SNS COLLEGE OF TECHNOLOGY

Coimbatore-35

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A++’ Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : 23ITT201 DATA STRUCTURES

II YEAR/ III SEMESTER

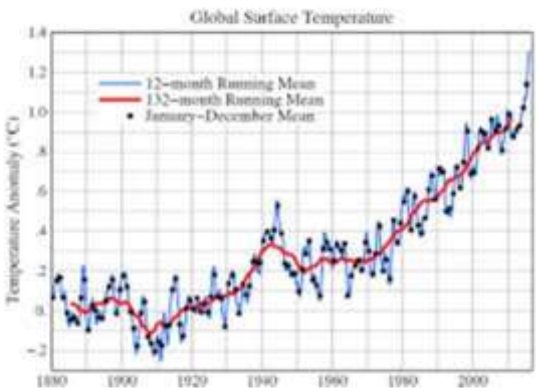
UNIT – IV MULTIWAY SEARCH TREES AND GRAPH

Topic: *GRAPH*

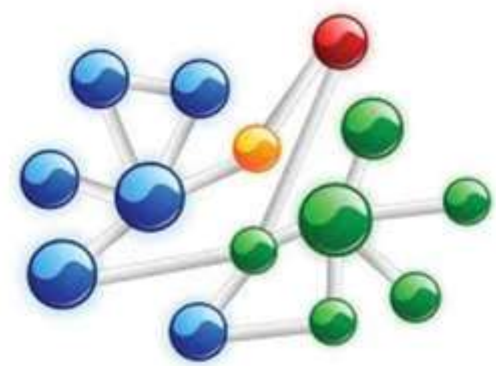
SNS COLLEGE OF TECHNOLOGY

(Autonomous)

COIMBATORE-35



UNIT IV

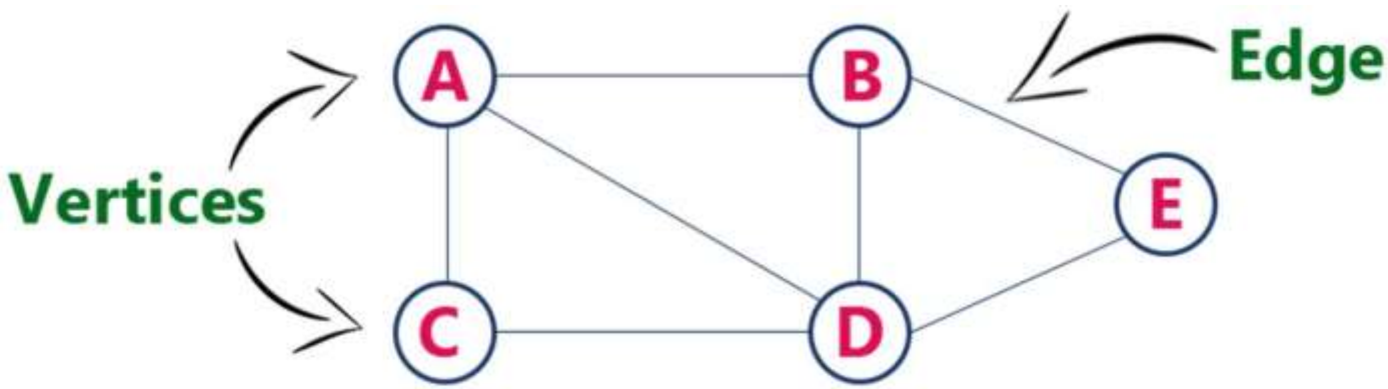


Introduction

- **Graph** is a collection of vertices and arcs which connects vertices in the graph
- Graph is a collection of nodes and edges which connects nodes in the graph
- Generally, a graph **G** is represented as **G = (V , E)**, where **V is set of vertices** and **E is set of edges**.

Introduction

The following is a graph with 5 vertices and 6 edges.
This graph G can be defined as G = (V , E)
Where V = {A,B,C,D,E} and E =
{(A,B),(A,C)(A,D),(B,D),(C,D),(B,E),(E,D)}.



Basic Terminologies

Vertex

A individual data element of a graph is called as Vertex. **Vertex** is also known as **node**. In above example graph, A, B, C, D & E are known as vertices

Edge

An edge is a connecting link between two vertices. **Edge** is also known as **Arc**. An edge is represented as (starting Vertex, ending Vertex).

The link between vertices A and B is represented as (A,B). In above example graph, there are 7 edges (i.e., (A,B), (A,C), (A,D), (B,D), (B,E), (C,D), (D,E)).

