Discrete Mathematics (CSA103)

DST-Centre for Interdisciplinary Mathematical Sciences

Institute of Science, Banaras Hindu University

gangaiitr11@gmail.com; gangacims@bhu.ac.in

Mathematical Logic: Propositional and Predicate Logic, Propositional Equivalences, Normal Forms, Predicates and Quantifiers, Nested Quantifiers, Rules of Inference.

Sets and Relations: Set Operations, Representation and Properties of Relations, Equivalence Relations, Partially Ordering. Counting,

Mathematical Induction and Discrete Probability: Basics of Counting, Pigeonhole Principle, Permutations and Combinations, Inclusion- Exclusion Principle, Mathematical Induction, Probability, Bayes Theorem.

Group Theory: Groups, Subgroups, Semi Groups, Product and Quotients of Algebraic Structures, Isomorphism, Homomorphism, Automorphism, Rings, Integral Domains, Fields, Applications of Group Theory.

Graph Theory: Simple Graph, Multigraph, Weighted Graph, Paths and Circuits, Shortest Paths in Weighted Graphs, Eulerian Paths and Circuits, Hamiltonian Paths and Circuits, Planner graph, Graph Coloring, Bipartite Graphs, Trees and Rooted Trees, Prefix Codes, Tree Traversals, Spanning Trees and Cut-Sets.

Boolean Algebra: Boolean Functions and its Representation, Simplifications of Boolean Functions.

Optimization: Linear Programming - Mathematical Model, Graphical Solution, Simplex and Dual Simplex Method, Sensitive Analysis; Integer Programming, Transportation and Assignment Models,

PERT-CPM: Diagram Representation, Critical Path Calculations, Resource Levelling, Cost Consideration in Project Scheduling.

Books Recommended:

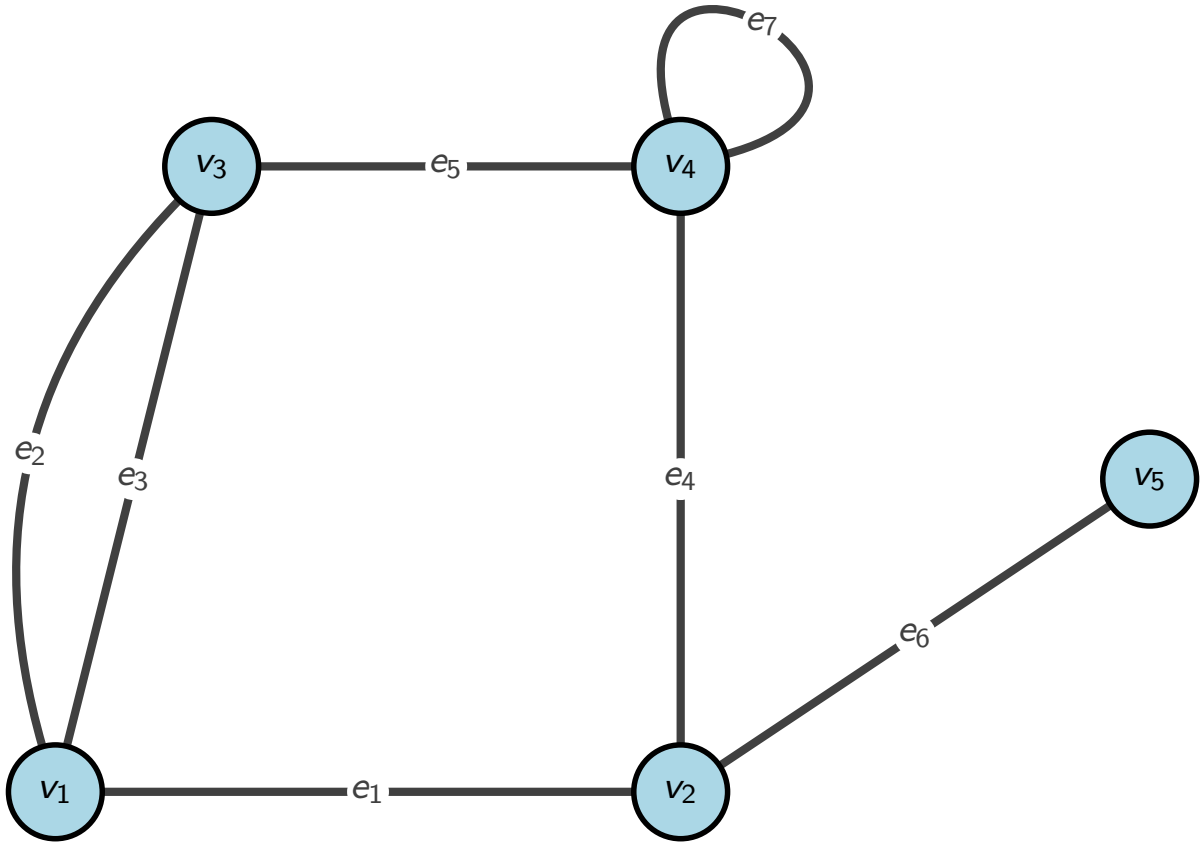
- J.P. Trembley and R.P.Manohar, Discrete Mathematical Structures with Applications to Computer Science, McGraw Hill.
- Dornhoff and Hohn, Applied Modern Algebra, McMillan.
- N. Deo, Graph Theory with Applications to Engineering and Computer Science, PHI.
- C.L. Liu, Elements of Discrete Mathematics, McGraw-Hill.
- Rosen, Discrete Mathematics, Tata McGraw Hill.
- K.L.P. Mishra, N. Chandrasekaran, Theory of Computer Science: Automata, Languages and Computation, PHI.

Graph: A algebraic Structures or collection $G = (V, E, \Phi)$ of sets V and E and a relation Φ,

- V is called the set of **vertices**
- E is called the collection of **edges**
- Φ is a **relation** means each edge has either one or two vertices associated with ordered or un-ordered pair of vertices

and these vertices are called **end points** is called a **graph.**

The number of vertices in a graph G is called **order** of a graph and notation $|V(G)| = n$. The number of edges in a graph G is called **size** of a graph and notation $|E(G)| = m$.


e_7
v_3
e_5
v_4
e_2
e_3
e_4
v_5
e_6
v_1
e_1
v_2

Types of Graphs: We have following types

Finite and infinite Graph:
Directed and undirected Graph:
Underlaying Graph:
Mixed Graph:
Simple Graph:
Trivial and Non-trivial Graph:
Null/Empty Graph:
General/Pseudo Graph:
Multi Graph:
Weighted Graph:

Edge Incident: Two edges are "incident" if they share a common vertex.

Vertex Incident: A vertex is incident with an edge if the vertex is one of the endpoints of that edge.

Adjacent Vertex/Simple Adjacency: If they are joined by exactly one edge.

Neighbor: Two adjacent Vertices are referred to as neighbors of each others.

Neighborhood: The set of neighbors of a vertex v. Notation $N_G(v)$ or $\phi N(v)$

Closed Neighborhood: The set $N[v] = N_G(v) \cup \{v\}$ or $\phi N(v) \cup \{v\}$

Adjacent Edges: Those two edges that have an end point in common.

Proper Edge: It is an edge that joint/connect two distinct vertices.

Multiple Edges/Parallel Edges: It is a collection of two or more edges having identical ends points.

Multiplicity: It is the number of edges between a pair of vertices say u and v.

Self Loop: It is an edge that joint/connect a single end point to itself.

Applications/Puzzles Models on Graph: These are followings

1. Königsberg Bridge Problem: The East Prussian city of Königsberg (now called Kaliningrad and located in Russia) occupied both banks of the River Pregel and the island of Kneiphof, lying in the river at a point where the river branches into two parts. There were seven bridges that spanned various sections of the river. This problem, asks whether there is a route that crosses each of these bridges exactly once.

