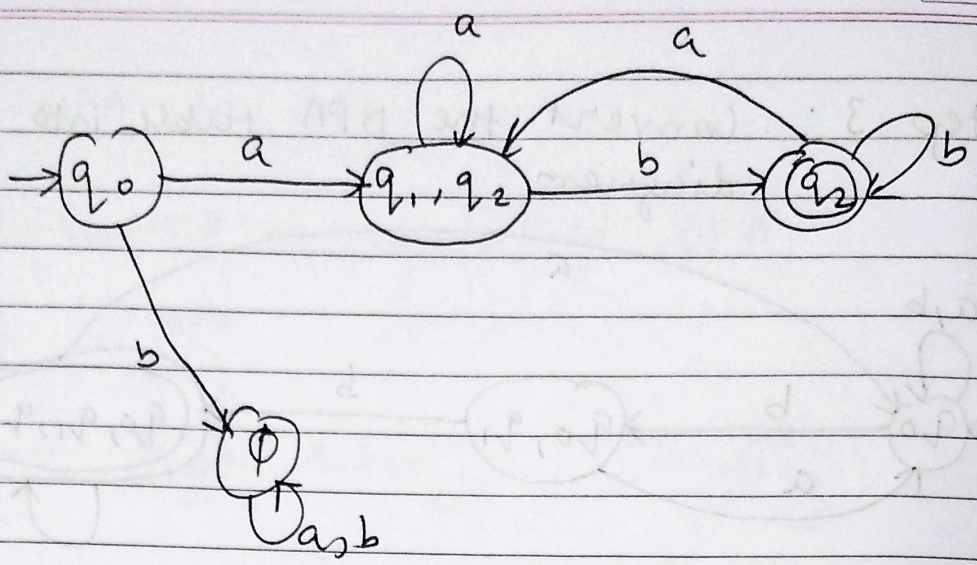


Step 1: Convert the NFA into a table

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

Step 2: Convert the table into DFA table

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



⇒ NFA with ϵ-moves (ϵ-NFA)

An automaton that consists of null transitions is called a Null-NFA i.e. we allow a transition on null means empty string.

ϵ-NFA is a 5-tuple $(Q, \Sigma, \delta, S, F)$ where:

- Q is a finite & non-empty set of states
- Σ is a finite non-empty set of finite input alphabet
- δ is a transition function:
  $\delta : (Q \times \{\Sigma \cup \epsilon\}) \longrightarrow 2^Q$
- S is initial state (always one) $(S \in Q)$
- F is a set of final states

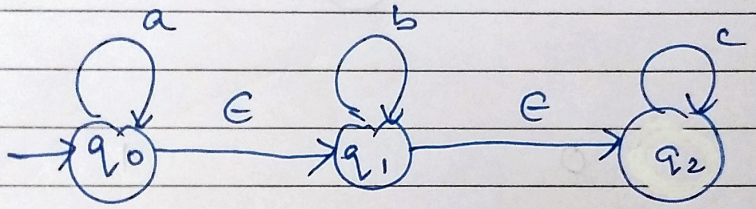
Page No.:
Date:

⇒ Null closure

Null closure of set Q is defined as a set of all the states, which are at zero distance from the state Q.

ϵ-closure (q_i) - The set of all states which are at zero distance from the state q_i is called ϵ-closure (q_i)

Eg:



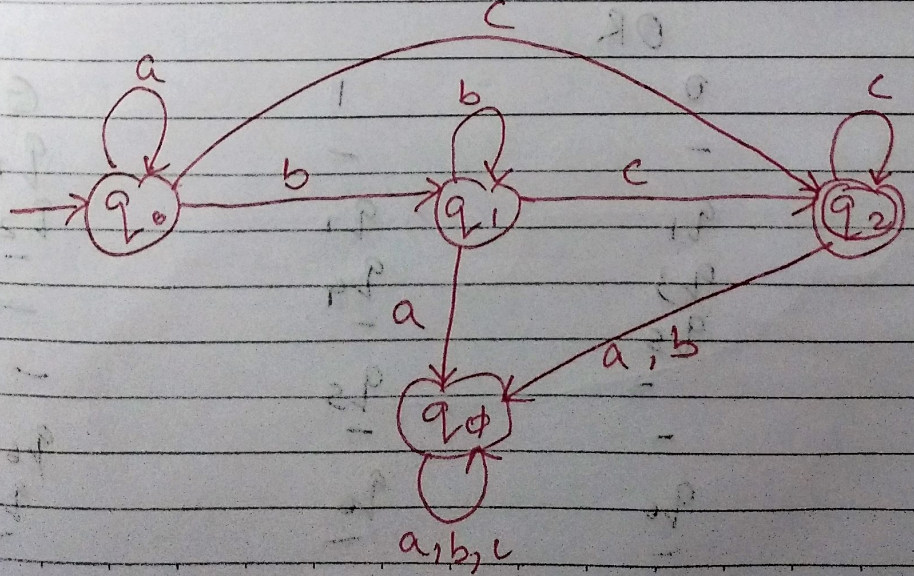
$L = \{a^n b^m c^q \mid m, n, q \geq 0\}$