Basic Terminologies

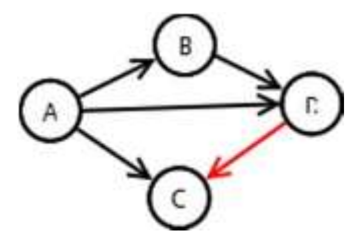
- **Undirected Edge -** An undirected edge is a bidirectional edge. If there is a undirected edge between vertices A and B then edge (A , B) is equal to edge (B , A).
- **Directed Edge -** A directed edge is a unidirectional edge. If there is a directed edge between vertices A and B then edge (A , B) is not equal to edge (B , A).
- **Weighted Edge -** A weighted egde is an edge with cost on it.

Undirected Graph

- A graph with only undirected edges is said to be undirected graph.

Directed Graph

- A graph with only directed edges is said to
- be directed graph.



Basic Terminologies

Mixed Graph

- A graph with undirected and directed edges is said to be mixed graph.

End vertices or Endpoints

- The two vertices joined by an edge are called the end vertices (or endpoints) of the edge.

Origin

- If an edge is directed, its first endpoint is said to be origin of it.

Destination

- If an edge is directed, its first endpoint is said to be origin of it and the other endpoint is said to be the destination of the edge.

Basic Terminologies

Adjacent

If there is an edge between vertices A and B then both A and B are said to be adjacent. In other words, Two vertices A and B are said to be adjacent if there is an edge whose end vertices are A and B.

Incident

An edge is said to be incident on a vertex if the vertex is one of the endpoints of that edge.

Outgoing Edge

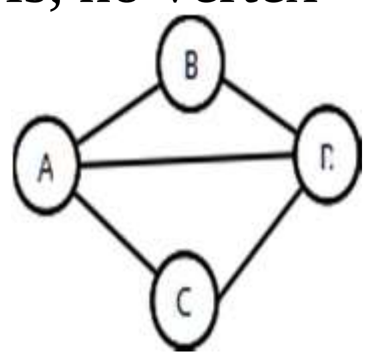
A directed edge is said to be outgoing edge on its orign vertex.

Incoming Edge

A directed edge is said to be incoming edge on its destination vertex.

Basic Terminologies

- **Degree of a Node**
  - In-degree: Number of edges pointing **to** a node
  - Out-degree: Number of edges pointing **from** a node
- **Path**: sequence of vertices in which each pair of successive vertices is connected by an edge
  Cycle: a path that starts and ends on the same vertex
  Simple path: a path that does not cross itself That is, no vertex is repeated (except first and last)

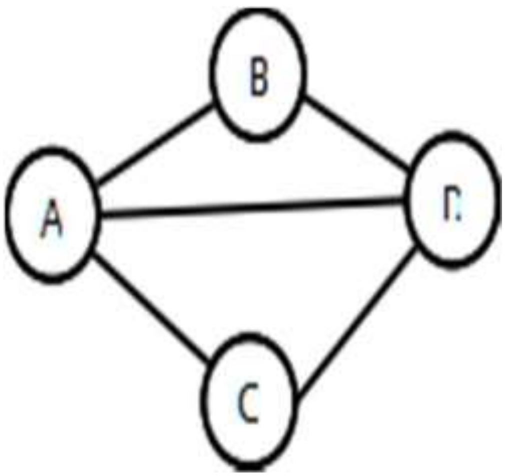


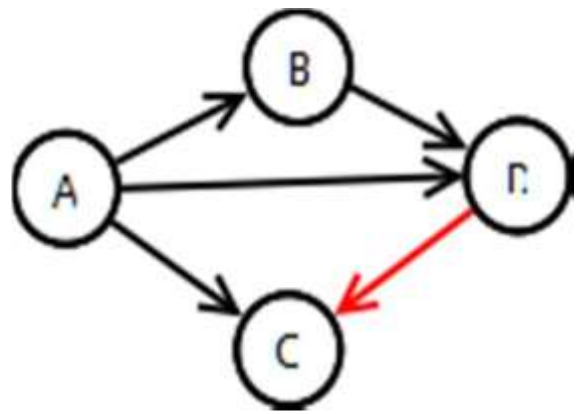
Simple paths cannot contain cycles
Length of a path: Number of edges in the

