1. Finite Automata and regular expression

1.1. Review of set theory, proof techniques

1.2 Finite state system/machine (FSM)

A Finite State Machine is a mathematical model that is used to explain and understand the behaviour of a digital system. FSM are used in many digital systems to control the behaviour of systems and data flow paths. They can be implemented in software or hardware to simplify complex problems.

It is a concept widely used in computer science, engineering and other fields to design, analyze and implement system with discrete and sequential behaviour.

A Finite State Machine or Finite State Automaton or simply a state machine is a model of behaviour i.e., it consists of a set of states, input and transition function that maps i/o symbols and current state to next state. In finite state machine, a current state is determined by the past state of the system. Transition indicates a state change and is described by a condition that would need to be full filled to.

HK1

enable transition. An action is a description of an activity i.e. to be performed at a given moment. The several types of actions are

(i) entry action
(ii) Input action
(iii) Transition action
(iv) Exit action

An automaton can be represented using a directed graph with vertices representing the states and edges representing the transiting the transition. A finite state machine is an abstract model of a machine with a primitive internal memory.

In computer science, Finite State Machine is widely used in modeling of application behaviour, design of hardware in digital system software engineering, compiler design, network protocols and so on.

HKQ

✡ Finite Automata

The finite automata resembles with real computer in the fact that it has CPU of fixed capacity i.e. Finite Memory.

Formal definition of an automata

Automaton is represented by 5-tuple $M = \{Q, \Sigma, \delta, q_0, F\}$, where

Q = Set of states

Σ = Finite set of symbols i.e. alphabet of the language that the automaton accept

$\delta = Q \times \Sigma \to Q$ is a function, called the transition function.

q_0 = It is an element of Q. It is called start state.

F = It is a subset of Q called the accepting state or Final state.

✡ Classes of Finite Automata

(i) Deterministic Finite Automata (DFA)

(ii) Non-Deterministic Finite Automata (NFA)

HK3

✡ Deterministic Finite Automata (DFA)

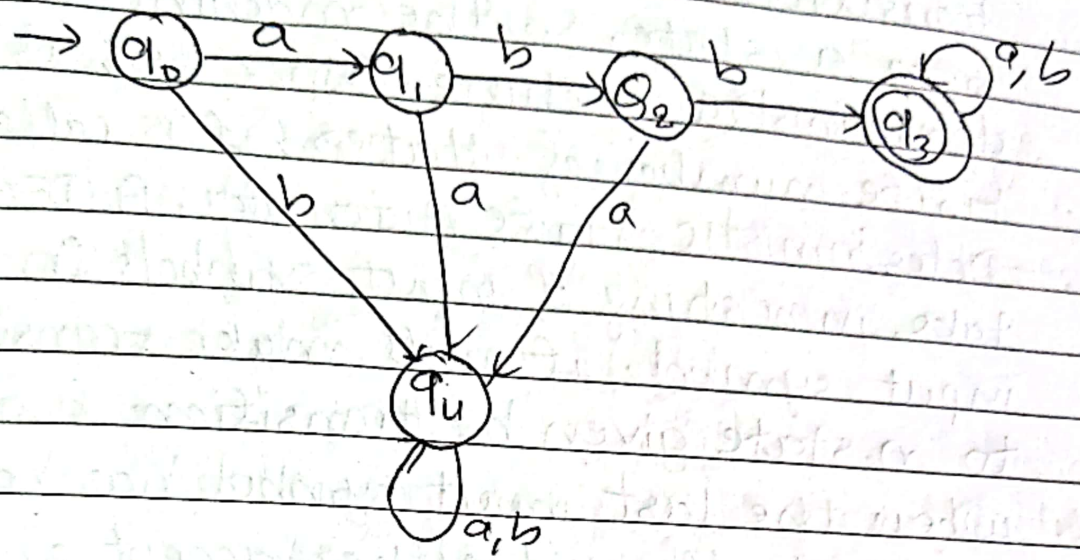
Deterministic Finite Automata is a Finite state machine that accepts or rejects strings of symbols and only produces a unique computation of the automaton for each input string. The Deterministic refers to the uniqueness of the computation. In DFA, for each input symbols, one can determine a unique state to which the machine will move.

DFA is defined by the quintuple i.e. 5 tuple $M=\{Q, \Sigma, \delta, q_0, F\}$, where

Q = Non-empty finite set of state
Σ = Non-empty finite set of input alphabet
δ = It is a transition function from $Q \times \Sigma$ to Q
q_0 = Start state
F = Final state or Accepting state

HKM

Example : DFA to accept ~~string~~ start with string abb.



Transition table

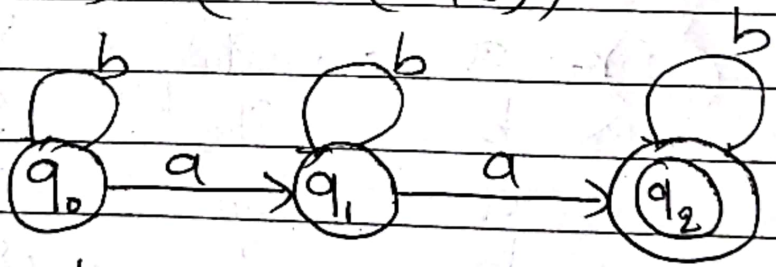
Current State	Input Symbol	Next State	Transition function
q_0	a	q_1	$\delta(q_0, a) = q_1$
q_0	b	q_4	$\delta(q_0, b) = q_4$
q_1	a	q_4	$\delta(q_1, a) = q_4$
q_1	b	q_2	$\delta(q_1, b) = q_2$
q_2	a	q_4	$\delta(q_2, a) = q_4$
q_2	b	q_3	$\delta(q_2, b) = q_3$
q_3	a	q_3	$\delta(q_3, a) = q_3$
q_3	b	q_3	$\delta(q_3, b) = q_3$
q_4	a	q_4	$\delta(q_4, a) = q_4$
q_4	b	q_4	$\delta(q_4, b) = q_4$

For each input symbol from a.

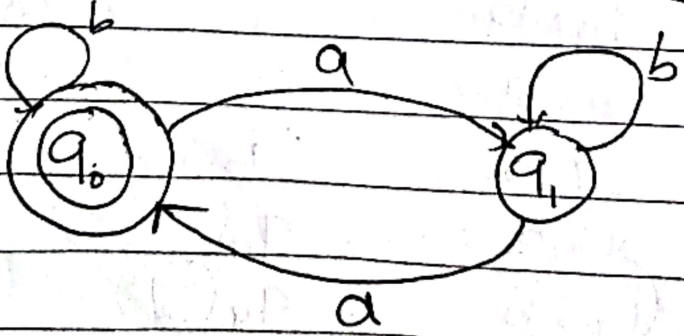
HKS

given state there is exactly one transition or there can be no transition from a state. So, the machine is called deterministic machine. Since, it has Finite number of states, it is called Deterministic Finite Automata. A DFA will take in a string of input symbols for each input symbol, it will make transition to n state given by transition function. When the last input symbol has been recieved, it will either accept or reject the string depending on whether it is an accepting state or non accepting state.