ϵ-closure $(q_0) = q_0, q_1, q_2$

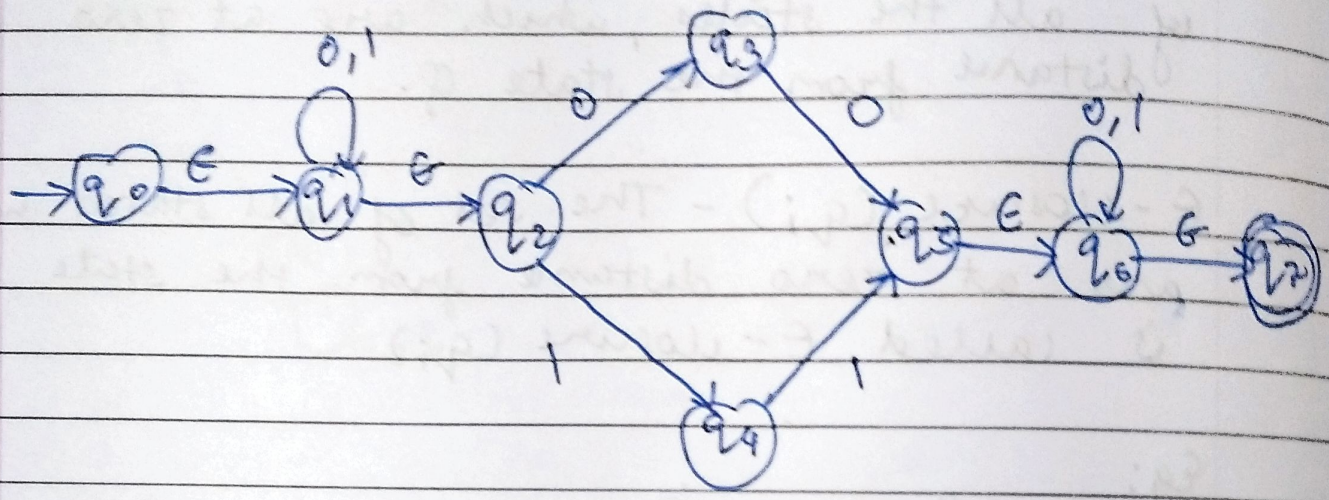
$q_1 = q_1, q_2$

$q_2 = q_2$

the DFA of the above dig is :



Q) Convert ε-NFA to NFA



⇒

	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	$\emptyset$
q_4	$\emptyset$	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	$\emptyset$	$\emptyset$

Page No.:
Date:

⇒ Moore and Mealy machine

★ Both moore and mealy machines are special case of DFA.

★ Both acts like O/P producers rather than language acceptors.

★ In moore and mealy machine, no need to define the final states.

★ No concepts of dead states and no concepts of final states.

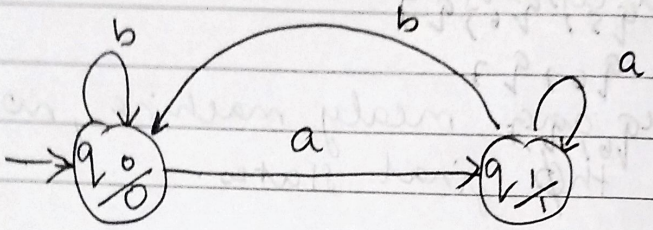
★ Mealy and moore machines are equivalent in power.

⇒ Moore machine

- A moore machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where

- Q is a finite set of states
- Σ is the input alphabet
- Δ is the output alphabet
- δ is the transition function $Q \times \Sigma$ into Q
- λ is the output function mapping Q into Δ
- q_0 is the initial state.