path Sometimes the sum of the weights of the edges

Basic Terminologies

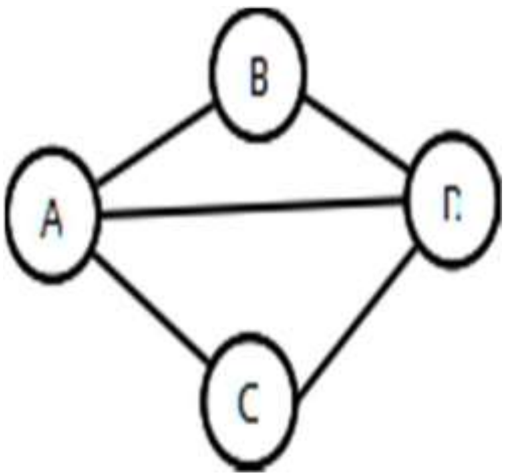
- A **Cyclic** graph contains cycles Example: roads (normally)
  An **acyclic** graph contains no cycles

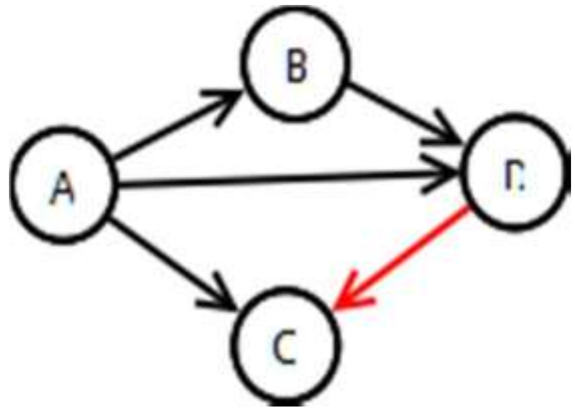




Basic Terminologies

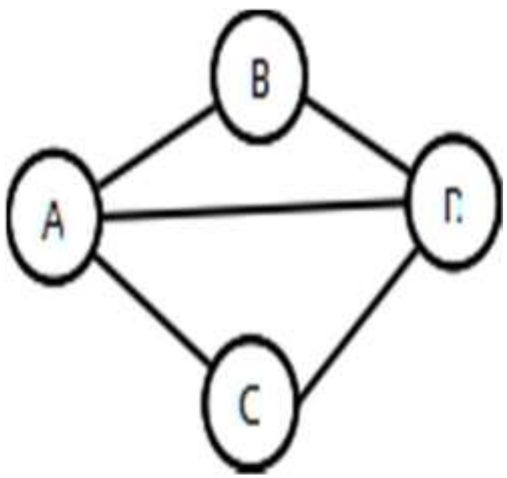
- A **Cyclic** graph contains cycles Example: roads (normally)
  An **acyclic** graph contains no cycles

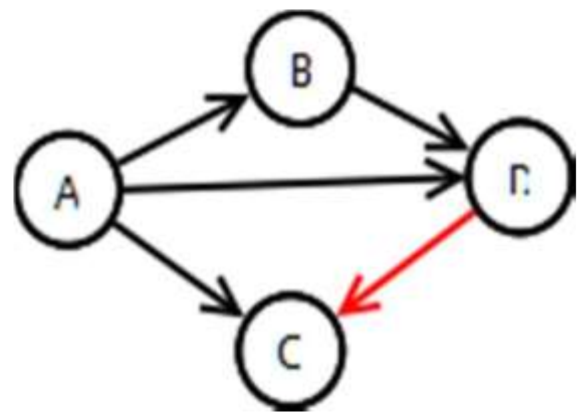




Basic Terminologies

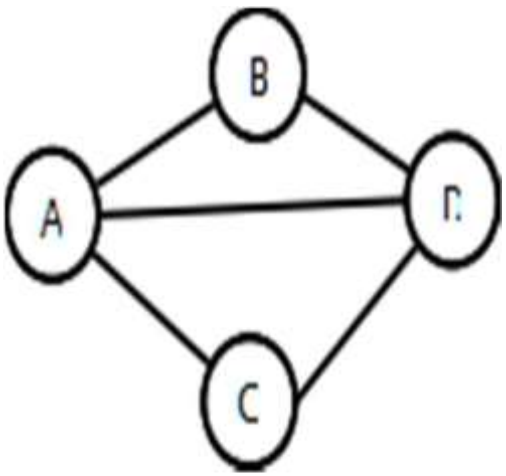
- A **Cyclic** graph contains cycles Example: roads (normally)
  An **acyclic** graph contains no cycles