Q.1. Design a DFA to accept even numbers of a's. ($\Sigma = (a, b)^*$)

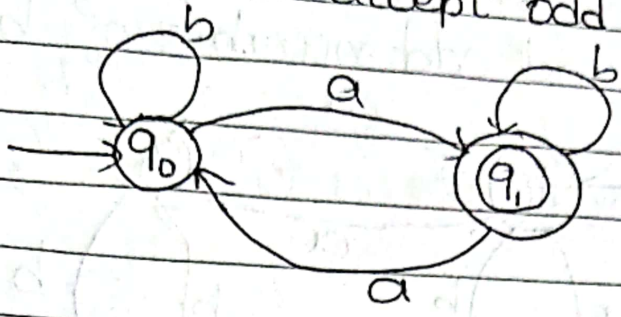


or,

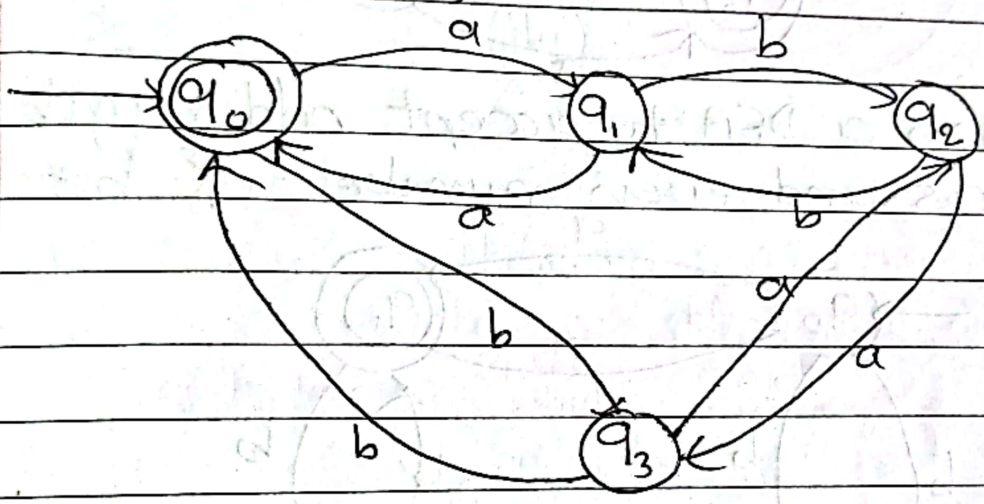


HK6

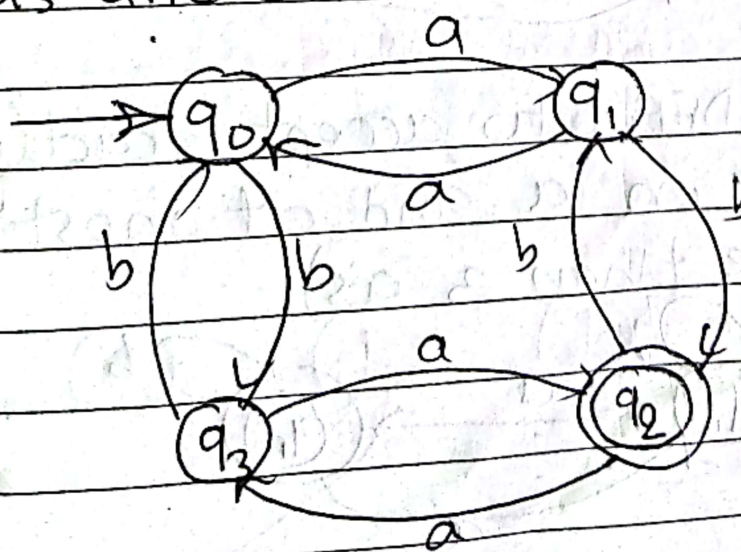
Q.2. Design a DFA to accept odd number of a's.



Q.3. Design a DFA to accept even number of a's and b's.

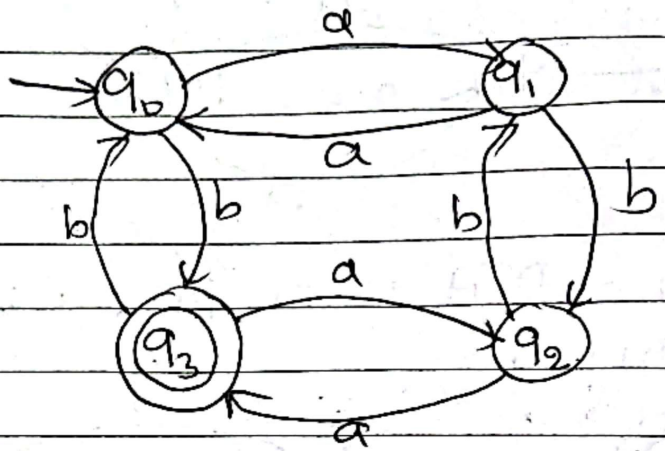


Q.4. Design a DFA to accept odd number of a's and b's.

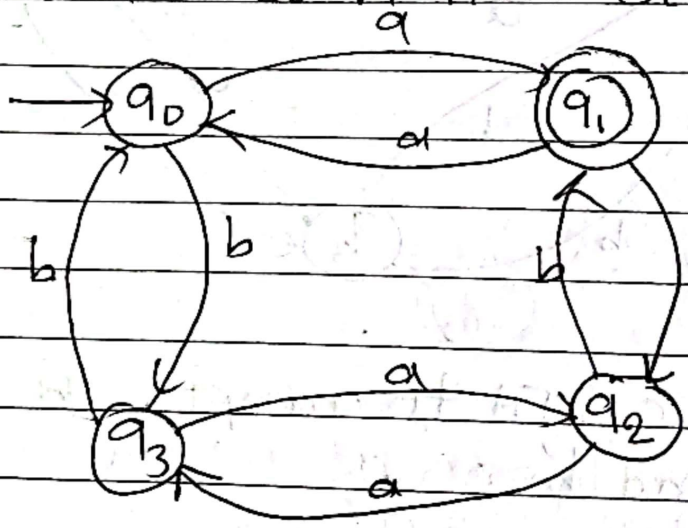


HK7

Q.5. Design a DFA to accept even number of a's and odd number of b's.



Q.6. Design a DFA to accept odd number of a's and even number of b's.