Graphical Formulation: Vertices represent the land areas and the edge represent the bridge.

Solution of Problem: As Euler showed the answer to this question is no.

Examples: Is it possible to take a route about Königsberg that crossed each bridge exactly twice?

2. Utilities Problem:
Graphical Formulation:
Solution of Problem:
Examples:

3. Electrical Network Problem:
Graphical Formulation:
Solution of Problem:
Examples:

4. Seating Problem:
Graphical Formulation:
Solution of Problem:
Examples:

5. A puzzle with multicolored cubes/Instant Insanity Puzzle: We are given four cubes. The six faces of every cube are variously colored blue (B), green (G), red (R), or white (W). Is it possible to stack the cubes one on top of another to form a column such that no color appears twice on any of the four sides of this column?

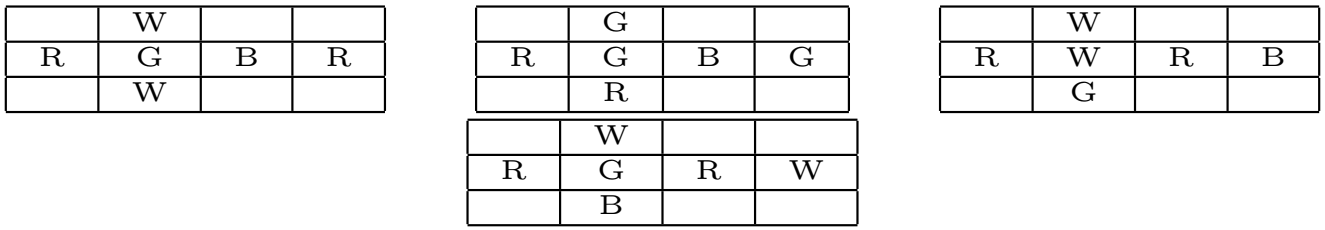
	W		
R	G	B	R
	W		

	G		
R	G	B	G
	R		

	W		
R	G	R	W
	B		

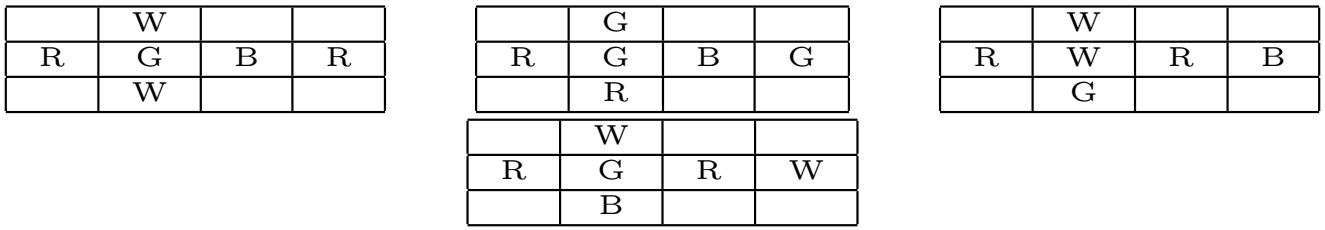
	W		
R	W	R	B
	G		

5. A puzzle with multicolored cubes/Instant Insanity Puzzle: We are given four cubes. The six faces of every cube are variously colored blue (B), green (G), red (R), or white (W). Is it possible to stack the cubes one on top of another to form a column such that no color appears twice on any of the four sides of this column?



Formulation: Draw a graph with fur vertices B, G, R and W as one for each color. Consider one cube and represent its three pairs of opposite faces by three edges, drawn between the vertices with appropriate colors.

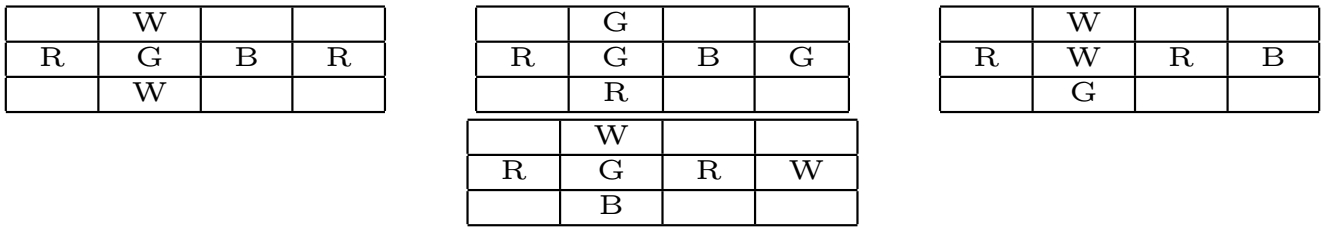
5. A puzzle with multicolored cubes/Instant Insanity Puzzle: We are given four cubes. The six faces of every cube are variously colored blue (B), green (G), red (R), or white (W). Is it possible to stack the cubes one on top of another to form a column such that no color appears twice on any of the four sides of this column?



Formulation: Draw a graph with fur vertices B, G, R and W as one for each color. Consider one cube and represent its three pairs of opposite faces by three edges, drawn between the vertices with appropriate colors.

Solution: If there exist two edge-disjoint subgraph, each with four edges, each of the edge labeled differently, and such that each vertex is of degree two.

5. A puzzle with multicolored cubes/Instant Insanity Puzzle: We are given four cubes. The six faces of every cube are variously colored blue (B), green (G), red (R), or white (W). Is it possible to stack the cubes one on top of another to form a column such that no color appears twice on any of the four sides of this column?



Formulation: Draw a graph with fur vertices B, G, R and W as one for each color. Consider one cube and represent its three pairs of opposite faces by three edges, drawn between the vertices with appropriate colors.

Solution: If there exist two edge-disjoint subgraph, each with four edges, each of the edge labeled differently, and such that each vertex is of degree two.

Example: Each side shows only one color.

Some Graph Models/Problems: Following are some models

Some Graph Models/Problems: Following are some models

Ecology Problems: Models involving the interaction of different species of animals.

Some Graph Models/Problems: Following are some models

Ecology Problems: Models involving the interaction of different species of animals.

Acquaintanceship: To represent various relationship between people.

Some Graph Models/Problems: Following are some models

Ecology Problems: Models involving the interaction of different species of animals.

Acquaintanceship: To represent various relationship between people.

Influence: In studies of group behavior it is observed that certain people can influence the thinking of others.

Some Graph Models/Problems: Following are some models

Ecology Problems: Models involving the interaction of different species of animals.

Acquaintanceship: To represent various relationship between people.

Influence: In studies of group behavior it is observed that certain people can influence the thinking of others.

Hollywood: When the actors have worked together on a movie.

Some Graph Models/Problems: Following are some models

Ecology Problems: Models involving the interaction of different species of animals.

Acquaintanceship: To represent various relationship between people.

Influence: In studies of group behavior it is observed that certain people can influence the thinking of others.

Hollywood: When the actors have worked together on a movie.

Round-Robin Tournament: A tournament where each team plays each other team exactly once.

Collaboration : These can be used to model joint authorship of academic papers.

Collaboration : These can be used to model joint authorship of academic papers.

Call/Road Map Network: These can be used to model telephone call/roadmaps made in network.

Collaboration : These can be used to model joint authorship of academic papers.

Call/Road Map Network: These can be used to model telephone call/roadmaps made in network.

Web: The World Wide Web can be modeled as a graph with each Web page is a vertex and edge is a link.

Collaboration : These can be used to model joint authorship of academic papers.

Call/Road Map Network: These can be used to model telephone call/roadmaps made in network.

Web: The World Wide Web can be modeled as a graph with each Web page is a vertex and edge is a link.

Precedence and concurrent Processing: Computer programs can be executed more rapidly by executing certain statements concurrently. The dependence of statements on previous statements can be represented by a graph.

New graph from old graph: How to construct a graph?

