UNIT-V

FINITE STATE AUTOMATA & GRAPH THEORY

FINITE AUTOMATION:-

A finite Automation is formerly defined as 5-tuple $(Q, \Sigma, \delta, q_0, F)$. where.

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

Σ - is input Alphabet.

q_0 - is initial state.

F - is a set of final states and $F \subseteq Q$

δ - is a transition function (or) mapping function $Q \times \Sigma \to Q$.

Using this the next state can be determined depending on the Current input.

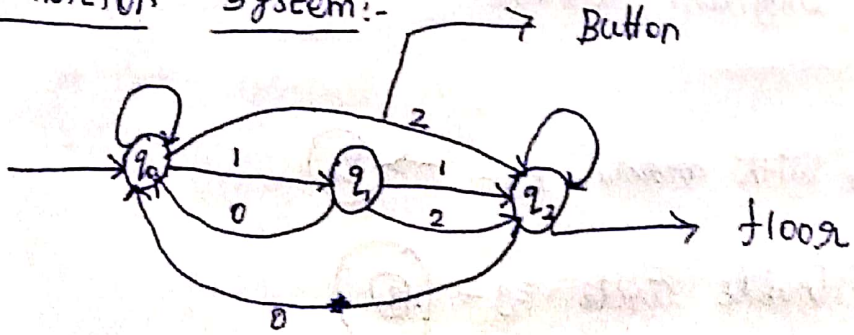
Ex: SFL ——— q_2 (2)

FFL ——— q_1 (1)

GFL —— ⊠ —— q_0 (0)

Example of Lift Control is given in the above picture.

Transition System:-



Transition Table:-

PS	0	1	2 i/Ps
q_0	q_0	q_1	q_2
q_1	q_0	q_1	q_2
q_2	q_0	q_1	q_2

Q - $\{q_0, q_1, q_2\}$

Σ - $\{0, 1, 2\}$

q_0 is initial state, $q_0 \in Q$

$F = \{q_2\}$, $F \subseteq Q$

δ - $\delta(q_0, 0) = q_0$, $\delta(q_0, 1) = q_1$ $\delta(q_0, 2) = q_2$

$\delta(q_1, 0) = q_0$, $\delta(q_1, 1) = q_1$, $\delta(q_1, 2) = q_2$

$\delta(q_2, 0) = q_0$, $\delta(q_2, 2) = q_1$, $\delta(q_2, 2) = q_2$.

Note:- Transition Diagram contains.

* ~~Set of states~~

* Start state q_0 with arrow, Eg - → (q_0)

* Final state by double circle, Eg - $((q))$

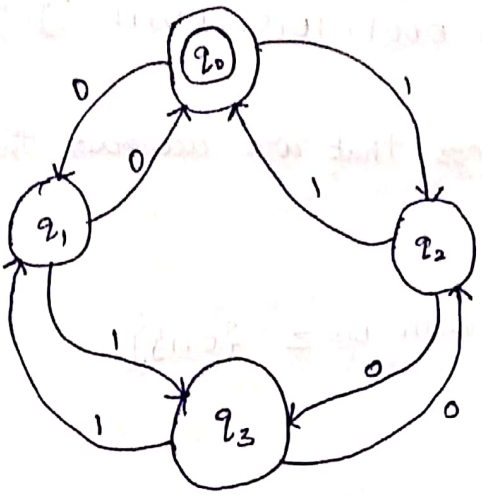
Language Acceptance

A string w is accepted by finite Automaton U given as $U = \{Q, \Sigma, \delta, q_0, F\}$ if $\delta(q_0, w) = p$ for some p in F.

~~Ex:1 Let us check if / no~~

Ex:1 Let us check if input string 1010 is accepted or not.

(a)



$q_0 \vdash^{1} q_2 \vdash^{0} q_3 \vdash\!\vdash q_1 \vdash^{0} q_0$

Here q_0 is the final state. Hence the string is accepted.

