Tree and Graph

Unit: 5

Discrete Structure

B. Tech 3rd Sem

RAHUL KUMAR
Assistant Professor
B.Tech CSE

NOIDA INSTITUTE OF ENGINEERING & TECHNOLOGY, GREATER NOIDA(An Autonomous Institute)

EVALUATION SCHEME SEMESTER-III

Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Schemes				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
WEEKS COMPULSORY INDUCTION PROGRAM													
1	AAS0303	Statistics and Probability	3	1	0	30	20	50		100		150	4
2	ACSE0306	Discrete Structures	3	0	0	30	20	50		100		150	3
3	ACSE0305	Computer Organization & Architecture	3	0	0	30	20	50		100		150	3
4	ACSE0302	Object Oriented Techniques using Java	3	0	0	30	20	50		100		150	3
5	ACSE0301	Data Structures	3	1	0	30	20	50		100		150	4
6	ACSDS0301	Foundations of Data Science	3	0	0	30	20	50		100		150	3
7	ACSE0352	Object Oriented Techniques using Java Lab	0	0	2				25		25	50	1
8	ACSE0351	Data Structures Lab	0	0	2				25		25	50	1
9	ACSDS0351	Data Analysis Lab	0	0	2				25		25	50	1
10	ACSE0359	Internship Assessment-I	0	0	2				50			50	1
11	ANC0301 / ANC0302	Cyber Security* / Environmental Science*(Non Credit)	2	0	0	30	20	50		50		100	0
12		MOOCs (For B.Tech. Hons. Degree)											
		GRAND TOTAL										1100	24

Subject Syllabus

B. TECH. SECOND YEAR (3rd Semester))-CSE/IT/CS/M.Tech. Integrated/Data Science/AI/AI-ML/IoT

Course code		L	T	P	Credits
Course title	DISCRETE STRUCTURES	3	0	0	3

Course objective:

The subject enhances one's ability to develop logical thinking and ability to problem solving. The objective of discrete structure is to enables students to formulate problems precisely, solve the problems, apply formal proofs techniques and explain their reasoning clearly.

Pre-requisites:

1. Basic Understanding of mathematics
2. Basic knowledge algebra.
3. Basic knowledge of mathematical notations

Subject Syllabus

Course Contents / Syllabus		
Unit 1	**Set Theory, Relation, Function**	**8 Hours**
Set Theory: Introduction to Sets and Elements, Types of sets, Venn Diagrams, Set Operations, Multisets, Ordered pairs. Proofs of some general Identities on sets. **Relations:** Definition, Operations on relations, Pictorial Representatives of Relations, Properties of relations, Composite Relations, Recursive definition of relation, Order of relations. **Functions:** Definition, Classification of functions, Operations on functions, Growth of Functions. **Combinatorics:** Introduction, basic counting Techniques, Pigeonhole Principle. **Recurrence Relation & Generating function**: Recursive definition of functions, Recursive Algorithms, Method of solving Recurrences. **Proof techniques:** Mathematical Induction, Proof by Contradiction, Proof by Cases, Direct Proof.		
Unit 2	**Algebraic Structures**	**8 Hours**
Algebraic Structures: Definition, Operation, Groups, Subgroups and order, Cyclic Groups, Cosets, Lagrange's theorem, Normal Subgroups, Permutation and Symmetric Groups, Group Homomorphisms, Rings, Internal Domains, and Fields.		

Subject Syllabus

Unit 3	Lattices and Boolean Algebra	8 Hours

Ordered set, Posets, Hasse Diagram of partially ordered set, Lattices: Introduction, Isomorphic Ordered set, Well ordered set, Properties of Lattices, Bounded and Complemented Lattices, Distributive Lattices.

Boolean Algebra: Introduction, Axioms and Theorems of Boolean Algebra, Algebraic Manipulation of Boolean Expressions, Simplification of Boolean Functions.

Unit 4	Propositional Logic	8 Hours

Propositional Logic: Introduction, Propositions and Compound Statements, Basic Logical Operations, Well-formed formula, Truth Tables, Tautology, Satisfiability, Contradiction, Algebra of Proposition, Theory of Inference.

Predicate Logic: First order predicate, Well-formed formula of Predicate, Quantifiers, Inference Theory of Predicate Logic.

Subject Syllabus

Unit 5	Tree and Graph	8 Hours

Trees: Definition, Binary tree, Complete and Extended Binary Trees, Binary Tree Traversal, Binary Search Tree.

Graphs: Definition and terminology, Representation of Graphs, Various types of Graphs, Connectivity, Isomorphism and Homeomorphism of Graphs, Euler and Hamiltonian Paths, Graph Coloring

Course outcome: After completion of this course students will be able to:

Unit 1	Apply the basic principles of sets, relations & functions and mathematical induction in computer science & engineering related problems.	K3
Unit 2	Understand the algebraic structures and its properties to solve complex problems.	K2
Unit 3	Describe lattices and its types and apply Boolean algebra to simplify digital circuit.	K2, K3
Unit 4	Infer the validity of statements and construct proofs using predicate logic formulas.	K3, K5
Unit 5	Design and use the non-linear data structure like tree and graphs to solve real world problems.	K3, K6

Application in CSE

1. Discrete Structures are useful in studying and describing objects and problems in branches of computer science such as computer algorithms, programming languages.
2. Computer implementations are significant in applying ideas from discrete mathematics to real-world problems, such as in operations research.
3. It is a very good tool for improving reasoning and problem-solving capabilities.
4. Discrete mathematics is used to include theoretical computer science, which is relevant to computing.
5. Discrete structures in computer science with the help of process algebras.

Course Objective

- The subject enhances one’s ability to develop logical thinking and ability to problem solving.

- The objective of discrete structure is to enables students to formulate problems precisely, solve the problems, apply formal proofs techniques and explain their reasoning clearly.

Course Outcome

Course Outcome (CO)	At the end of course , the student will be able to	Bloom's Knowledge Level (KL)
CO1	Apply the basic principles of sets, relations & functions and mathematical induction in computer science & engineering related problems	K3
CO2	Understand the algebraic structures and its properties to solve complex problems	K2
CO3	Describe lattices and its types and apply Boolean algebra to simplify digital circuit.	K2,K3
CO4	Infer the validity of statements and construct proofs using predicate logic formulas.	K3,K5
CO5	Design and use the non-linear data structure like tree and graphs to solve real world problems.	K3,K6

Content

- Course Objective
- Course Outcome
- CO-PO Mapping
- Syllabus
- Prerequisite and Recap
- Graph and digraph
- Incidence and adjacency matrix
- Isomorphism
- Eulerian path and circuits in graphs
- Hamiltonian path and circuits
- Trees
- Four color theorem

Content

- Planar graphs
- Clique number
- Chromatic number
- Video links
- Daily Quiz
- Weekly Assignment
- MCQ
- Old Question papers
- Expected Question for University Exam
- Summary
- References.

Course Objective

- The subject enhances one's ability to develop logical thinking and ability to problem solving.
- Demonstrate the ability to write and evaluate a proof or outline the basic structure of and give examples of each proof technique described.
- Apply logical reasoning to solve a variety of problems.
- Use Mathematically correct terminology and notation.

Course Outcome

Course Outcome (CO)	At the end of course , the student will be able to	Bloom's Knowledge Level (KL)
CO1	Apply the basic principles of sets, relations & functions and mathematical induction in computer science & engineering related problems	K3
CO2	Understand the algebraic structures and its properties to solve complex problems	K2
CO3	Describe lattices and its types and apply Boolean algebra to simplify digital circuit.	K2,K3
CO4	Infer the validity of statements and construct proofs using predicate logic formulas.	K3,K5
CO5	Design and use the non-linear data structure like tree and graphs to solve real world problems.	K3,K6

Program Outcome

Engineering Graduates will be able to Understand:

1. Engineering knowledge
2. Problem analysis
3. Design/development of solutions
4. Conduct investigations of complex
5. Modern tool usage
6. The engineer and society
7. Environment and sustainability
8. Ethics
9. Individual and team work
10. Communication
11. Project management and finance
12. Life-long learning

CO-PO Mapping

CO-PO correlation matrix Discrete Structures (ACSE0306)

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
ACSE0306.1	**2**	**2**	**3**	**3**	**2**	**2**	**-**	**-**	**2**	**1**	**-**	**3**
ACSE0306.2	1	3	2	3	2	2	-	1	1	1	2	2
ACSE0306.3	2	2	3	2	2	2	-	2	2	1	2	3
ACSE0306.4	2	2	2	3	2	2	-	2	2	1	1	3
ACSE306.5	3	2	2	2	2	2	-	2	1	1	1	2
Average	2	2.2	2.4	2.6	2	2	-	1.4	1.6	1	1.2	2.6

End Semester Question Paper Templates (Offline Pattern/Online Pattern

Printed page: .... Subject Code: ....

Roll No:

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY ,GREATER NOIDA

(An Autonomous Institute Affiliated to AKTU, Lucknow)

B.Tech/B.Voc./MBA/MCA/M.Tech (Integrated)

(SEM: ..... THEORY EXAMINATION (2020-2021)

Subject ....

Time: 3 Hours **Max. Marks:100**

General Instructions:

- All questions are compulsory. Answers should be brief and to the point.
- This Question paper consists of ....pages & ...8.........questions.
- It comprises of three Sections, A, B, and C. You are to attempt all the sections.
- Section A -Question No- 1 is objective type questions carrying 1 mark each, Question No- 2 is very short

- **Section B** - Question No-3 is Long answer type -I questions with external choice carrying 6 marks each. You need to attempt any five out of seven questions given.
- **Section C** - Question No. 4-8 are Long answer type –II (within unit choice) questions carrying 10 marks each. You need to attempt any one part *a or b.*
- Students are instructed to cross the blank sheets before handing over the answer sheet to the invigilator.
- No sheet should be left blank. Any written material after a blank sheet will not be evaluated/checked.

End Semester Question Paper Templates (Offline Pattern/Online Pattern

		SECTION – A		CO
1.	Attempt all parts-		[10×1=10]	
	1-a.	Question-	(1)	
	1-b.	Question-	(1)	
	1-c.	Question-	(1)	
	1-d.	Question-	(1)	
	1-e.	Question-	(1)	
	1-f.	Question-	(1)	
	1-g.	Question-	(1)	
	1-h.	Question-	(1)	
	1-i.	Question-	(1)	
	1-j.	Question-	(1)	

End Semester Question Paper Templates (Offline Pattern/Online Pattern

2.	Attempt all parts-		[5×2=10]	CO
	2-a.	Question-	(2)	
	2-b.	Question-	(2)	
	2-c.	Question-	(2)	
	2-d.	Question-	(2)	
	2-e.	Question-	(2)	

End Semester Question Paper Templates (Offline Pattern/Online Pattern

SECTION – B				CO
3.	Answer any five of the following-		[5×6=30]	
	3-a.	Question-	(6)	
	3-b.	Question-	(6)	
	3-c.	Question-	(6)	
	3-d.	Question-	(6)	
	3-e.	Question-	(6)	
	3-f.	Question-	(6)	
	3-g.	Question-	(6)	

End Semester Question Paper Templates (Offline Pattern/Online Pattern

SECTION – C				CO
4	Answer any one of the following-		[5×10=50]	
	4-a.	Question-	(10)	
	4-b.	Question-	(10)	
5.	Answer any one of the following-			
	5-a.	Question-	(10)	
	5-b.	Question-	(10)	

End Semester Question Paper Templates (Offline Pattern/Online Pattern

6.	**Answer any one of the following-**			
	6-a.	**<u>Question-</u>**	**(10)**	
	6-b.	**<u>Question-</u>**	**(10)**	
7.	**Answer any one of the following-**			
	7-a.	**<u>Question-</u>**	**(10)**	
	7-b.	**<u>Question-</u>**	**(10)**	
8.	**Answer any one of the following-**			
	8-a.	**<u>Question-</u>**	**(10)**	
	8-b.	**<u>Question-</u>**	**(10)**	

Prerequisite & Recap

Prerequisite

- Knowledge of Mathematics upto 12^{th} standard.