Q.7. Design a DFA's to accept exactly on a, at least on a and at most 3 a's (not more than 3 a's)

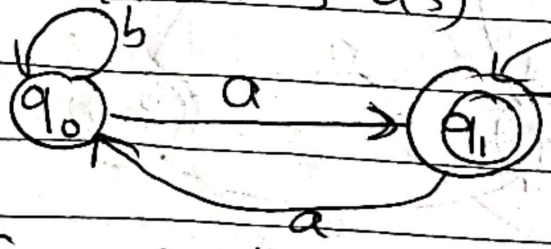


fig: DFA to accept exactly one a

HKB

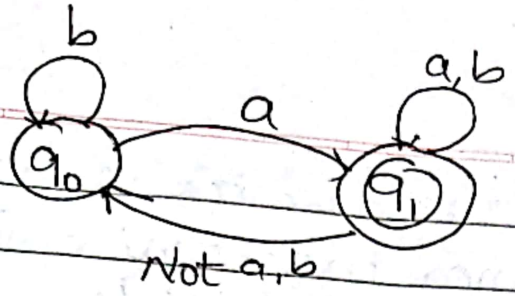


fig: DFA to accept at least one a

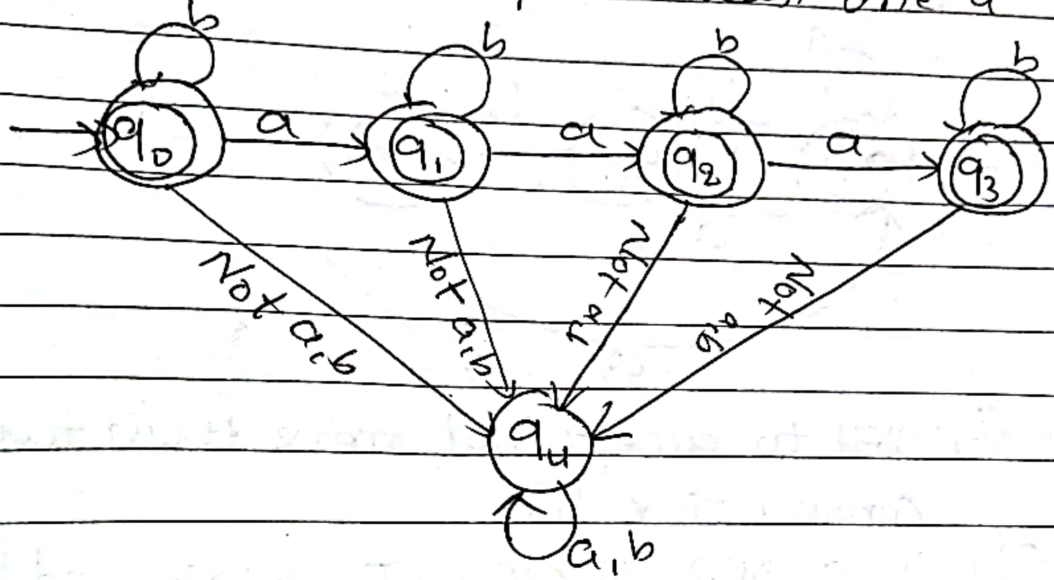


fig: DFA to accept not more than 3 a's

Q.8. Design a DFA to accept string 011 over the alphabet $\Sigma = \{0, 1\}$.

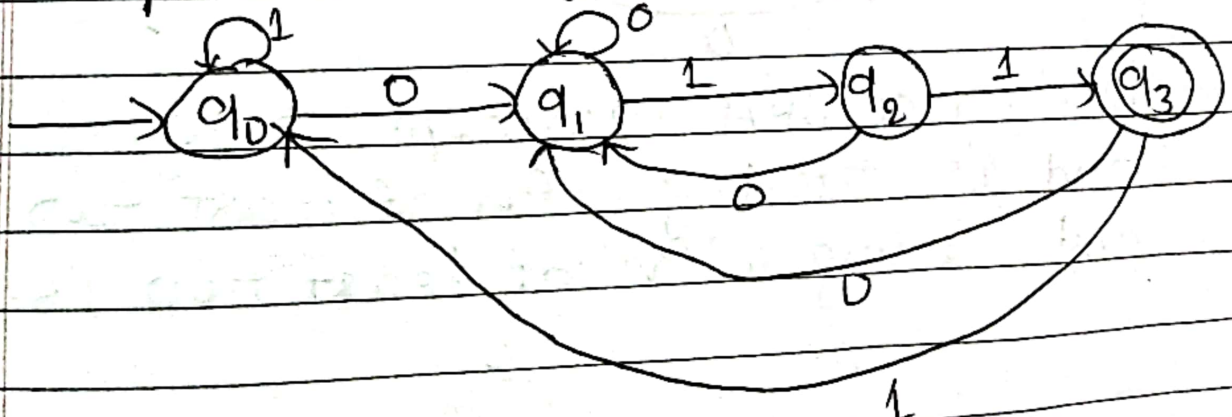


fig: DFA ~~to~~ accept string 011

HKG

Q.9. Design a DFA to recognize string which contain not more than two consecutive b's over alphabet $\Sigma = \{a, b\}$
(at most two consecutive b's)

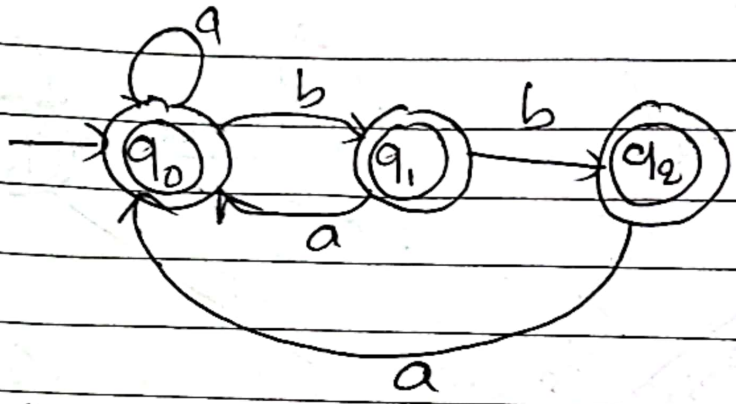
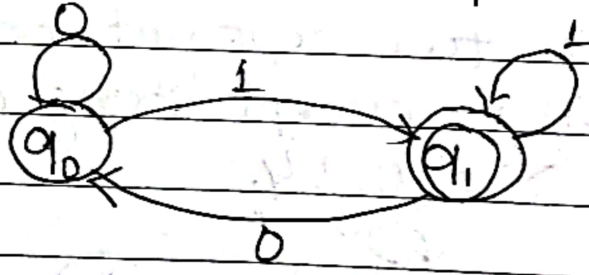


fig: DFA to accept not more than two consecutive b's.

Q.10. Obtain a DFA to accept binary odd numbe



Q.11. Design a DFA to accept string of 0's and 1's starting with at least two 0's and ending with at least two 1's.

