SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

AUTONOMOUS INSTITUTION

Accredited by NAAC–UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMET OF COMPUTER SCIENCE AND ENGINEERING

19CS502- AUTOMATA THEORY AND COMPILER DESIGN

QUESTION BANK

1.Define automata.

Ans.: Automata are the kind of machines which take some string as input. This input goes through a finite number of states and may enter in the final state.

2. List the applications of automata theory.

Ans.: 1. Automata theory is the base for the formal languages and these formal languages are useful of the programming languages.

2. Automata theory plays an important role in compiler design.

3. To prove the correctness of the program automata theory is used.

4. In switching theory and design and analysis of digital circuits automata theory is applied.

5. Automata theory deals with the design finite state machines.

3. Construct a finite automata for the language $\{0^n \mid n \bmod 3 = 2, n \geq 0\}$. AU:
Ans.: While testing divisibility by 3 we group the input as

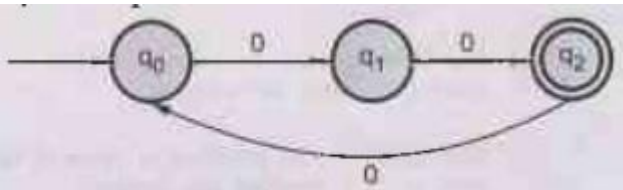


Fig. 1.13.1

q_0 remainder 0 state.

q_1 remainder 1 state.

q_2 remainder 2 state.

The q_2 is remainder 2 state hence we will make it as final state.

4. **What is a finite automation? Give two examples.**

Ans.: A finite automation is a collection of 5-tuples (Q, Σ, δ, q_0, F) where

• Q is a finite set of states, which is a non-empty one.

• Σ is input alphabet, indicates input set.

• q_0 in Q is the initial state.

• F is a set of final states.

• δ is a transition function.

For example -

1. The FA which accepts only those strings which start with 1 and end with 0.

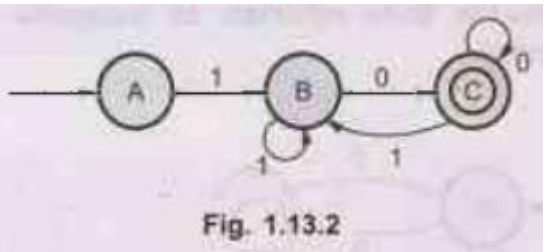


Fig. 1.13.2

2. The FA which accepts the only input 101.

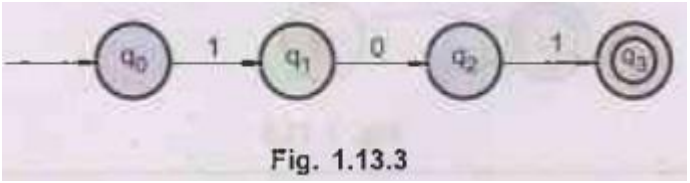


Fig. 1.13.3

5. Enumerate the differences between DFA and NFA.

Sr. No.	DFA	NFA
1.	Every input string leads to the unique state of finite automata.	For the same input there can be more than one next states.
2.	Conversion of regular expression to DFA is complex.	Regular expression can be easily converted to NFA using Thompson's construction.
3.	The DFA requires more memory for storing the state information.	The NFA requires more computations to match RE with input.
4.	The DFA it is not possible to move to next state without reading any symbol.	In NFA we can move to next state without reading any symbol.

Q.13 Construct a DFA over Σ = (a, b) which produces not more than 3a's.

Ans.: In the given language at the most 3a's are allowed and there is no restriction on number of b's. Such a DFA can be as shown in Fig. 1.13.4.