1. Subgraph Method
    1. Subgraph
    2. Proper Subgraph
    3. Spanning Subgraph
    4. Edge-Disjoint Subgraphs
    5. Vertex-Disjoint Subgraphs
    6. Induced Subgraph
2. Operational Method
    1. Primary/Uniry
    2. Second/Binary

- Primary/Uniry
  - Adding vertices: $G \cup \{v\}$

- Primary/Uniry
  - Adding vertices: $G \cup \{v\}$
  - Deleting vertices: $G - \{v\}$

- Primary/Uniry
  - Adding vertices: $G \cup \{v\}$
  - Deleting vertices: $G - \{v\}$
  - Adding edges: $G \cup \{e\}$

- Primary/Uniry
  - Adding vertices: $G \cup \{v\}$
  - Deleting vertices: $G - \{v\}$
  - Adding edges: $G \cup \{e\}$
  - Deleting edges: $G - \{e\}$

- Primary/Uniry
  - Adding vertices: $G \cup \{v\}$
  - Deleting vertices: $G - \{v\}$
  - Adding edges: $G \cup \{e\}$
  - Deleting edges: $G - \{e\}$
  - Fusion: $G - (a, b)$

- Second/Binary

- Second/Binary
  - Union: $G = G_1 \cup G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$

- Second/Binary
  - Union: $G = G_1 \cup G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$
  - Intersection: $G = G_1 \cap G_2 \Rightarrow V(G) = V(G_1) \cap V(G_2)$ and $E(G) = E(G_1) \cap E(G_2)$

- Second/Binary
  - Union: $G = G_1 \cup G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$
  - Intersection: $G = G_1 \cap G_2 \Rightarrow V(G) = V(G_1) \cap V(G_2)$ and $E(G) = E(G_1) \cap E(G_2)$
  - Ring-sum: $G = G_1 \oplus G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) - E(G_1) \cap E(G_2)$

- Second/Binary
  - Union: $G = G_1 \cup G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$
  - Intersection: $G = G_1 \cap G_2 \Rightarrow V(G) = V(G_1) \cap V(G_2)$ and $E(G) = E(G_1) \cap E(G_2)$
  - Ring-sum: $G = G_1 \oplus G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) - E(G_1) \cap E(G_2)$
  - Edge-complement: $\bar{G} = G - E(G) \Rightarrow V(\bar{G}) = V(G)$ and If (u, v) is an edge of $\bar{G}$ iff it is not an edge of G.

- Second/Binary
  - Union: $G = G_1 \cup G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$
  - Intersection: $G = G_1 \cap G_2 \Rightarrow V(G) = V(G_1) \cap V(G_2)$ and $E(G) = E(G_1) \cap E(G_2)$
  - Ring-sum: $G = G_1 \oplus G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) - E(G_1) \cap E(G_2)$
  - Edge-complement: $\bar{G} = G - E(G) \Rightarrow V(\bar{G}) = V(G)$ and If (u, v) is an edge of $\bar{G}$ iff it is not an edge of G.
  - Complement of a subgraph: If $g \subset G$, then $\bar{g} = G - g = G \oplus g$

- Second/Binary
  - Union: $G = G_1 \cup G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$
  - Intersection: $G = G_1 \cap G_2 \Rightarrow V(G) = V(G_1) \cap V(G_2)$ and $E(G) = E(G_1) \cap E(G_2)$
  - Ring-sum: $G = G_1 \oplus G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) - E(G_1) \cap E(G_2)$
  - Edge-complement: $\bar{G} = G - E(G) \Rightarrow V(\bar{G}) = V(G)$ and If (u, v) is an edge of $\bar{G}$ iff it is not an edge of G.
  - Complement of a subgraph: If $g \subset G$, then $\bar{g} = G - g = G \oplus g$
  - Decomposition: If $g_1 \cup g_2 = G$ and $g_1 \cap g_2 =$ null graph.

- Second/Binary
  - Union: $G = G_1 \cup G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$
  - Intersection: $G = G_1 \cap G_2 \Rightarrow V(G) = V(G_1) \cap V(G_2)$ and $E(G) = E(G_1) \cap E(G_2)$
  - Ring-sum: $G = G_1 \oplus G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) - E(G_1) \cap E(G_2)$
  - Edge-complement: $\bar{G} = G - E(G) \Rightarrow V(\bar{G}) = V(G)$ and If (u, v) is an edge of $\bar{G}$ iff it is not an edge of G.
  - Complement of a subgraph: If $g \subset G$, then $\bar{g} = G - g = G \oplus g$
  - Decomposition: If $g_1 \cup g_2 = G$ and $g_1 \cap g_2 =$ null graph.
  - Join/Suspension: $G = G_1 + G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) \cup \{(u, v) : u \in V(G_1), v \in V(G_2)\}$

- Second/Binary
  - Union: $G = G_1 \cup G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$
  - Intersection: $G = G_1 \cap G_2 \Rightarrow V(G) = V(G_1) \cap V(G_2)$ and $E(G) = E(G_1) \cap E(G_2)$
  - Ring-sum: $G = G_1 \oplus G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) - E(G_1) \cap E(G_2)$
  - Edge-complement: $\bar{G} = G - E(G) \Rightarrow V(\bar{G}) = V(G)$ and If (u, v) is an edge of $\bar{G}$ iff it is not an edge of G.
  - Complement of a subgraph: If $g \subset G$, then $\bar{g} = G - g = G \oplus g$
  - Decomposition: If $g_1 \cup g_2 = G$ and $g_1 \cap g_2 =$ null graph.
  - Join/Suspension: $G = G_1 + G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) \cup \{(u, v) : u \in V(G_1), v \in V(G_2)\}$
  - Cartesian Product: $G = G_1 \times G_2 \Rightarrow V(G) = V(G_1) \times V(G_2)$ and $E(G) = E(G_1) \times V(G_2) \cup V(G_1) \times E(G_2)$ where two distinct vertices (u, v) and (x, y) of G are adjacent if either

$$(1) u = x \ \textit{and} \ vy \in E(G_2) \ \textit{or} \ (2) v = y \ \textit{and} \ ux \in E(G_1).$$

- Second/Binary
  - Union: $G = G_1 \cup G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$
  - Intersection: $G = G_1 \cap G_2 \Rightarrow V(G) = V(G_1) \cap V(G_2)$ and $E(G) = E(G_1) \cap E(G_2)$
  - Ring-sum: $G = G_1 \oplus G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) - E(G_1) \cap E(G_2)$
  - Edge-complement: $\bar{G} = G - E(G) \Rightarrow V(\bar{G}) = V(G)$ and If (u, v) is an edge of $\bar{G}$ iff it is not an edge of G.
  - Complement of a subgraph: If $g \subset G$, then $\bar{g} = G - g = G \oplus g$
  - Decomposition: If $g_1 \cup g_2 = G$ and $g_1 \cap g_2 =$ null graph.
  - Join/Suspension: $G = G_1 + G_2 \Rightarrow V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) \cup \{(u, v) : u \in V(G_1), v \in V(G_2)\}$
  - Cartesian Product: $G = G_1 \times G_2 \Rightarrow V(G) = V(G_1) \times V(G_2)$ and $E(G) = E(G_1) \times V(G_2) \cup V(G_1) \times E(G_2)$ where two distinct vertices (u, v) and (x, y) of G are adjacent if either

$$(1) u = x \ \textit{and} \ vy \in E(G_2) \ \textit{or} \ (2) v = y \ \textit{and} \ ux \in E(G_1).$$

  - **Assignment:** m-fold self Join/Product-

Degree of vertex:-

Degree of a vertex: The degree of v is the number of edges incident with v. If v is a vertex of a simple graph G of order n, then

$$0 \leq \delta(G) \leq deg(v) \leq \Delta(G) \leq n-1.$$

Average degree of a graph G: The average degree of a graph G of order n

$$\frac{\text{Sum of degree of G}}{n}$$

Degree Sequence: The list of the degrees of the graph's vertices in non-increasing order

Graphic: A sequence $< d_1, (1), d_n >$ is called graphic if it is the degree sequence of a simple graph.