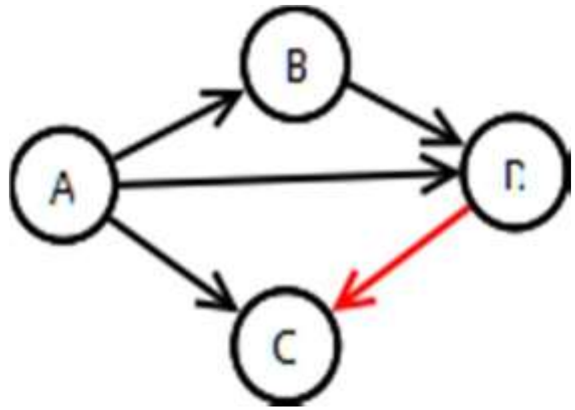




Basic Terminologies

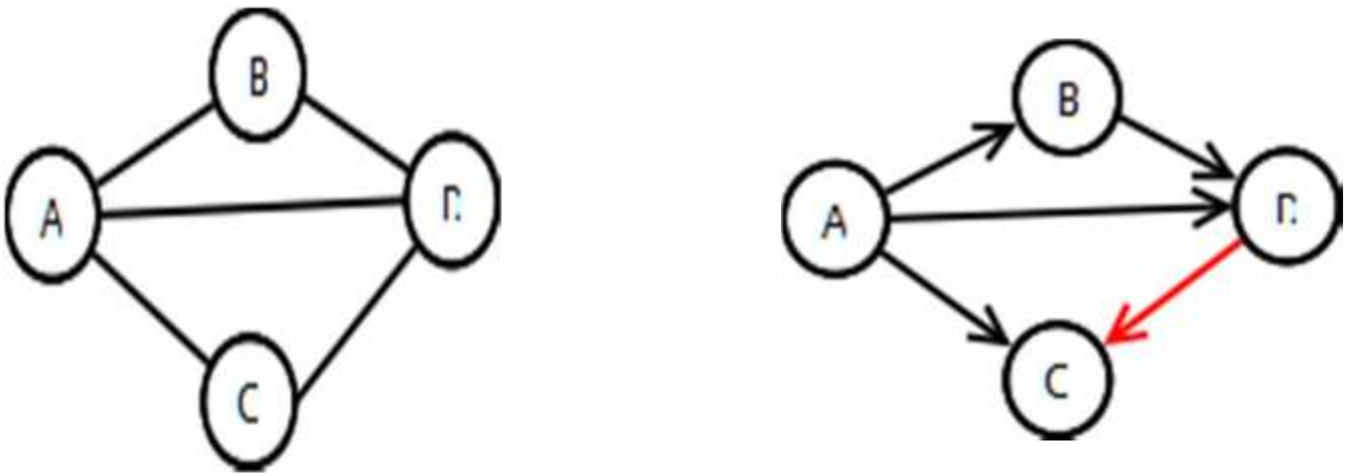
- A **Cyclic** graph contains cycles Example: roads (normally)
  An **acyclic** graph contains no cycles





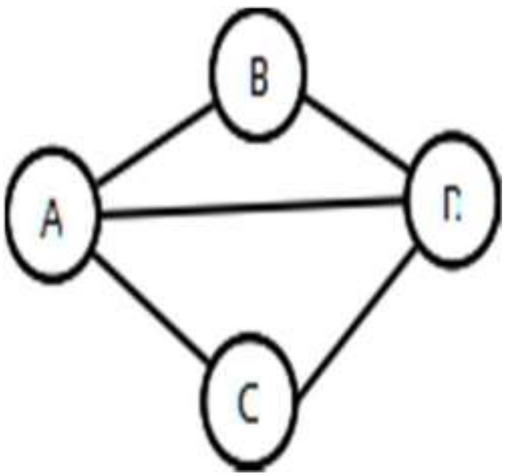
Basic Terminologies

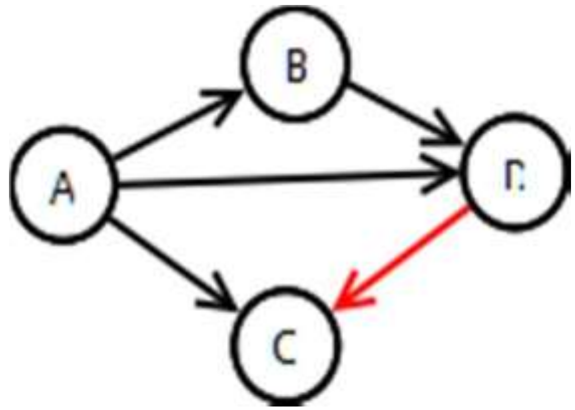
- A **Cyclic** graph contains cycles Example: roads (normally)
  An **acyclic** graph contains no cycles