Recap

- The fundamental concepts of Sets, Relations and Functions, Logic, Probability and Boolean Algebra.

Brief Introduction about the subject with video

Discrete mathematics is the study of mathematical structures that are fundamentally discrete rather than continuous. In contrast to real numbers that have the property of varying "smoothly", the objects studied in discrete mathematics – such as integers, Trees, graphs, and statements in logic.

- https://www.youtube.com/watch?v=Dsi7x-A89Mw&list=PL0862D1A947252D20&index=28
- https://www.youtube.com/watch?v=74l6t4_4pDg&list=PL0862D1A947252D20&index=29
- https://www.youtube.com/watch?v=4d2XEn1j_q4&list=PL0862D1A947252D20&index=30

Unit Content

- Tree
- Binary Tree
- Tree Traversal
- Binary Search Tree
- Graphs
- Types of Graph
- Graph coloring

Unit Objective: Tree and Graph (CO5)

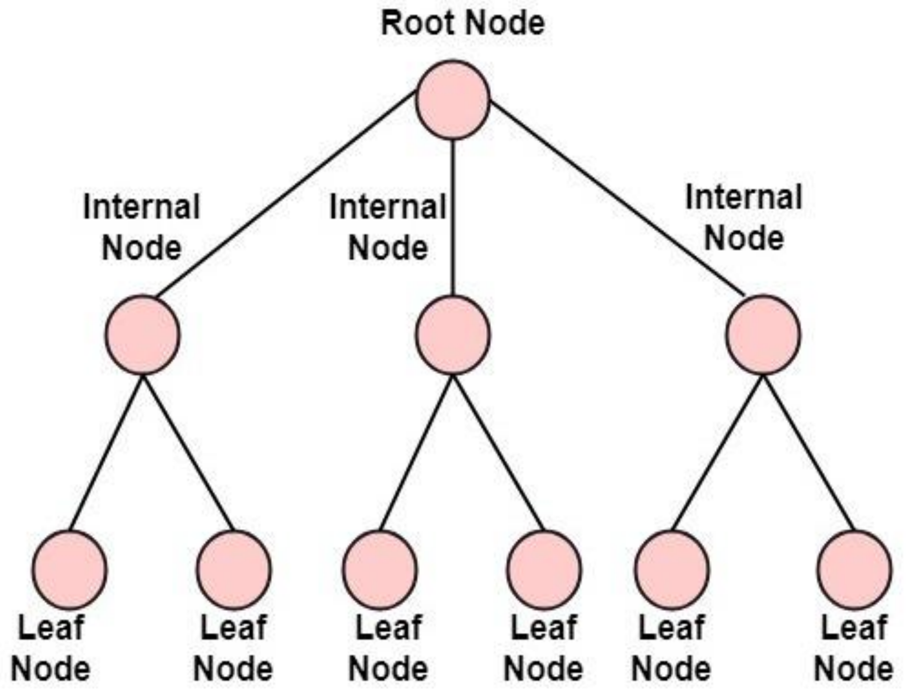
- Define how Tree and graphs will be serve as models for many standard problems.
- How Graph theory will be used in Computer networks to minimize the cost and time of delivery of data.
- To distinguish between two chemical compounds with the same molecular formula but different structures.
- To solve shortest path problems between cities.
- To schedule exams and assign channels to television stations.

Trees (CO5)

- A **tree** is a connected graph containing no cycles.
- A graph T is a tree if and only if between every pair of distinct vertices of T there is a unique path.
- Let T be a tree with v vertices and e edges. Then $e=v-1$.
- Example:

  $G=(V,E)$ with $V=\{a,b,c,d,e\}$ and $E=\{\{a,b\},\{b,c\},\{c,d\},\{d,e\}\}$ is a tree.

Trees terminology (CO5)



Binary Trees (CO5)

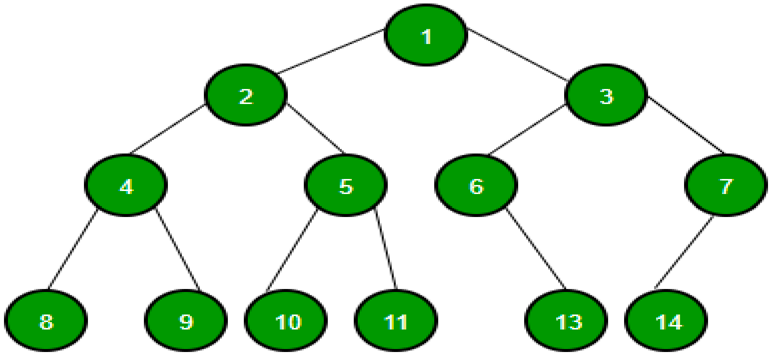
A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

A Binary Tree node contains following parts.

1.Data

2.Pointer to left child

3.Pointer to right child



Binary Tree(Properties)

1) The maximum number of nodes at level ‘l’ of a binary tree is 2^l.

Here level is the number of nodes on the path from the root to the node (including root and node). Level of the root is 0.

This can be proved by induction.

For root, l = 0, number of nodes = $2^0 = 1$

Assume that the maximum number of nodes on level ‘l’ is 2^l

Since in Binary tree every node has at most 2 children, next level would have twice nodes, i.e. $2 * 2^l$

2. The Maximum number of nodes in a binary tree of height ‘h’ is $2^{h+1} – 1$.

3. In a Binary Tree with N nodes, minimum possible height or the minimum number of levels is $\log_2(N+1) - 1$

Types of Binary Tree

1. Complete Binary Tree
2. Full Binary Tree
3. Balanced Binary Tree

Trees (CO5)

Complete Binary Tree: A Binary Tree is a complete Binary Tree if all the levels are completely filled except possibly the last level and the last level has all keys as left as possible

The following are examples of Complete Binary Trees

```
               18
             /     \
          15         30
         /  \        /  \
      40     50    100    40

               18
             /     \
          15         30
         /  \        /  \
      40     50    100    40
     /  \   /
    8    7 9
```

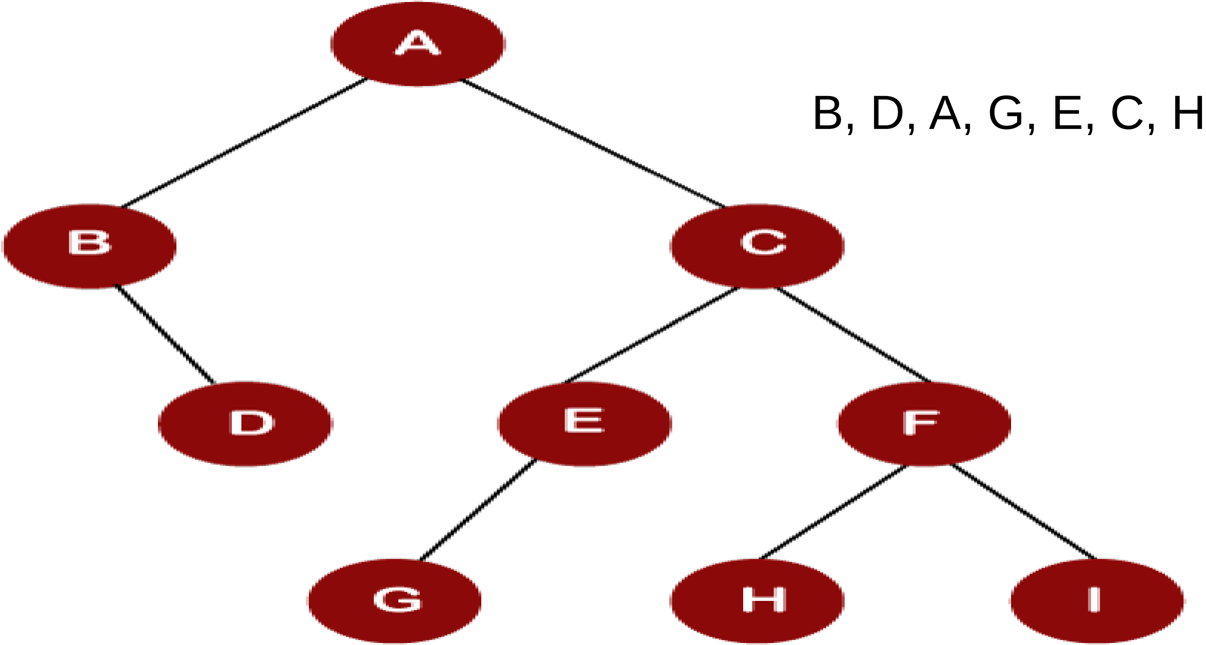
Binary Trees Traversal (CO5)

There are three ways which we use to traverse a tree

- In-order Traversal
- Pre-order Traversal
- Post-order Traversal

Inorder traversal (CO5)

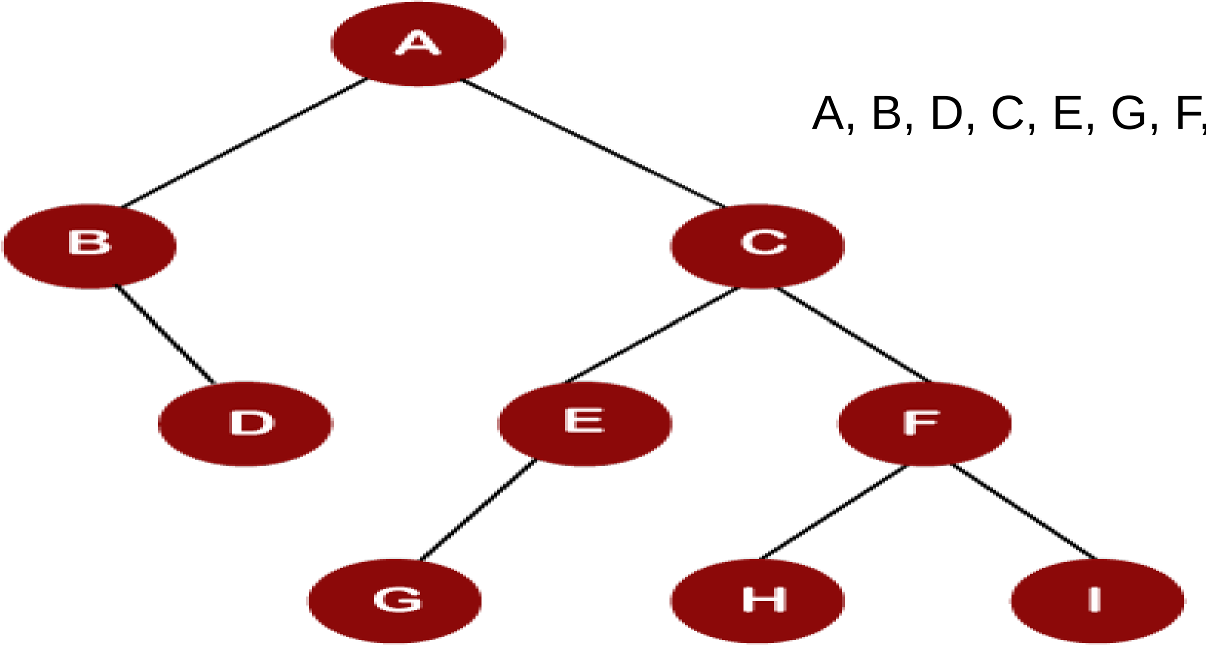
Inorder traversal : Left Root Right.



B, D, A, G, E, C, H, F, I.