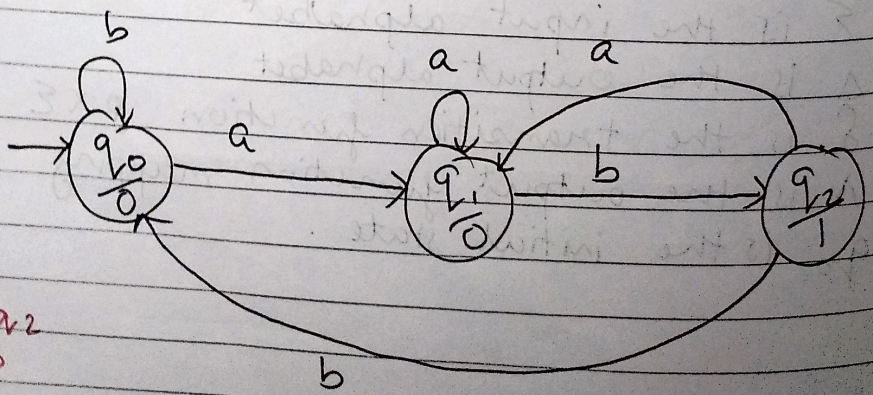
Q1) Construct a moore machine take all the string of a's and b's as input and counts the number of a's in the input string in terms of 1
$\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$



everytime an a is given 1 should be generated

	q_1	q_0	q_0	q_1
q_0	a	b	b	a
↓	↓	↓	↓	↓
0	1	0	0	1

Q2) Construct a Moore machine take all strings of a's and b's as input and counts the number of occurence of sub-string 'ab' in terms of 1, $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$



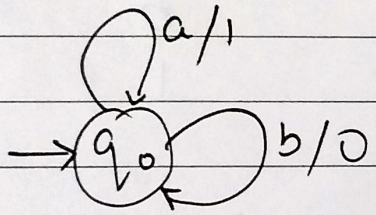
	q_1	q_2	q_0	q_1	q_2
q_0	a	b	b	a	b
↓	↓	↓	↓	↓	↓
0	0	1	0	0	1

Page No.:
Date:

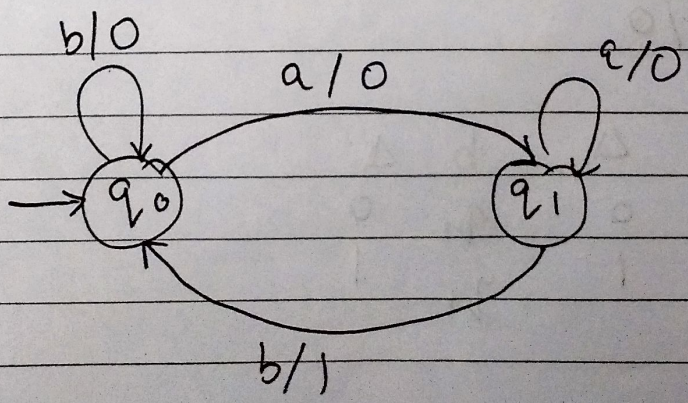
⇒ Mealy machine

- In case of mealy machine, the output symbol depends on the transition.
- Mealy machine do not response for empty string ϵ

Q1) Construct a mealy machine take all the string of a's and b's as input and counts the number of a's in the input string in terms of 1, $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$

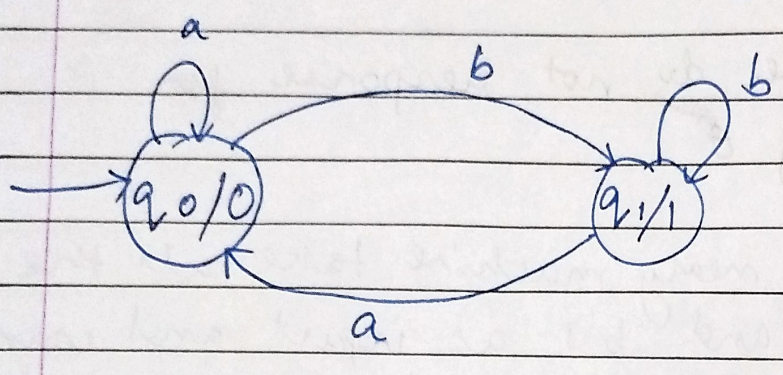


Q2) Construct a mealy machine take all strings of a's and b's as input and counts the number of occurence of sub-string 'ab' in terms of 1, $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$