Basic Terminologies

- A **Cyclic** graph contains cycles Example: roads (normally)
  An **acyclic** graph contains no cycles





Basic Terminologies

Degree

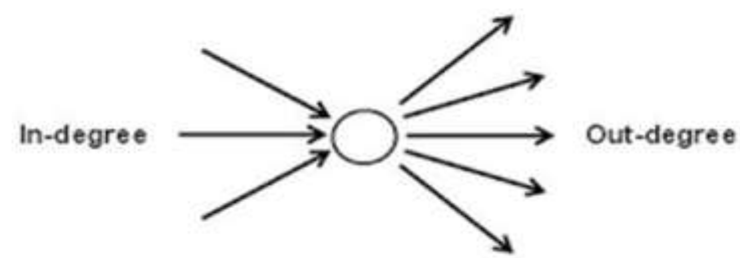
- Total number of edges connected to a vertex is said to be degree of that vertex.

Indegree

- Total number of incoming edges connected to a vertex is said to be indegree of that vertex.

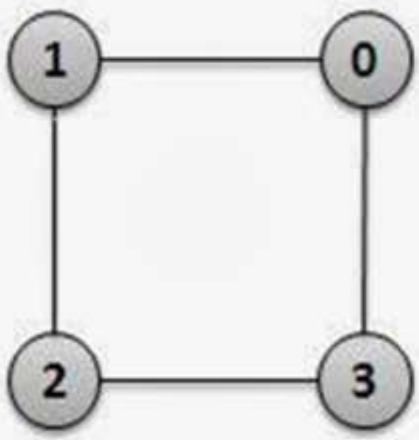
Outdegree

- Total number of outgoing edges connected to a vertex is said to be outdegree of that vertex.