Preorder traversal (CO5)

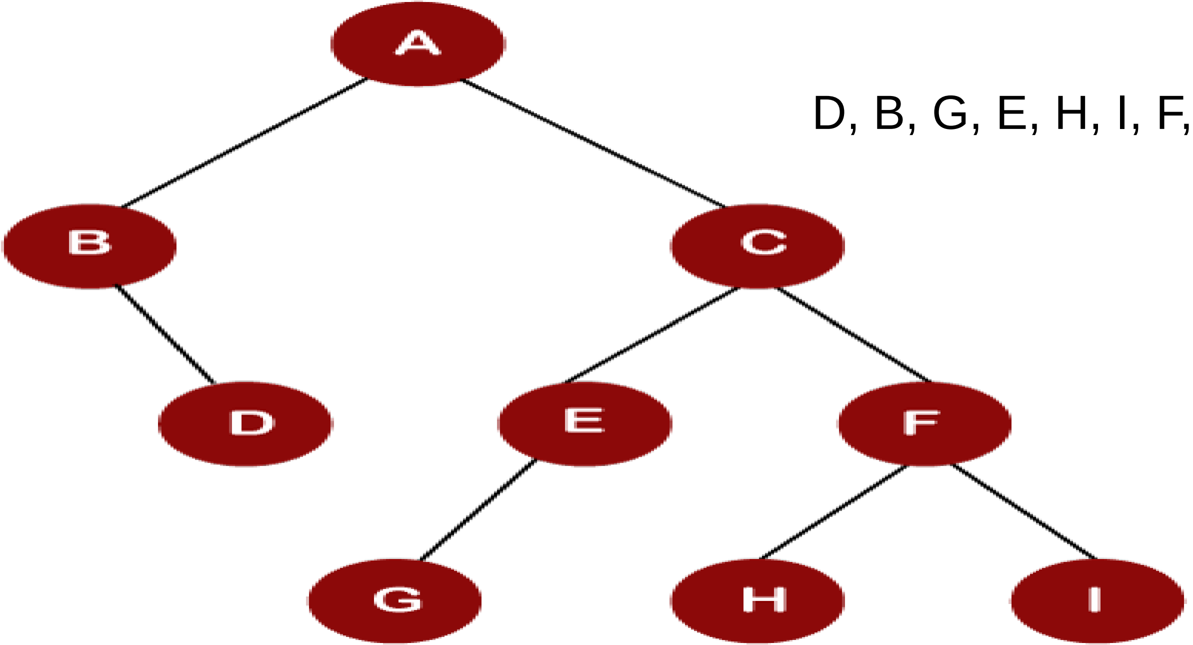
Preorder traversal : Root Left Right.



A, B, D, C, E, G, F, H, I

Postorder traversal (CO5)

Postorder traversal : Left Right Root

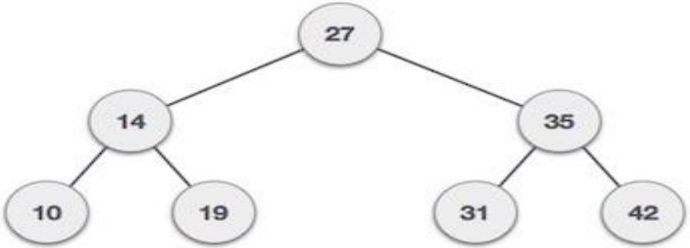


D, B, G, E, H, I, F, C, A.

Binary Search Tree (CO5)

Binary Search tree exhibits a special behavior.

A node's left child must have a value less than its parent's value and the node's right child must have a value greater than its parent value.



BST Basic Operations

The basic operations that can be performed on a binary search tree data structure, are the following −

- **Insert** − Inserts an element in a tree/create a tree.
- **Search** − Searches an element in a tree.
- **Preorder Traversal** − Traverses a tree in a pre-order manner.
- **Inorder Traversal** − Traverses a tree in an in-order manner.

- **Postorder Traversal** − Traverses a tree in a post-order manner.

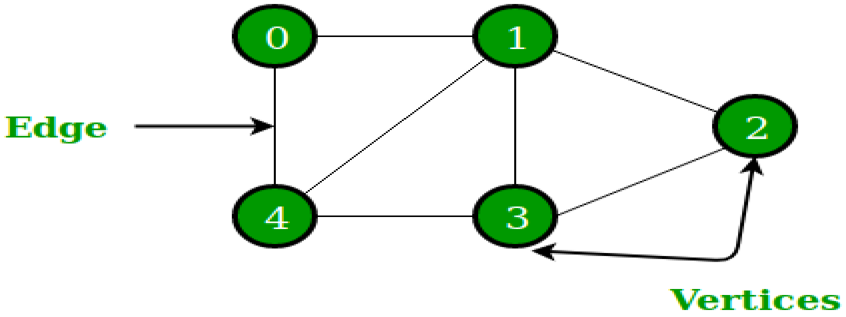
We shall learn creating (inserting into) a tree structure and searching a data item in a tree in this chapter. We shall learn about tree traversing methods in the coming chapter.

Definition of a Graph(CO 5)

- A generalization of the simple concept of a set of dots, links, edges or arcs.

- Representation:

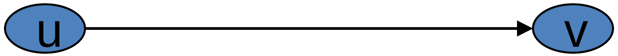
Graph G =(V, E) consists set of vertices denoted by V, or by V(G) and set of edges E, or E(G)



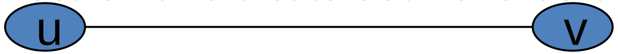
In the above Graph, the set of vertices V = {0,1,2,3,4} and the set of edges E = {01, 12, 23, 34, 04, 14, 13}.

Definitions – Edge Type(CO 5)

Directed: Ordered pair of vertices. Represented as (u, v) directed from vertex u to v.

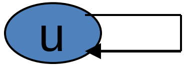


Undirected: Unordered pair of vertices. Represented as {u, v}. Disregards any sense of direction and treats both end vertices interchangeably

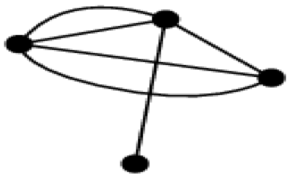


Definitions – Edge Type(CO 5)

Loop: A loop is an edge whose endpoints are equal i.e., an edge joining a vertex to it self is called a loop. Represented as {u, u} = {u}



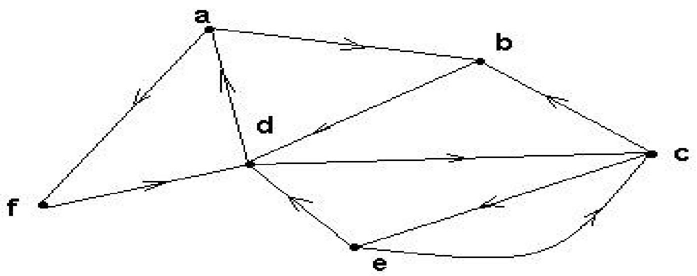
Multiple Edges: Two (; the same pair of vertices



nonsimple graph with multiple edges

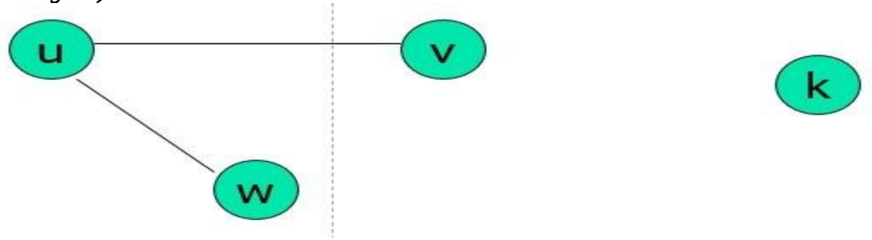
Directed graphs(Digraph) (CO 5)

G is a directed graph or digraph if each edge has been associated with an ordered pair of vertices, i.e. each edge has a direction



Terminology – Undirected graphs(CO5)

- u and v are **adjacent** if {u, v} is an edge, e is called **incident** with u and v. u and v are called **endpoints** of {u, v}
- **Degree of Vertex (deg (v)):** the number of edges incident on a vertex. A loop contributes twice to the degree (why?).
- **Pendant Vertex:** deg (v) =1
- **Isolated Vertex:** deg (v) = 0



Representation Example: For V = {u, v, w} , E = { {u, w}, {u, w}, (u,v) }, deg (u) = 2, deg (v) = 1, deg (w) = 1, deg (k) = 0, w and v are pendant , k is isolated

Terminology – Undirected graphs(CO5)

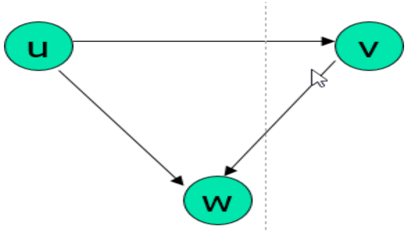
- For the edge (u, v), u is **adjacent to** v OR v is **adjacent from** u, u – **Initial vertex**, v – **Terminal vertex**

- **In-degree (deg^- (u)):** number of edges for which u is terminal vertex

- **Out-degree (deg^+ (u)):** number of edges for which u is initial vertex

Note: A loop contributes 1 to both in-degree and out-degree (why?)

Representation Example: For V = {u, v, w} , E = { (u, w), (v, w), (u, v) }, deg^- (u) = 0, deg^+ (u) = 2, deg^- (v) = 1,
deg^+ (v) = 1, and deg^- (w) = 2, deg^+ (u) = 0



Handshaking Theorem (CO4)

Handshaking Theorem

Handshaking theorem states that **the sum of degrees of the vertices of a graph is twice the number of edges**. If G=(V,E) be a graph with E edges, then-

$$\Sigma \deg \ G(V) = 2E$$

Proof-

Since the degree of a vertex is the number of edges incident with that vertex, the sum of degree counts the total number of times an edge is incident with a vertex. Since every edge is incident with exactly two vertices, each edge gets counted twice once at each end. Thus the sum of the degrees is equal twice the number of edges.

This theorem applies even if multiple edges and loops are present. The theorem holds this rule that if several people shake hands, **the total number of hands shake must be even** that is why the theorem is called **handshaking theorem**.

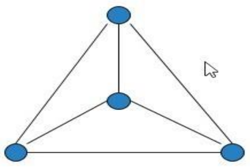
Summary of Graph Types(CO5)

Type	Edges	Multiple Edges Allowed ?	Loops Allowed ?
Simple Graph	undirected	No	No
Multigraph	undirected	Yes	No
Pseudograph	undirected	Yes	Yes
Directed Graph	directed	No	Yes
Directed Multigraph	directed	Yes	Yes

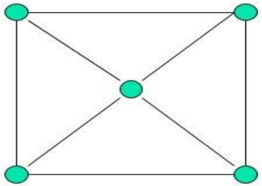
Simple graph – special case(CO 5)

Wheels: W_n, obtained by adding additional vertex to Cn and connecting all vertices to this new vertex by new edges.

Representation Example: W_3, W_4



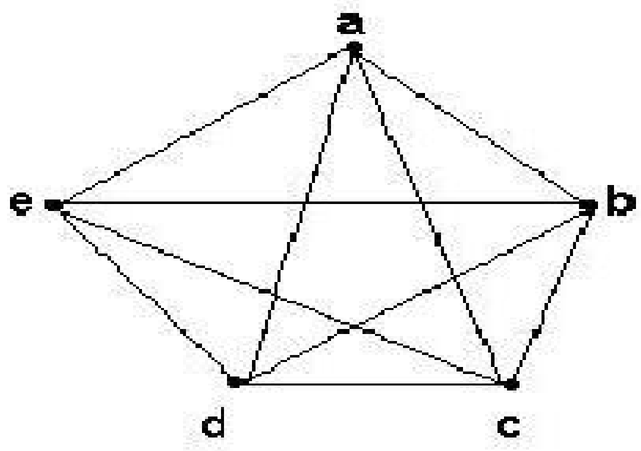
W_3



W_4

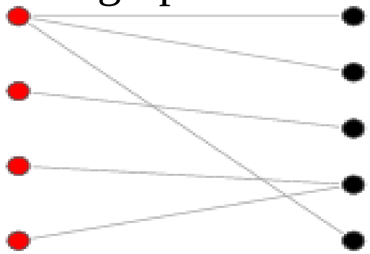
Complete graph K_n (CO 5)

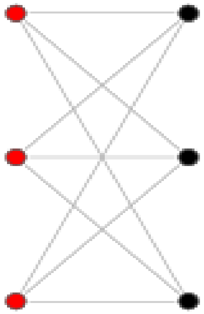
- Let $n \geq 3$
- The *complete graph* K_n is the graph with n vertices and every pair of vertices is joined by an edge.
- The figure represents K_5

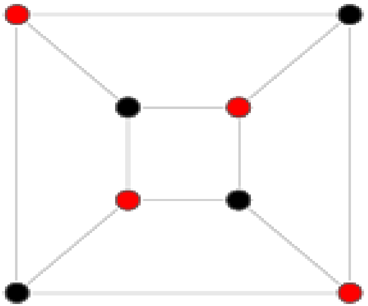


Bipartite graphs(CO 5)

A bipartite graph, also called a bi-graph, is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent. A bipartite graph is a special case of a *k*-partite graph with *k*=2 .