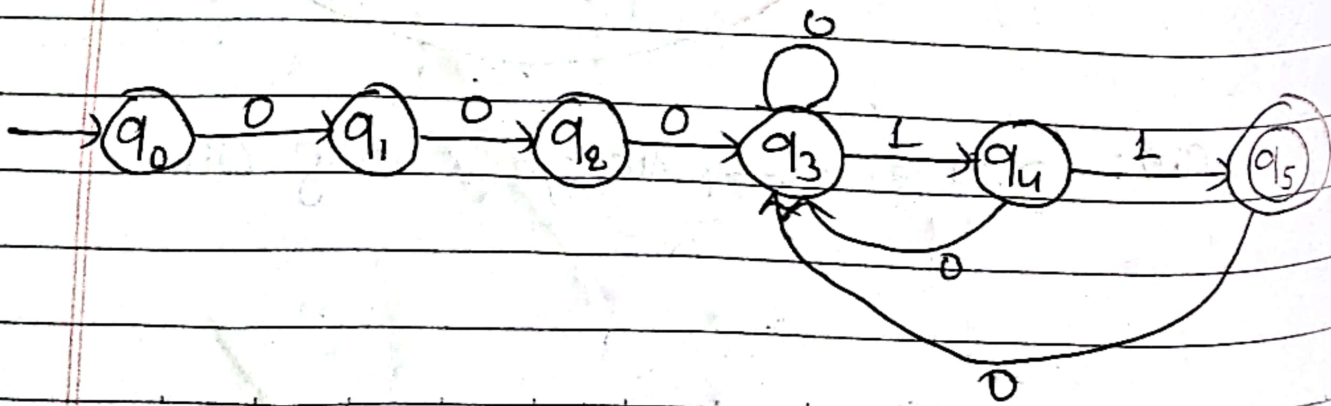
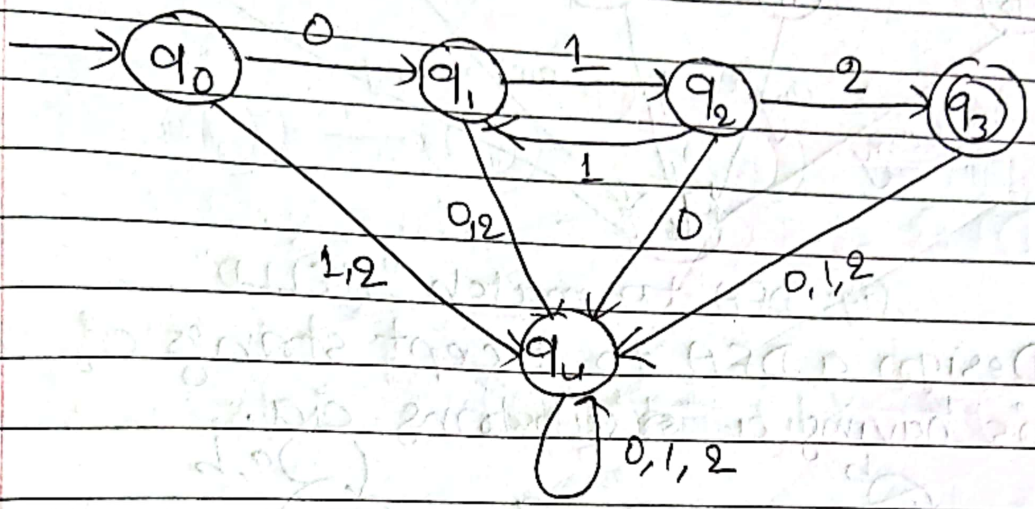


fig: DFA to accept starting with at least two 0s and ending with at least two 1's

Q.12. Design a DFA to accept string of 0's, ~~and~~ 1's and 2's beginning with '0' followed by odd number of 1's and ending with a 2'.



Q.13. Design a DFA to accept strings of a's and b's except (besides) those containing the substring aab.

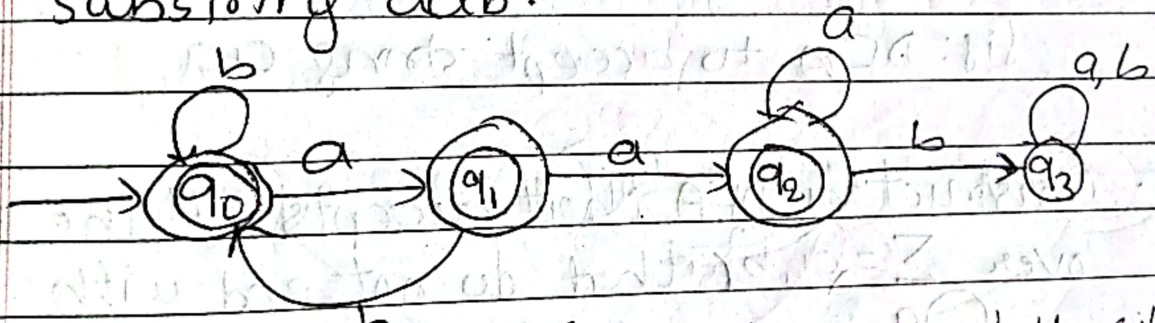


Fig:- DFA to accept the string except the substring aab.

Q.14. Obtain a DFA to accept string of a's and b's starting with the string ab.

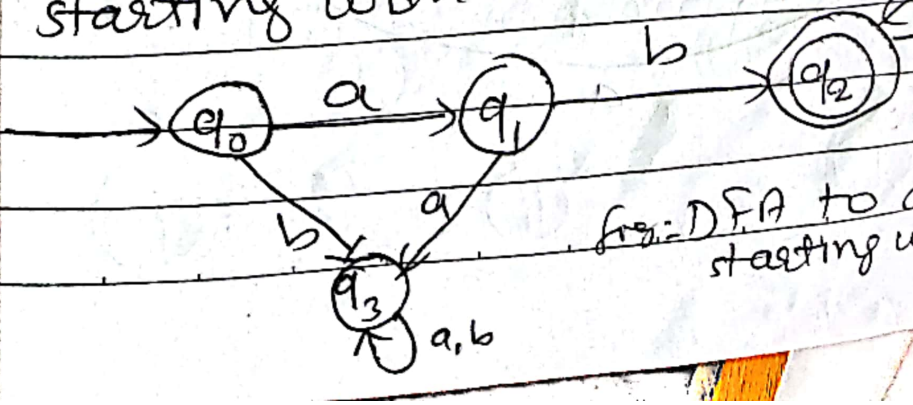


Fig:- DFA to accept string which starting with ab.

Q.15. Design a DFA to accept the string "HELLO"

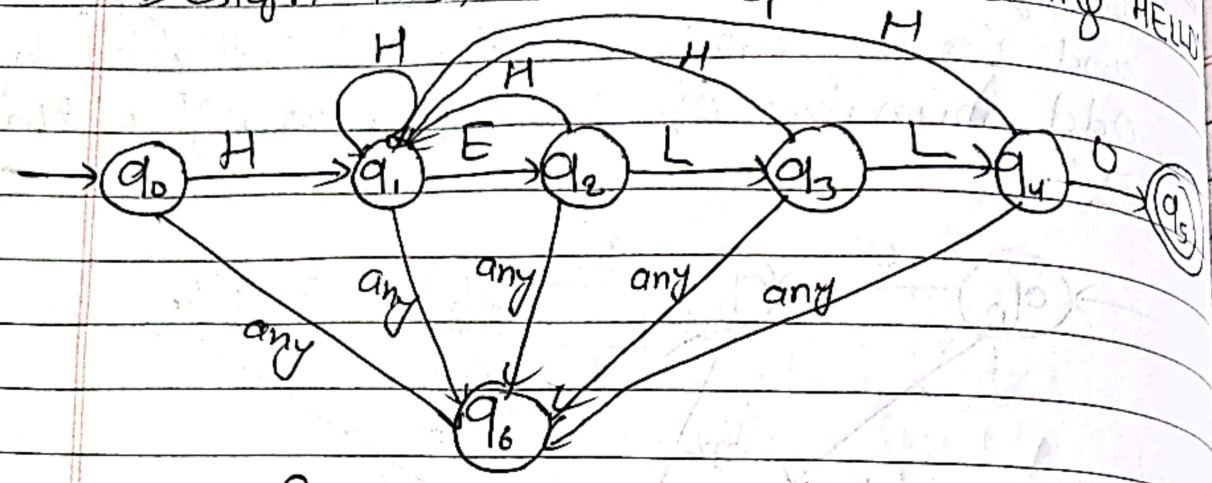


fig: DFA to match "HELLO"

Q.16. Design a DFA to accept strings of a's or b's having a substring aa.