(b) CHECH 11111

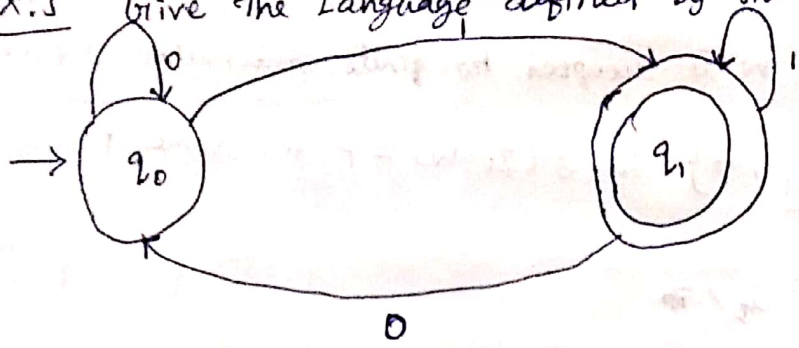
$q_0 \vdash^{1} q_2 \vdash^{1} q_0 \vdash^{1} q_2 \vdash^{1} q_0 \vdash^{1} q_2$

$q_2 \in F$

$\therefore q_0$ is not reached the final state.

$\therefore$ It is rejected.

EX:5 Drive the Language defined by the following FA.



Solution:-

If we list different strings accepted by this automation, we get $\{1, 01, 0001, 10101, 011111 \cdots\}$

If we observe all strings that are accepted they always end with 1.

$L(M) = \{w/w$ ends with on $\Sigma = \{0,1\}\}$.

FINITE AUTOMATON IS OF TWO TYPES:-

(a) Deterministic Finite Automaton (DFA)

Definition:-

Deterministic finite automaton can be defined as a quintuple $M = (Q, \Sigma, \delta, q_0, F)$, where,

Q = non empty finite set of states.

Σ = input alphabet.

q_0 = initial start state

F = Set of final states.

δ = Transition function that takes two arguments: a. state and input symbol and returns output as state ie, $\delta: Q \times \Sigma \to Q$.
ie, $\delta(q, a) = q_1$.

DFA Design

Ex:1 Design a DFA that accepts even number of 0's and even number of 1's.

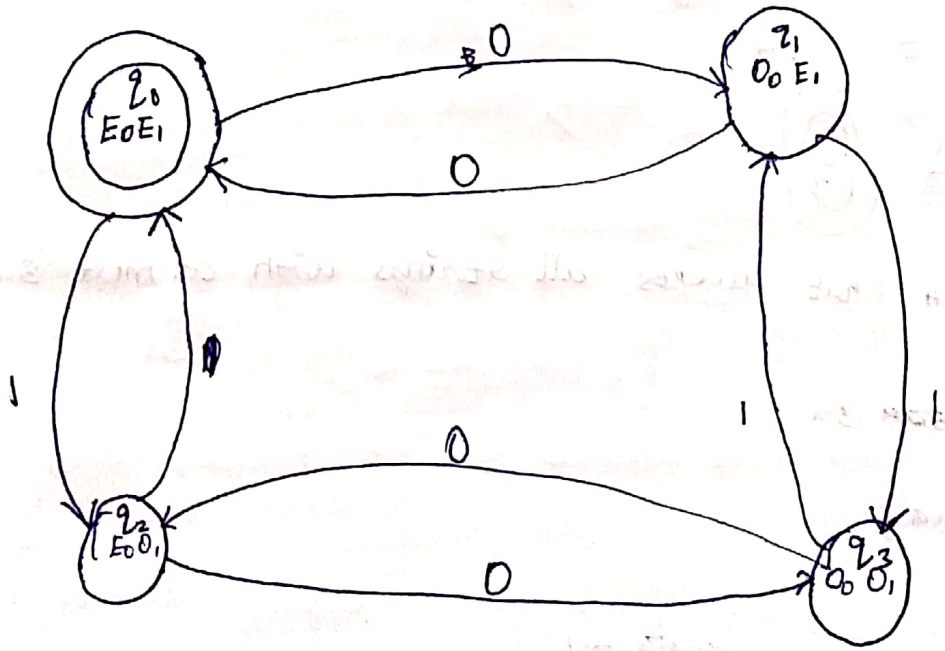
Solution:- This FA will have four different possibilities while reading 0's and 1's as input. The possibilities could be

Even number of 0's and even number of 1's — q_0

odd number of 0's and even number of 1's — q_1

Even number of 0's and odd number of 1's — q_2

Odd number of 0's and odd number of 1's — q_3



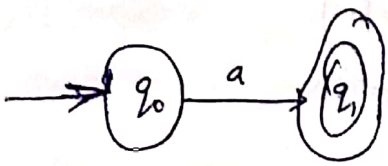
Pattern recognition:-

1. Identify the minimum string.
2. Identify the alphabets.
3. Draw the skeleton DFA.
4. Define the Transition functions.
5. Construct the DFA.

Ex:1 Draw a DFA accept strings of a's having atleast one a.