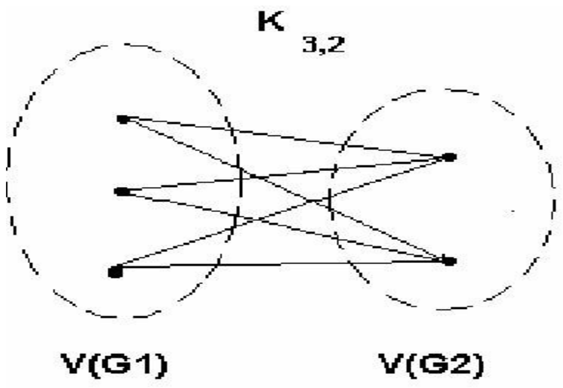


The illustration above shows some bipartite graphs, with vertices in each graph colored based on to which of the two disjoint sets they belong.

Bipartite graphs are equivalent to two-colorable graphs. All acyclic graphs are bipartite. A cyclic graph is bipartite iff all its cycles are of even length

Complete bipartite graph $K_{m,n}$ (CO 5)

- A bipartite graph is the *complete* bipartite graph $K_{m,n}$ if every vertex in $V(G_1)$ is joined to a vertex in $V(G_2)$ and conversely,
- $|V(G_1)| = m$
- $|V(G_2)| = n$



Representation of Graphs(CO5)

Incidence (Matrix): Most useful when information about edges is more desirable than information about vertices.

Adjacency (Matrix/List): Most useful when information about the vertices is more desirable than information about the edges. These two representations are also most popular since information about the vertices is often more desirable than edges in most applications.

Representation- Incidence Matrix(CO5)

- G = (V, E) be an unditected graph. Suppose that $v_1, v_2, v_3, \ldots, v_n$ are the vertices and $e_1, e_2, \ldots, e_m$ are the edges of G. Then the incidence matrix with respect to this ordering of V and E is the n x m matrix $M = [m_{ij}]$, where

$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i \\ 0 & \text{otherwise} \end{cases}$$

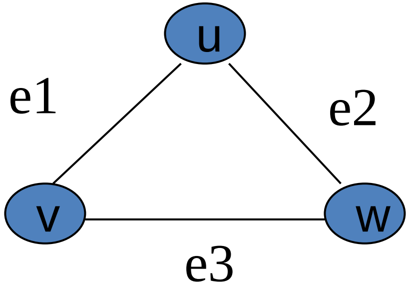
Can also be used to represent :

Multiple edges: by using columns with identical entries, since these edges are incident with the same pair of vertices

Loops: by using a column with exactly one entry equal to 1, corresponding to the vertex that is incident with the loop

Representation-Incidence matrix(CO5)

- Representation Example: G = (V, E)



	e_1	e_2	e_3
v	1	0	1
u	1	1	0
w	0	1	1

Representation- Adjacency Matrix(CO5)

- There is an N x N matrix, where |V| = N , the Adjacenct Matrix (NxN) A = $[a_{ij}]$

For undirected graph

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G \\ 0 & \text{otherwise} \end{cases}$$

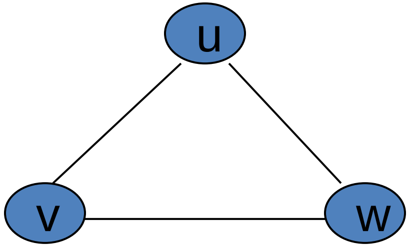
For directed graph

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \text{ is an edge of } G \\ 0 & \text{otherwise} \end{cases}$$

This makes it easier to find sub-graphs, and to reverse graphs if needed.

Representation-Adjacency matrix(CO5)

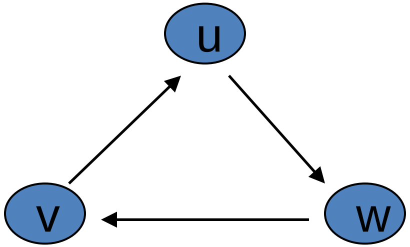
- Representation of undirected graph G = (V, E)



	v	u	w
v	0	1	1
u	1	0	1
w	1	1	0

Representation-Adjacency matrix(CO5)

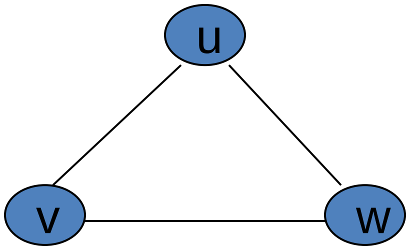
- Representation of directed graph G = (V, E)



	v	u	w
v	0	1	0
u	0	0	1
w	1	0	0

Representation-Adjacency list (CO5)

- Each node (vertex) has a list of which nodes (vertex) it is adjacent
- Representation of undirected graph G = (V, E)



node	Adjacency List
u	v , w
v	w, u
w	u , v

Graph - Isomorphism(CO5)

- G1 = (V1, E2) and G2 = (V2, E2) are isomorphic if:
- There is a one-to-one and onto function f from V1 to V2 with the property that

  --a and b are adjacent in G1 if and only if f (a) and f (b) are adjacent in G2, for all a and b in V1.
- Function f is called isomorphism.

- Application Example: In chemistry, to find if two compounds have the same structure

Graph - Isomorphism(CO5)

Representation example: G1 = (V1, E1) , G2 = (V2, E2)

$f(u_1) = v_1$, $f(u_2) = v_4$, $f(u_3) = v_3$, $f(u_4) = v_2$,

Graph - Homomorphism(CO5)

A homomorphism from a graph ***G*** to a graph ***H*** is a mapping (May not be a bijective mapping) h: G → H such that − (x, y) ∈ E(G) → (h(x), h(y)) ∈ E(H). It maps adjacent vertices of graph G to the adjacent vertices of the graph ***H***.

Properties of Homomorphisms

- A homomorphism is an isomorphism if it is a bijective mapping.
- Homomorphism always preserves edges and connectedness of a graph.
- The compositions of homomorphisms are also homomorphisms.
- To find out if there exists any homomorphic graph of another graph is a NPcomplete problem.

Connectivity in Graph (CO5)

- **Basic idea of connectivity: In a Graph reachability among vertices can be acheived by traversing the edges**

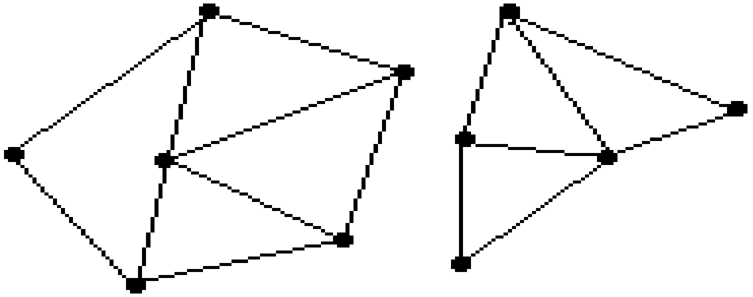
  Application Example:

  - In a city to city road-network, if one city can be reached from another city.

  - Problems if determining whether a message can be sent between two computer using intermediate links

  - Efficiently planning routes for data delivery in the Internet

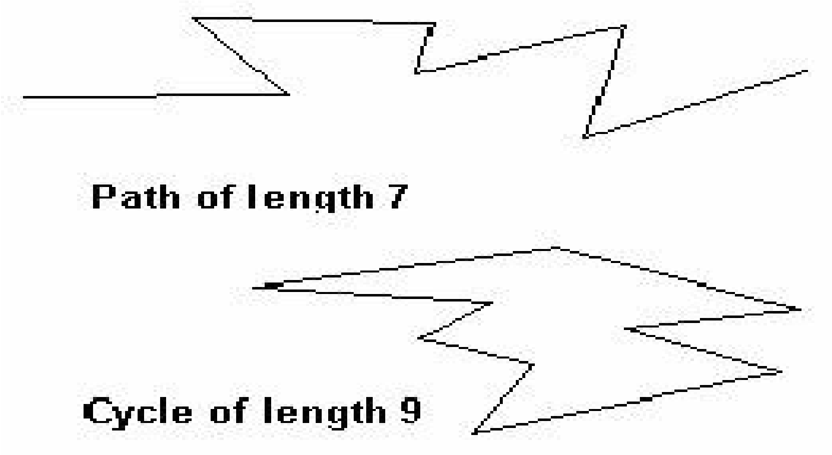
- A graph is *connected* if every pair of vertices can be connected by a path.
- A *connected graph* is an undirected graph in which every unordered pair of vertices in the graph is connected. Otherwise, it is called a *disconnected graph*.
- Each connected subgraph of a non- connected graph G is called a *component* of G.



2 connected components

Paths and cycles in Graph (CO 5)

- A *path of length n* is a sequence of n + 1 vertices and n consecutive edges.
- A *cycle* is a path that begins and ends at the same vertex.

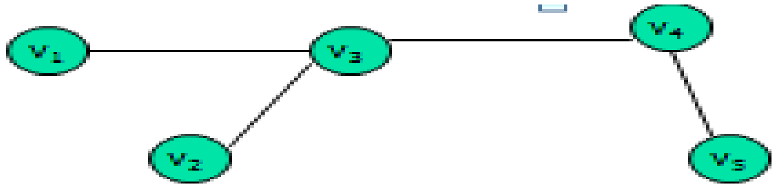


Connectivity- Undirected Graph(CO5)

Undirected Graph

An undirected graph is connected if there exists is a simple path between every pair of vertices.

Representation Example: G (V, E) is connected since for V = {v_1, v_2, v_3, v_4, v_5}, there exists a path between {v_i, v_j}, $1 \leq i, j \leq 5$



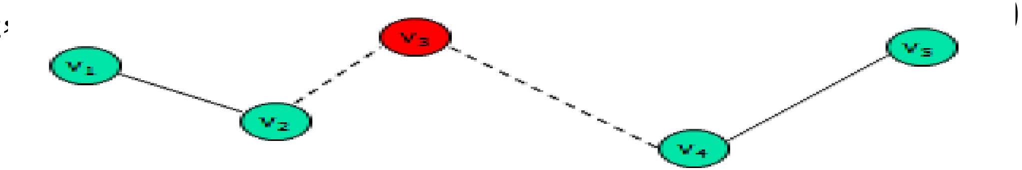
Connectivity- Undirected Graph(CO4)

Undirected Graph

- **Articulation Point (Cut vertex):** removal of a vertex produces a subgraph with more connected components than in the original graph. The removal of a cut vertex from a connected graph produces a graph that is not connected

- **Cut Edge:** An edge whose removal produces a subgraph with more connected components than in the original graph.

