

Graph Theory

Ujjwal Maulik

Department of Computer Science & Engineering

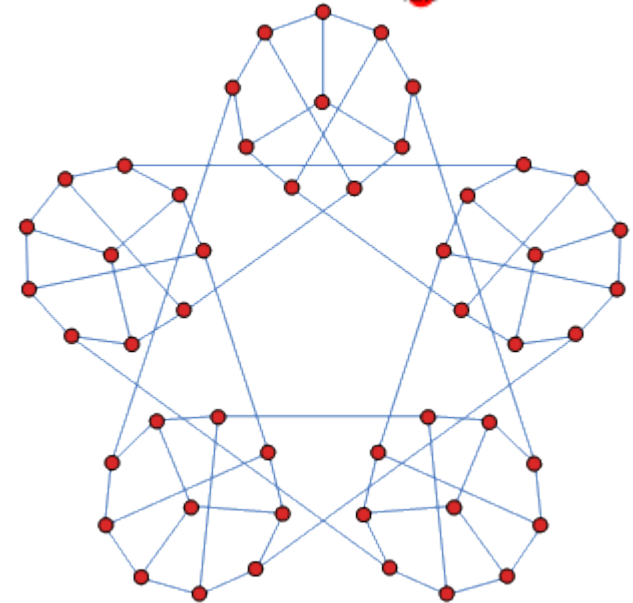
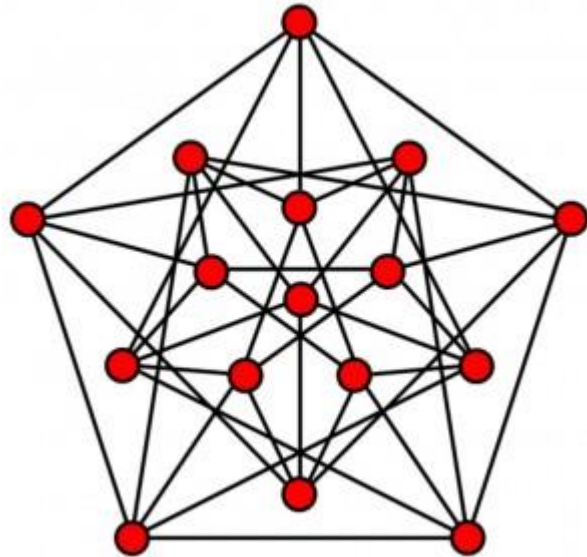
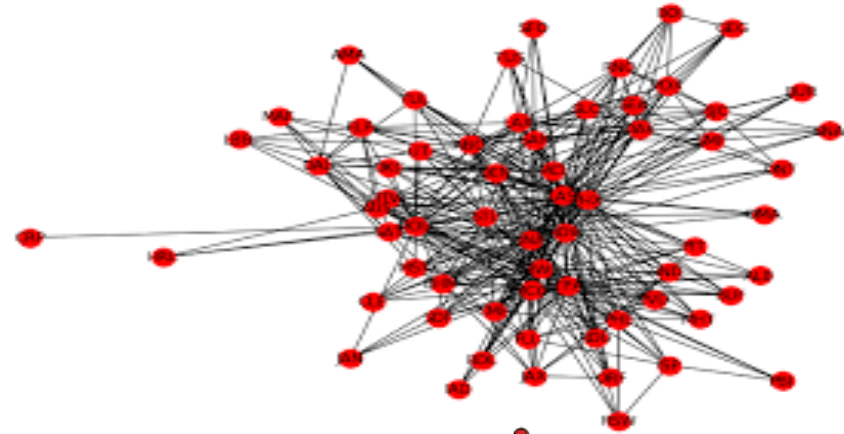
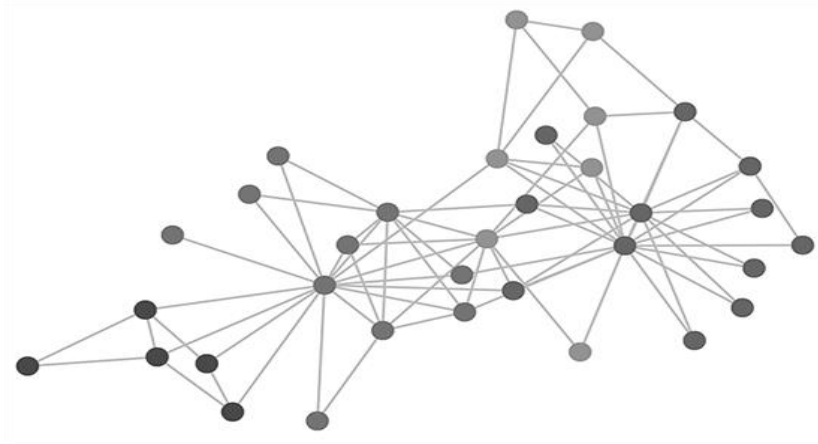
Jadavpur University, Kolkata

Definition

- Graph $\rightarrow G(V, E)$
 - Defined by a set of Vertices V and Set of Edges E where
 - Each edge is defined by two vertices which may be same/different.
- Example
 - Social Network, Biological Network, Computer Network ...