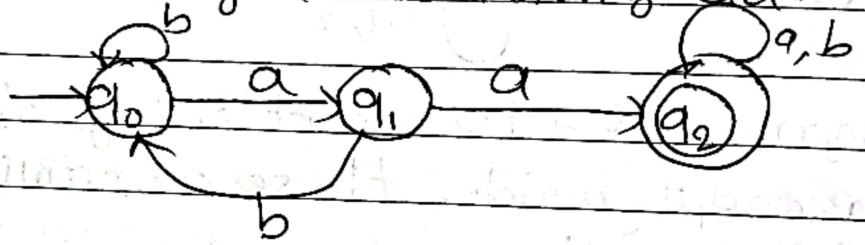
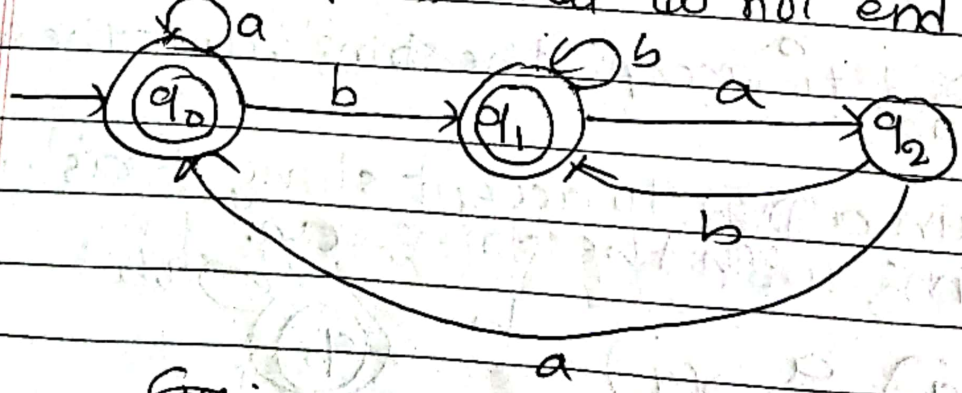


fig: DFA to accept string aa

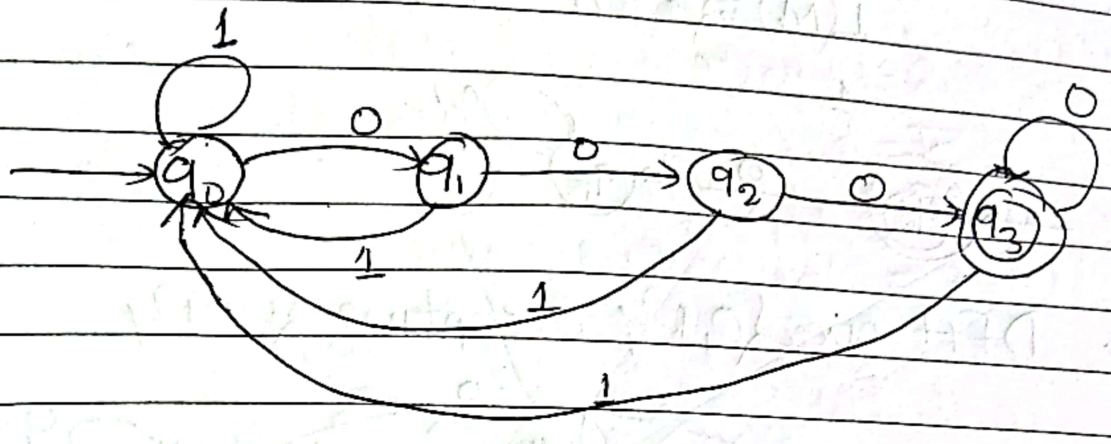
Q.17. Construct a DFA that accepts all the strings over $\Sigma = \{a, b\}$ that do not end with ba



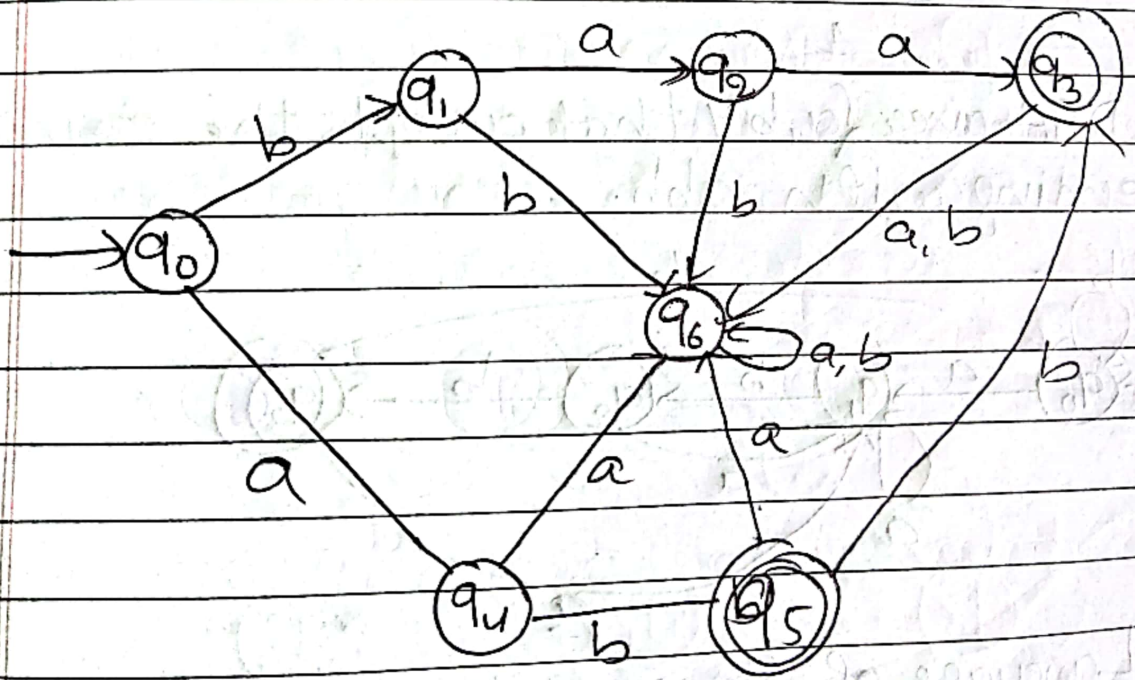
Soln:

HK12

Q.18. DFA to accepting all string over $\Sigma = \{0,1\}$ ending with 3 consecutive 0's.