  Representation example: G (V, E), v_3 is the articulation point or edge $\{v_2,$



Connectivity- Directed Graph(CO5)

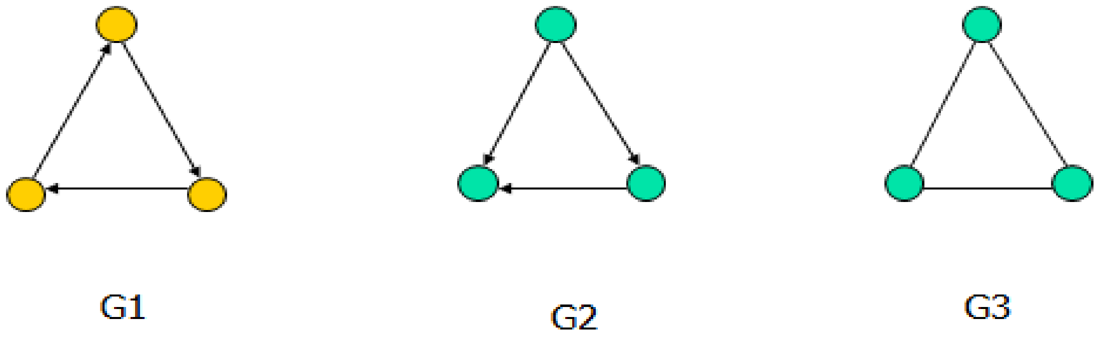
Directed Graph

- 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
- A directed graph is **weakly connected** if there is a (undirected) path between every two vertices in the underlying undirected path

A strongly connected Graph can be weakly connected but the vice-versa is not true (why?)

Directed Graph

Representation example: G1 (Strong component), G2 (Weak Component), G3 is undirected graph representation of G2 or G1

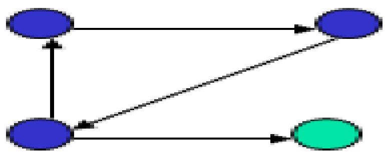


Connectivity- Directed Graph(CO5)

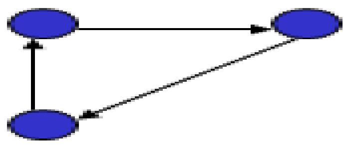
- **Directed Graph**

  Strongly connected Components: subgraphs of a Graph G that are strongly connected

  Representation example: G1 is the strongly connected component in G



G



G1

Topic objective : Euler Graph(CO5)

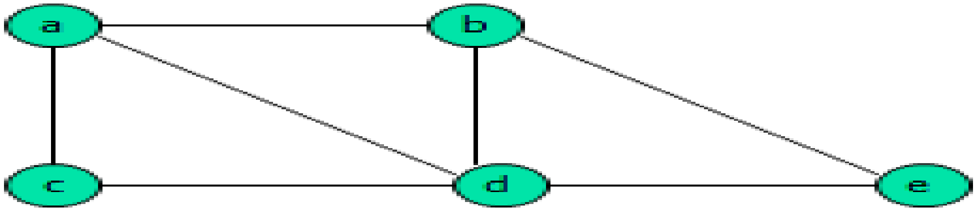
- **Eulerian graphs** will be used to solve many practical problems like Konisberg Bridge problem.
- They can also be used to by mail carriers who want to have a route where they don’t retrace any of their previous steps.

Euler- Definitions(CO5)

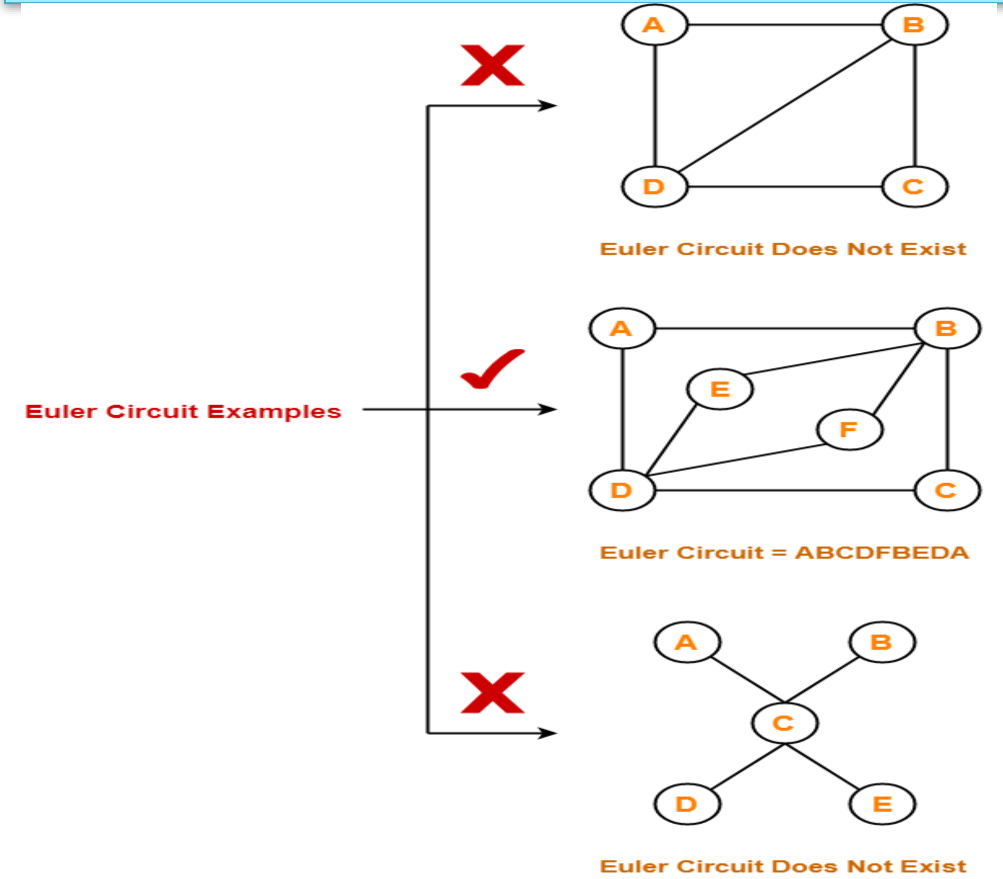
- An **Eulerian path** (**Eulerian trail**, **Euler walk**) in a graph is a path that uses each edge precisely once. If such a path exists, the graph is called **traversable**.

- An **Eulerian cycle** (**Eulerian circuit**, **Euler tour**) in a graph is a cycle that uses each edge precisely once. If such a cycle exists, the graph is called **Eulerian** (also **unicursal**).

- Representation example: G1 has Euler path a, c, d, e, b, d, a, b

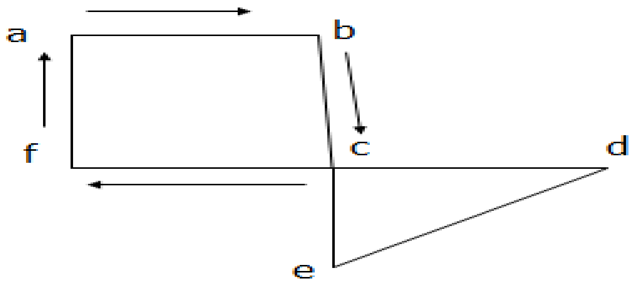


Euler Circuit(CO5)



Euler Theorem1(CO5)

1. A connected graph G is Eulerian if and only if G is connected and has no vertices of odd degree.



<u>Building a simple path</u>:
{a,b}, {b,c}, {c,f}, {f,a}

Euler circuit constructed if all edges are used. True here?

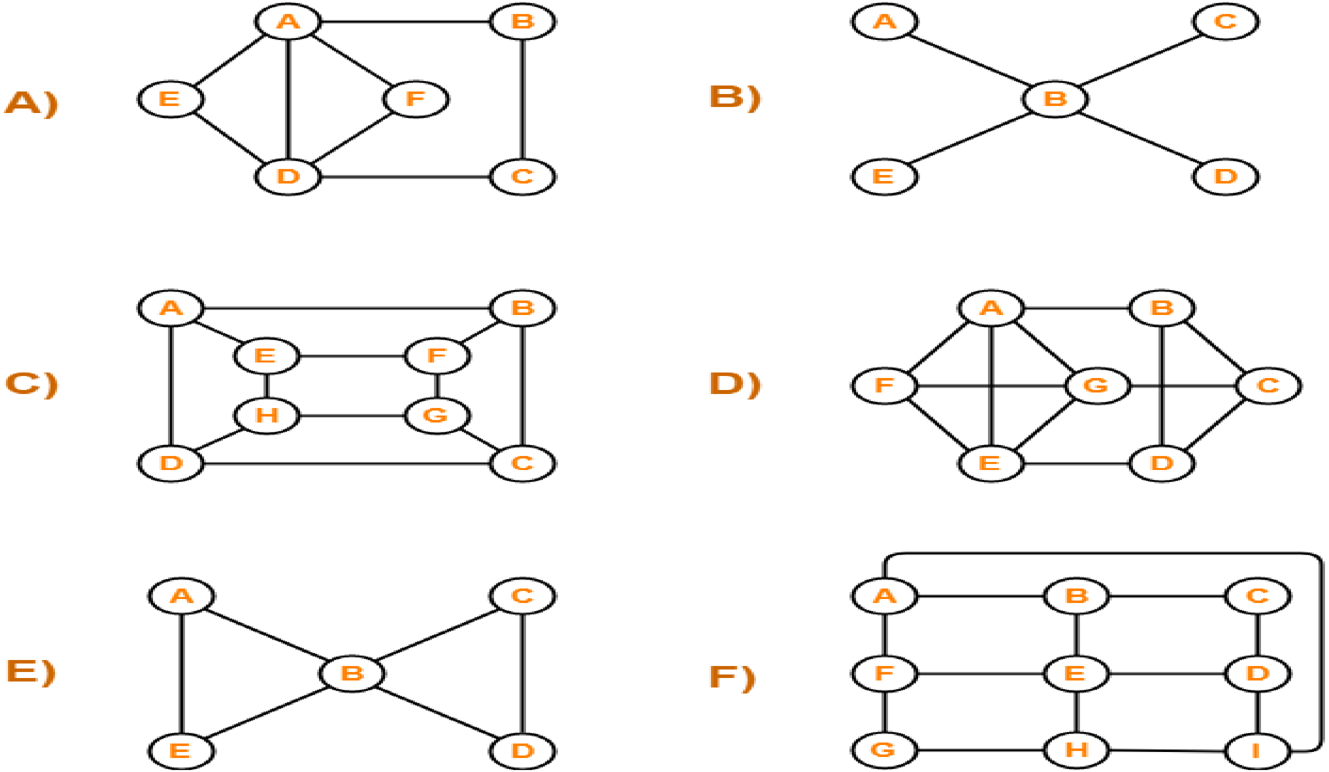
Euler Theorem2 (CO5)

- A connected graph G is has an Euler trail from node *a* to some other node b if and only if G is connected and a ≠ b are the only two nodes of odd degree.
- Assume *G* has an Euler trail *T* from node *a* to node *b* (*a* and *b* not necessarily distinct).
- For every node besides *a* and *b*, *T* uses an edge to exit for each edge it uses to enter. Thus, the degree of the node is even.

1. If $a = b$, then *a* also has even degree. → Euler circuit

2. If $a \neq b$, then *a* and *b* both have odd degree. → Euler path

A and E are Eulerian Graphs.



Topic objective: Hamiltonian Graph(CO5)

- It will be used in various fields such as Computer Graphics, electronic circuit design, mapping genomes, and operations research.
- To plan bus route to pick up students (node->student, road-> edges, bus path-> Hamiltonian path)
- To combine many tiny fragments of genetic code in genome mapping.