Havel–Hakimi Theorem: A sequence $s : d_1, d_2, ..., d_n$ of nonnegative integers with $\Delta = d_1 \geq d_2 \geq \ \geq d_n$ and $\Delta \geq 1$ is graphical if and only if the sequence

$$s_1 : d_2 - 1, d_3 - 1, ..., d_{\Delta+1} - 1, d_{\Delta+1}, ..., d_n$$

is graphical.

Series Edges: If their comman vertex is of degree two.

Even and odd Vertices: A vertex in a graph G is even or odd, according to whether its degree in G is even or odd.

Isolated Vertex: A vertex of degree 0.

Pendant Vertex/leaf/end or edge: A vertex of degree 1 is an end-vertex or a leaf. An edge incident with an end-vertex is called a pendant edge.

First Theorem of Graph Theory/ Handshaking Lemma: If G is a graph of size m, then

$$\sum_{v \in V(G)} deg(v) = 2m$$

Proof: Hint When summing the degrees of the vertices of G, each edge of G is counted twice, once for each of its two incident vertices.

Theorem: Every graph has an even number of odd vertices. or The number of vertices of odd degree in a graph G is always even.

Proof: Hint Since the sum of the degrees of the even vertices of G is even, the sum of the degrees of the odd vertices of G must be even as well, implying that G has an even number of odd vertices.

Assignment:- (i) Maximum number of edges in simple graph with order n
(ii) Minimum number of edges in graph with order n

Some Special Graphs:- We have following graphs

Dipole: It is a loopless graph with two vertices and n edges joining them.
Bauquet: It is a graph with one vertex and n self-loop.
Dumbbell: It is a graph with two vertices, one edge joining them and two self-loop at end point of edge.
Complete: A simple graph in which every two distinct vertices are adjacent.
Regular: If the vertices of G have the same degree and is regular of degree r if this degree is r. Such graphs are also called r-regular.

Cubic: A 3-regular graph.
Petersen: It is special cubic graphs.
Cycle: A connected 2-regular. C_n
Path: $C_n - \{e\}$
Wheel: It is obtained from C_{n-1}, $n \geq 4$ by adding a vertex v called "Hub" inside C_{n-1}, and connecting it to every vertex in C_{n-1}.
Notation W_n
Hypercube/Cube: It is the graph that has vertices representing the 2^n bit strings of length n such that two vertices are adjacent if and only if the bit strings that they represent differ in exactly one bit position. $Q_n = N - cube = Q_{n-1} \times P_2$

Bipartite: If its vertex set V can be partitioned into two disjoint non-empty sets V_1 and V_2 such that $V_1 \cup V_2 = V$ and every edge in E has one end in V_1 and the other end in V_2.
Complete Bipartite: A bipartite graph G with the bipartition V_1 and V_2 is called complete bipartite if every vertex in V_1 is adjacent to every vertex in V_2. $K_{m,n}$
Assignment:
k-partite:
Complete k-partite Graphs/Complete Multipartite Graphs:

Star: Any graph that is $K_{1,n}$ or S_{n+1} and claw $(K_{1,3})$
Platonic Graphs: The graph formed by the vertices and edges of the five regular (platonic) solids -Tetrahedron, Octahedron, Cube, Icosahedron, Decahedron.
Book: A book graph is defined as $B_m = S_{m+1} \times P_2$
Irregular Graph: If $deg(u) \neq deg(v)$ for every two vertices u and v of G.

Some applications of special graphs: These are listed below.

Job assignment ($K_{m,n}$)

Local Area Network ($S_{n+1}(K_{1,n}), C_n, W_n$)

Inter connection Network (P_{n+1}, Q_n)

Dynamical system ($S_{n+1}(K_{1,n})$)

Connect-ness of graph

Walk/trail: A finite alternating sequence of vertices and edges, beginning and ending with vertices such that each edge is incident with vertices preceding and following it.

- No edge appear more then once vertex may be repeat.
- Vertices with which a walk begins and ends are called its terminal vertices.
- Number of edges in the walk is called **length of walk:**

Closed walk/Circuit/polygon/cycle: If the terminal vertices of walk is same then the walk is called closed walk. e.g. triangle, even and odd circuit,

Open walk: If terminal vertices are different then walk are called open walk.

Path: An open walk in which no vertices appear more then once.

Girth and Circumference: The length of a smallest and longest cycle in a graph G. Denoted as $g(G)$ and $c(G)$

Connected graph: If there is a walk/path between every pair of vertices.

Disconnected graph: If there is no walk/path between any pair of vertices.

Component: A disconnected graph consists two or more connected subgraphs called components.

Theorem: A simple graph with order n and k components can have at most $\frac{(n-k)(n-k+1)}{2}$ edges.

Proof: Hint: $n_1 + n_2 + \cdots + n_k = n, \quad n_{\geq} 1$

$$[\sum_{i=1}^{k}(n_i - 1)]^2 = (n-k)^2$$

$$\sum_{i=1}^{k}(n_i^2 - 2n_i) + k + \text{cross terms} = (n-k)^2.$$

Maximum number of edges in i^{th} component $\frac{(n_i - 1)n_i}{2}$.

Maximum number of edges in G

$$\sum_{i=1}^{k}\frac{(n_i - 1)n_i}{2}$$

Simplifying, get the result.

Euler Line/Circuit: A closed walk which contains all the edges of the graph.

Euler Graph: A graph which have Euler line.

Open-Euler Line/Unicursal Line: A open walk which contains all the edges of the graph.

Unicursal Graph: A graph which have open Euler line.

Euler First Theorem on Graph: A given connected graph G is an Euler Graph if and only if all vertices of G are of even degree.

Proof: Hint: Suppose that G is an Euler graph.

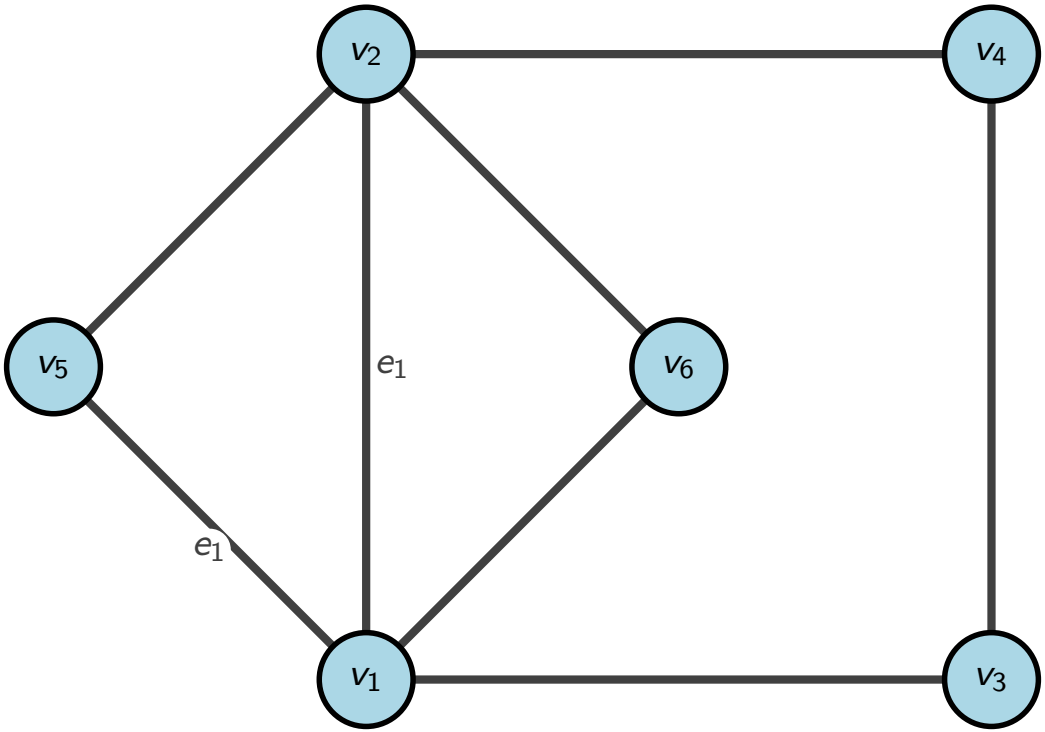
- In tracing this walk we observe that every time the walk meets a vertex v it goes through two new edges incident on vwith one we entered v and with the other exited.
- This is true not only of all intermediate vertices of the walk but also of the terminal vertex, because we exited and entered the same vertex at the beginning and end of the walk, respectively.