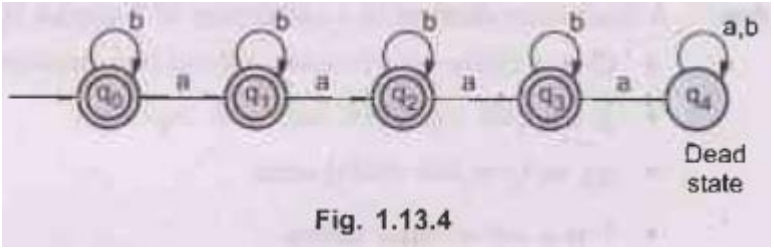


Fig. 1.13.4

The above DFA accepts {ε, a, aa, aaa, ab, aba,...}. The states q_0, q_1, q_2 and q_3 are final states but state q_4 is a non-final state. Thus the above DFA accepts 3a's and not more than that.

Q.14 Define the languages described by NFA and DFA.

• The finite automata is called Deterministic Finite Automata if there is only one path for a specific input from current state to next state. For example, the DFA can be shown in Fig. 1.6.1.

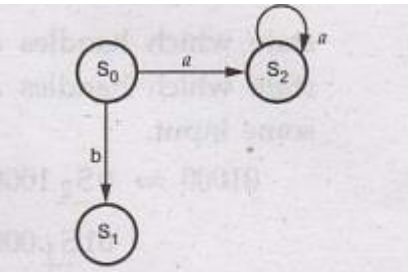


Fig. 1.6.1 Deterministic finite automata

• The concept of Non-deterministic Finite Automata is exactly reverse of Deterministic Finite Automata. The Finite Automata is called NFA when there exists many paths for a specific input from current state to next state. The NFA can be shown as in Fig. 1.7.1

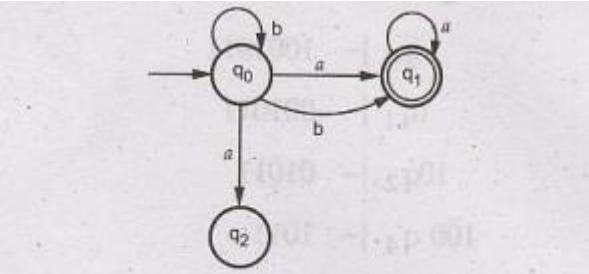


Fig. 1.7.1 Non-deterministic finite automata

Q.15 Construct deterministic finite automata to recognise odd number of 1's and even number of zero's.

Ans. :

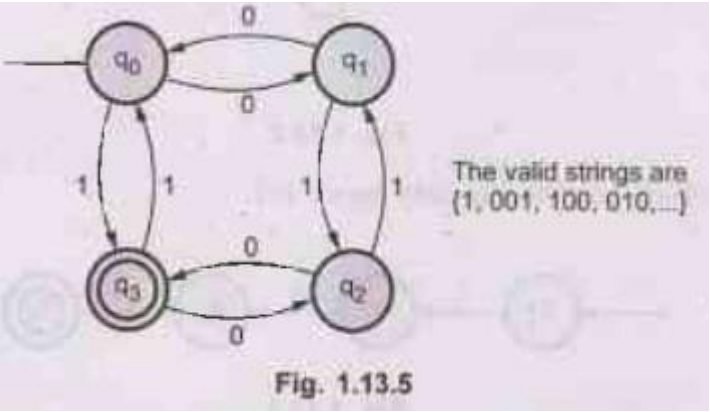
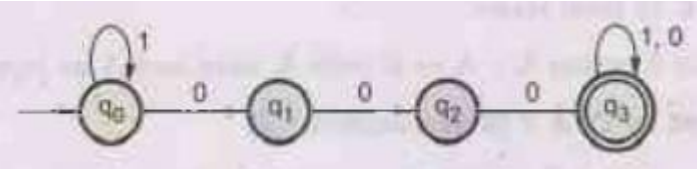


Fig. 1.13.5

Q.16 Construct a DFA for the language over {0, 1}* such that it contains "000" as a substring.