1. Min string = a
2. $\Sigma = \{a\}$
3. No of states = min string length + 1

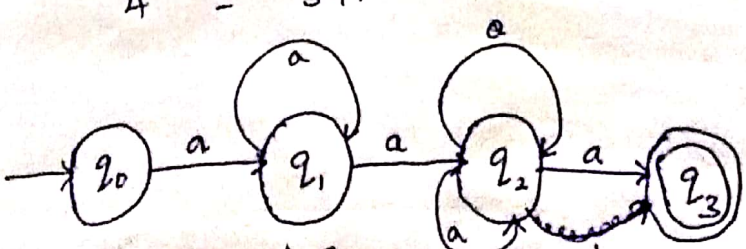
2 = 1 + 1



Ex:2 Design a DFA that accepts all strings with at most 3a's

1. min string = ~~aaa~~ 3a
2. $\Sigma = \{a, aa, aaa\}$
3. No of states = min string length + 1

4 = 3 + 1



$\delta(q_0, a) = q_1$ | $\delta(q_1, a) = q_2$ | $\delta(q_2, a) = q_3$

$\delta(q_1, a) = q_1$ | $\delta(q_2, a) = q_2$ | $\delta(q_2, a) = q_3$

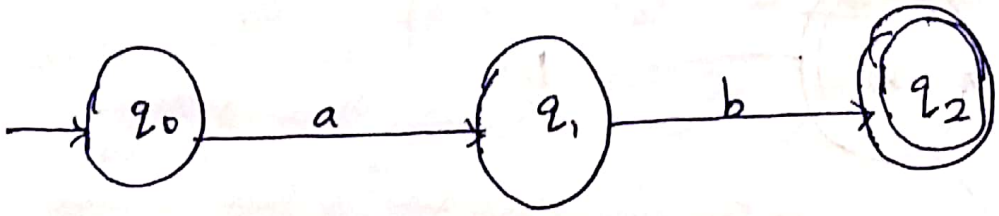
Ex:3 Draw a DFA that recognises the set of all strings with $\Sigma = \{a, b\}$ starting with prefix ab.

1. Min string = 2 (ab)
2. $\Sigma = \{ab, abba, abab, abaab, \ldots\}$
3. No of states = min string ~~length~~ + 1

= 3.

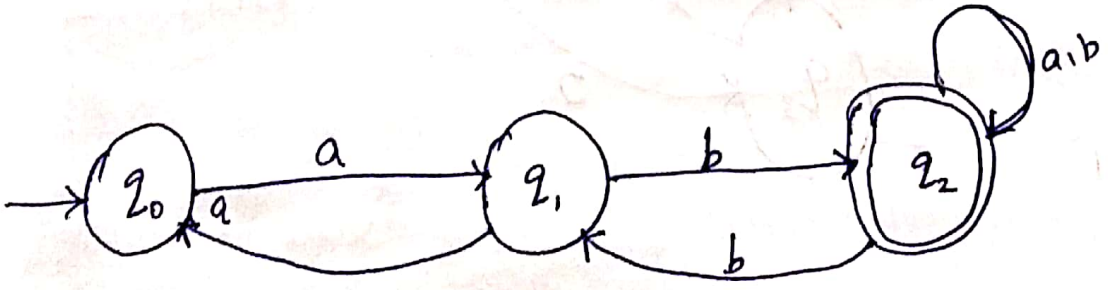
= q_0, q_1, q_2 { Our assumption on Giving Name of the States).

(i) ab



$\delta(q_0, a) = q_1$
$\delta(q_1, b) = q_2$ } for ab.