Conversely,

- Construct a walk starting at an arbitrary vertex v and going through the edges of G such that no edge is traced more than once.
- We can exit from every vertex we enter; the tracing cannot stop at any vertex but say v.
- If this closed walk say h we just traced includes all the edges of G.
- If not, we remove from G all the edges in h and obtain a subgraph h' of G formed by the remaining edges.
- Subgraph h' must touch h at least at one vertex say a,
- Starting from a, we can again construct a new walk in graph h'.
- This new walk in h' must terminate at vertex a.
- This new walk in h' can be combined with h to form a new walk, which starts and ends at vertex v.
- This process can be repeated until we obtain a closed walk that traverses all the edges of G.

Fleury Algorithm: An algorithm for finding the Euler Line in given Euler graph is called Fleury's algorithm.

Algorithm:



Chinese Postman Problem: A postman start from his post office in a town to deliver his letters and returns to his starting point every day. What routs should he take so that he visits each road at least once and the total distance covered is least.

Formulation: Vertices represent the cities and the edge represent the routs.

Solution: A closed rout (finite sequence of vertices and edges) which contains all the edges of the graph at-least once and the total weight of the walk is minimum.

Example:

Hamiltonian Circuit A closed walk that passes through each vertex exactly once is called a Hamiltonian Circuit.

Hamiltonian Graph A graph which have Hamiltonian Circuit.

Hamiltonian Path: A path that passes through each vertex exactly once is called a Hamiltonian Path.

Ore's Theorem: Let G be a simple graph of order $n \geq 3$. If

$$\deg u + \deg v = \sigma(G) \geq n$$

where $\sigma(G)$ denotes the minimum degree sum of two non-adjacent vertices of G, then G is a Hamiltonian Graph.

Verify :

Dirac's Theorem: If G be a simple graph of order $n \geq 3$, such that the degree of every vertex in G is at least $\frac{n}{2}$, then G is a Hamiltonian Graph.

Verify : Assignment

Travelling Salesman Problem: A salesman is required to visit a number of cities. What is the route he should take if he has to start at his home city, visit each city exactly once and then return home travelling the minimum distance.

Formulation: Vertices represent the cities and the edge represent the routs.

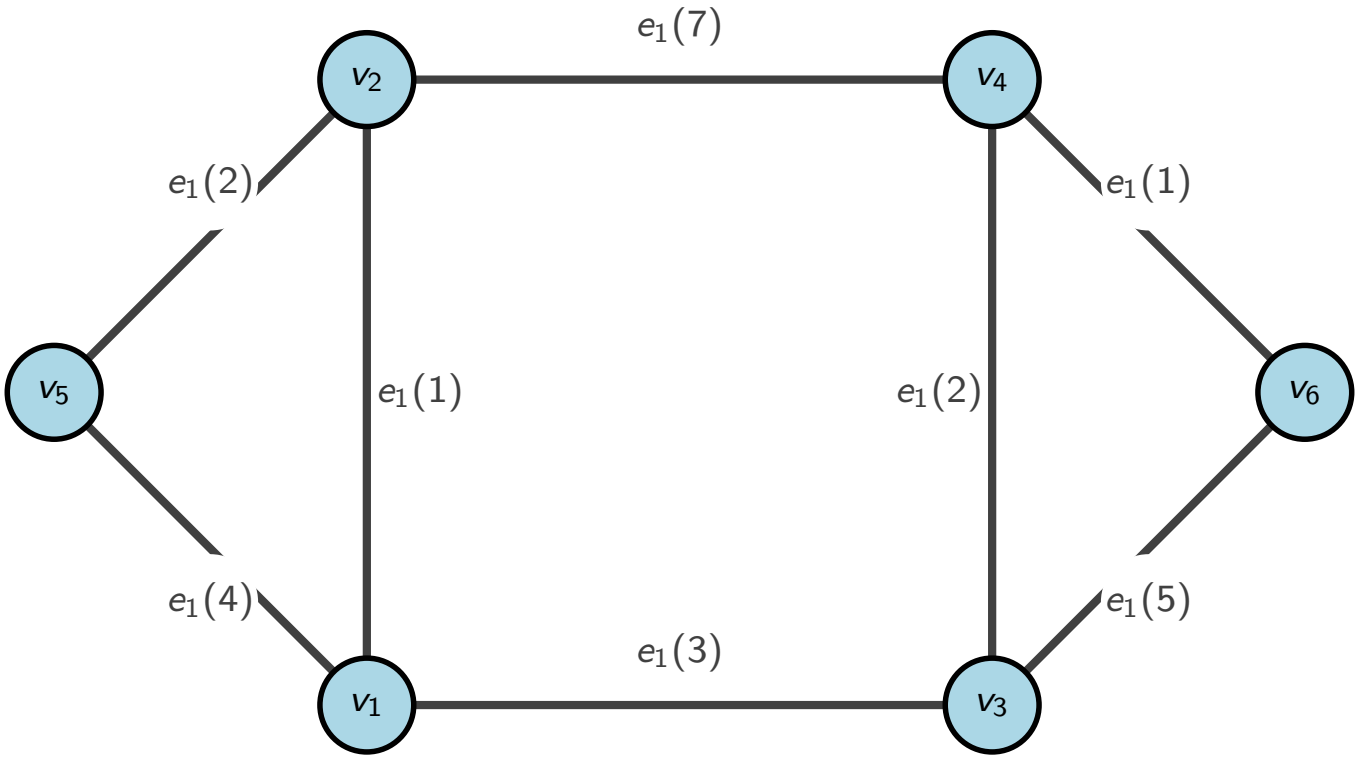
Solution: He has to find a closed walk that passes through each vertex exactly once and the total weight of the walk is minimum.

Example:

Shortest Path: A path of least length between two given vertices in weighted graph.

Dijkstra's Algorithm: An algorithm for finding the shortest path between two given vertices in weighted graph is called Dijkstra's algorithm.
Relaxation: $d(u) + c(u, v) < d(v)$ then $d(v) = d(u) + c(u, v)$.

Ans.-9



Trees in Graph

Trees: A connected graph without circuit.

Tree of a graph: A subgraph which is a tree

Leaf/Pandent Vertex: If $deg(v) = 1$.

Tree edge and non-tree edge: If it is the edge of a tree.

Frontier Edge: A non-tree edge with one end point in tree T and one endpoint not in tree T.

Minimally connected graph: If removal of any one edge from G disconnects the graph.

Some special tree

Path: A tree order $n \geq 2$ containing exactly two vertices of degree one and other vertices of degree two (if exist).

Star: A tree order $n \geq 2$ containing exactly one vertex which is not a leaf.

Double star: A tree containing exactly two vertices that are not leaves.

Caterpillar: A tree T of order 3 or more, the removal of whose leaves produces a path.

Spine of Tree: A path obtained from removal of leaves in a tree.

Applications/Models of tree

Saturated Hydrocarbons: To enumerate the isomers of compound of molecules C_nH_{2n+2}

Representing Organizations: The structure of a large organization can be modeled using a tree.