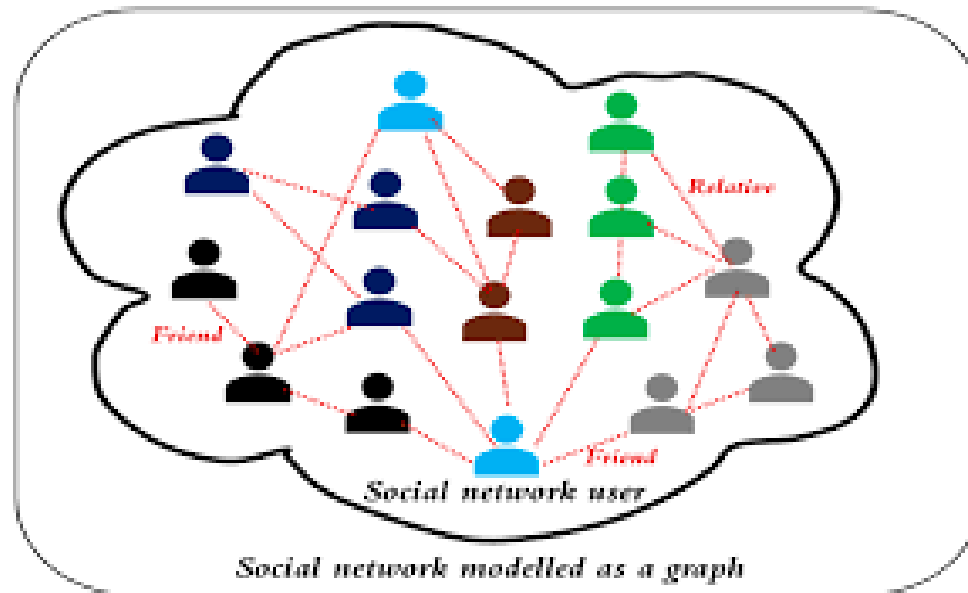
Example of Graphs

Ack: Wikipedia



Social Network

Ack: Wikipedia



Graph Theory

Basic Concepts

- Weighted/Unweighted
 - Weight may or may not be associated with the edges
- Directed/Undirected
 - There may or may not be direction associated with the edges
- Connected/disconnected
 - Disconnected graph form Forest.
- Computer representation of a Graph
 - Adjacency Matrix, Link List (better if the adjacency matrix is sparse)

Some theorems

- Total degree of all the vertices in a Graph is even.
 - Each edge contributes 2 degrees. So total degree is $2 \times E$ where E is the number of edges in the graph.
- Number of odd degree vertices in a Graph is even.
 - Total Degree = Degree of (even degree vertices + odd degree vertices).
 - So Even = Even + Degree of odd degree vertices
- If there is only two odd degree vertices in a graph they will be in the same component