Ans.:

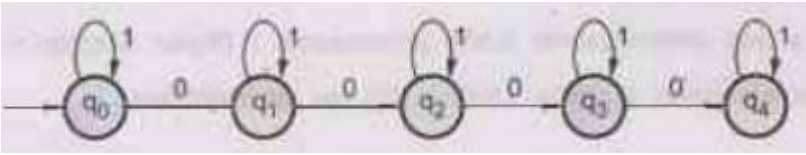


Q.17 Construct a DFA for the following:

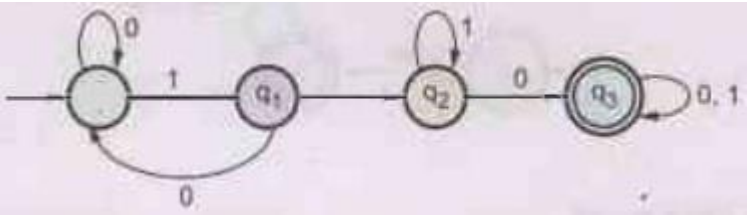
a) All strings that contain exactly 4 - zeros.

b) All strings that don't contain the substring 110.

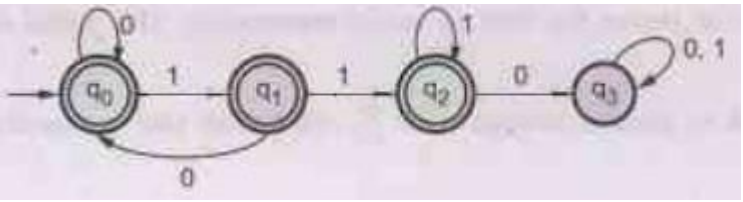
Ans. a) The DFA will be



b) For required DFA, we will first create a DFA that contains the substring 110.



Now change non-final state to final state and final state to non-final state.



Q.18 Find the set of strings accepted by the finite automata.

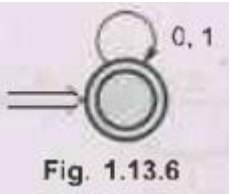


Fig. 1.13.6

Ans.: Set of strings accepted by given finite automata are

S = {ε, 0, 1, 00, 000, 11, 111, 01, 10, 101, 110, 001, ...)

Q.19 Define a) Finite Automaton (FA) b) Transition diagram.

Ans.: a) Finite Automaton (FA): Refer answer of Q.11.

Ans.: A finite automation is a collection of 5-tuples (Q, Σ, δ, q_0, F) where

• Q is a finite set of states, which is a non-empty one.

• Σ is input alphabet, indicates input set.

• q_0 in Q is the initial state.

• F is a set of final states.

• δ is a transition function.

b) Transition diagram: A transition diagram can be defined as collection of -

1) Finite set of states K

2) Finite set of symbols

3) A non empty set S of K which is called start state.

4) A set F $\subseteq$ K of final states.

5) A transition function K × A → K with K state and A as input from Σ*.

Q.20 What is meant by DFA ?

The finite automata is called Deterministic Finite Automata if there is only one path for a specific input from current state to next state. For example, the DFA can be shown in Fig. 1.6.1.

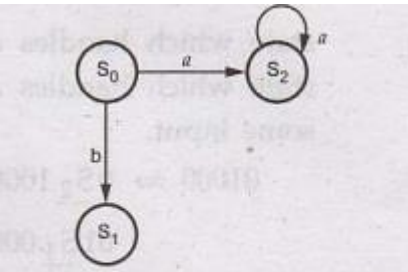


Fig. 1.6.1 Deterministic finite automata

From state S_0 for input 'a' there is only one path, going to S_2. Similarly from S_0 there is only one path for input b going to S_1.

The DFA can be represented by the same 5-tuples described in the definition of FSM.

Q.21 Define the term epsilon transition.

Ans.: In the non deterministic finite state machine, the transition that does not require input symbols for state transition and is capable of transiting to zero or more states with & is called epsilon transition.