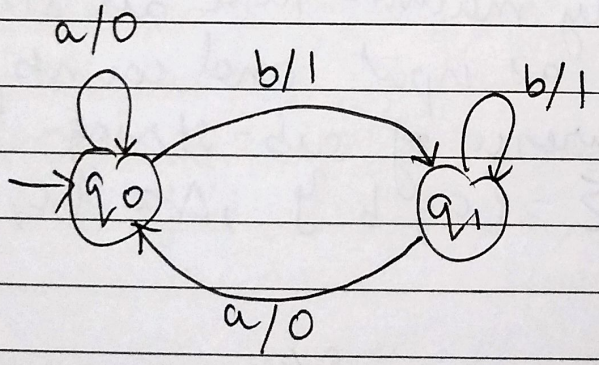
⇒ Conversion of Moore to Mealy machine

Q) Convert the following Moore machine into its equivalent Mealy machine.



Q	a	b	Output (λ)
q_0	q_0	q_1	0
q_1	q_0	q_1	1

⇒

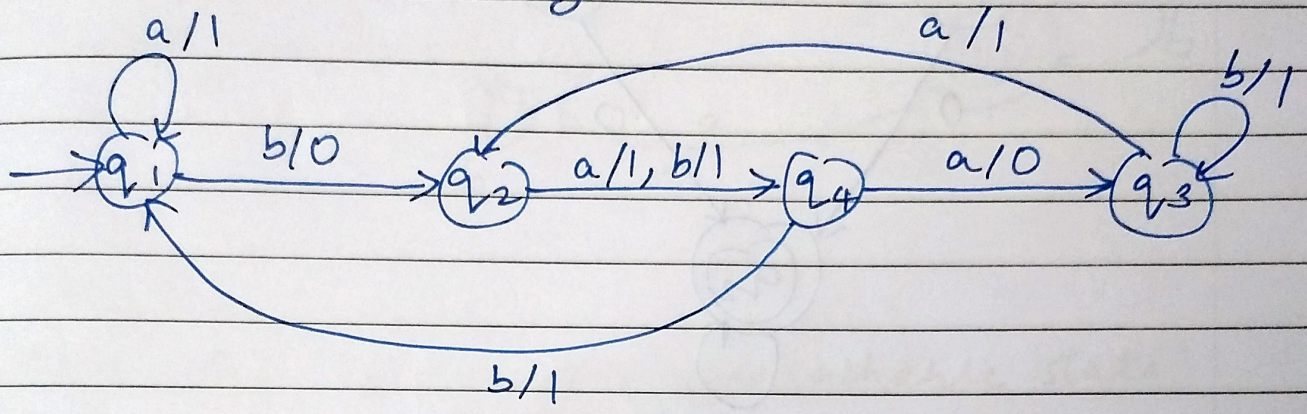


Q	a	Δ	b	Δ
q_0	q_0	0	q_1	0
q_1	q_0	1	q_1	1

Page No.:
Date:

⇒ Procedure for transforming a mealy machine to moore machine

Q) Consider the Mealy Machine:



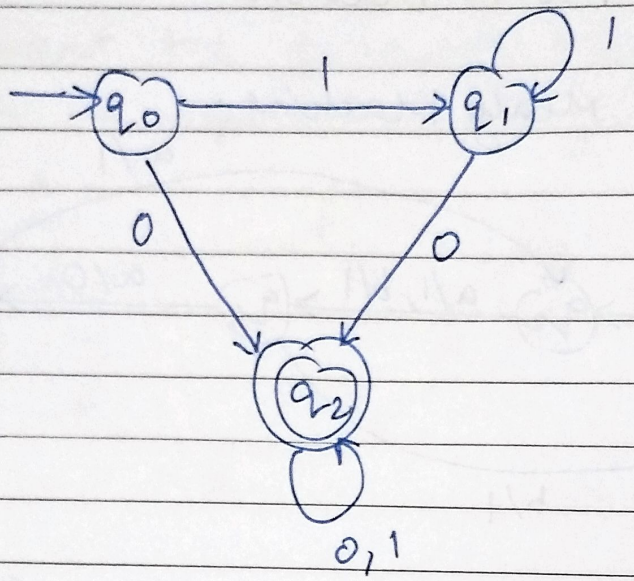
Present state	Next state			
	a		b	
	State	O/P	State	O/P
q_1	q_1	1	q_2	0
q_2	q_4	1	q_4	1
q_3	q_2	1	q_3	1
q_4	q_3	0	q_1	1