Graph Theory

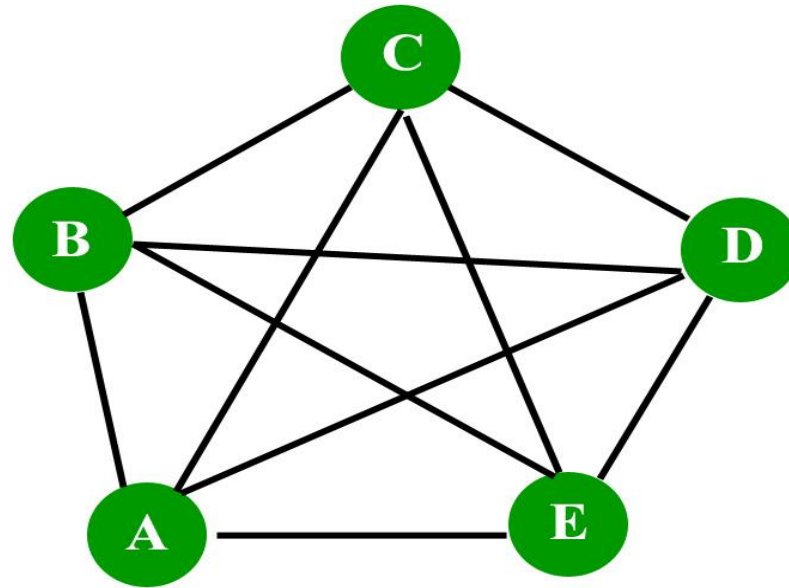
Basic Concepts

- Path and Circuit
 - Succession of adjacent edges in a graph without repeated edges are known as path.
 - A circuit is a path which joins a node to itself.
- Euler Graph, Euler Path and Euler Circuit
 - *A connected graph whose all vertices are of even degree is called Euler Graph.*
 - *In the connected graph Euler Path is a trail which contains all the edges of the graph. A closed Euler trail is known as an Euler Circuit.*
- Hamiltonian
 - A path that passes through each vertex of any graph exactly once is known as Hamiltonian path. A graph possessing a Hamiltonian cycle is called Hamiltonian/Hamilton graph.

Complete Graph and Planner Graph

- Complete Graph
 - There is an edge or direct connection between any two vertices
 - n^2 edges for a graph with n vertices
- Clique and Quasi-Clique
 - Complete subgraph of a graph is known as clique. A almost complete subgraph of a graph is known Q-clique
 - Example: Strong Community is a social network
- Planner/non Planer
 - A graph which can be drawn in a plane such a way that any two edges are not intersecting.
 - Complete graph with more than 4 vertices are non planner
 - Useful for VLSI floor planning

Complete Graph



Complete Graph

Clique and Quasi Cliques

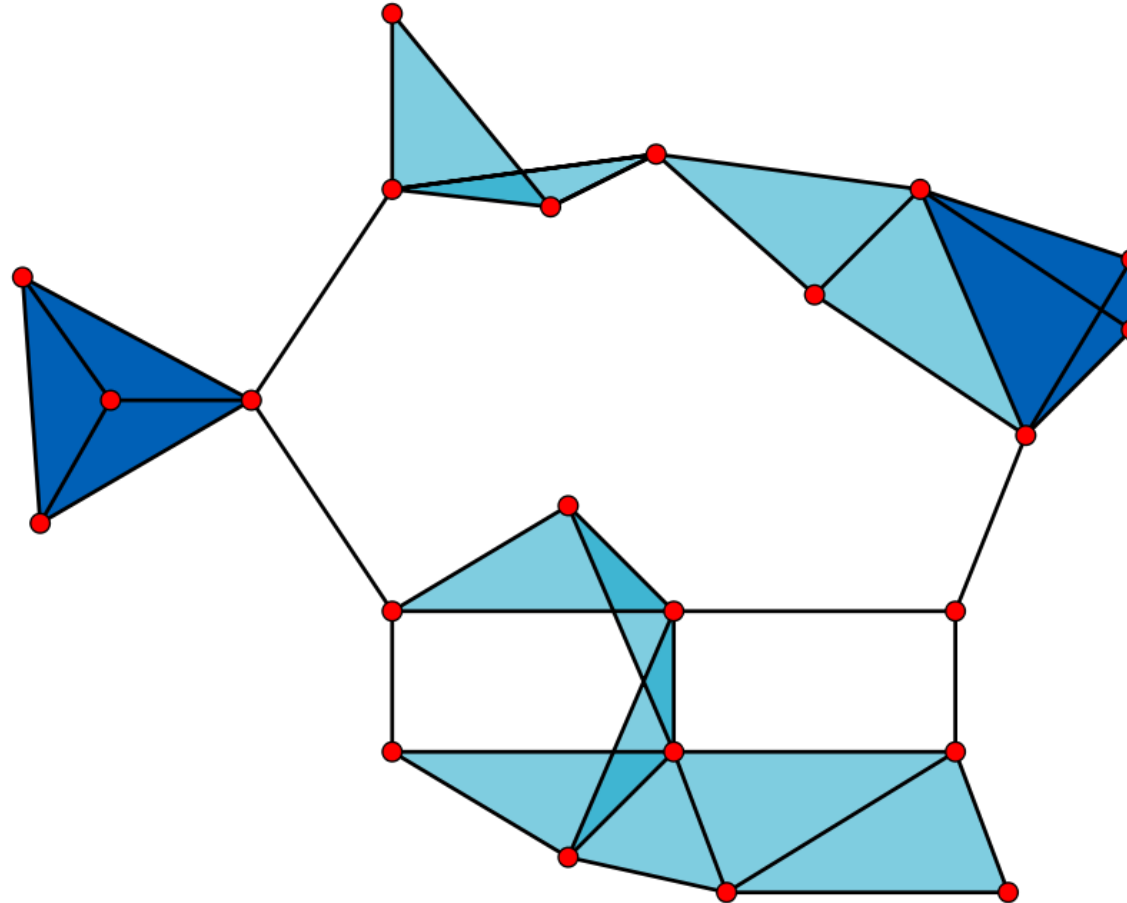
- Clique
 - Complete Subgraph of a Graph
 - If subgraph $G^*(V^*, E^*)$ of Graph (V, E) is a clique then
 - V^* and E^* are subset of V and E respectively
 - If cardinality(V^*) = m then cardinality of (E^*) is $mC2$
 - We are mostly interested to find Maximum Clique
- Quasi Clique (Q-Clique)
 - Almost Complete Subgraph of a Graph
 - If subgraph $G^{**}(V^{**}, E^{**})$ of Graph (V, E) is a quasi clique then.
 - V^{**} and E^{**} are subset of V and E respectively
 - If cardinality(V^{**}) = m then cardinality of (E^{**}) is $< mC2$
 - The Quasi part may be defined by user.