Data Structures for Representing Graphs
Adjacency Matrix –Undirected Graph

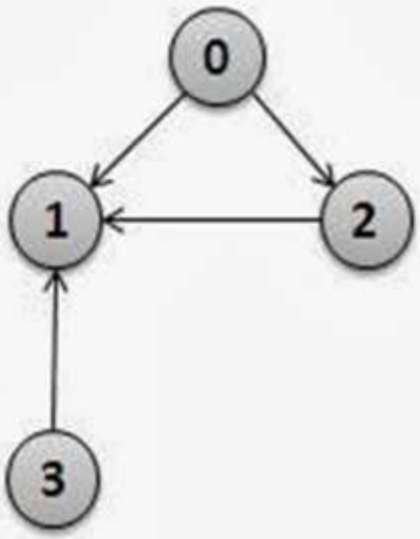
- Adjacency matrix



	0	1	2	3
0	0	1	0	1
1	1	0	1	0
2	0	1	0	1
3	1	0	1	0

Adjacency Matrix Representation of Undirected Graph

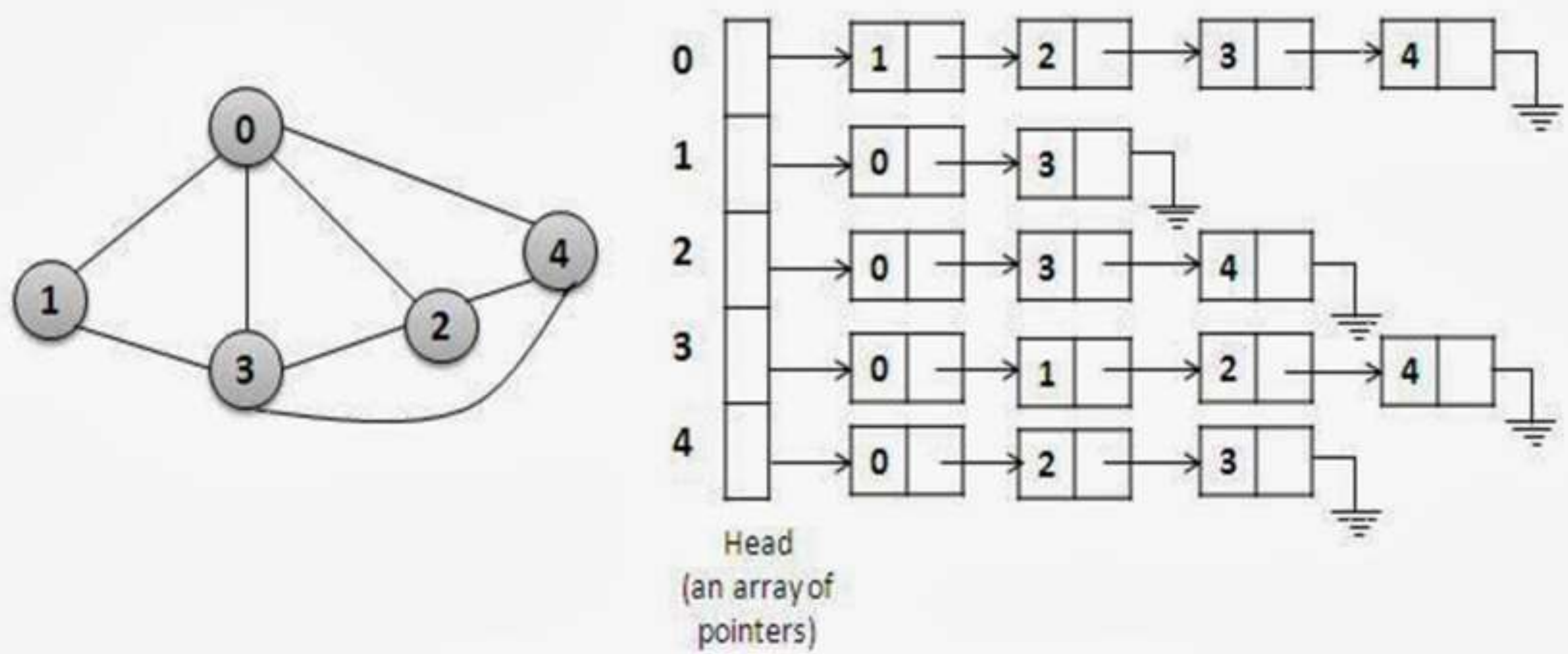
Data Structures for Representing Graphs
Adjacency Matrix –Directed Graph



	0	1	2	3
0	0	1	1	0
1	0	0	0	0
2	0	1	0	0
3	0	1	0	0

Adjacency Matrix Representation of Directed Graph

Data Structures for Representing Graphs
Adjacency List



Adjacency List Representation of Graph

Summary of Graph Introducution

- Basic Terminologies
- Representation of Graph
- Adjacency Matrix
- Adjacency List

Recap

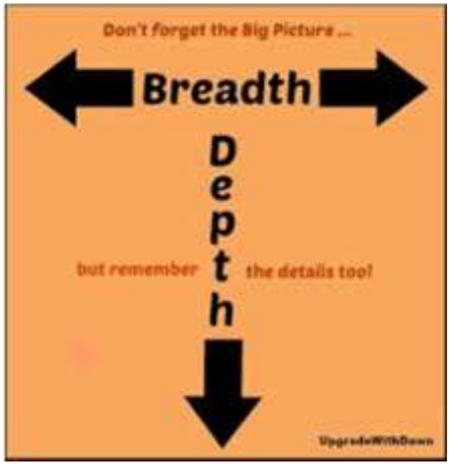
Graphs

- Basic Terminologies
- Representation of Graph

SNS COLLEGE OF TECHNOLOGY

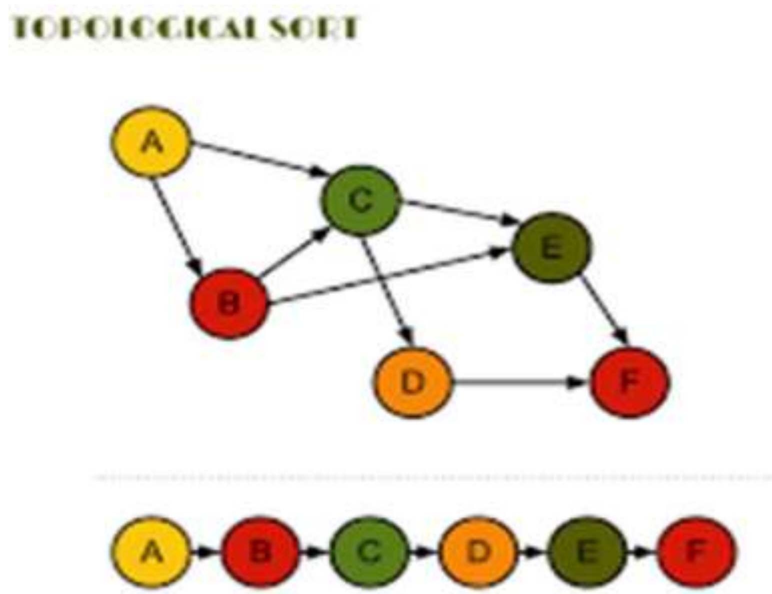
(Autonomous)

