Graphs

Recap: Applications of Trees

- **Why Tree?**
  Unlike Array and Linked List, which are linear data structures, tree is hierarchical (or non-linear) data structure.

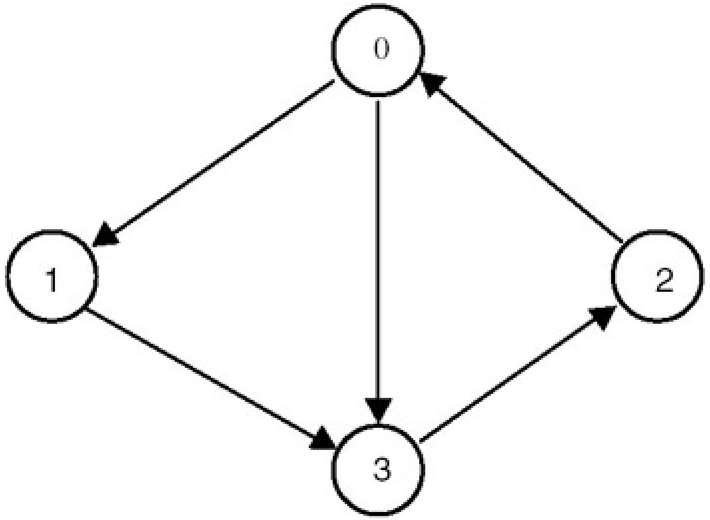
Why Trees?

- One reason to use trees might be because you want to store information that naturally forms a hierarchy.
- For example,
  - The file system on a computer: file system
  - Organization chart of a large organization.
  - Different tree data structures allow quicker and easier access to the data as it is a non-linear data structure.
    - Suffix Tree: For quick pattern searching in a fixed text.
    - Syntax Tree: Used in Compilers.

GRAPHS

Graphs: Term definition

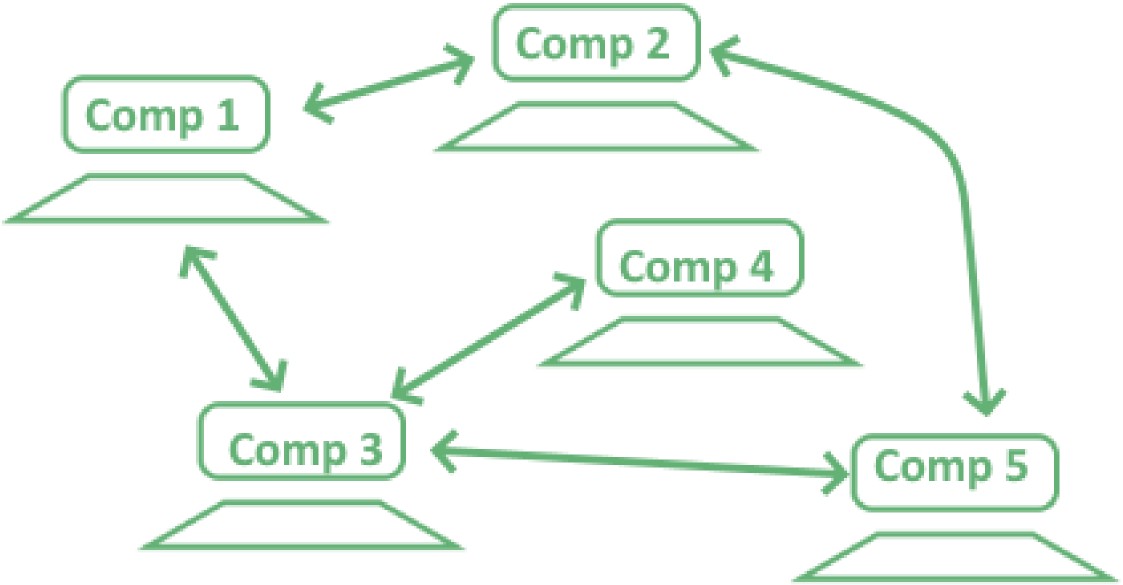
- *Graphs* are natural models that are used to represent arbitrary relationships among data objects.