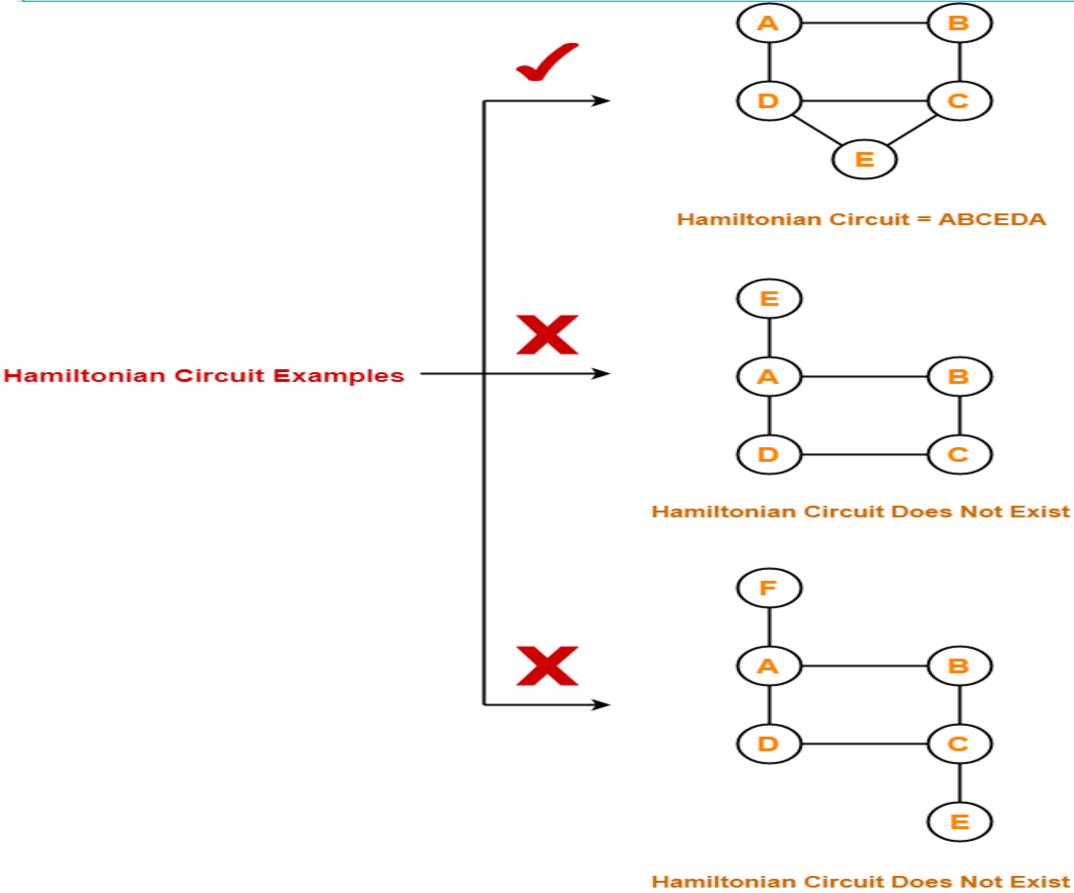
Hamiltonian Graph(CO5)

- **Hamiltonian path** (also called *traceable path*) is a path that visits each vertex exactly once.

- A **Hamiltonian cycle** (also called *Hamiltonian circuit*, *vertex tour* or *graph cycle*) is a cycle that visits each vertex exactly once (except for the starting vertex, which is visited once at the start and once again at the end).

- A graph that contains a Hamiltonian path is called a **traceable graph**. A graph that contains a Hamiltonian cycle is called a **Hamiltonian graph**. Any Hamiltonian cycle can be converted to a Hamiltonian path by removing one of its edges, but a Hamiltonian path can be extended to Hamiltonian cycle only if its endpoints are adjacent.

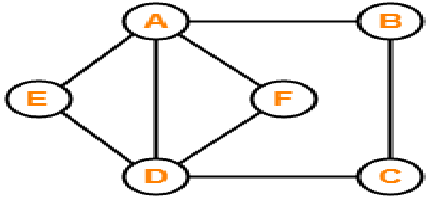
Hamiltonian Circuit(CO5)



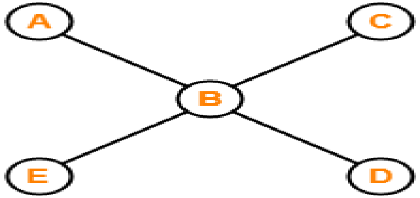
Hamiltonian Graph(CO5)

C, D and F are a Hamiltonian Graphs.

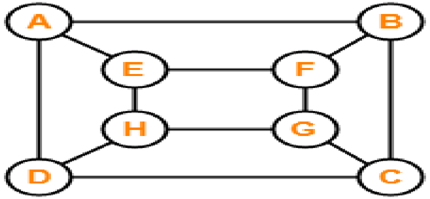
A)



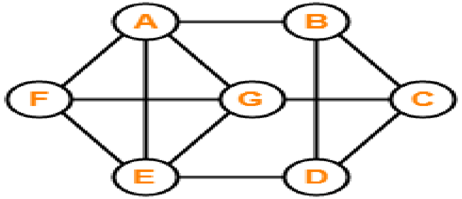
B)



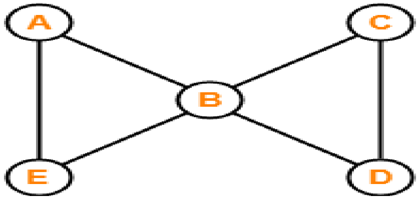
C)



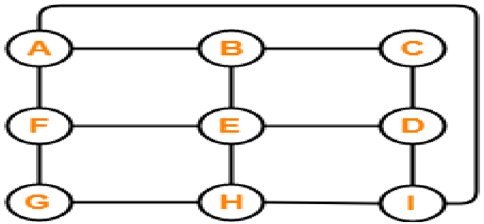
D)



E)



F)



Hamiltonian Graph Theorems(CO5)

- **DIRAC'S Theorem:** if G is a simple graph with n vertices with $n \geq 3$ such that the degree of every vertex in G is at least n/2 then G has a Hamilton circuit.

- **ORE'S Theorem:** if G is a simple graph with n vertices with $n \geq 3$ such that $\deg(u) + \deg(v) \geq n$ for every pair of nonadjacent vertices u and v in G, then G has a Hamilton circuit.

Topic objective: Planar Graphs(CO4)

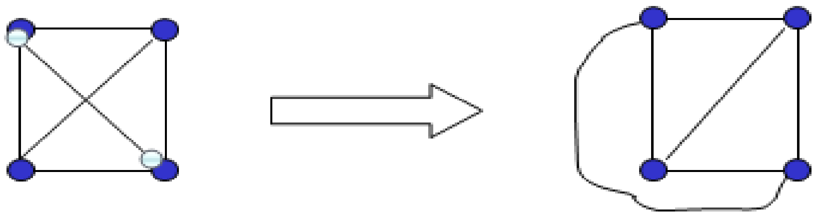
To design or to provide a solution for a **graph** structure in which crossing edges are a nuisance, including design problems for circuits, subways, utility lines.

Planar Graphs(CO5)

A graph (or multigraph) G is called *planar* if G can be drawn in the plane with its edges intersecting only at vertices of G, such a drawing of G is called an *embedding* of G in the plane.

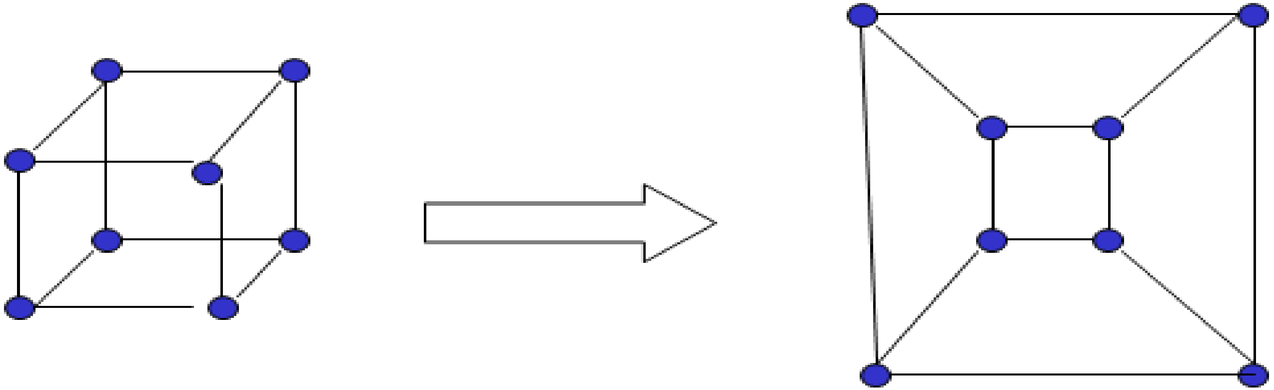
Application Example: VLSI design (overlapping edges requires extra layers), Circuit design (cannot overlap wires on board)

Representation examples: $K1, K2, K3, K4$ are planar, Kn for $n>4$ are non-planar

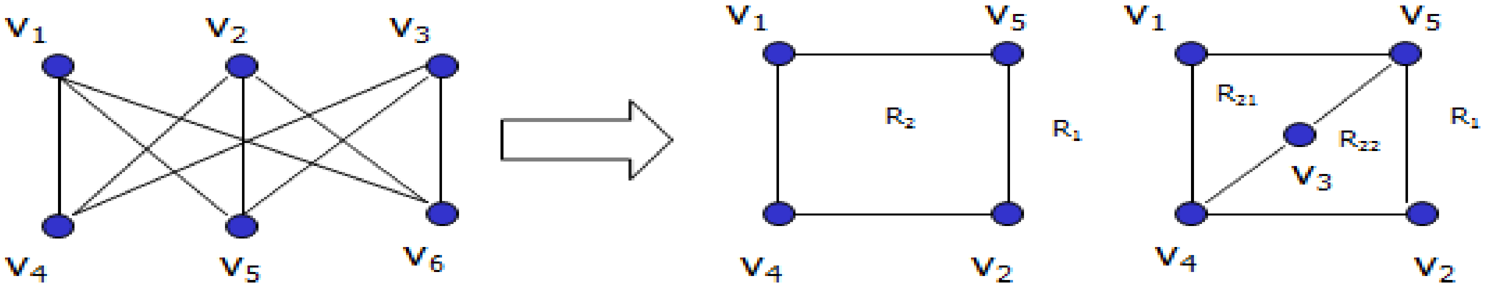


K_4

- Representation example



- Representation examples: $K_{3,3}$ is Non-planar



Planar Graph Theorem(CO5)

Theorem : *Euler's planar graph theorem*

For a **connected** planar graph or multigraph:

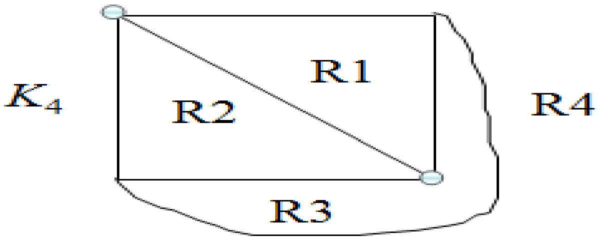
$$v - e + r = 2$$

number of vertices

number of edges

number of regions

Example of Euler's theorem



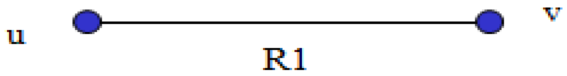
A planar graph divides the plane into several regions (faces), one of them is the infinite region.

$$v=4, e=6, r=4,\ v-e+r=2$$

Planar Graphs(CO5)

- Proof of Euler’s formula: By Induction

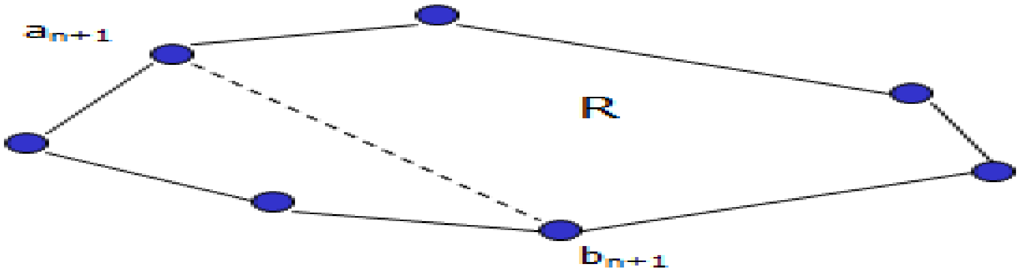
  Base Case: for G1 . $e_1 = 1$. $v_1 = 2$ and $r_1 = 1$



n+1 Case: Assume, $r_n = e_n – v_n + 2$ is true. Let {an+1, bn+1} be the edge that is added to Gn to obtain Gn+1 and we prove that $r_n = e_n – v_n + 2$ is true. Can be proved using two cases.