Q.22 What is a non deterministic finite automation?

• The concept of Non-deterministic Finite Automata is exactly reverse of Deterministic Finite Automata. The Finite Automata is called NFA when there exists many paths for a specific input from current state to next state. The NFA can be shown as in Fig. 1.7.1

• Note that the NFA shows from q_0 for input a there are two next states q_1 and q_2. Similarly, from q_0 for input b the next states are q_0 and q_1.

• Thus it is not fixed or determined that Fig. 1.7.1 Non-deterministic finite automata with a particular input where to go next. Hence this FA is called non-deterministic finite automata.

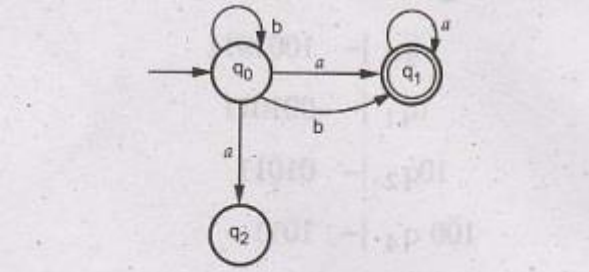
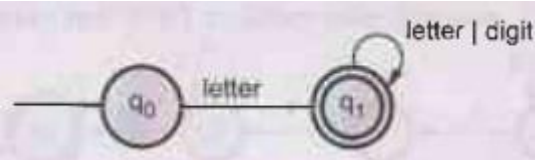


Fig. 1.7.1 Non-deterministic finite automata

Q.23 Draw the transition diagram (automata) for an identifier.

Ans. :



Q.25 Design DFA to accept strings over Σ = (0,1) with two consective O's.
Ans.:

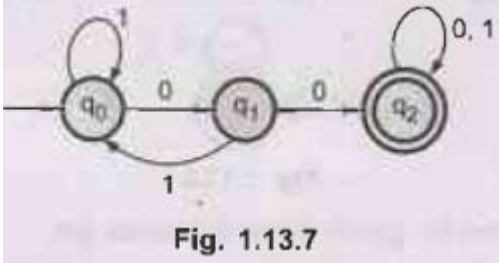


Fig. 1.13.7

Q.26 Define Deterministic Finite Automata (DFA).

Ans.:

The Deterministic Finite Automata is a collection of following things:

1) Finite set of states - Q

2) Finite set of input symbols - Σ

3) The start state q_0 such that $q_0 \in Q$.

4) The set of final states F such that $F \in Q$

5) The mapping function δ.

The DFA is denoted by,

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

Q.27 Draw a non-deterministic automata to accept a string containing substring 0101.

Ans.:

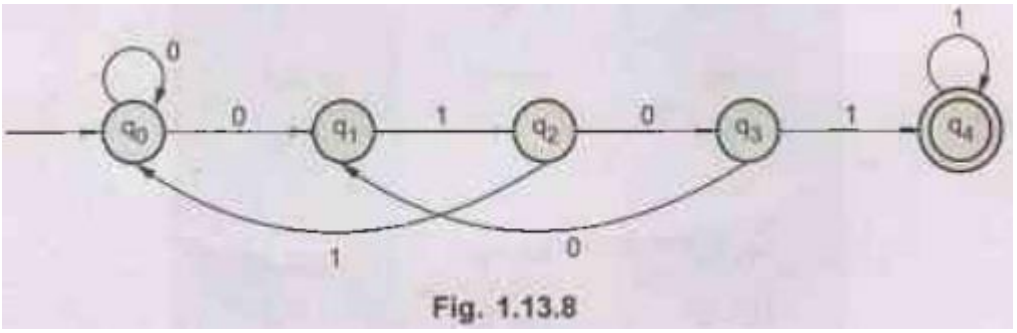


Fig. 1.13.8

Q.28 Obtain the DFA equivalent to the following NFA.