Computer Files Systems: Files in computer memory can be organized into directories.

Connected Parallel Processor: Tree described several interconnection network for parallel processing.

Terminology of genealogical Origins: The genealogy of a family is often represented by means of a family tree.

Representing/Organizing Data: Trees appear in representing or organizing the data

- Representing tributaries and sub-tributaries of a rivers.
- The sorting of mail/punched cards (Decision tree or Sorting tree)
- Finding the largest monotonically increasing subsequence in sequence.

Theorem: A graph G is a tree if and only if there is one and only one path between every pair of vertices.

Proof: Graph G is a tree T

- Connected graph $\Leftrightarrow$ there exist least one path
- **Suppose** that between two vertices u and v of T there are two distinct paths.
- The union of these two paths will contain a circuit.

Conversely: There is a unique path in G

- G is connected.
- **Suppose** G has circuit.
- There are two distinct paths between one pair of vertices.

Theorem: A tree with n vertices has $n-1$ edges.

Proof: Mathematical induction

- It is easy to see that the theorem is true for $n = 1, 2$, and 3.
- **Assume** that the theorem holds for all trees with fewer than n vertices.
- Consider a tree T with n vertices e_k be an edge with end vertices v_i and v_j.
- There is no other path between v_i and v_j except e_k.
- $T - e_k$ consists of exactly two components, say T_1 and T_2 having vertices fewer than n vertices each.
- Each component contains one less edge than the number of vertices in it.
- Thus $T - e_k$ consists of $n-2$ edges.

Theorem: A graph is a tree if and only if it is minimally connected.

Proof: G is tree

- Connected and circuitless.
- Remove edge e_k from G
- $G - e_k$ disconnected.

Conversely

- **Assume** that G is not minimally connected.
- $G - e_k$ is connected for any e_k.
- That is $e_k \in Circuit$.
- G is not a tree.

Rooted Tree

Root: A particular one vertex of a tree which is distinguished from all the other vertices and every edge is reached from that particular vertex to other vertices is called a root in tree.
Rooted tree: A tree with root is called a rooted tree.
Terminology used for rooted tree

- **Parent:** The parent of v (v is not root) is a unique vertex u such that there is a direct edge from u to v and
- **Child:** v is called child of u.
- **Siblings:** Vertices with the same parent.
- **Ancestors:** The ancestors of v (v is not root) are the vertices in the path from the root to this vertex.
- **Descendant:** Vertices that have v as an ancestor.
- **Leaf:** The vertex has no children.
- **Internal Vertices:** Vertices that have children.

The $m-$ ary tree: If every internal vertex has no more than m children.

Full $m-$ ary tree: If every internal vertex has exactly m children.

Note: If $m = 2$ then $m-$ ary tree is called binary tree.

Sub rooted tree: A subgraph of a rooted tree.

Ordered rooted tree: The children of each internal vertices are ordered from left to right.

Theorem A Full $m-$ ary tree with

1. i internal Vertices has $n = mi + 1$ vertices and $l = (m-1)i + 1$ leaves.
2. n vertices has $i = \dfrac{n-1}{m}$ internal vertices and $l = \dfrac{(m-1)n+1}{m}$ leaves.
3. l laves has $n = \dfrac{ml-1}{m-1}$ vertices and $i = \dfrac{l-1}{m-1}$ internal Vertices.

Proof: Hint:1. Let $n =$ vertices, $i =$ internal vertices and $l =$ leaves.

- Each internal vertices has m children.
- Total vertices without root are mi.
- But $n = l + i$

Assignment :

Level of a vertex v (L): The length of the unique path from the root to this vertex v.

Height: The maximum of the levels of the vertices e.i. length of the longest path from the root.

Balanced: All the leaves are at levels L or $L-1$.

Path length and Weighted Path Length: The sum of the path lengths from the root to all pendant vertices, and weighted path length is defined as

$$\sum w_j L_j$$

Labeled Graph: A graph in which each vertex is assigned a unique name or label (i.e., no two vertices have the same label).

Theorem There are at most m^H laves in an $m-$ary tree of height H.

Verification:

Cayley's Theorem The number of labeled trees with n vertices $(n \geq 2)$ is n^{n-2}.

Verification:

Assignment:

- Minimum possible height of an n vertex binary tree.
- Maximum possible height of an n vertex binary tree.

Spanning Tree/skeleton/ scaffolding/maximal tree subgraph A tree T is called a spanning tree of a connected graph G if T is a subgraph of G and T contains all vertices of G. i.e

$$V(T) = V(G), \quad T \subseteq G$$

Spanning forest: A disconnected graph G with k component is called spanning forest if it has k spanning tree. i.e

$$\sum V(T_i) = V(G), \quad i = 1, 2, \ldots k$$

Branch: An edge of a spanning tree.

Chord/tie/link: An edge of a graph G that is not in a spanning tree.

Fundamental numbers: A graph has following fundamental numbers

$$n, \quad e \quad k$$

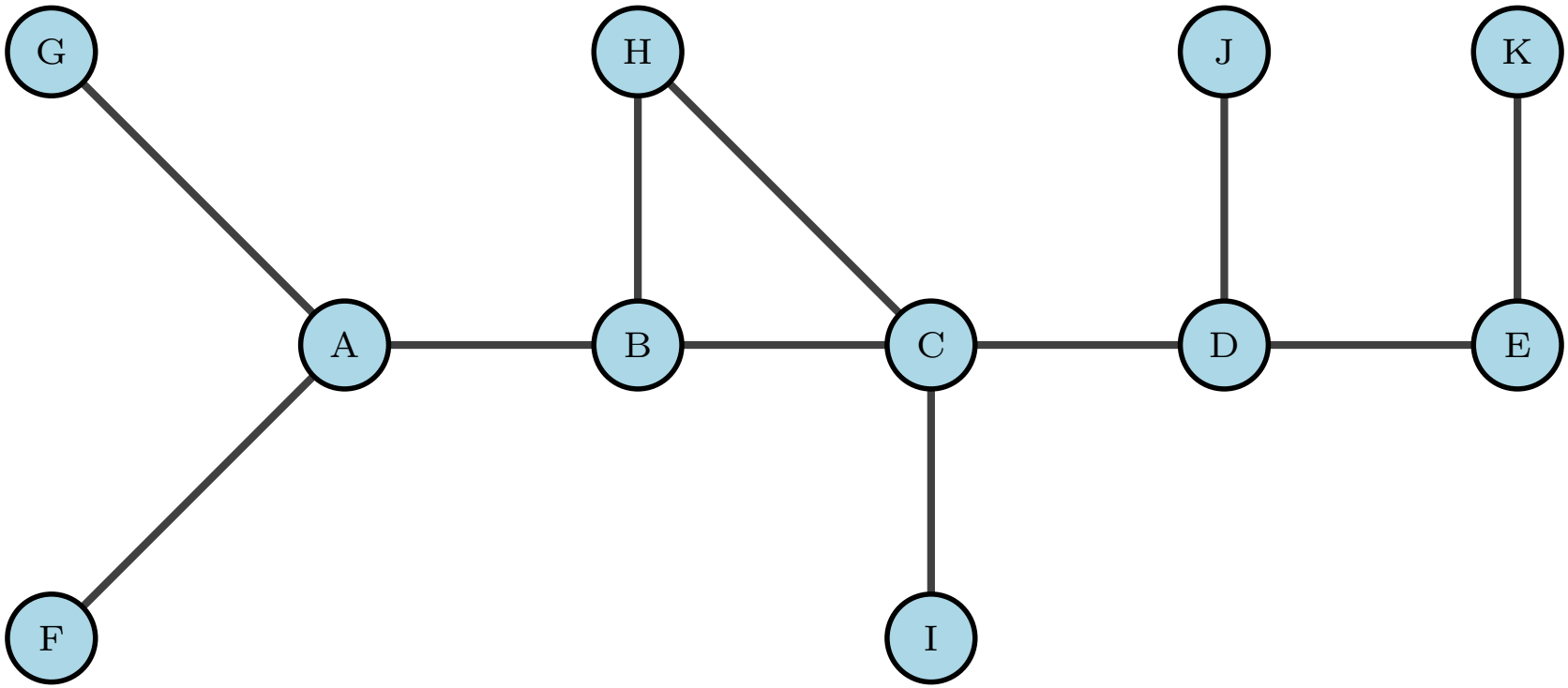
Rank of a graph: It is defined as $r = n - k$.