(ii) abbaab



Ex:4

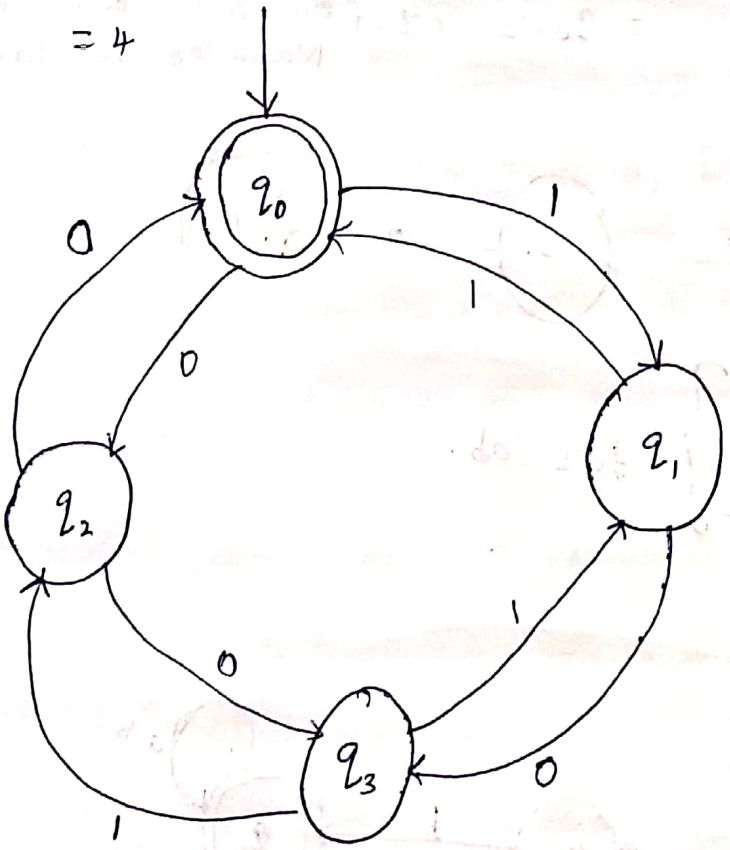
Design a DFA that accepts even number of 0's and even number of one's.

$\Sigma = \{ 001, 0011, 1100, 101, 110, 100, 011, \ldots, 000111, \ldots \}$

Min string = 3

No of states = 3+1
= 4

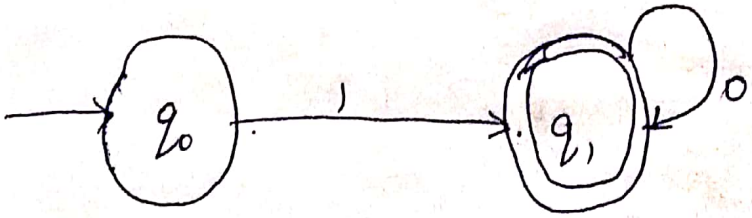
1010



Ex:5 Design a DFA that accepts an odd number of 1's on $\{0,1\}$

$\Sigma = \{ 1, 10, 01, 001, 100, \ldots \}$

Min string = 1 , No. of states = 1+1 = 2.



Non-deterministic Finite Automaton (NFA)

Definition:-

Non-deterministic Finite Automaton can be defined as a quintuple. $M = (Q, \Sigma, \delta, q_0, F)$, where,

Q = Non-empty finite set of states.

Σ = input alphabet.

q_0 = initial start state.

F = set of final states.

δ = transition function that takes two arguments: a state and input symbol and returns output as state.

ie, $\delta: Q \times \Sigma \rightarrow 2^Q$.

Note:- It is easier to construct NFA than DFA. but the Processing time of strings is more than in DFA.

Ex:1 Design a NFA accepting all strings ending with 01 over.

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

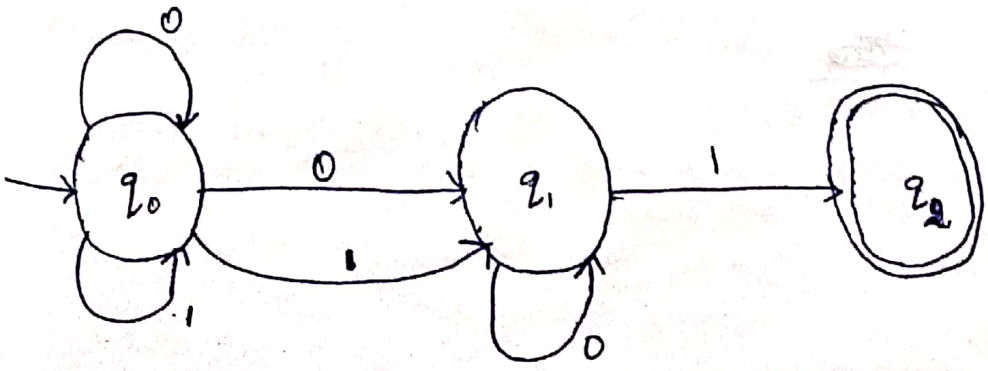
Solution:-

Min length = 2

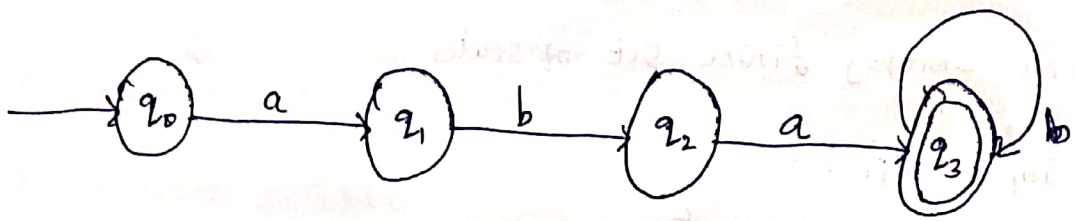
$\Sigma = \{01, 001, 101, 11001, \ldots\}$