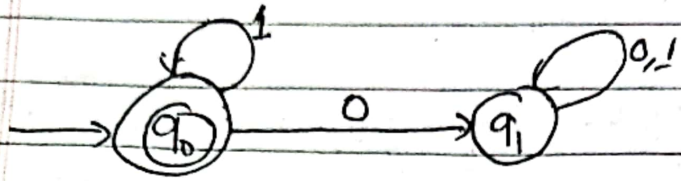
Q.19. DFA over (a,b) accepting {baa, ab, abb}



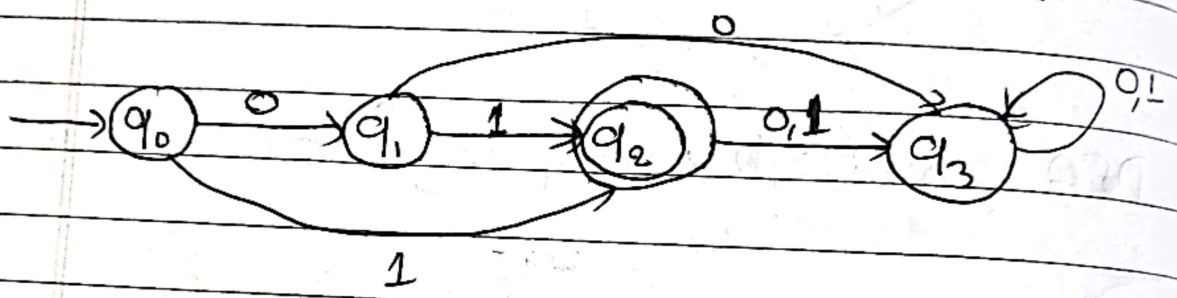
HK 13

Q.20 DFA ~~over~~ accepting zero or more consecutive 1's.

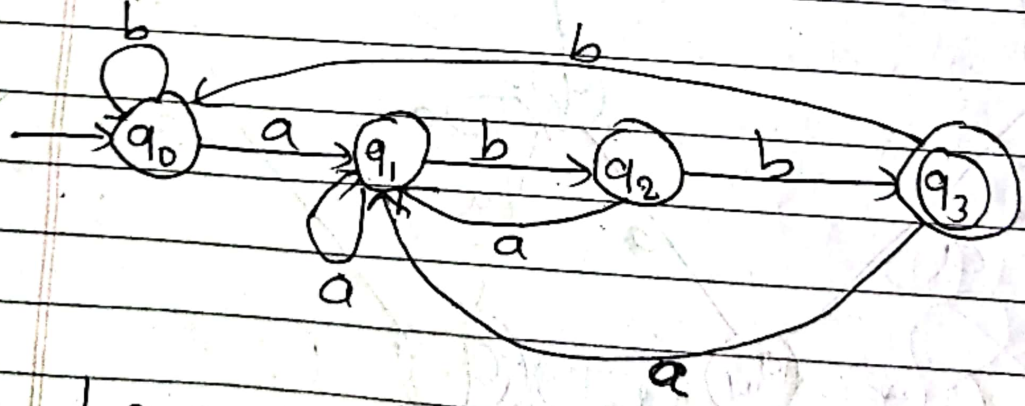
$$L(M) = \{1^n / n = 0, 1, 2, \ldots\}$$



Q.21. DFA over {0,1} accepting {1,01}

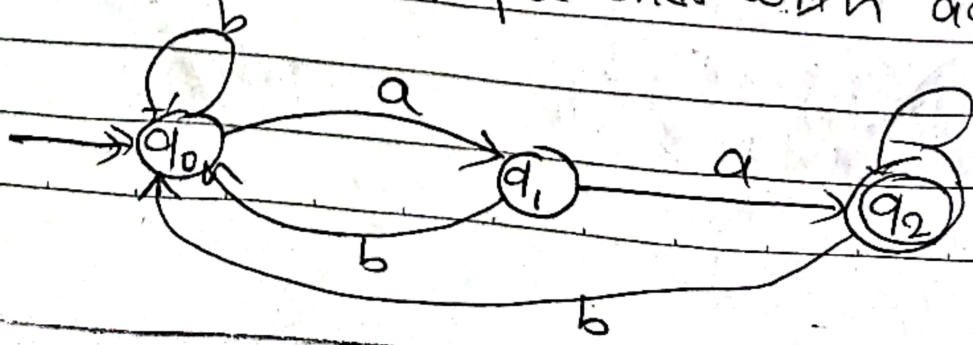


Q.22. DFA over {a,b} that accepts the strings ending with abb.



Q.23. Language of Strings ending in aa i.e.

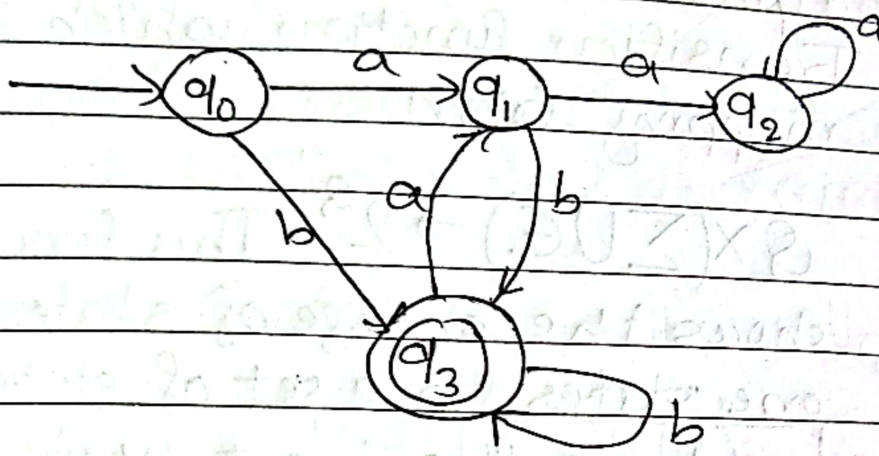
$$L = \{x \in \{a,b\}^* \mid x \text{ ends with } aa\}$$



HK'14

Q.24. Language of Strings Ending in b and Not Containing the Substring aa i.e.

$L2 = \{x \in \{a,b\}^* \mid x$ ends with b and does not contain the substring aa$\}$



Non-Deterministic Finite Automata (NFA)

NFA is an automaton machine in which for a particular input symbol, the machine can move to any combination of the states in the machine. In other words, the exact state to which the machine moves cannot be determined, hence the term non-deterministic. The transitions move from a a state can be to zero or more states taking an input.

HK 15

An NFA is a 5-tuple or quintuple $M = \{Q, \Sigma, \delta, q_0, F\}$ where,

Q = Finite set of states

Σ = Non-empty finite set of input alphabets.

δ = Transition function which is mapping from

$Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$. This function shows the change of states from one states to a set of states based on the input symbol.

$q_0 \in Q$ is the start state

$F \subseteq Q$ is set of final states.