- <u>Case 1:</u>

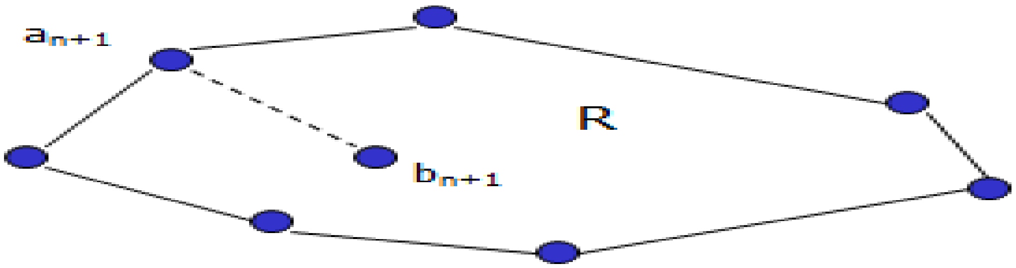
$r_{n+1} = r_n + 1, e_{n+1} = e_n + 1, v_{n+1} = v_n \Rightarrow r_{n+1} = e_{n+1} - v_{n+1} + 2$



Planar Graphs(CO5)

- <u>Case 2:</u>

$$r_{n+1} = r_n,\ e_{n+1} = e_n + 1,\ v_{n+1} = v_n + 1 \Rightarrow r_{n+1} = e_{n+1} - v_{n+1} + 2$$



Planar Graph Corollary1(CO5)

Corollary 1: Let $G = (V, E)$ be a connected simple planar graph with $|V| = v$, $|E| = e > 2$, and r regions. Then $3r \leq 2e$ and $e \leq 3v - 6$

Proof: Since G is loop-free and is not a multigraph, the boundary of each region (including the infinite region) contains at least three edges. Hence, each region has degree ≥ 3.

Degree of region: No. of edges on its boundary; 1 edge may occur twice on boundary -> contributes 2 to the region degree.

Each edge occurs exactly twice: either in the same region or in 2 different regions

Planar Graphs(CO5)

Each edge occurs exactly twice: either in the same region or in 2 different regions

$\Rightarrow$2e = sum of degree of r regions determined by 2e

$\Rightarrow 2e \geq 3r$. (since each region has a degree of at least 3)

$\Rightarrow r \leq (2/3)\, e$

$\Rightarrow$From Euler's theorem, $2 = v - e + r$

$\Rightarrow 2 \leq v - e + 2e/3$

$\Rightarrow 2 \leq v - e/3$

$\Rightarrow$So $6 \leq 3v - e$

$\Rightarrow$or $e \leq 3v - 6$

Planar Graph Corollary2 (CO5)

Corollary 2: Let G = (V, E) be a connected simple planar graph then G has a vertex degree that does not exceed 5

Proof: If G has one or two vertices the result is true

If G has 3 or more vertices then

by Corollary 1, $e \leq 3v - 6$

$\Rightarrow 2e \leq 6v - 12$

If the degree of every vertex were at least 6:by Handshaking theorem: $2e = \text{Sum}(\deg(v))$

$\Rightarrow 2e \geq 6v$.

$\Rightarrow$But this contradicts the inequality $2e \leq 6v - 12$

$\Rightarrow$There must be at least one vertex with degree no greater than 5

Planar Graph Corollary 3(CO5)

Corollary 3: Let G = (V, E) be a connected simple planar graph with v vertices ($v \geq 3$) , e edges, and no circuits of length 3 then $e \leq 2v - 4$

Proof: Similar to Corollary 1 except the fact that no circuits of length 3 imply that degree of region must be at least 4.

Topic objective: Graph coloring(CO5)

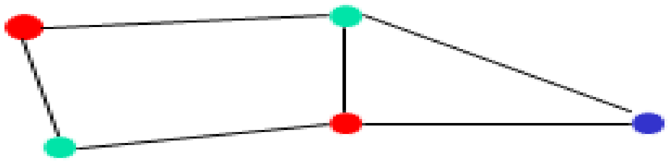
- It will be used for making Schedule or Time Table.
- Register Allocation: assigning a large number of target program variables onto a small number of CPU registers
- Mobile Radio Frequency Assignment: every tower represents a vertex and an edge between two towers represents that they are in range of each other.
- Sudoku: There is an edge between two vertices if they are in same row or same column or same block.
- It will check if a graph is Bipartite or not by coloring the graph using two colors

Graph coloring(CO5)

- **Graph coloring** is an assignment of *"colors"*, almost always taken to be consecutive integers starting from 1 without loss of generality, to certain objects in a graph. Such objects can be vertices, edges, faces, or a mixture of the above.
- **Vertex coloring** is the most common graph coloring problem. The problem is, given m colors, find a way of coloring the vertices of a graph such that no two adjacent vertices are colored using same color.
- **Application examples**: scheduling, register allocation in a microprocessor, frequency assignment in mobile radios, and pattern matching.

Vertex Coloring Problem(CO5)

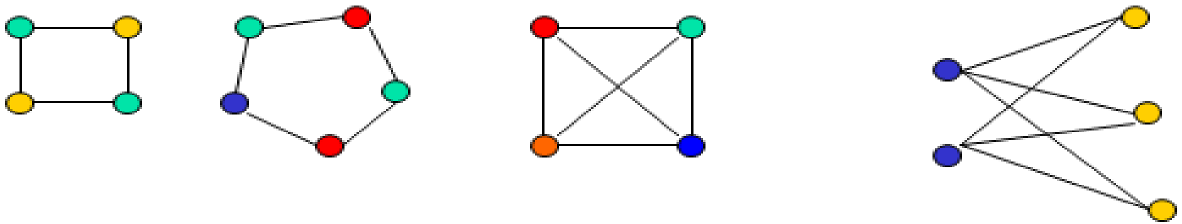
- Assignment of colors to the vertices of the graph such that proper coloring takes place (no two adjacent vertices are assigned the same color)
- **Chromatic number**: least number of colors needed to color the graph
- A graph that can be assigned a (proper) k-coloring is **k-colorable**, and it is **k-chromatic** if its chromatic number is exactly k.



3-chromatic

Vertex Coloring Problem(CO5)

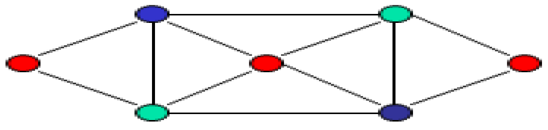
- The problem of finding a minimum coloring of a graph is NP-Hard
- The corresponding decision problem (Is there a coloring which uses at most *k* colors?) is NP-complete
- The chromatic number for C_n = 3 (n is odd) or 2 (n is even), K_n = n, $K_{m,n}$ = 2
- Cn: cycle with n vertices; Kn: fully connected graph with n vertices; Km,n: complete bipartite graph



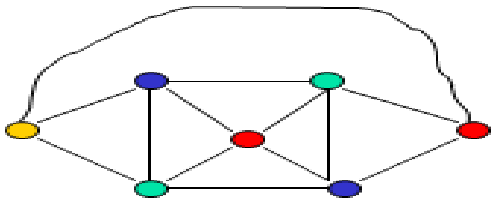
C_4 C_5 K_4 $K_{2,3}$

The Four color theorem(CO5)

- the Four color theorem, or the four color map theorem, states that, given any separation of a plane into contiguous regions, producing a figure called a map, no more than four colors are required to color the regions of the map so that no two adjacent regions have the same color
- The Four color theorem: the chromatic number of a planar graph is no greater than 4
- Example: G1 chromatic number = 3, G2 chromatic number = 4
- (Most proofs rely on case by case analysis).



G1



G2

Breadth first search (BFS) (CO5)

BFS Algorithm

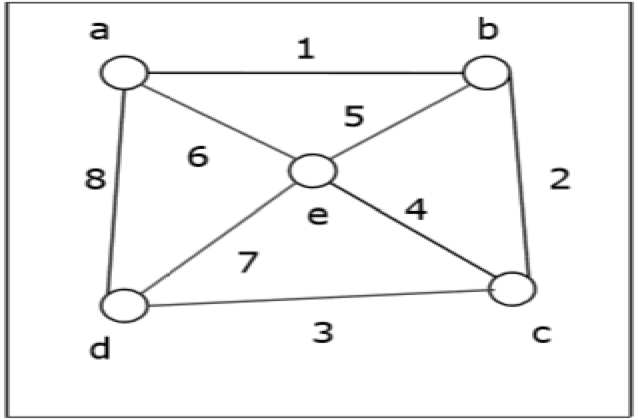
The concept is to visit all the neighbor vertices before visiting other neighbor vertices of neighbor vertices.

Steps:

- Initialize status of all nodes as “Ready”.
- Put source vertex in a queue and change its status to “Waiting”.
- Repeat the following two steps until queue is empty −
- Remove the first vertex from the queue and mark it as “Visited”.
- Add to the rear of queue all neighbors of the removed vertex whose status is “Ready”. Mark their status as “Waiting”.

Breadth first search (BFS) (CO5)

Let us take a graph (Source vertex is ‘a’) and apply the BFS algorithm to find out the traversal order.



- Initialize status of all vertices to “Ready”.
- Put *a* in queue and change its status to “Waiting”.
- Remove *a* from queue, mark it as “Visited”.
- Add *a*’s neighbors in “Ready” state *b*, *d* and *e* to end of queue and mark them as “Waiting”.

NIET Greater Noida

Breadth first search (BFS) (CO5)

- Remove *b* from queue, mark it as “Visited”, put its “Ready” neighbor *c* at end of queue and mark *c* as “Waiting”.
- Remove *d* from queue and mark it as “Visited”. It has no neighbor in “Ready” state.
- Remove *e* from queue and mark it as “Visited”. It has no neighbor in “Ready” state.
- Remove *c* from queue and mark it as “Visited”. It has no neighbor in “Ready” state.
- Queue is empty so stop.

- So the traversal order is $a \rightarrow b \rightarrow d \rightarrow e \rightarrow c$
- Complexity analysis of BFS is O(|V|+|E|).O(|E|)

Application of BFS (CO5)

- Finding the shortest path
- Minimum spanning tree for un-weighted graph
- GPS navigation system
- Detecting cycles in an undirected graph
- Finding all nodes within one connected component

Depth first search (DFS) (CO5)

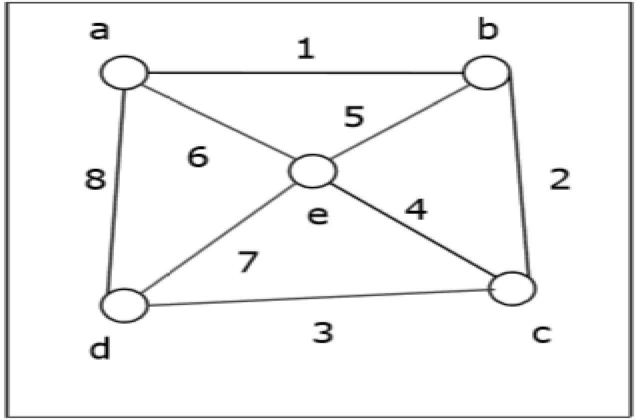
DFS Algorithm

The concept is to visit all the neighbor vertices of a neighbor vertex before visiting the other neighbor vertices.