No of states = 2+1

= 3



Ex:2 Design a NFA for $\{abab^n / n>=0\}$



GRAPHS

A Graph Gr is a pair (V, E), where V is finite set and E is a relation on V. the elements of V are called nodes. (or) vertices. The elements of E are called edges (or) ares.

Notes:-

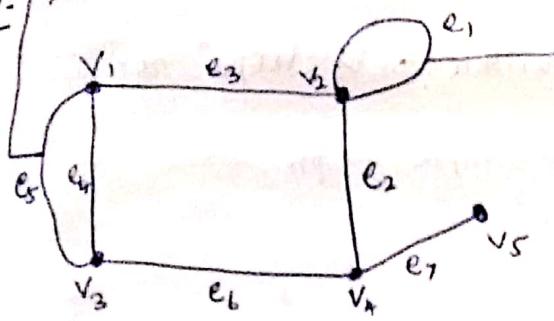
*. In Gr if $\langle u, v \rangle$ is an edge then u is a predecessor of v. and v is successor. of u.

*. The graph is said to be ordered if some ordering is assumed on the predecessors of each node, and on the successors of each node.

Path

Parallel edges (because in has a common vertices).

Ex:-



vertex ⇒ node, junction point,

edge ⇒ branch, line, arc, ...

Type of Graph.

(1) Null Graph:-

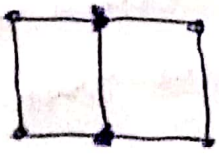
A Graph without any edges is called null graph.

Ex:

(2) Undirected graph:-

The Graph which have no, directions is called undirected. Graph

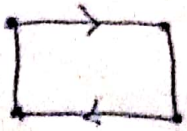
Ex:



(3) Directed Graph:-

It has Directions.

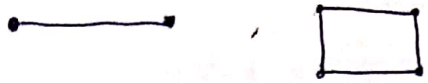
Ex:



(4) Simple graph:-

A graph that has neither self loops nor Parallel edges is called a simple graph.

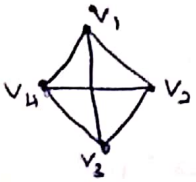
Ex:



(5) Connected graph:-

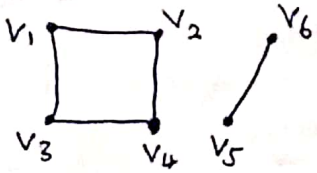
A Graph is said to be connected if there exists atleast one path between every pair of vertices

Ex:-



(6) Disconnected Graph:-

A Graph can be split into an independent components, then it is called as disconnected graph.



∵ A null graph with more than one vertex is also a disconnected graph.

(7) A Graph $G_1 = (V_1, E_1)$ is said to be a subgraph of a graph $G_2 = (V_2, E_2)$ if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

WALKS

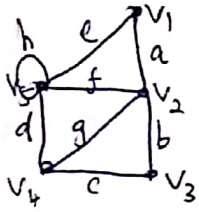
It is defined as a finite ~~alternating~~ alternating sequence of vertices and edges beginning and ending with vertices, such that each edge is incident with the vertices preceding and following it.

*. No edge appears more than once in a walk.

*. A vertex however, appear more than once.

*. A walk is also referred as edge train (or) a chain.

EX:



⇒ terminal vertices of walk. (Starting and ending vertices).

$V_1 a V_2 f V_5 h V_5 d V_4 g V_2 b V_3$

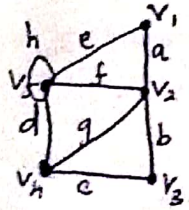
PATH

*. A walk that the terminal vertices are distinct. then it is called an open walk.

*. An open walk in which no vertex appears more than once is called a path. In other word a path does not intersect itself.

A walk begin and end with the same vertex is called a closed walk.

EX:-



$V_1 a V_2 f V_5 d V_4 c V_3$

The no. of edges in path is called the length of the path.