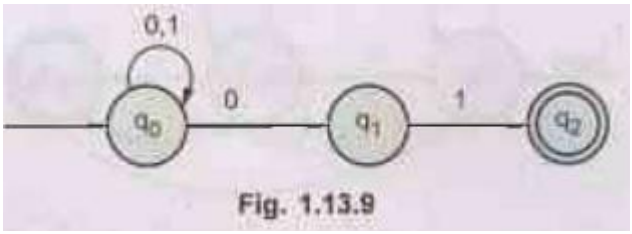


Fig. 1.13.9

Ans.: The transition table for given NFA can be drawn as follows.

States	0	1
$\{q_0\}$	$\{q_0\}, \{q_1\}$	$\{q_0\}$
$\{q_1\}$	–	$\{q_2\}$
$\{q_2\}$	–	–

To construct equivalent DFA.

$\delta\ (q_0, 0) = \{q_0, q_1\}$ a new state

$\delta\ \{q_0, 1\} = \{q_0\}$

$\delta\ \{q_1, 1) =$ -

$\delta (q_1, 1\} = \{q_2\}$

$\delta \{q_2, 0\} = -$

$\delta \{q_2, 1\} = -$

$\delta \{\{q_0, q_1), 0\} = \{q_0, q_1\}$

$\delta \{\{q_0, q_1), 1\} = \{q_0, q_2\}$ a new state

$\delta \{\{q_0, q_2\}, 0\} = \{q_0, q_1\}$

$\delta \{\{q_0, q_2\}, 1\} = \{q_0\}$

Hence the DFA is

States	0	1
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	–	$\{q_2\}$
$\{q_2\}$	–	–
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

The transition diagram can be drawn as follows.

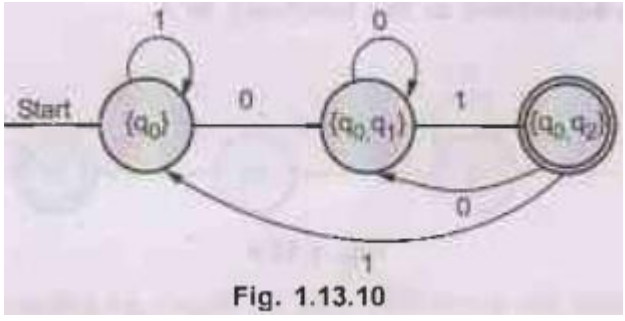


Fig. 1.13.10

Q.29 Obtain the NFA without ε transition to the following NFA with ε transition.

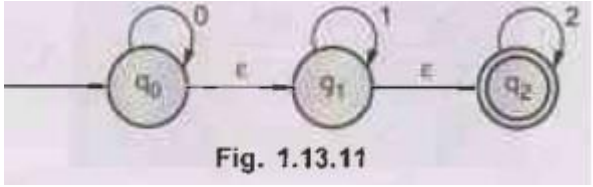


Fig. 1.13.11

Ans.: Remove ε transition from q_0 to q_1.

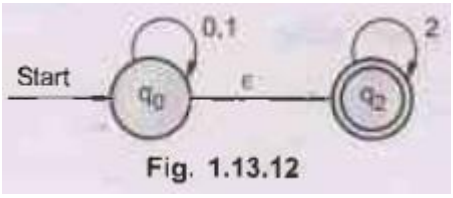


Fig. 1.13.12

Now remove ε transition from q_0 to q_2.

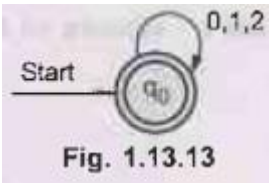


Fig. 1.13.13

As q_0 to q_2 is ε transition q_0 will become start and final state both.

Q.30 Obtain the ε - closure of states q_0 and q_1 in the following NFA with ε transition.