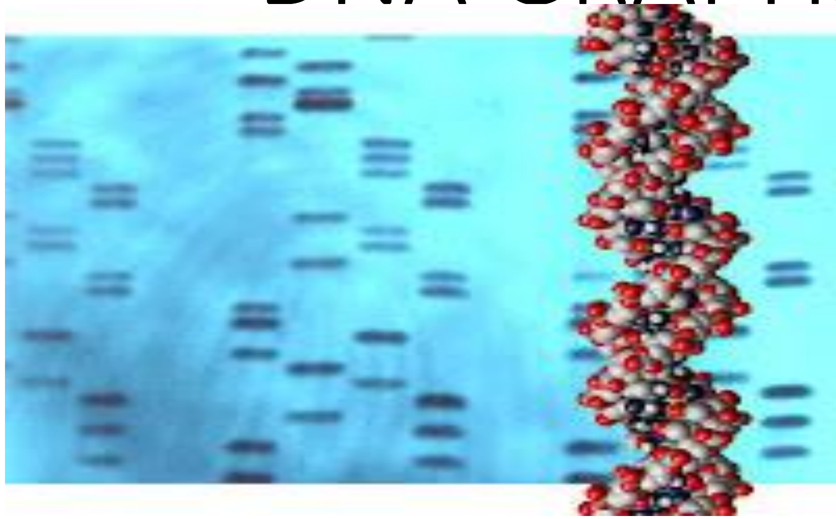
Road Networks

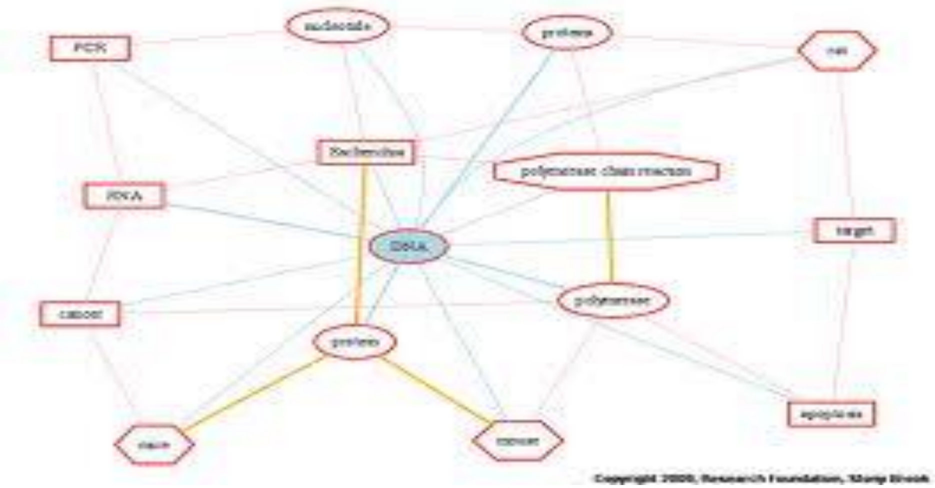
- In modeling a road network, the vertices may represent the cities or junctions, certain pairs of which are connected by roads/edges.