(Write Code for Clique and Q-Clique – Lab Exercise)

Example of Clique

Ack:

[https://en.wikipedia.org/wiki/Clique_\(graph_theory\)#/media/File:VR_complex.svg](https://en.wikipedia.org/wiki/Clique_(graph_theory)#/media/File:VR_complex.svg)



Finding K-Clique (using k-1 size clique)

- Every edge of a Graph is 2-clique
 - $A \leftarrow$ Find the clique of size 3 from two cliques of size 2.
 - $B \leftarrow$ Take the Union of above two cliques of size 2.

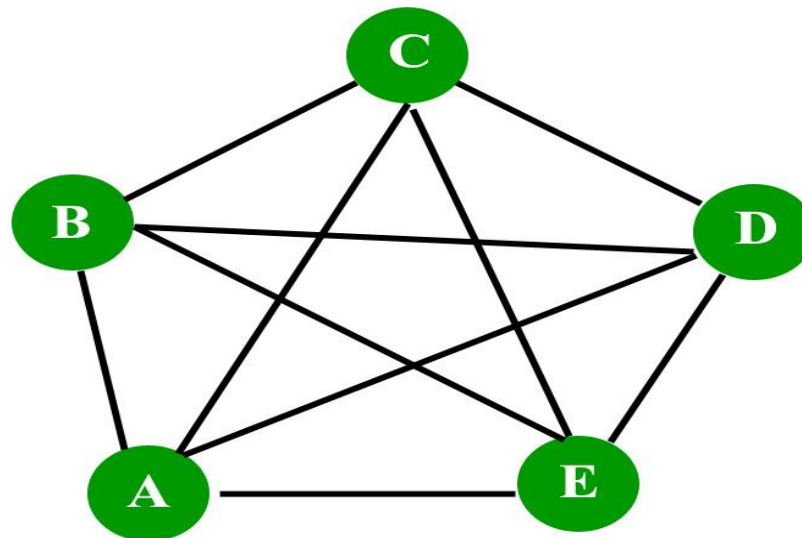
If (A-B) is an edge and that is present in the graph then A is clique of size 3.
- To generate 3-cliques from 2-cliques
- Find intersection of each combination pair of 2-cliques and take intersection of the pair, if the intersection is an edge and it is present in the graph then the union of the pair is a clique of size 3.
- By doing intersection of the pair we find the missing edge so that the 2-clique can be extended to 3-clique, and if the edge is present in the graph then we extend the 2-clique pair into 3-clique and store it. In similar way we generate k+1-clique from k-clique.

Clique- Algorithm

Remove Edge AB and BC from the Following Graph and then find the clique of Size 4

As we are looking for clique of size 4 remove all vertices less than degree 3.