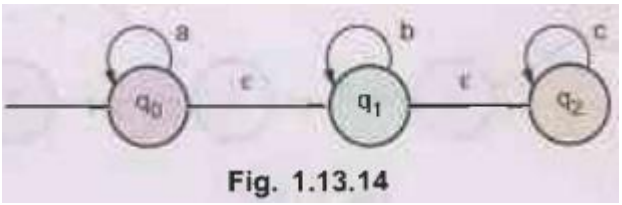


Fig. 1.13.14

Ans.: ε - closure $\{q_0\} = \{q_0, q_1, q_2\}$

ε - closure $\{q_1\} = \{q_1, q_2\}$

These are ε - reachable states from q_0 and q_1.

Q.31 Obtain ε - closure of each state in the following NFA with ε move. AU:

Ans.: The ε - closure of each state means collection of ε - reachable states.

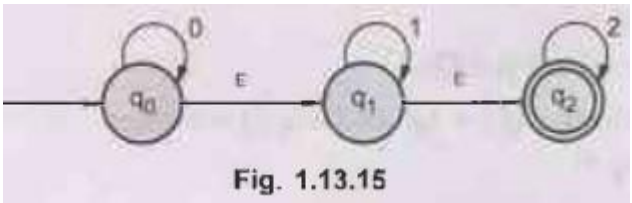


Fig. 1.13.15

ε - closure $(q_0) = \{q_0, q_1, q_2\}$ as we have ε transition from q_0 to q_0, q_1, q_2.

ε - closure (q_1) = {q_1, q_2}

ε - closure {q_2} = {q_2}

Q.32 Find the language accepted by the DFA given below.

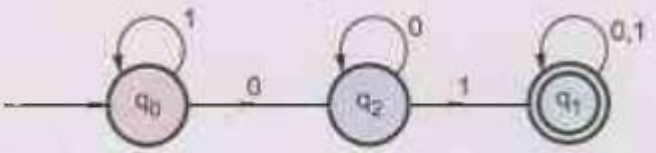


Fig. 1.13.16

Ans.: The regular expression for this DFA is

r.e. = 1*00*1 (0+1)*

= 1*0*1 (0+1)*

That means this is a language containing all the strings which consist of atleast one single pair of 01.

Q.33 Find the closure of the states 1, 2 and 4 in the following transition diagram

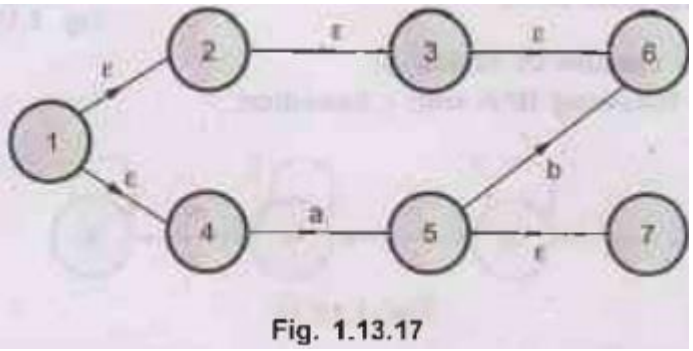


Fig. 1.13.17

Ans.: ε - closure {1} = {1, 2, 3, 6, 4}

ε - closure {2} = {2, 3, 6}

ε - closure {4} = {4}

Q.34 Define ε - closure.

Ans.: The ε - closure (p) is a set of all states which are reachable from state p on ε - transitions such that:

i) ε - closure (p) = p where p ϵ Q.

ii) If there exists ε - closure (p) = {q} and δ (q, ε) = r then

ε - closure (p) = {q, r}

Q.35 Define ε - closure (q) with an example.

Ans. Definition: Refer answer of Q.34.

Ans.: The ε - closure (p) is a set of all states which are reachable from state p on ε - transitions such that:

i) ε - closure (p) = p where p ϵ Q.

ii) If there exists ε - closure (p) = {q} and δ (q, ε) = r then

ε - closure (p) = {q, r}

Example: Refer answer of Q.31.