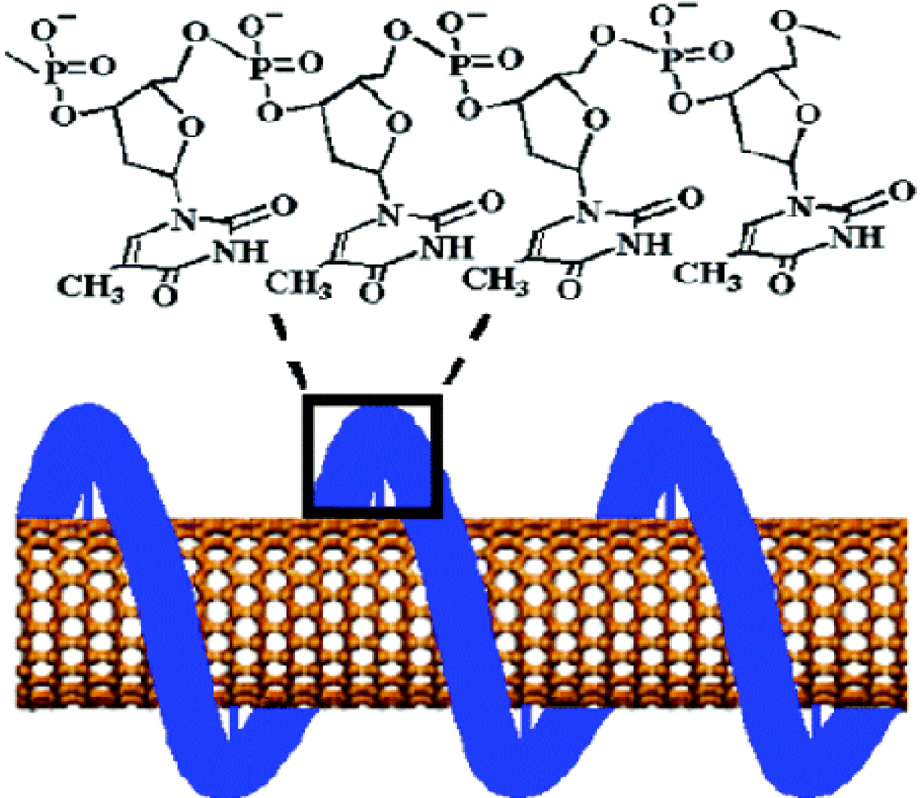
Network of Computers

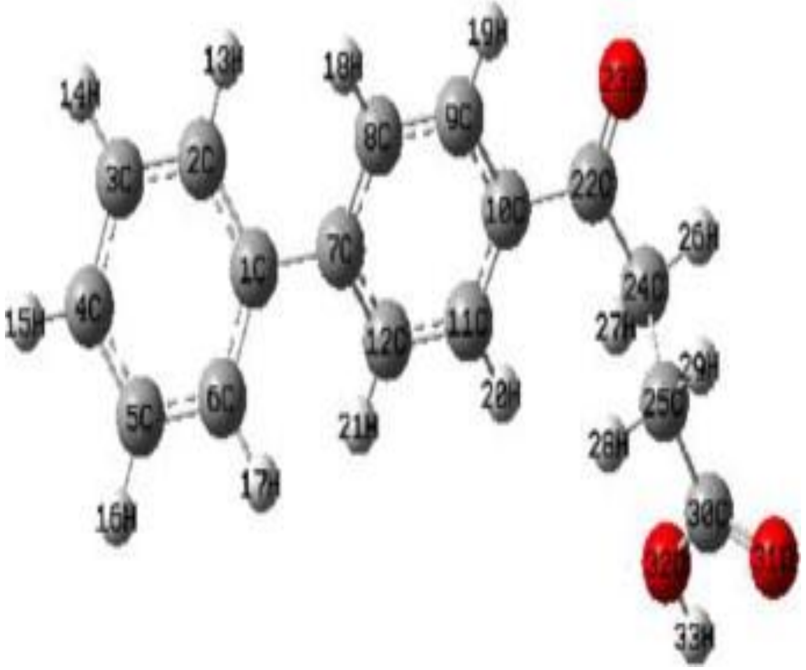


DNA GRAPHS



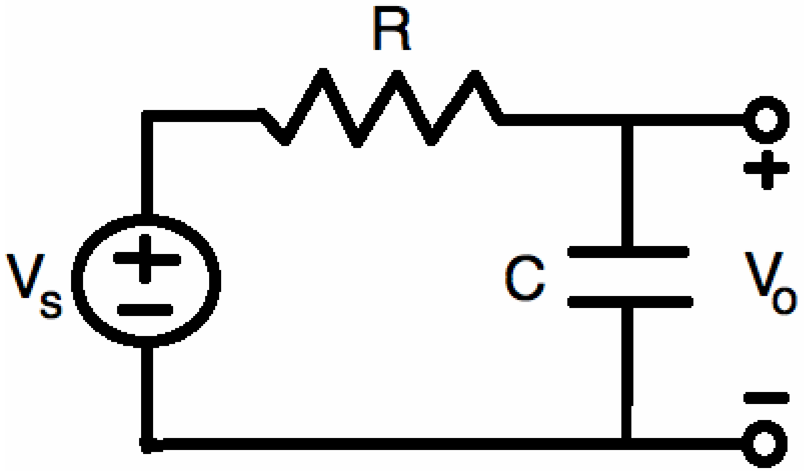


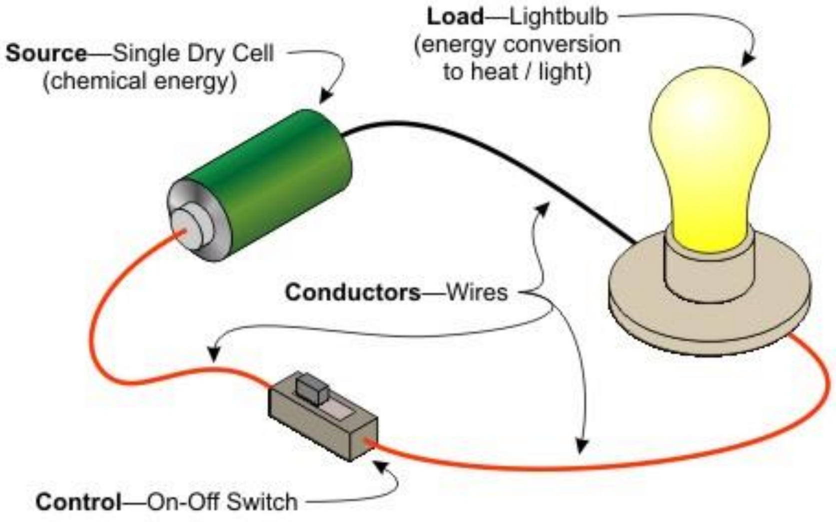




Electronic Circuits

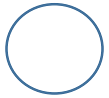
- In an electronic circuit, with junctions as vertices as components as edges.

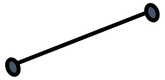




Graphs –Intuitive Notion

A graph is:

- a bunch of vertices (or nodes) represented by:
  - circles 
- which are connected by edges, represented by:
  - line segments .



Definition:

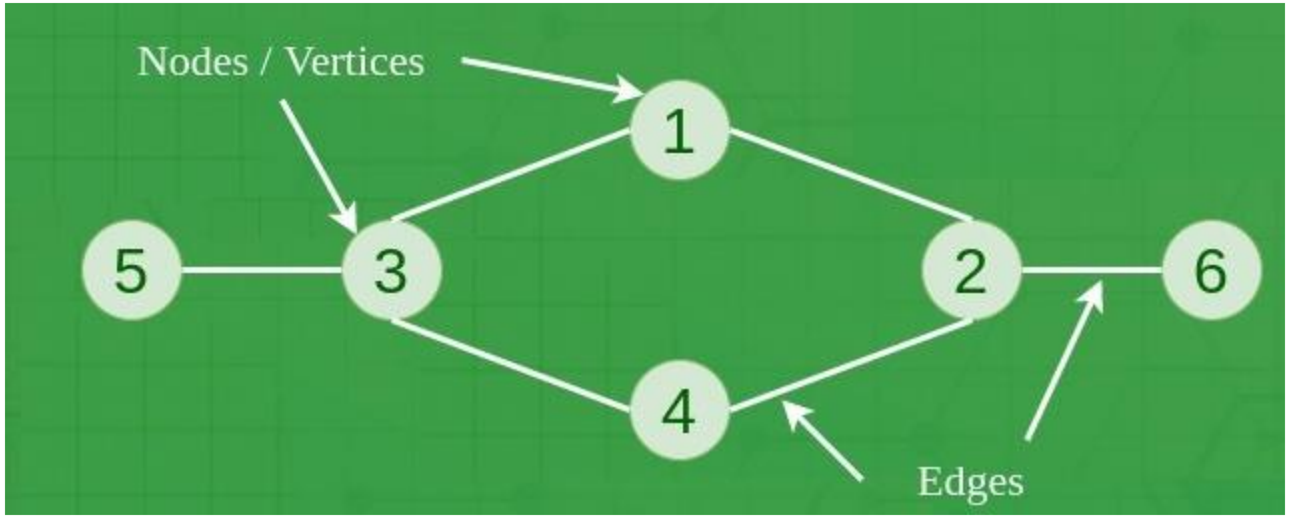
A graph G is defined as a set of two tuples that is:

G = (V,E),

where

V represents set of vertices of G and

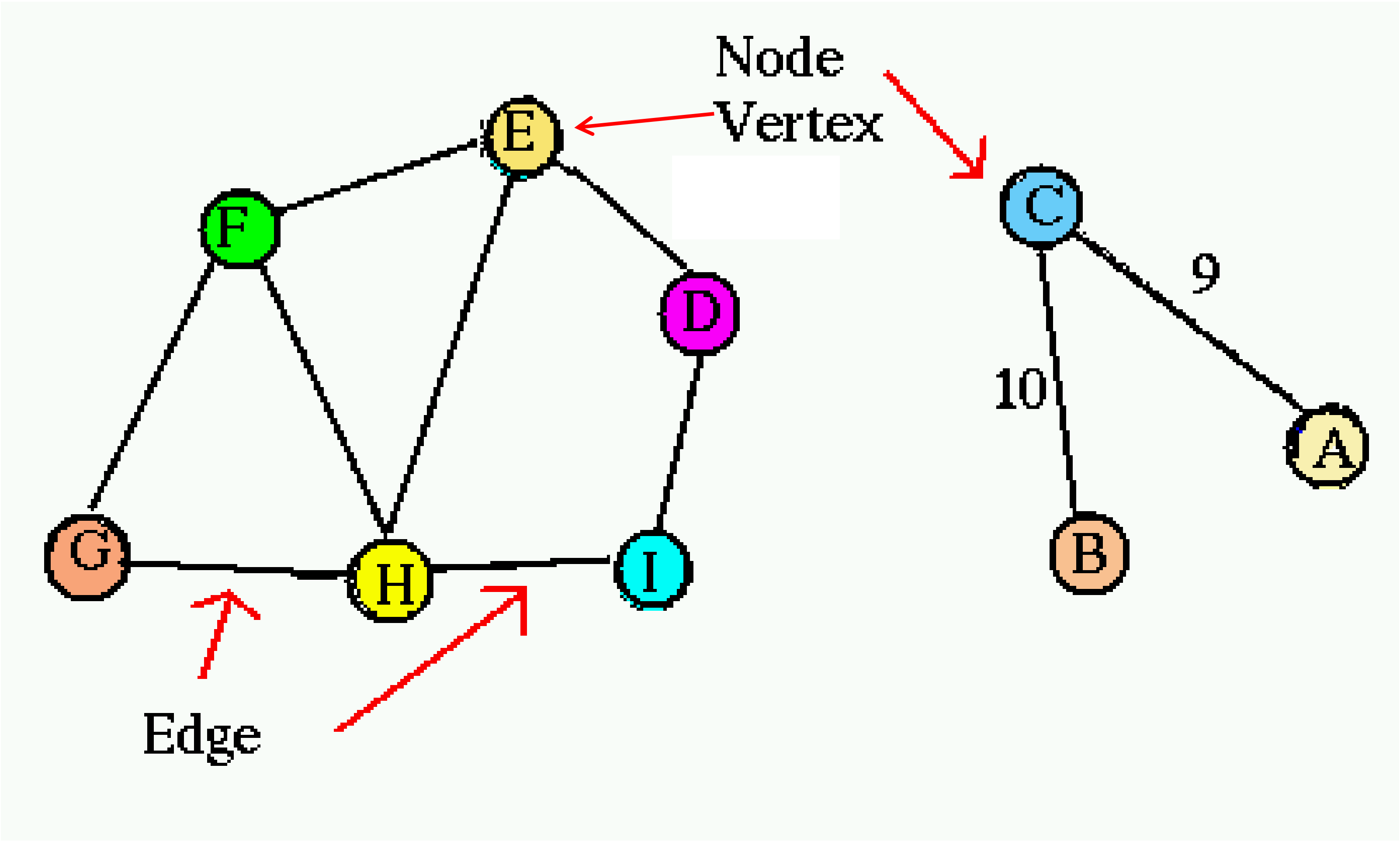
E represents the set of edges of G.



Components of a Graph

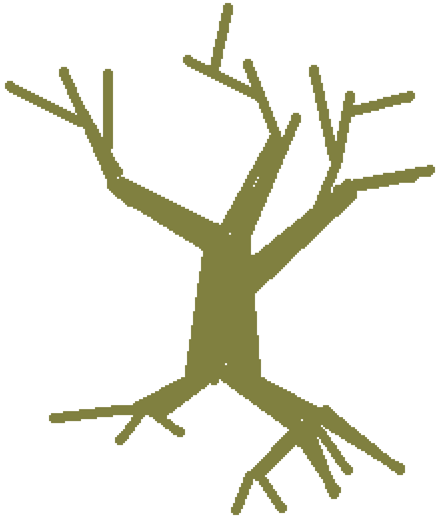
- **Vertices**: Vertices are the fundamental units of the graph. Sometimes, vertices are also known as vertex or nodes. Every node/vertex can be labeled or unlabelled.
- **Edges**: Edges are drawn or used to connect two nodes of the graph. It can be ordered pair of nodes in a directed graph. Edges can connect any two nodes in any possible way. There are no rules. Sometimes, edges are also known as arcs. Every edge can be labeled/unlabelled.