CIRCUIT

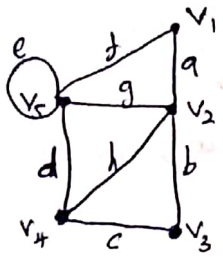
*. A closed walk in which no vertex except the initial and final vertex appears more than once is called a circuit.

*. Circuit is a closed, non intersecting walk.

*. Cycle, circular path, polygon are called another name of circuit.

*. Every self loop is a circuit but every circuit is a not a self loop.

Ex:

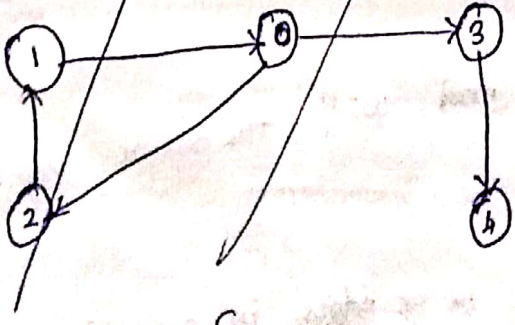


$V_1 f V_5 d V_4 c V_3 V_2 a V_1$

Reachable

If there is a path from u to v then v is said to be reachable from u.

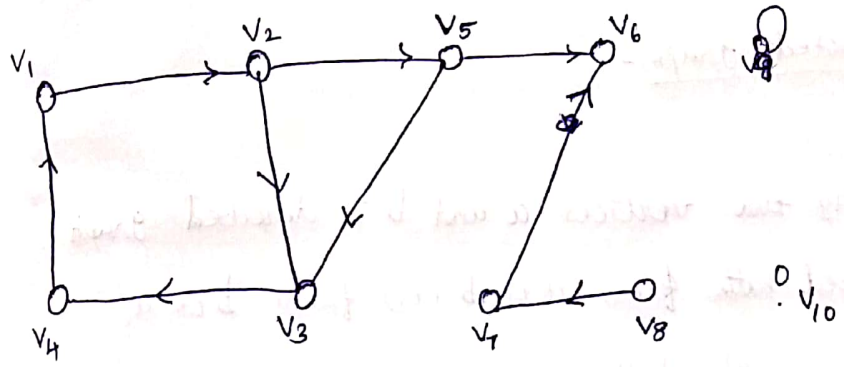
Ex:



Ex: $R(1) = \{0,$

Reachability

A node v of a simple digraph is said to be reachable from the node u of the same digraph. if there exist a path from u to v.



$R(V_1) = \{V_1, V_2, V_3, V_4, V_5, V_6\}$ — reachable set.

$R(V_2) = \{V_1, V_2, V_3, V_4, V_5, V_6\}$

$R(V_3) = \{V_1, V_2, V_3, V_4, V_5, V_6\}$

$R(V_4) = \{V_1, V_2, V_3, V_4, V_5, V_6\}$

$R(V_5) = \{V_1, V_2, V_3, V_4, V_5, V_6\}$

$R(V_6) = \{V_6\}$

$R(V_7) = \{V_6, V_7\}$, $R(V_8) = \{V_6, V_7, V_8\}$.

$R(V_9) = \{V_9\}$

$R(V_{10}) = \{V_{10}\}$

Connectedness in Directed Graphs

Strongly Connected:-

A directed graph is strongly connected if there is a path from a to b and from b to a, whenever a and b are vertices in the graph.

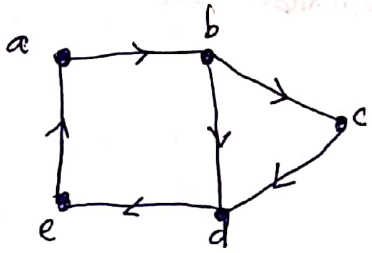
Weakly Connected:-

A directed graph is weakly connected if there is path between every two vertices in the underlying undirected graph.

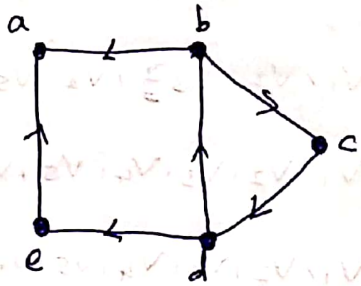
Unilaterally Connected Graph:-

For any two vertices a and b in directed graph there is a directed path from a to b (or) from b to a but not necessarily both.

Ex:



G1



H

* G1 is strongly connected.
* G1 is also weakly connected.
* The Graph H is not strongly connected. because there is no directed Path from a to b in this graph.
* H is weakly connected. because there is a path between any two vertices in the underlying undirected graph of H.
* H is unilaterally connected.