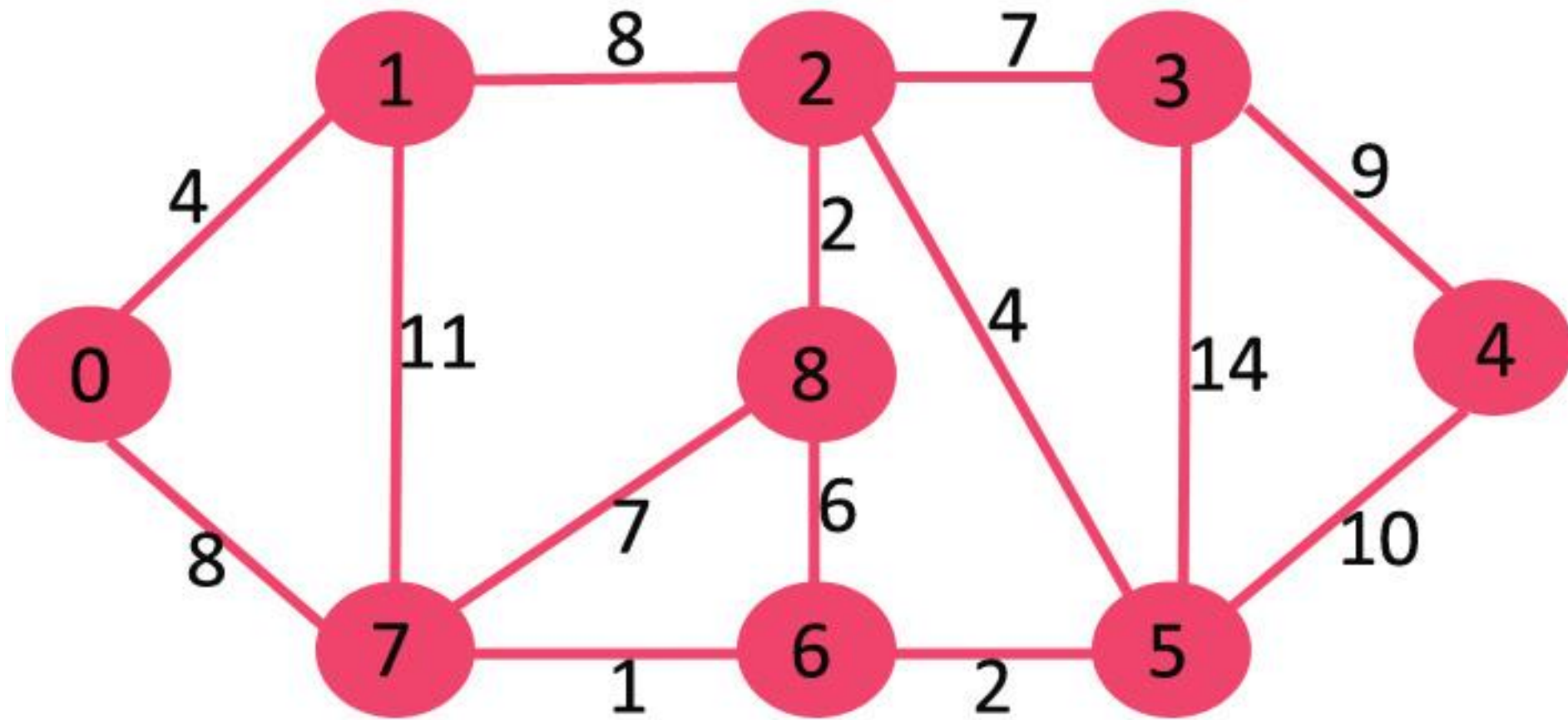
So remove vertices B and all edges from B. Continue this process.



Minimum Spanning Tree

- Spanning Tree
 - Spanning Tree of a Graph is a Tree that contain all the vertices of the Graph
- Minimum Spanning Tree
 - Among all the Spanning Tree of a Graph the tree/trees whose sum of total edge weight is minimum
 - Greedy Algorithms
 - Kruskal Algorithm
 - Prims Algorithm

Weighted Graph



Kruskal's Algorithm

- Kruskal Algorithm(G, G^*)
 G – Input Graph
 G^* - MST
- Sort all the edges of the G in ascending order and insert in *Set S*.
- while S is nonempty or G^* is not yet spanning
 - remove an edge with minimum weight from S
 - Insert the edge along with the vertices in G^* if
 - insertion of the edge does not form a cycle in G^* .

(Write Code for Kruskal's Algorithm – Lab Exercise)

Prim's Algorithm

- Prim's Algorithm(G, G^*)
 G – Input Graph
 G^* - MST
- Initialize G^* with a vertex, chosen arbitrarily from G .
- while G^* is not yet spanning
 - Add one more edge which is minimum weight and also connect G^* to vertices not yet in the tree (*no cycle in G^**).

(Write Code for Prim's Algorithm – Lab Exercise)

Graph Colouring

- Colouring all the vertices of a graph is such a way such that two adjacent vertices do not have same colour.
 - Example: Colour Map of world/country/State
- 2 Colours is needed to Colour a Tree

Shortest Path Algorithm

- **Dijkstra's algorithm**

- Find the shortest path between any two nodes in a weighted graph where the weighted are +ive. The algorithm was developed by Edsger W. Dijkstra in 1956.