- Initialize status of all nodes as “Ready”
- Put source vertex in a stack and change its status to “Waiting”
- Repeat the following two steps until stack is empty −
- Pop the top vertex from the stack and mark it as “Visited”
- Push onto the top of the stack all neighbors of the removed vertex whose status is “Ready”. Mark their status as “Waiting”.

Depth first search (DFS) (CO5)

Let us take a graph (Source vertex is 'a') and apply the DFS algorithm to find out the traversal order.



Depth first search (DFS) (CO5)

- Initialize status of all vertices to "Ready".
- Push *a* in stack and change its status to "Waiting".
- Pop *a* and mark it as "Visited".
- Push *a*'s neighbors in "Ready" state *e*, *d* and *b* to top of stack and mark them as "Waiting".
- Pop *b* from stack, mark it as "Visited", push its "Ready" neighbor *c* onto stack.
- Pop *c* from stack and mark it as "Visited". It has no "Ready" neighbor.
- Pop *d* from stack and mark it as "Visited". It has no "Ready" neighbor.
- Pop *e* from stack and mark it as "Visited". It has no "Ready" neighbor.
- Stack is empty. So stop.
- So the traversal order is $a \rightarrow b \rightarrow c \rightarrow d$

- Complexity Analysis of DFS is $\Theta(|V|+|E|)$

Application of DFS (CO5)

- Detecting cycle in a graph
- To find topological sorting
- To test if a graph is bipartite
- Finding connected components
- Finding the bridges of a graph
- Finding bi-connectivity in graphs
- Solving the Knight’s Tour problem
- Solving puzzles with only one solution

Daily Quiz(CO 5)

1. For which of the following combinations of the degrees of vertices would the connected graph be eulerian?
   a) 1,2,3 b) 2,3,4
   c) 2,4,5 d) 1,3,5
2. A graph with all vertices having equal degree is known as a __________
   a) Multi Graph b) Regular Graph
   c) Simple Graph d) Complete Graph
3. Which of the following ways can be used to represent a graph?
   a) Adjacency List and Adjacency Matrix
   b) Incidence Matrix
   c) Adjacency List, Adjacency Matrix as well as Incidence Matrix
   d) No way to represent
4. Consider an undirected random graph of eight vertices. The probability that there is an edge between a pair of vertices is 1/2. What is the expected number of unordered cycles of length three?
   a) 1/8 b)1
   c) 7 d)8

Daily Quiz(CO 5)

5. A graph is a collection of.... ?

a) Row and columns
b) Vertices and edges
c) Equations
d) None of these

6. The degree of any vertex of graph is .... ?

a) The number of edges incident with vertex
b) Number of vertex in a graph
c) Number of vertices adjacent to that vertex
d) Number of edges in a graph

7. A graph with no edges is known as empty graph. Empty graph is also known as... ?

a) Trivial graph
b) Regular graph
c) Bipartite graph
d) None of these

Daily Quiz(CO 5)

8. If the origin and terminus of a walk are same, the walk is known as... ?

a) Open
b) Closed
c) Path
d) None of these

9. Radius of a graph, denoted by rad(G) is defined by.... ?

a) max {e(v): v belongs to V }
b) min { e(v): v belongs to V}
c) max { d(u,v): u belongs to v, u does not equal to v }
d) min { d(u,v): u belongs to v, u does not equal to v }

10. A graph G is called a ..... if it is a connected acyclic graph

a) Cyclic graph
b) Regular graph
c) Tree
d) Not a graph

11. A graph is a collection of

a) Row and columns
b) Vertices and edges
c) Equations
d) None of these

Daily Quiz(CO 5)

12. How many relations are there on a set with n elements that are symmetric and a set with n elements that are reflexive and symmetric ?
a) 2n(n+1)/2 and 2n.3n(n–1)/2
b) 3n(n–1)/2 and 2n(n–1)
c) 2n(n+1)/2 and 3n(n–1)/2
d) 2n(n+1)/2 and 2n(n–1)/2

13. In a graph if e=(u, v) means
a) u is adjacent to v but v is not adjacent to u
b)e begins at u and ends at v
c) u is processor and v is successor
d) both b and c

14. A minimal spanning tree of a graph G is
a) A spanning sub graph
b) A tree
c) Minimum weights
d) All of above

Weekly Assignment(CO 5)

Q1.Explain different type of graph with example.
Q2.Explain different terminology of graph with example
Q3. Define incidence and adjacency matrix of graph with example.
Q4.Explain graph and digraph with example.
Q5. Explain planar graph with example.
Q6.Explain Euler circuit and Euler path.
Q7. what is isomorphic graph.
Q8.Explain chromatic number .
Q9. For the expression (7-(4*5))+(9/3) which of the following is the post order tree traversal?
Q10. Define planar graph. Prove that for any connected planar graph, v – e+ r = 2 Where v, e, r is the number of vertices, edges, and regions of the graph respectively.

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details (CO 5)

Youtube/other Video Links:

- https://www.youtube.com/watch?v=Dsi7x-A89Mw&list=PL0862D1A94725 2D20&index=28
- https://www.youtube.com/watch?v=74l6t4_4pDg&list=PL0862D1A947252D20&index=29
- https://www.youtube.com/watch?v=4d2XEn1j_q4&list=PL0862D1A947252D20&index=30
- https://www.youtube.com/watch?v=qvw1GX93JSY&list=PL0862D1A947252D20&index=32
- https://www.youtube.com/watch?v=ImLM1Vsr35c&list=PL0862D1A947252D20&index=33

MCQs(CO 5)

Q1. Which of the following statements for a simple graph is correct?

a) Every path is a trail

b) Every trail is a path

c) Every trail is a path as well as every path is a trail

d) Path and trail have no relation

Q2. What is the number of edges present in a complete graph having n vertices?

a) (n*(n+1))/2

b) (n*(n-1))/2

c) n

d) Information given is insufficient

Q3. In a simple graph, the number of edges is equal to twice the sum of the degrees of the vertices.

a) True

b) False

Q4. A connected planar graph having 6 vertices, 7 edges contains_____________ regions.

a) 15 b) 3

c) 1 d) 11

Q5. If a simple graph G, contains n vertices and m edges, the number of edges in the Graph G'(Complement of G) is ___________

a) (n*n-n-2*m)/2 b)(n*n+n+2*m)/2

c) (n*n-n-2*m)/2 d) (n*n n+2*m)/2

Q6. Which of the following properties does a simple graph not hold?

a) Must be connected b) Must be unweighted

c) Must have no loops or multiple edges d) Must have no multiple edges

Q7. What is the maximum number of edges in a bipartite graph having 10 vertices?

a) 24 b) 21

c) 25 d) 16

MCQs(CO 5)

Q8 An undirected graph G which is connected and acyclic is called ____________
a) bipartite graph
b) cyclic graph
c) tree
d) forest

Q9. An n-vertex graph has ______ edges.
a) n^2
b) n-1
c) n*n
d) n*(n+1)/2

Q10. The tree elements are called __________
a) vertices
b) nodes
c) points
d) edges

MCQs(CO 5)

Q11. What is a bipartite graph?
a) a graph which contains only one cycle
b) a graph which consists of more than 3 number of vertices
c) a graph which has odd number of vertices and even number of edges
d) a graph which contains no cycles of odd length

Q12. How many cycles are there in a wheel graph of order 5?
a) 6
b) 10
c) 25
d) 7

Q13. The time complexity to find a Eulerian path in a graph of vertex V and edge E is ______________
a) $O(V^2)$
b) $O(V+E-1)$
c) $O(V+E)$
d) $O(E+1)$

MCQs(CO 5)

Q14. In preorder traversal of a binary tree the second step is ____________

a) traverse the right subtree

b) traverse the left subtree

c) traverse right subtree and visit the root

d) visit the root

Q15. What is the minimum height for a binary search tree with 60 nodes?

a) 1

b) 3

c) 4

d) 2

Q16. For the expression (7-(4*5))+(9/3) which of the following is the post order tree traversal?

a) *745-93/+

b) 93/+745*-

c) 745*-93/+

d) 74*+593/-

GLOSSARY QUESTION (CO5)

1. n-vertex graph has edges	1. directed acyclic graph
2. undirected graph G which is connected and acyclic	2. tree
3. polytree is called	3. n-1
4. The tree elements are called	4. nodes
5. an n-ary tree, each vertex has at most children	5. n
6. graph which consists of disjoint union of trees is called	6. 360
7. 7-node directed cyclic graph, the number of Hamiltonian cycle	7. forest
8. G has degree at most 23 then G can have a vertex colouring of	8. bipartite graph
9. the vertex set and the edge set are finite sets	9. 24
10. A Tree is a connected graph	10. acyclic undirected

Q1 Construct a binary tree from the given two Travels (IGDTUW, 2020)

In order 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Post order 1, 4, 3, 2, 6, 8, 9, 11, 10, 7, 5

Q2 Define Hamiltonian graph with suitable example. (Anna Univ, 2017)

Q3 Define planar graph. Prove that for any connected planar graph, v – e+ r = 2 Where v, e, r is the number of vertices, edges, and regions of the graph respectively. (AKTU, 2017)

Q4. What are different ways to represent a graph. Define Euler circuit and Euler graph. Give necessary and sufficient conditions for Euler circuits and paths. (KTU,2017)

Q5. For the expression (7-(4*5))+(9/3) which of the following is the post order tree traversal?

Q6. What are application of a Depth First Search traversal ? (Anna Univ, 2017)

Q7. What is the Worst case complexity of Breadth First Search traversal ? (NIET Autonomous, 2021)

Q8. What are the applications of Breadth First Search traversal? (KTU, 2017)

For more Previous year Question papers:

https://drive.google.com/drive/folders/1xmt08wjuxu71WAmO9Gxj2iDQ0lQf-so1

Expected Questions for Exam(CO 5)

Q1. Define Hamiltonian graph with suitable example

Q2. Define planar graph. Prove that for any connected planar graph, v – e+ r = 2 Where v, e, r is the number of vertices, edges, and regions of the graph respectively.

Q3. What are different ways to represent a graph. Define Euler circuit and Euler graph. Give necessary and sufficient conditions for Euler circuits and paths

Q4.Construct a binary tree from the given two Travels

In order	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Post order	1, 4, 3, 2, 6, 8, 9, 11, 10, 7,

Q5. For the expression (7-(4*5))+(9/3) which of the following is the post order tree traversal?

Q6. What are application of a Depth First Search traversal ?

Q7. What is the Worst case complexity of Breadth First Search traversal ?

Q8. What are the applications of Breadth First Search traversal?

Q9. For the expression (7-(4*5))+(9/3) which of the following is the post order tree traversal?

RECAP OF UNIT (CO5)

- **Tree Terminology**
- A tree is a hierarchical data structure defined as a collection of nodes. Nodes represent value and nodes are connected by edges. A tree has the following properties:
- The tree has one node called root. The tree originates from this, and hence it does not have any parent.
- Each node has one parent only but can have multiple children.
- Each node is connected to its children via edge.
- Binary Tree: In a Binary tree, every node can have at most 2 children, left and right. In diagram below, B & D are left children and C, E & F are right children.

- Balanced Tree: If the height of the left and right subtree at any node differs at most by 1, then the tree is called a balanced tree.
- GRAPH- In discrete mathematics, a graph is **a collection of points, called vertices, and lines between those points, called edges**.

RECAP OF UNIT (CO5)

- The concept of graphs in graph theory stands up on some basic terms such as point, line, vertex, edge, degree of vertices, properties of graphs, etc.
- Graph theory has several application in real world.
- Graph theory used in Computer networks to minimize the cost and time of delivery of data.
- To distinguish between two chemical compounds with the same molecular formula but different structures.
- To solve shortest path problems between cities.
- To schedule exams and assign channels to television stations.

Thank You