GRAPHS



Trees

A very important type of graph in Computer Science is called a *tree*:

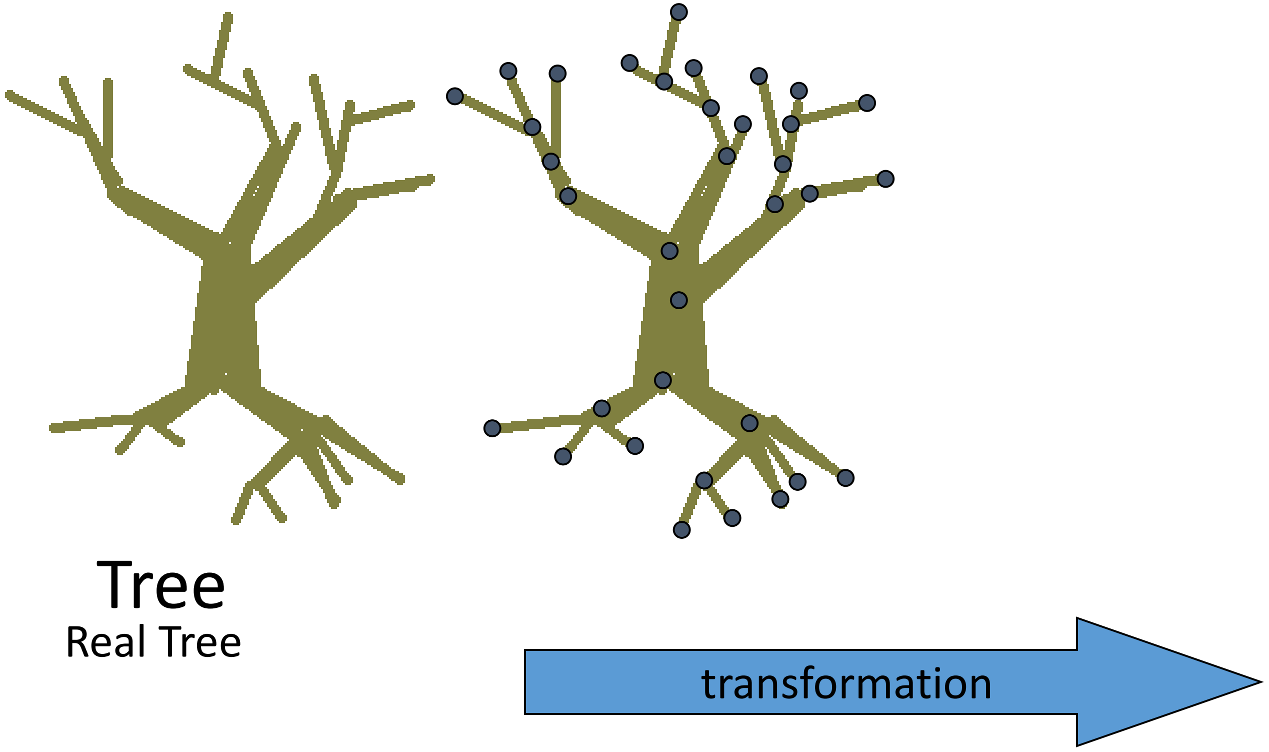


Real Tree

Real Tree

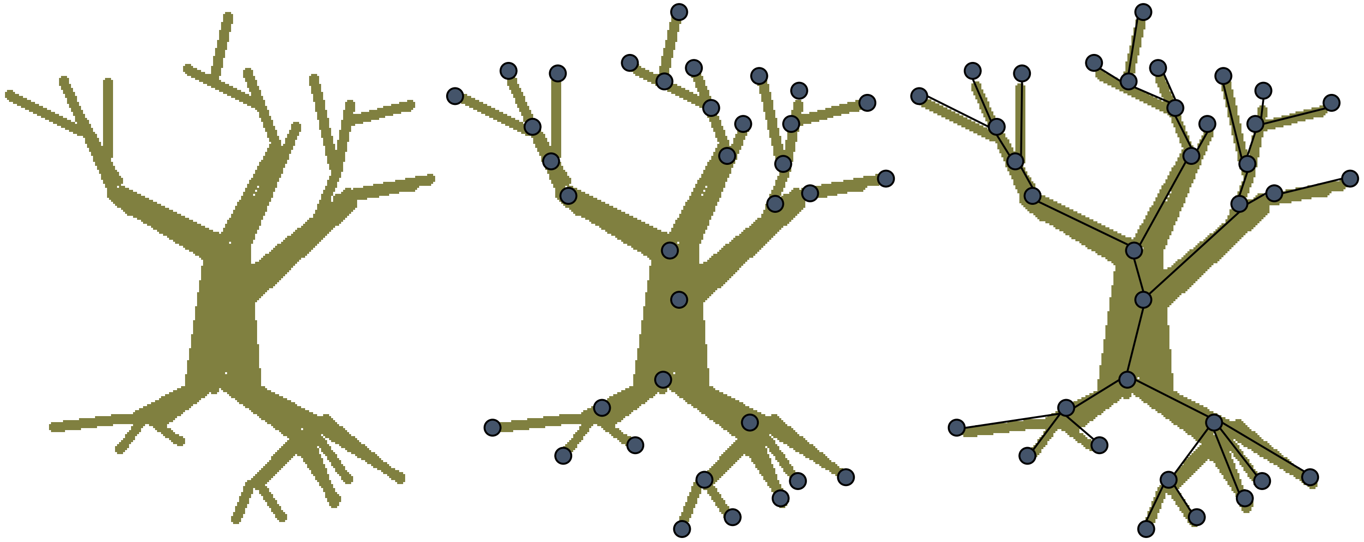
Trees

A very important type of graph in CS is called a *tree*:



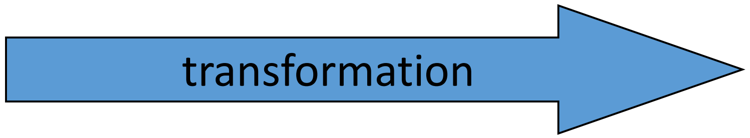
Trees

A very important type of graph in CS is called a *tree*:



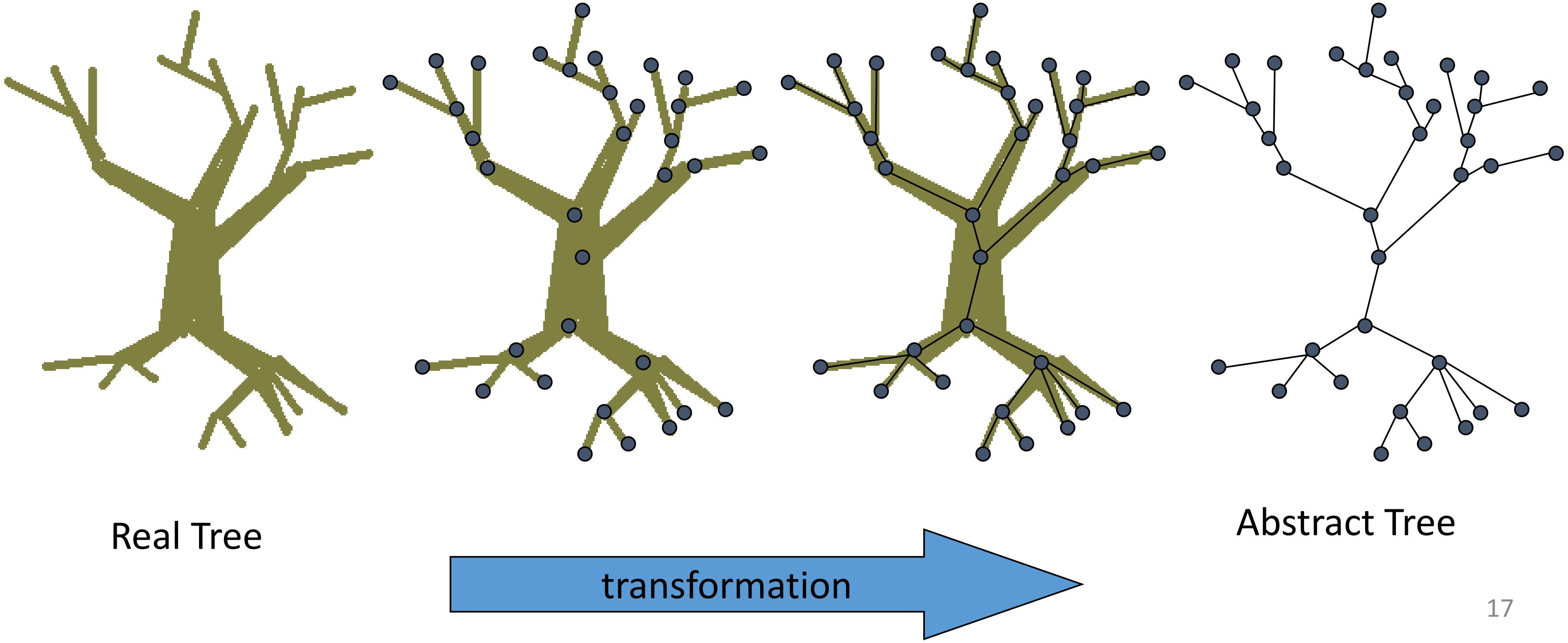
Tree

Real Tree



Trees

A very important type of graph in CS is called a *tree*:



Types Of Graph

Types of Graphs

- Empty Graph
- Null Graph
- Trivial Graph
- Directed Graphs
- Undirected Graphs
- Labeled Graphs
- Acyclic Graphs
- Cyclic Graphs
- Connected Graphs
  - Strongly Connected
  - Weakly Connected
- Complete Graphs
- Simple Graphs
- Multi-Graphs
- Pseudo-Graphs

empty graph

- **An *empty* graph** is one that does not contain any vertex. A *non-empty* graph is one that contains at least one vertex.

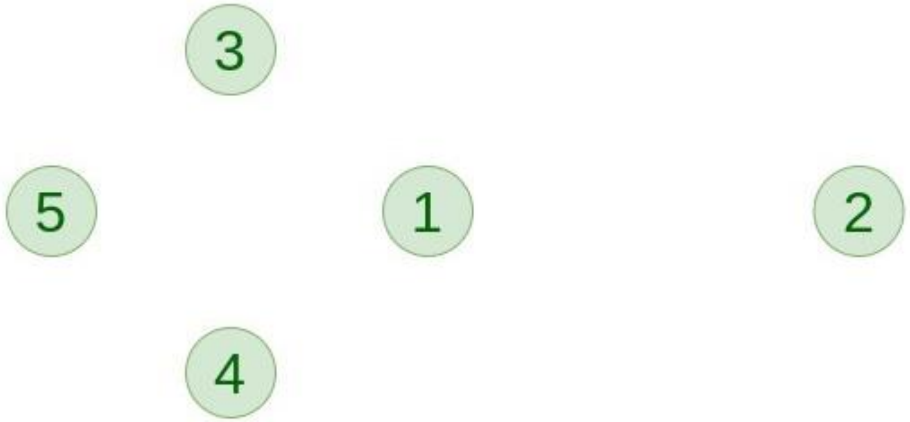
Null and Trivial Graphs

1. Null Graph

- A graph is known as a null graph if there are no edges in the graph.

2. Trivial Graph

- Graph having only a single vertex, it is also the smallest graph possible.