May watch the link for online videos

<https://www.youtube.com/watch?v=XB4MlexjvY0>

https://www.youtube.com/watch?v=JcN_nq1EAr4

Algorithm

- STEP I: Form an **unvisited set** with all nodes. Assign **Zero** to start node and **infinity** for all other nodes. Mark the initial node as **current node**.
- STEP II: For the **current node**, consider all of its unvisited neighbors and compute their *tentative* distances through the current node. Compare the newly calculated *tentative* distance to the current assigned value and assign the smaller one.

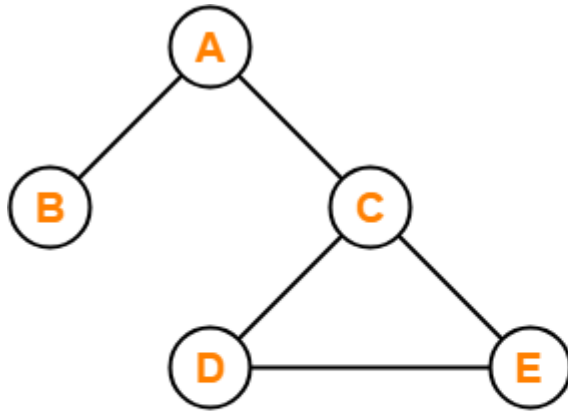
Example, Let **current node C** is marked with a distance of 10 and the edge connecting it with a neighbor **N** has length 5. Then the distance to **N** through **A** will be $10 + 5 = 15$. If **N** was previously marked with a distance greater than 15 then change it to 15. Otherwise, keep the current value.

- STEP III: After considering all of the unvisited neighbors of the **current node**, mark the **current node** as visited and remove it from the *unvisited set*. A visited node will never be checked again.
- STEP IV: Stop when the destination node is marked or if the smallest tentative distance among the nodes in the *unvisited set* is infinity.
- STEP V: Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new **current node**, and go back to STEP II.

Shortest Path Algorithm

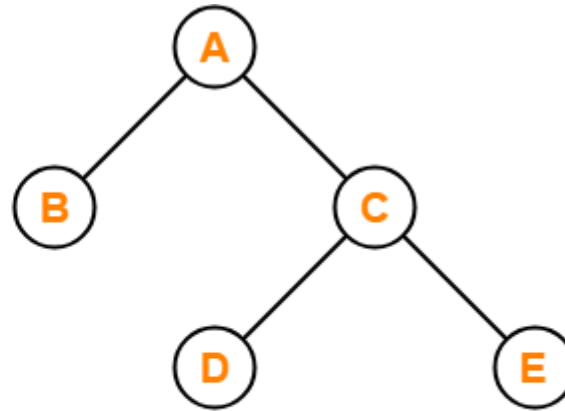
- Bellman–Ford algorithm
 - It can find the shortest path even when the edge weights are –ive but fail when there is a cycle with negative weights.
 - Detail
 - https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm
 - <https://www.tutorialspoint.com/Bellman-Ford-Algorithm-for-Shortest-Paths#:~:text=Bellman%2DFord%20algorithm%20is%20used,we%20can%20handle%20it%20easily.>
 - <https://www.youtube.com/watch?v=FtN3BYH2Zes>

Tree: Acyclic Graph



X

This graph is not a Tree



✓

This graph is a Tree

Tree Search

- BFS – Breadth First Search

- Uses Queue Data Structure
- Suitable when solutions are available in any depth.
- Find best solution with minimum depth.
- suitable when searching vertices are closer to the given source