Example: Consider NFA over $\{0, 1\}$

→

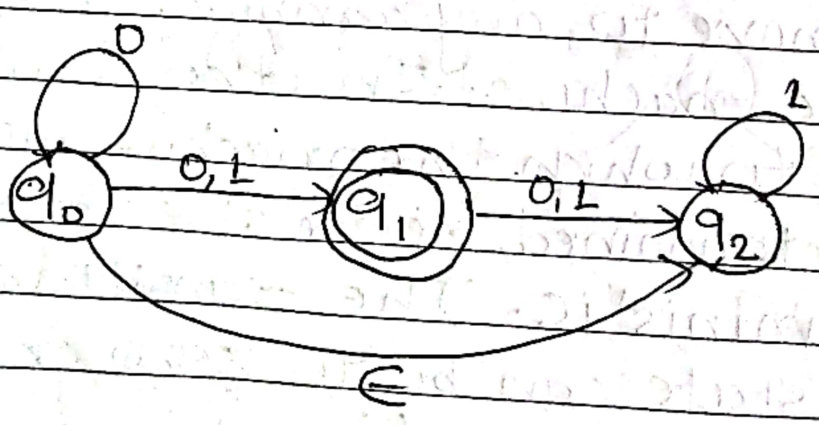


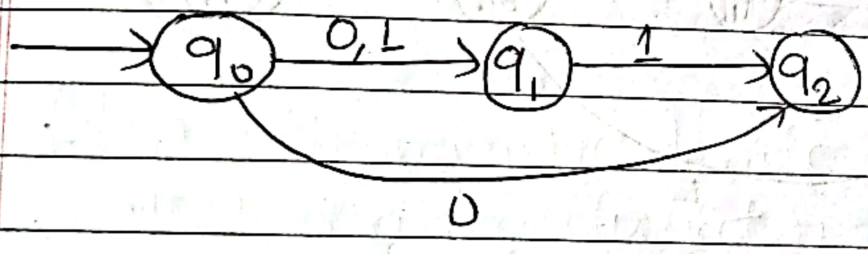
fig: NFA

HK'16

Current state	Input Symbol	Next state	Transition function
q_0	0	q_0, q_1	$\delta(q_0, 0) = (q_0, q_1)$
q_0	1	q_1	$\delta(q_0, 1) = q_1$
q_0	ϵ	q_2	$\delta(q_0, \epsilon) = q_2$
q_1	0	q_2	$\delta(q_1, 0) = q_2$
q_1	1	q_2	$\delta(q_1, 1) = q_2$
q_2	1	q_2	$\delta(q_2, 1) = q_2$
q_2	0	ϕ	

Example:- NFA over $\{0,1\}$ accepting strings $\{0, 01, 11\}$

Soluⁿ→

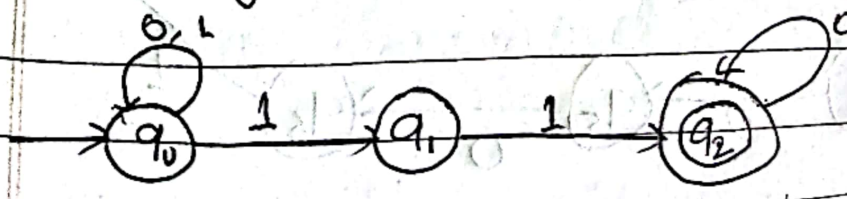


Transition Table

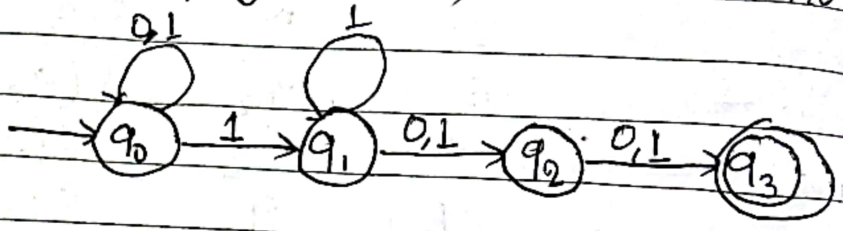
δ	0	1
q_0	$\{q_0, q_2\}$	$\{q_1\}$
q_1	ϕ	$\{q_2\}$
q_2	ϕ	ϕ

Q.3- Construct a NFA over $\{0,1\}$ that accepts strings having 11 as substring.

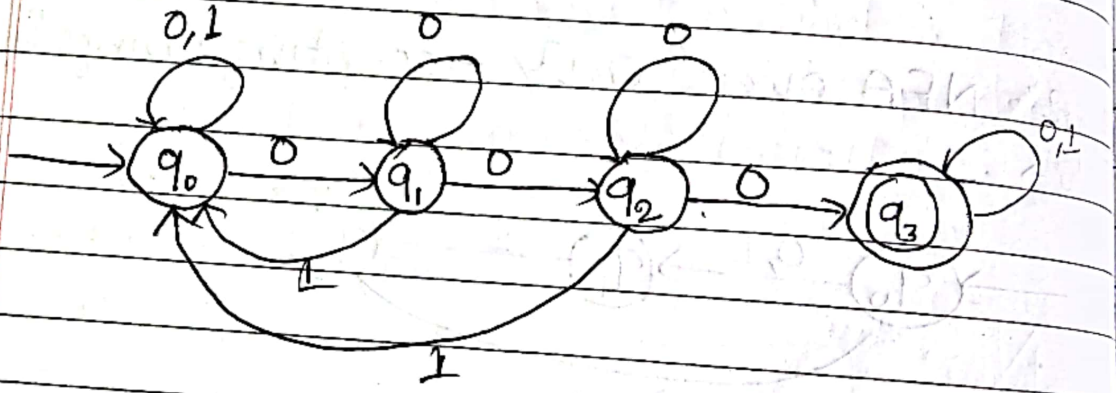
Soluⁿ→



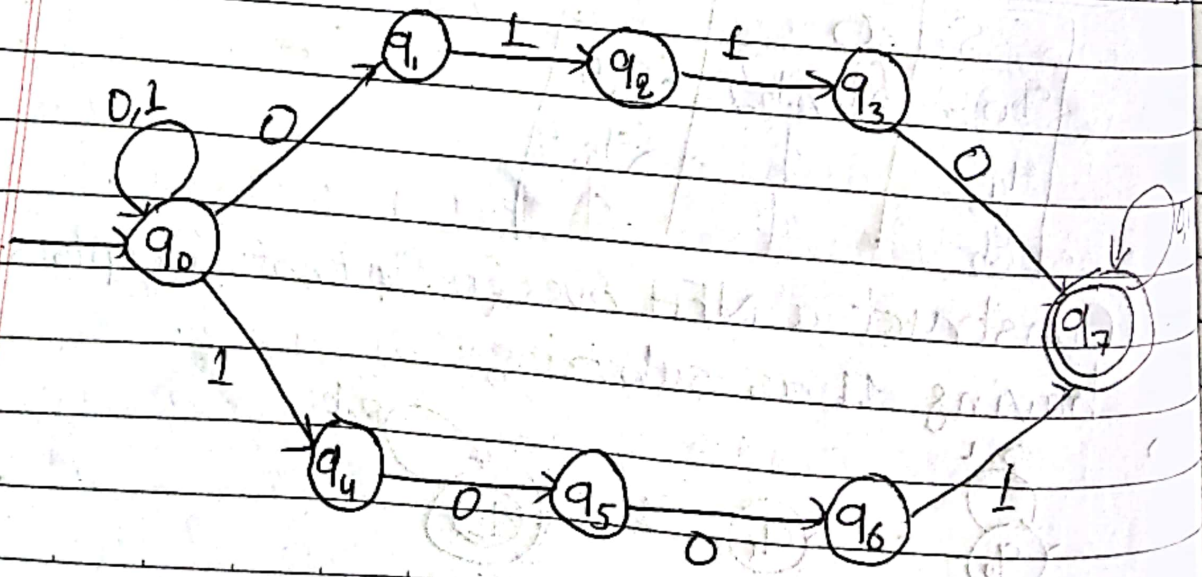
Q.4. Construct a NFA over {0,1} that accepts strings having 1 in the third position from the end {e.g., 00100, 010101 but not 00101}