COIMBATORE-35



Topological Sort

Linear ordering of Vertices in a directed acyclic graph such that if there is a path from Vi to Vj then Vj appears after Vi in the linear ordering



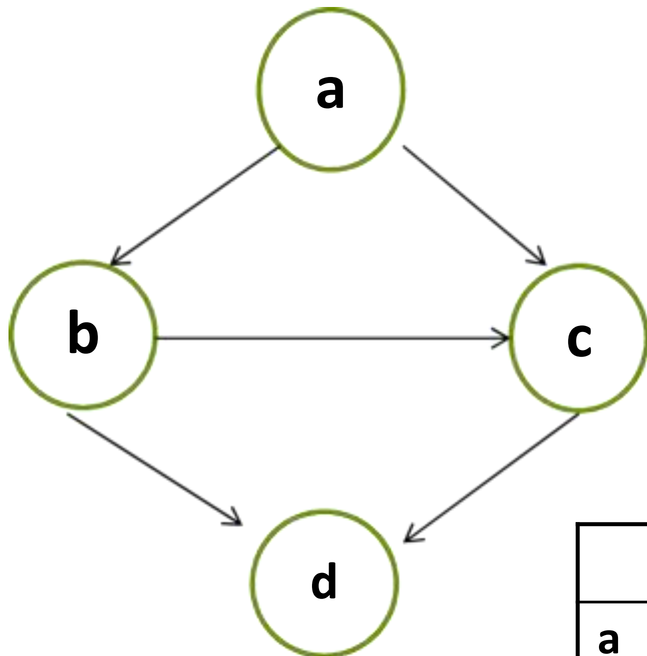
Steps in Topological Sort

Step 1	Find the indegree for every vertex
Step 2	Place the vertices whose indegree is '0' on the empty queue
Step 3	Dequeue the vertex V and decrement the indegree's of all the adjacent vertices
Step 4	Enqueue the vertex on the queue,if its indegree falls to zero
Step 5	Repeat from step 3 until the queue becomes empty
Step 6	The topological ordering is the order in which the vertices dequeued

Routine for Topological Sort

```
Void Topsort(Graph G)
{
Queue Q;
Int counter=0;
Q=CreateQueue (Num Vertex)
Makeempty(Q);
For each vertex V
If (indegree[V]==0)
Enqueue(V,Q);
While(!IsEmpty(Q))
{
V=Dequeue(Q);
TopNum[V]=+counter;
For each W adjacent to V
IF(--Indegree[W]==0
Enqueue(W,Q);
}
If(counter !=Num Vettex)
Error("Graph has a cycle");
DisposeQueue(Q);
}
```

Topological Sort - Example



	a	b	c	d
a	0	1	1	0
b	0	0	1	1
c	0	0	0	1
d	0	0	0	0

Step 1	Indegree[a]=0 Indegree[b]=1 Indegree[c]=2 Indegree[d]=2
Step 2	Enqueue the Vertex,whose Indegree is’0’ Vertex ‘a’ is 0,so place it on the queue
Step 3	Dequeue the vertex ‘a’ from the queue and decrement the indegree’s of all the adjacent vertices’b’ & ‘c’ Hence, **Indegree[b]=0 and Indegree[c]=1** Now,Enqueue the vertex ‘b’ as its indegree becomes zero
Step 4	Dequeue the vertex ‘b’ from Q and decrement the indegree’s of its adjacent vertices’c’ & ‘d’ Hence**, Indegree[c]=0 and Indegree[d]=1** Now,Enqueue the vertex ‘c’ as its indegree becomes