Ans.: The ε - closure of each state means collection of ε - reachable states.

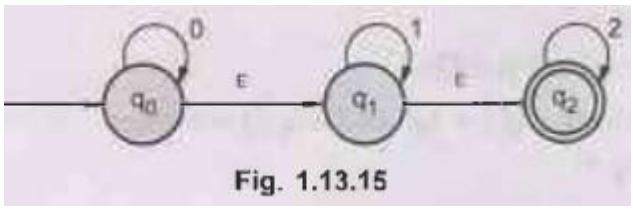


Fig. 1.13.15

ε - closure (q_0) = {q_0, q_1, q_2} as we have ε transition from q_0 to q_0, q_1, q_2.

ε - closure (q_1) = {q_1, q_2}

ε - closure {q_2} = {q_2}

QUESTIONS

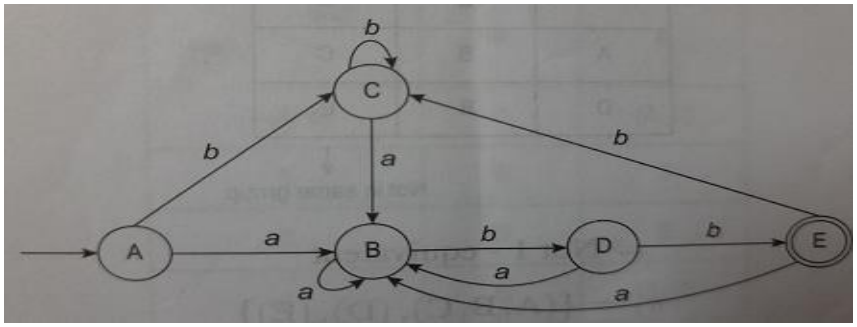
1. What is Finite automata?
2. Define Regular Expression. Give examples..
3. Enumerate the differences between DFA and NFA
4. Solve the following grammar is ambiguous: S→aSbS / bSaS / €, input: aabb
5. How will you represent a NFA?
6. Construct a DFA for the language over {0, 1} such that it contains "000" as a substring.
7. List the applications of automata theory.
8. Find the First () for the given Grammar

   E→E+T | T

   T→T*F | F

   F→ (E) | a | b
9. Outlines the properties of parse tree?
10. What is meant by yield of parse tree?
11. Define Sentential form.
12. Construct a NFA for the language over {a, b} such that it contains "aa" as a substring.
13. Construct a DFA for the language over {0, 1} such that it ending with 0 and starting with 11.
14. Construct a DFA for the language over {0, 1} such that all the strings which are having prime nos.
15. Define Regular language.
16. Draw the Transition diagram for identifier.
17. Draw the Transition diagram for digit.
18. Define CFG.
19. How will you represent a context free grammar?
20. Define derivation and its types.

1. construct the NFA for the following DFA



2. Solve NFA for the regular expression ab*/ab

3. Define the language accepted by FA. Convert the following NFAinto DFA.

	0	1
p	{p,q}	{p}
q	{r}	{r}
r	{s}	ϕ
*s	{s}	{s}

4. Draw the NFA for the augmented regular expression
(a | b)*.

5. Explain in detail about phases of compiler and also explain the applications of lexical analysis.

6. What is Leftmost derivation and Rightmost derivation?.
7. Draw leftmost derivation and Rightmost derivation for the following.
E->E+E|E*E| id input: id+ id*id

8. For the following grammar.
S→ (L) | a
L→ L, S | S
Give a rightmost and leftmost derivation and, parse tree, ambiguous for input string (a, (a, a))

9.Convert the following NFA to DFA

10.Difference between ambiguous and unambiguous grammar.

11.Construct the NFA for the given regular expression (a / b) * a b b (a / b)

12.Construct the DFA for the given NFA