⇒

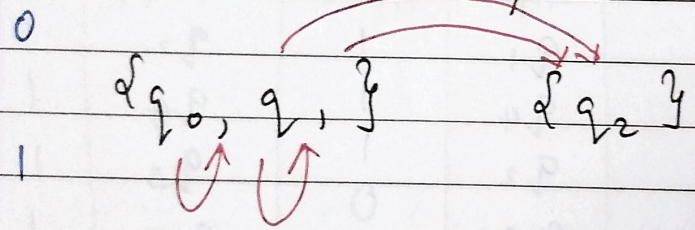
q	a	b	O/P (λ)
q_1	q_1	q_{20}	1
q_{20}	q_4	q_4	0
q_{21}	q_4	q_4	1
q_{30}	q_{21}	q_{31}	0
q_{31}	q_{21}	q_{31}	1
q_4	q_{30}	q_1	1

⇒ Minimization of DFA

Q1)

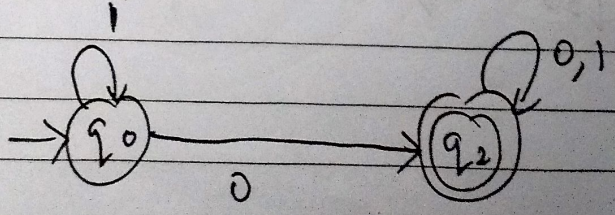


⇒ Step 1: Group them in dead state, unreachable state, initial state, final state

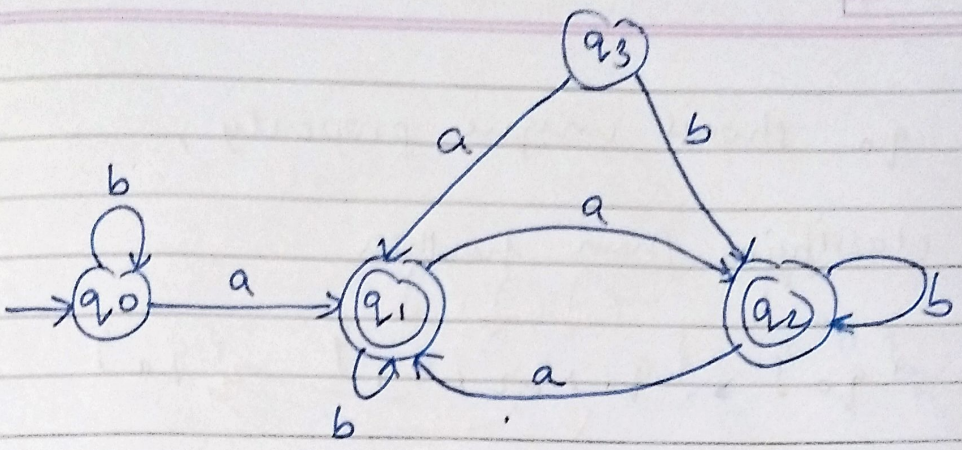


Step 2: Eliminate if found same characteristic, else divide in another group

Here we will eliminate, q_1 and remove its outgoing arrows, the incoming arrows will be given to the same group state



Q2)



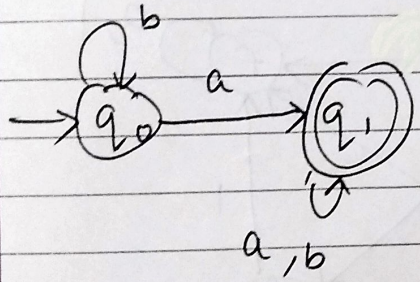
=> Step 1:

Here q_3 is an unreachable state

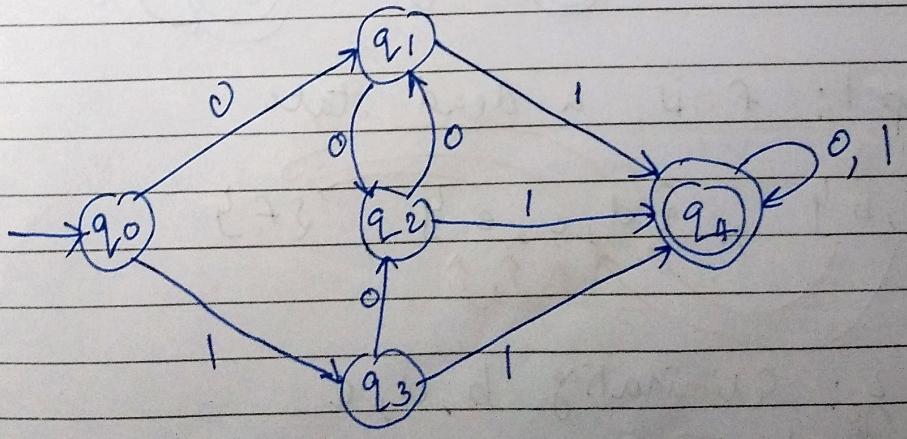
	a	b
$\{q_0\}$	$\{q_1, q_2\}$	
$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_1, q_2\}$
$\{q_3\}$		

Step 2:

Eliminating q_3 (unreachable state) & q_2



Q3)

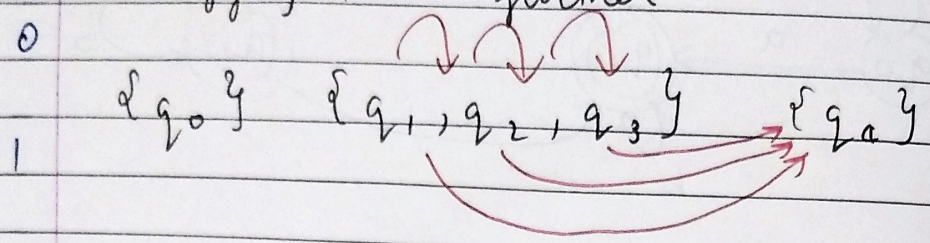


Step 1:

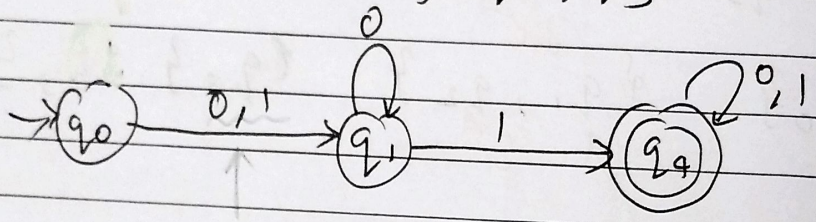
	0	1
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_4\}$
$\{q_4\}$	$\{q_4\}$	$\{q_4\}$

q_0 shows unique property,

classifying them further

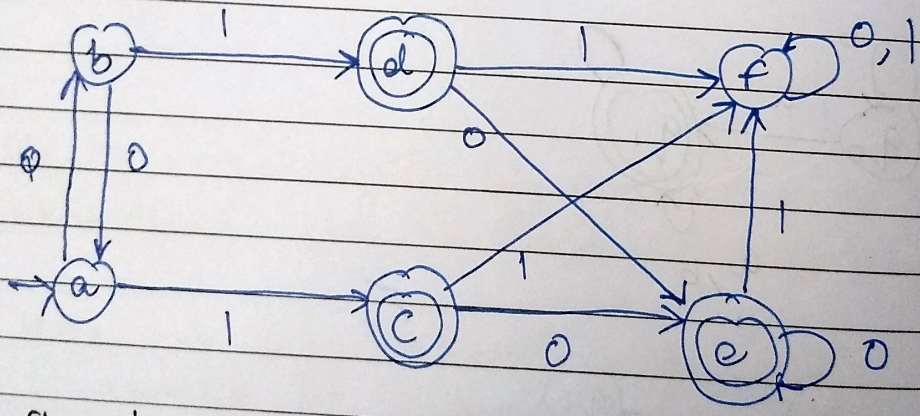


Step 2: Eliminating q_2, q_3

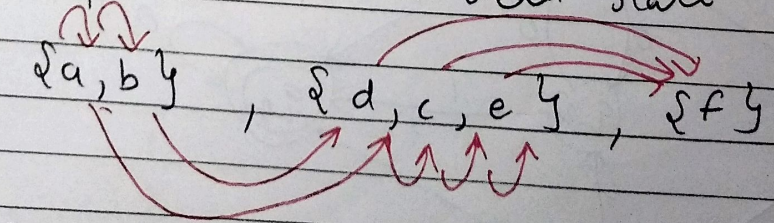


will go to the same group

q4)



⇒ Step 1: f is a dead state



Step 2: Eliminating b, d, e