Q.5. Construct an NFA to accept those strings containing three consecutive zeros

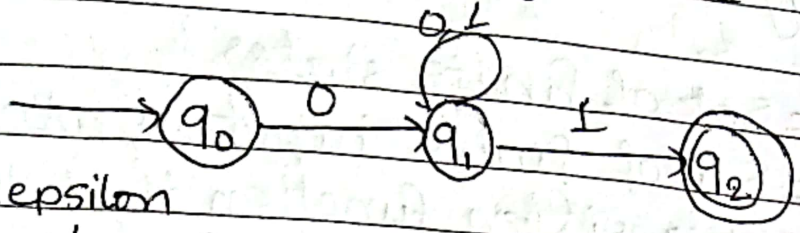


Q.6. NFA for strings over {0,1} that 0110 or 1001



11-18

8.7. NFA over {a,b} that accepts strings starting with 0 and ending with 1.



ε-Non-Deterministic Finite Automata (ε-NFA)

→ This is another extension of Finite Automata called ε-NFA. The new feature of ε-NFA is that, it allows a transition on ε, the empty string which means it can switch to new states without reading an input symbol. This capability doesnot expand the class of languages that can be accepted by finite automata, but it doesnot give some added "programming convenience".

→ epsilon-NFA (ε-NFA) is a NFA that can move to another state without consuming an input symbol. These transitions are called ε-transitions or λ transitions.

→ It's useful for constructing automata, especially when there are choices in the regular expression.

→ It's easier to create an ε-NFA for a language described by a regular expression that it is not create a DFA.

HK19

→ Definition: A NFA with ε-transition is defined by five tuples $(Q, \Sigma, \delta, q_0, F)$, where,

Q = Set of finite states

Σ = Set of finite input symbols

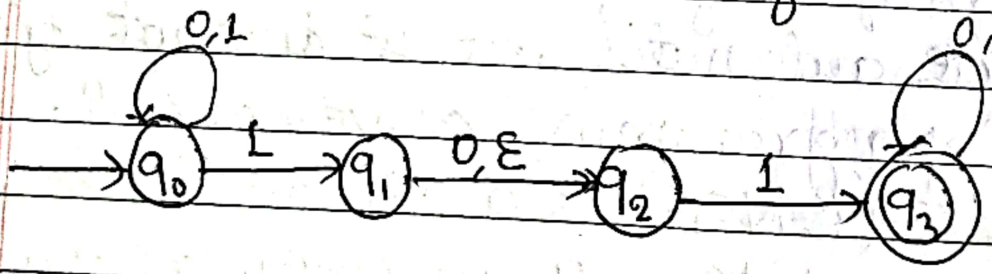
δ = Transition function that two arguments:

(i) A state in Q, and

(ii) A member of $\Sigma \cup \{\varepsilon\}$, that is, either an input symbol or the symbol ε. As input and produces a set of states as output i.e.

$$Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$$

Example Consider the following



The formal description of ε-NFA is $(Q, \Sigma, \delta, q_0, F)$ where,

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

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

δ = is given by

HK20

δ:	0	1	ε
$\to q_0$	$\{q_0\}$	$\{q_0, q_1\}$	ϕ
q_1	$\{q_2\}$	ϕ	$\{q_2\}$
q_2	ϕ	$\{q_3\}$	ϕ
q_3	$\{q_3\}$	$\{q_3\}$	ϕ

$\{q_0\}$ is the start state

$F = \{q_3\}$

HK21

1.4 NFA to DFA conversion

DFA and NFA recognises the same class of languages. That is; non-determinism does not make a finite automaton more powerful.

To show DFA and NFA recognize the same class of language; we will describe the method of converting an arbitary NFA into its equivalent DFA. This method is known as subset construction method.

Subset Construction Method

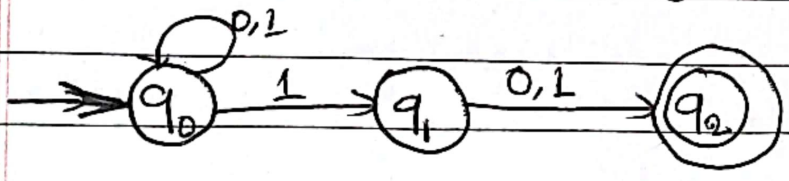
The formal conversion of a NFA $N = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ into its equivalent DFA $D = \{Q_D, \Sigma, \delta_D, q_0, F_D\}$ is done as follows:

1. The start state of D is the set of start state of N i.e, if q_0 is start state of N then D has start state as $\{q_0\}$.
2. Q_D is set of subsets of Q_N i.e. $Q_D = 2^{Q_N}$, so Q_D is power set of Q_N. So, if Q_N has n states then Q_D will have 2^n states. However, all of these states may not be accessible from start state of Q_D so they can be eliminated. So, Q_D will have less than 2^n states.
3. F_D is set of subsets S of Q_N such that $S \cap F_N \neq \phi$ i.e. F_D is all sets of N's states

that include at least one final state of N.

4. For each set $S \subseteq Q_N$ and each input $a \in \Sigma$, $\delta_D(S,a) = \bigcup_{p \in S} \delta_N(p,a)$
i.e., to compute $\delta_D(S,a)$ look at all the states p in S then find the states that N goes after reading input from p and take the union of those states.

Example:- Convert NFA to DFA of all binary string in which 2nd last bit is 1.



Sol^n Transition table

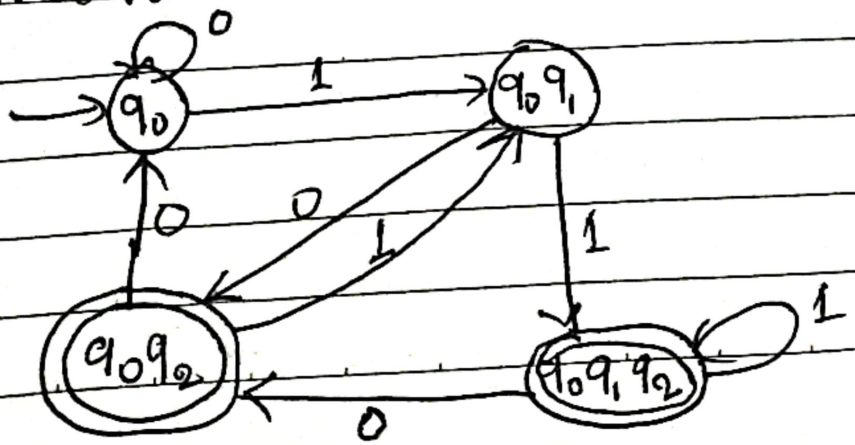
NFA

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

states	0	1
→ 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	$q_0 q_1$
($q_0 q_1 q_2$)	$q_0 q_2$	$q_0 q_1 q_2$

($n \to 2^n$)
($3 \to 2^3$ 8 max)

Equivalent DFA



HK-23.