Nullity of a graph: It is defined as $\mu = e - r = e - n + k$.

Fundamental Circuit: A circuit, formed by adding a chord to a spanning tree T, is called a fundamental circuit.

Algorithm for finding spanning tree: We have following algorithm

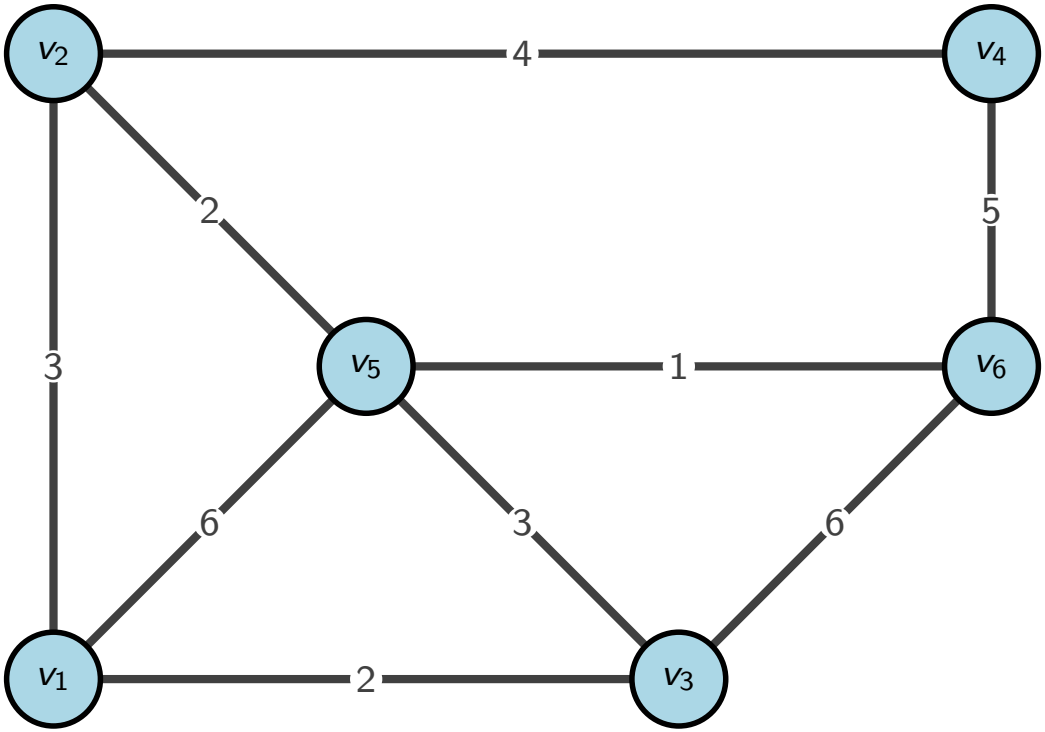
- Cyclic interchange /Elementary tree transformation
- Depth-first search/Backtracking
- Breadth-first search



Distance between two spanning trees: The number of edges of G present in one tree but not in the other.

Minimal spanning tree or shortest spanning tree or shortest distance spanning tree: A spanning tree with the smallest weight in a weighted graph is called a shortest spanning tree or shortest-distance spanning tree or minimal spanning tree.

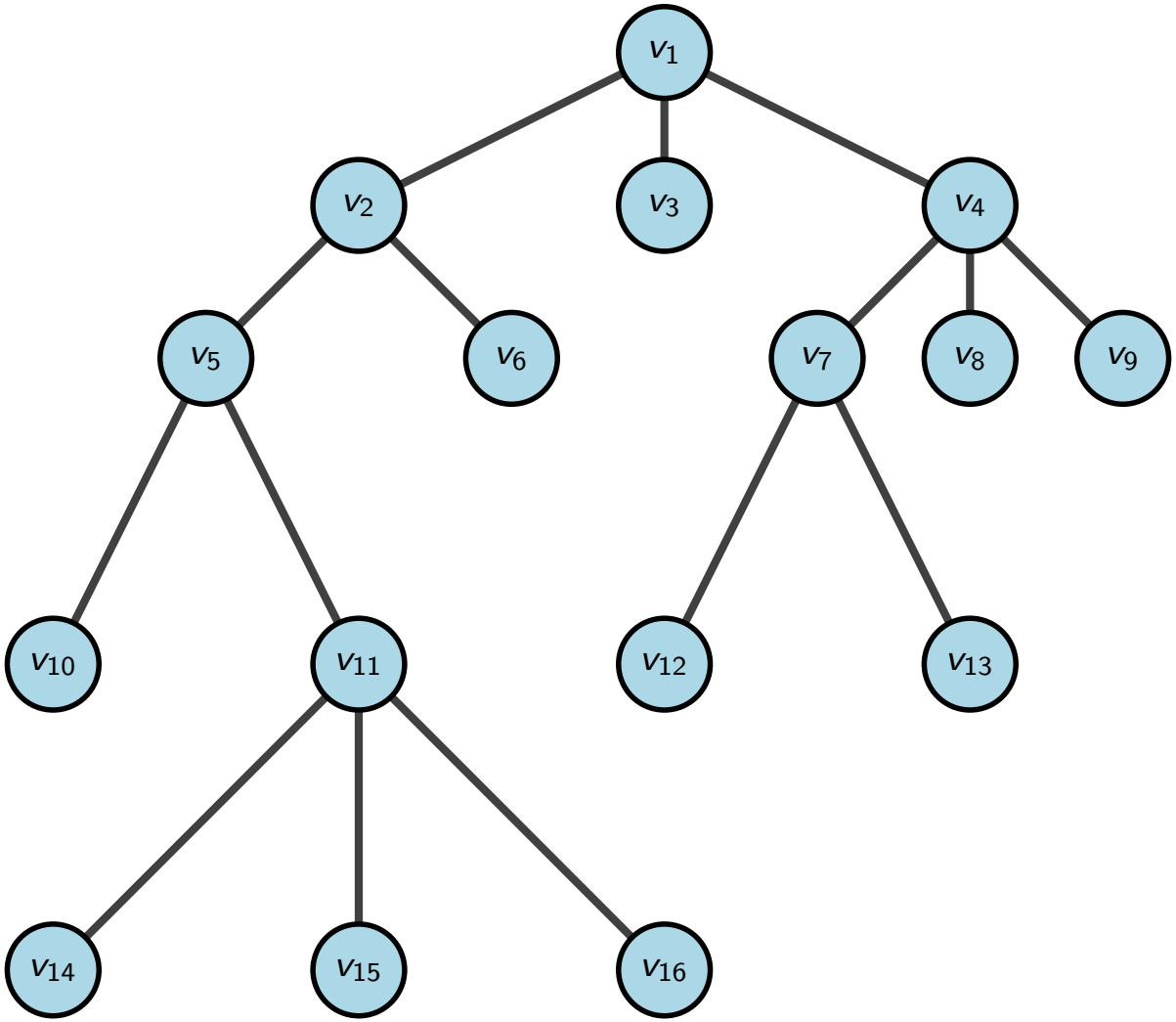
- **Kruskal's Algorithm:** List all edges of the graph G in order of nondecreasing weight. Next, select a smallest edge of G. Then for each successive step select (from all remaining edges of G) another smallest edge that makes no circuit with the previously selected edges. Continue until n - 1 edges have been selected, and these edges will constitute the desired shortest spanning tree.
- **Prim's Algorithm:** Start from vertex v1 and connect it to its nearest neighbor (i.e., to the vertex which has the smallest entry in row 1 of the table), say vk . Now consider v1 and vk as one subgraph, and connect this subgraph to its closest neighbor (i.e., to a vertex other than v1and vk that has the smallest entry among all entries in rows 1 and k). Let this new vertex be vi . Next regard the tree with vertices v1 , vk , and vi as one subgraph, and continue the process until all n vertices have been connected by n - 1 edges.