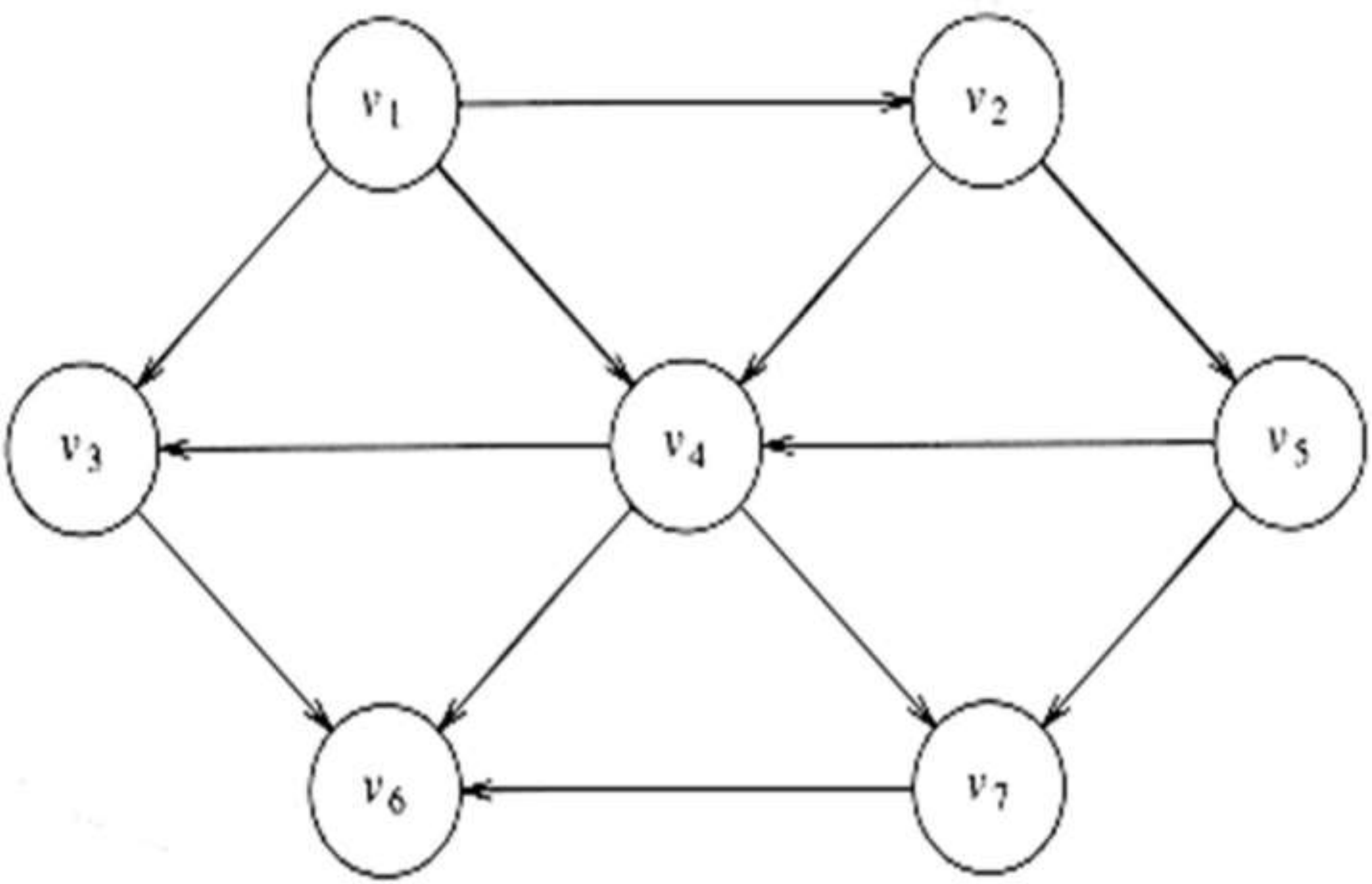
Steps in Topological Sort

Step 5	Dequeue the vertex ‘c’ from Q and decrement the indegree’s of its adjacent vertices’d’ Hence, **Indegree[d]=0** Now,Enqueue the vertex ‘d’ as its indegree becomes zero
Step 6	Dequeue the vertex ‘d’
Step 7	As the queue becomes empty, topological ordering is performed , which is nothing but the order in which the vertices are dequeued

Result of the Graph

Vertex	1	2	3	4
a	**0**	0	0	0
b	1	**0**	0	0
c	2	1	**0**	0
d	2	2	1	**0**
Enqueue	a	b	c	d
Dequeue	a	b	c	d

Example 2



Result of the Graph

Indegree Before Dequeue #							
Vertex	1	2	3	4	5	6	7
v_1	0	0	0	0	0	0	0
v_2	1	0	0	0	0	0	0
v_3	2	1	1	1	0	0	0
v_4	3	2	1	0	0	0	0
v_5	1	1	0	0	0	0	0
v_6	3	3	3	3	2	1	0
v_7	2	2	2	1	0	0	0
enqueue	v_1	v_2	v_5	v_4	v_3	v_7	v_6
dequeue	v_1	v_2	v_5	v_4	v_3	v_7	v_6

•

Summary of Toplogical Sort

- Linear Ordering of Vertices
- Find the Indegree of all vertices
- Topological sort algorithim
- Topological ordering of vertices