Null Graph

Trivial Graph

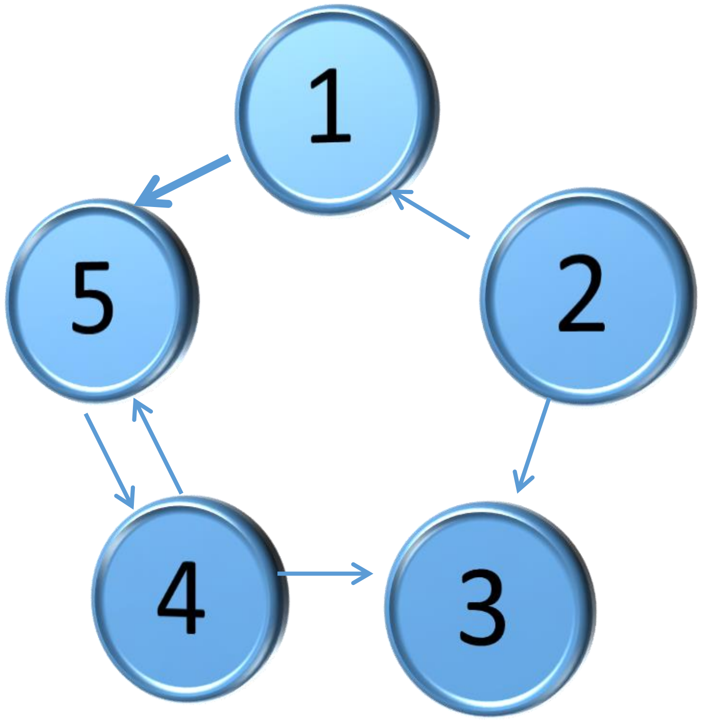
Directed GRAPH

A **directed graph** is one in which every edge (u, v) has a direction, so that (u, v) is different from (v, u). The direct pair (u, v) in directed graph is such that, u is a tail and v is the head of the edge.

There are two possible situations that can arise in a directed graph between vertices u and v.

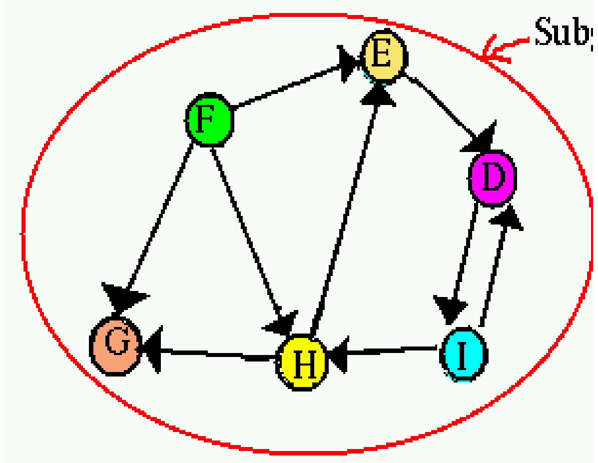
i) only one of (u, v) and (v, u) is present.

ii) both (u, v) and (v, u) are present.



Edges: {1,5}, {5,4} & {4,5}, {4,3}, {2,3} , {2,1}

Definitions

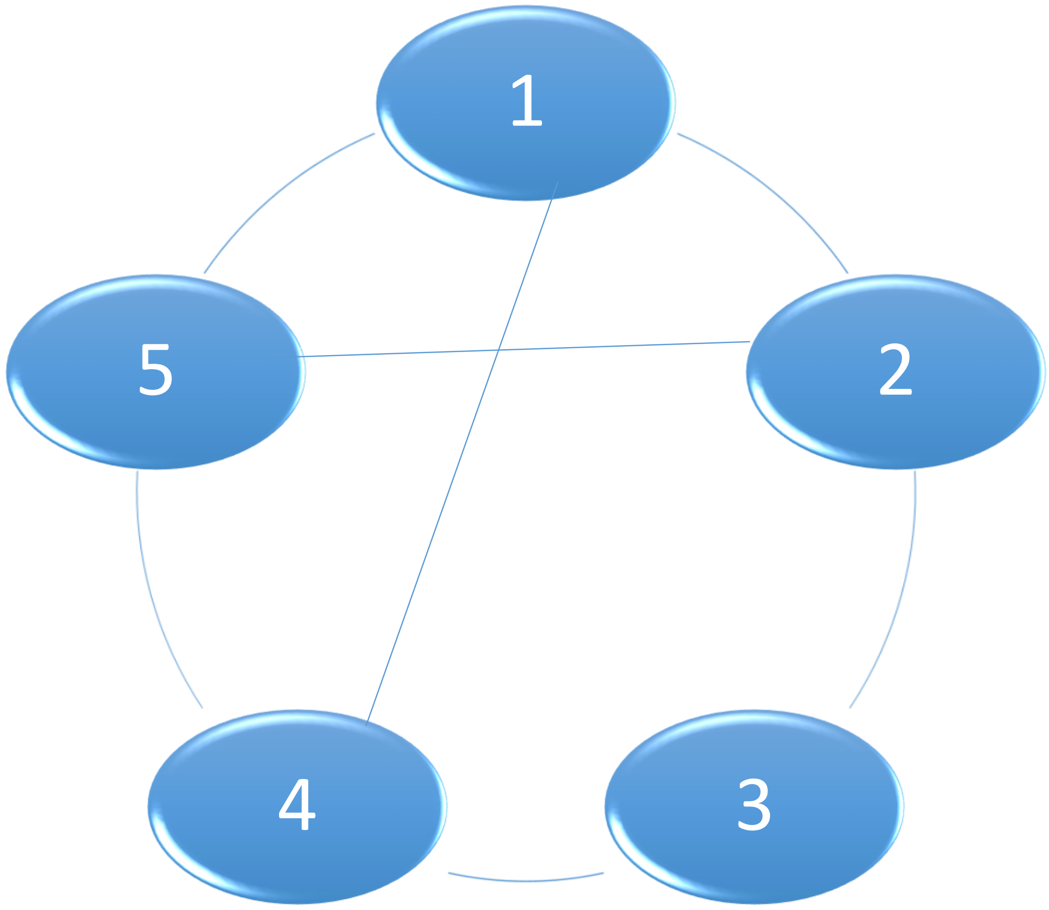


Degree of a Vertex:

The Degree of a Vertex is the number of edges incident to that vertex.