v_2
4
v_4
2
5
3
v_5
1
v_6
6
3
6
v_1
2
v_3

Traversal in Ordered Rooted Tree

Tree traversal: A systematic method for visiting every vertex of an ordered rooted tree is called traversal algorithm.
Let T be an ordered rooted tree with root r. If T consists only of r, then r is pre-order or in-order,or post-order. Otherwise, suppose that $T_1, T_2, (1), T_n$ are the subtrees at r from left to right. Then

- **Pre-order:** begins by visiting r. It continues by traversing T_1 in pre-order, then T_2 in pre-order, and so on, un-till T_n is traversed in pre-order.
- **In-order:** begins by traversing T_1 in-order, then visiting r. It continues by traversing T_2 in in-order, then T_3 in in-order,and so on, un-till T_n is traversed in in-order.
- **Post-order:** begins by traversing T_1 post-order, then T_2 in post-order, and so on and then T_n in post-order, and ends by visiting r.


v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
v_{10}
v_{11}
v_{12}
v_{13}
v_{14}
v_{15}
v_{16}

Some complicated expressions, such as compound propositions, arithmetic expressions, and combinations of sets using the ordered rooted tree.

Types of form Notations:

- **Infix form:** We obtain the infix form of an expression by traversing its binary tree in in-order.
- **Prefix form (Polish notation):** We obtain the prefix form of an expression by traversing its binary tree in pre-order.
- **Postfix form (Reverse Polish notation):** We obtain the postfix form of an expression by traversing its binary tree in post-order.

Example: (x+y)/(x+3)

Codeword and Code A codeword is a sequence of symbols taken from an output alphabet (denoted as Σ) to encode each character of a set of characters (from an input alphabet). A code is a set C such that for all $c \in C$, where c is a codeword generated over Σ.

A $q-$ary code is a code over an alphabet of q symbols, conventionally, $\{0, 1, 2, (1), q-1\}$.

Example: Binary code $C = \{000, 010, 111\}$, where $000, 010, 111$ are its codewords with fixed length 3.

Type of Coding Schemes:

- Fixed length code
- Pre-fix (a variable length) code

The coding is represented using a binary tree.

Connectivity in Graph

Cut Edge/Bridge: A edge e in a connected graph G is called cut-edge of G if G-e makes the G disconnected. Also named as Cut Arc or Cut Edges or Bridge.

Cut set (Edge Cut set:) A set of minimum number edges of G which, if removed (or "cut"), disconnects the graph (i.e., forms a disconnected graph). It is a set of edge of a graph G whose deletion increases the graph's number of connected components.

Edge Connectivity: The minimum number of edges whose removal makes 'G' disconnected is called edge connectivity of G. Notation - $\lambda(G)$

Cut vertex: A vertex v in a connected graph G is called cut-vertex of G if G-v makes the G disconnected.

Cut vertex set: A set of minimum number vertices of G which, if removed (or "cut"), disconnects the graph (i.e., forms a disconnected graph).

Vertex Connectivity: The minimum number of vertices whose removal makes 'G' disconnected is called vertex connectivity of G. Notation - $\kappa(G)$

Separable Graph: If $\kappa(G) = 1$.

k-connected Graph: If $\kappa(G) = k$.

Fundamental cut-set/Basis: A cut-set S containing exactly one branch of spanning tree of a connected graph G.

Result: A connected graph G may have atmost $n-2$ cut vertices.

Result: The maximum number of cut edges possible in a connected graph G is $n-1$.

Network flow Problem:

Cut-set with respect to pair of vertices:

Max Flow Min cut Theorem: The maximum flow possible between two vertices a and b in a network is equal to the minimum of the capacities of all cut sets with respect to a and b.

Proof:

Graph on Surfaces

Geometric representation of a Graph: A different diagram form.

Embedding: A drawing of a geometric representation of a graph on any surface such that no edges intersect.

Plane Representation: An embedding of a graph on a plane.

Planar Graph: If a graph has a plane representation.

Kuratowski's Graphs: We have following graphs

- A complete graph with five vertices K_5
- A regular graph with six vertices and nine edges $K_{3,3}$

Some Results: We have following results

- K_5 and $K_{3,3}$ are non-planer.
- K_5 and $K_{3,3}$ are regular.
- $K_5 - e$ and $K_{3,3} - e$ and $K_5 - v$ and $K_{3,3} - v$ planer.
- K_5 is smallest number of vertices which is non-planer.
- $K_{3,3}$ is smallest number of edges which is non-planer.

Regin/window/faces/meshes: A plane representation divides the plane into many parts these parts are called region.

Euler Formula: A connected planer graph with n vertices and e edges has $e - n + 2$ region.

Proof:

Coloring in Graph

Coloring: Painting all the vertices/edges. or $f : v(G) \to C$, (Set of color)

Proper coloring of a graph: Painting all the vertices of a graph with color that no two adjacent vertices have the same color.

Properly colored graph: A graph having proper coloring.

Chromatic number: Minimum number of color required for a proper coloring. Notation $\kappa(G)$.

$\kappa-$Chromatic Graph/$\kappa-$ colorable graph: A graph having minimum number of color required for a proper coloring.

Some Observations: Some observations that follow directly from the definitions:

1. For coloring problems we need to consider only simple, connected graphs.
2. Graph G is $\kappa-$chromatic graph for $n = 1$.
3. Graph G is atleat $2-$chromatic graph for $e \geq 1$ (no self-loop).
4. Graph G is atleat $\omega-$chromatic graph if $K_{(\omega)} \subseteq G$.
5. Graph C_n is $2-$chromatic graph if n is even and $3-$chromatic graph if n is odd respectively.

Theorem: Every tree with two or more vertices is 2− chromatic.

Proof: Hint:

- Consider T as a rooted tree at vertex v.
- Paint v with color 1. Paint all vertices adjacent to v with color 2.
- Paint the vertices adjacent to these (those that just have been colored with 2) using color 1.
- Continue this process till every vertex in T has been painted.
- Observe that all vertices at odd distances from v have color 2, while v and vertices at even distances from v have color 1.
- Observe that along any path in T the vertices are of alternating colors.
- Since there is one and only one path between any two vertices in a tree, no two adjacent vertices have the same color.
- Thus T has been properly colored with two colors.

Matching/Assignment: A set of edges of a graph, in which no two edges are adjacent.

Covering: A set of edges of a graph, if every vertex in graph is incident on at-least one edge.

Theorem: Every tree with two or more vertices is 2− chromatic.

Proof:

Matrix representation of a graph:

A matrix is a convenient and useful way of representing a graph to a computer.

Incidence Matrix: An n by e matrix $A = [a_{ij}]$ of a graph with no self-loop defined as

$$\begin{aligned} a_{ij} &= 1 \quad \text{if jth edge is inciednt on ith vertex} \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Adjancey Matrix: An n by n matrix $A = [a_{ij}]$ of a graph with no parrel edge defined as

$$\begin{aligned} a_{ij} &= 1 \quad \text{if there is an edge ith and jth vertex} \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Following matrices are frequently used in a graph.

- Circuit matrix
- Fundamental Circuit matrix
- Edge Cut set matrix
- Fundamental Edge Cut set matrix
- Path matrix

Questions/Query ?