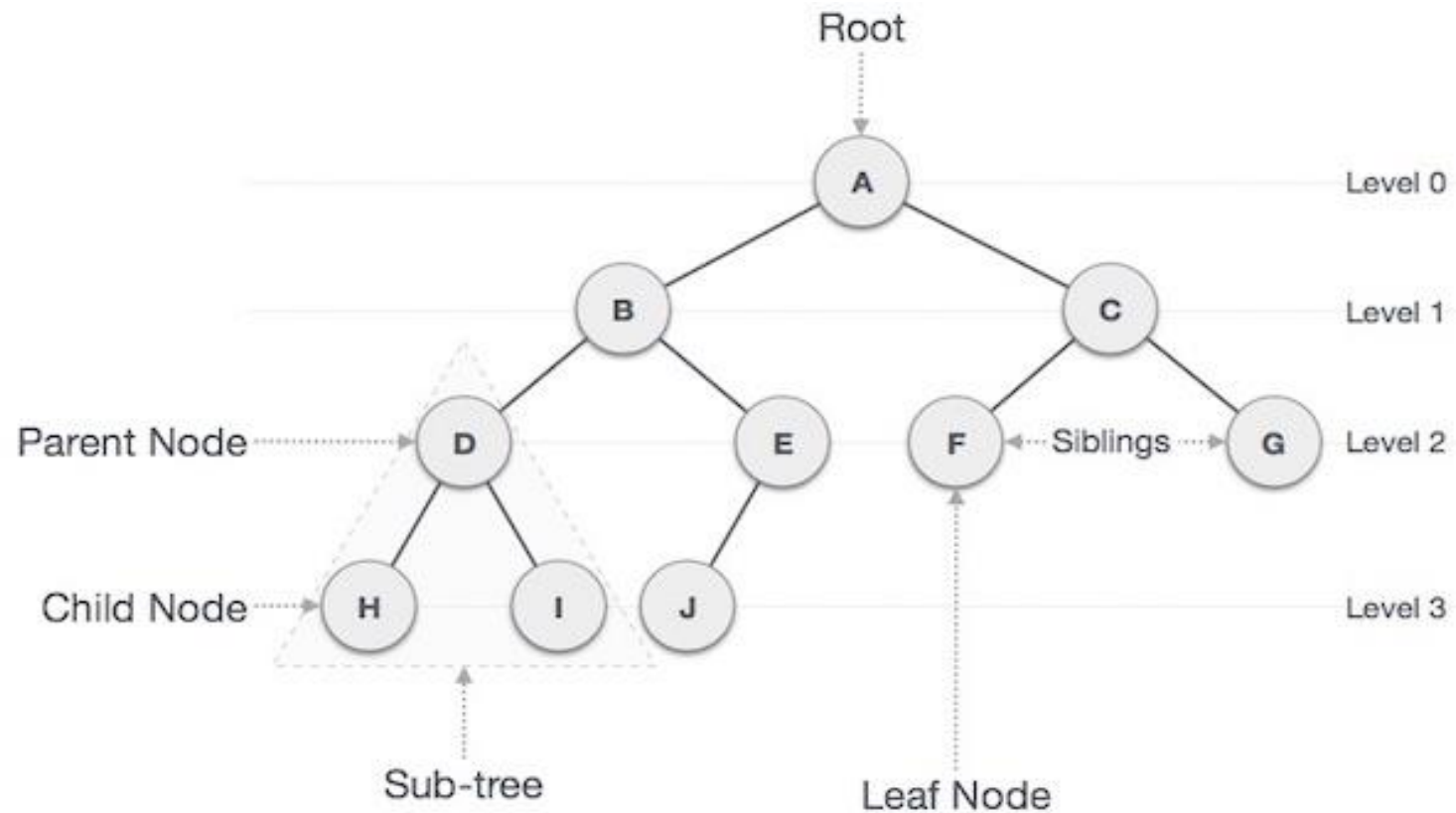
- DFS – Depth First Search

- Uses Stack Data Structure
- suitable when searching vertices are far from source
- Suitable when solutions are available only in the leaf nodes.
Example - games or puzzles

Example of DFS and BFS

DFS → ABDHIEJCFG

BFS → ABCDEFGHIJ



ISOMORPHISM

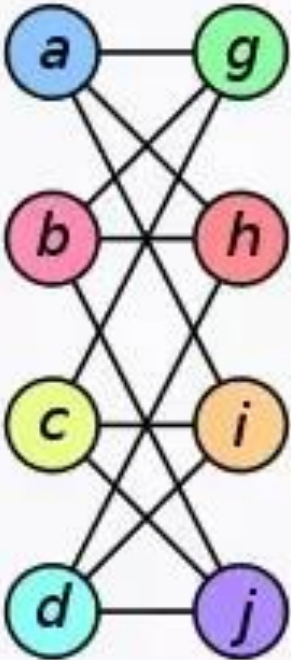
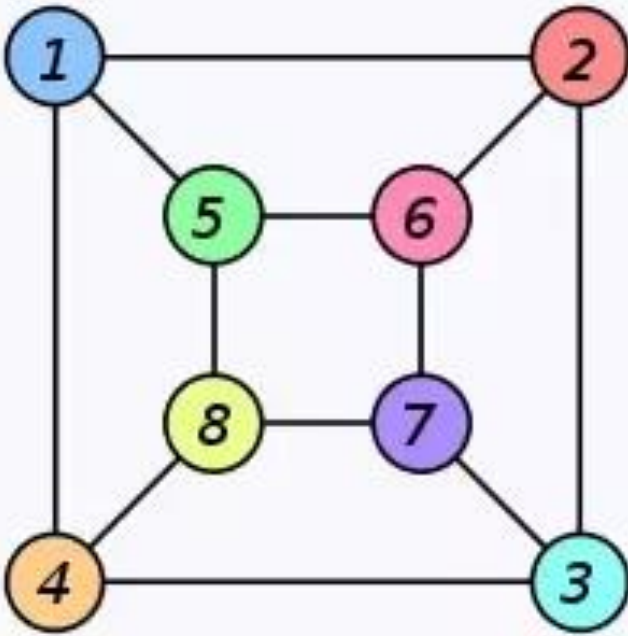
- Isomorphism is a structure-preserving mapping between two structures of the same type that can be reversed by an inverse mapping. Two mathematical structures are isomorphic if an isomorphism exists between them.

(Ack: <https://en.wikipedia.org/wiki/Isomorphism>)

- Isomorphism is a **one-to-one mapping/correspondence** between two sets while preserving binary relationships between the elements of the two sets.

Example: Isomorphism

(Ack: <https://qph.fs.quoracdn.net/main-qimg-16c40327f5a5eec560705ddf2e8eceb9.webp>)

Graph G	Graph H	An isomorphism between G and H
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$

Application: Isomorphism

- Automata:
 - Two languages are same/equal.
- Compiler:
 - Performing Different optimizations.
- Cryptography:
 - Zero knowledge proofs.
- Software/Hardware Verification:
 - Programs, Logic Proofs, Electronic Circuits.
- Search Engine:
 - linking two facebook's accounts of the same person,
 - recognizing web users based on their behavior,
 - recognizing plagiarism in students solutions,
 - database or medical images.
- Civil engineering, city planning, building interior planning
- Analysis of social structures

Bipartite Graph

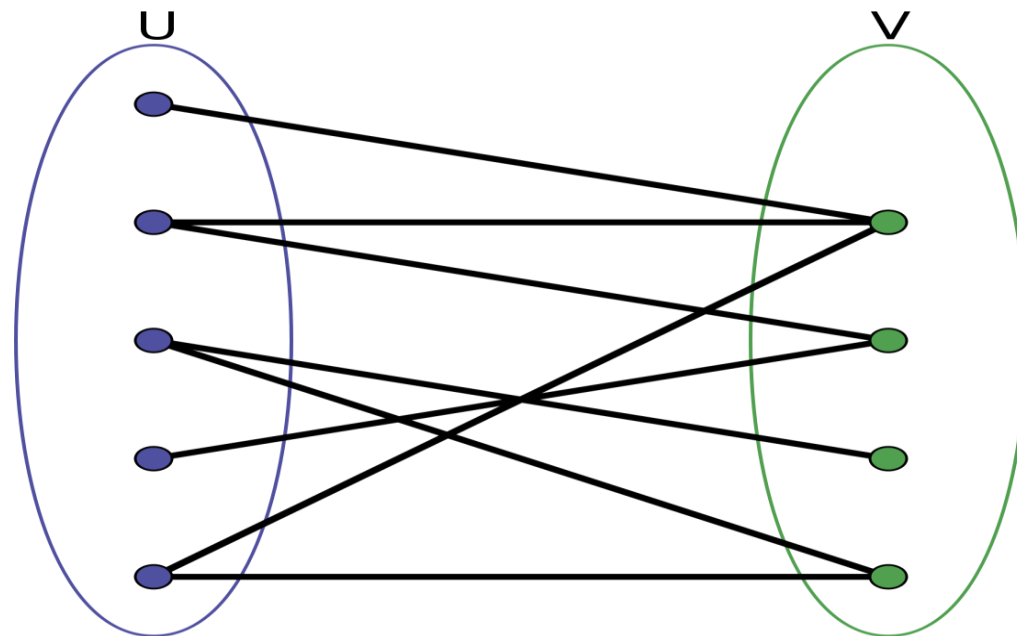
- A bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint and two independent sets U and V such that every edge connects a vertex in U to one in V .

(ack: https://en.wikipedia.org/wiki/Bipartite_graph)

Example: Bipartite Graph

Connections between

- two social groups
- Human/Virus proteins



References

- *Deo, Narsingh (1974). Graph Theory with Applications to Engineering and Computer Science. Englewood, New Jersey: Prentice-Hall. ISBN: 0-13-363473-6.*
- *Chartrand, Gary (1985). Introductory Graph Theory. Dover. ISBN: 0-486-24775-9.*
- *Gibbons, Alan (1985). Algorithmic Graph Theory. Cambridge University Press*