- **Indegree**: In directed graph, each edge of the vertices has indegree defined as number of edges for which v is head
- **Outdegree**: In directed graph, each edge of the vertices has outdegree defined as number of edges for which v is tail

Undirected Graph



An **Undirected graph** is one in which the edges (u, v) has no direction, so that, there is no difference between (u, v) and (v, u).

Degree of Vertex?

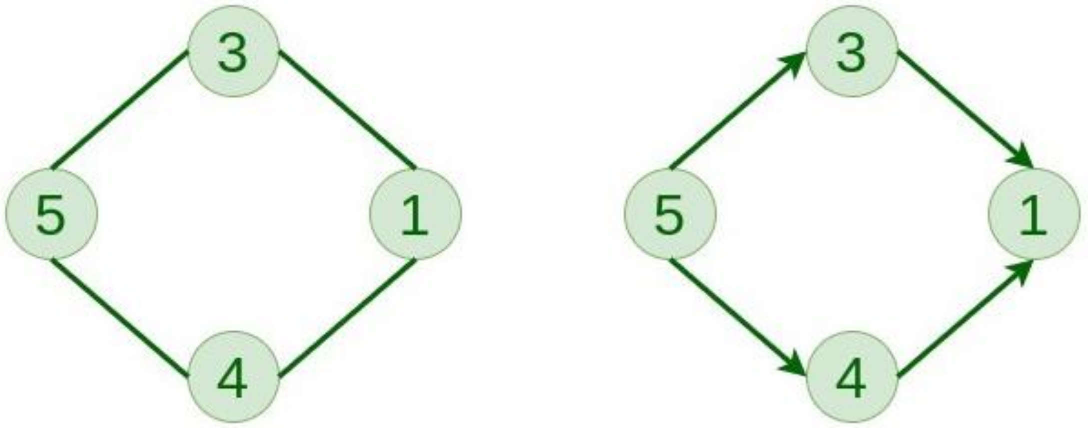
Directed and Undirected Graphs

3. Undirected Graph

- A graph in which edges do not have any direction. That is the nodes are unordered pairs in the definition of every edge.

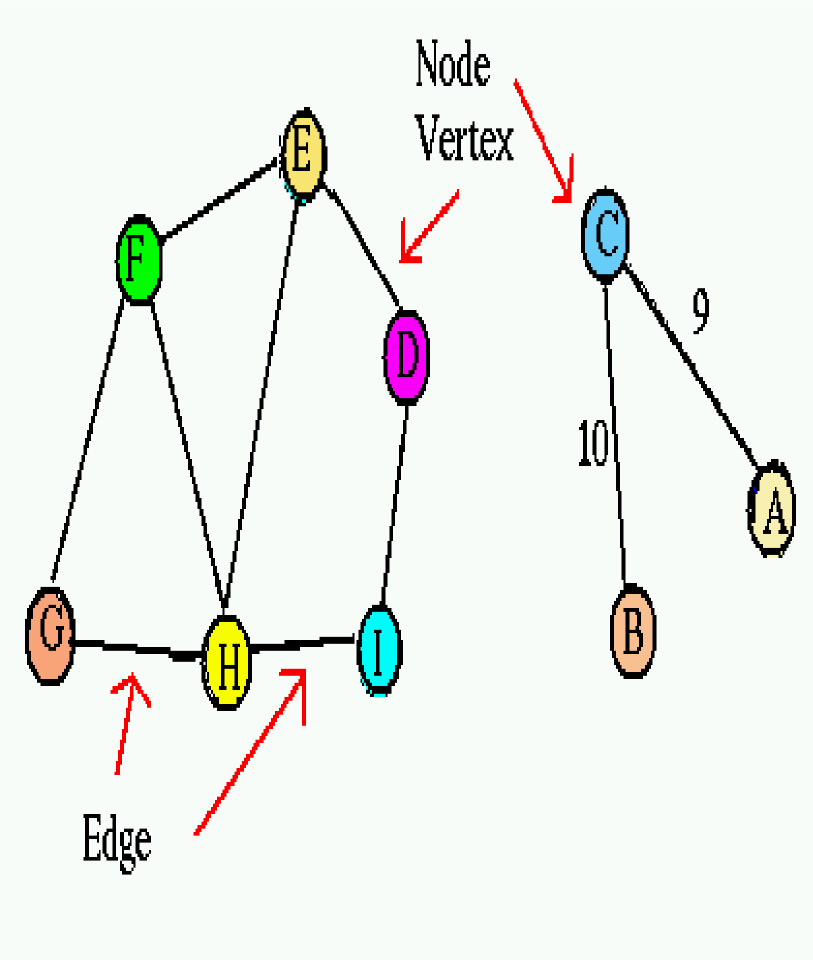
4. Directed Graph

- A graph in which edge has direction. That is the nodes are ordered pairs in the definition of every edge.



Undirected Graph Directed Graph

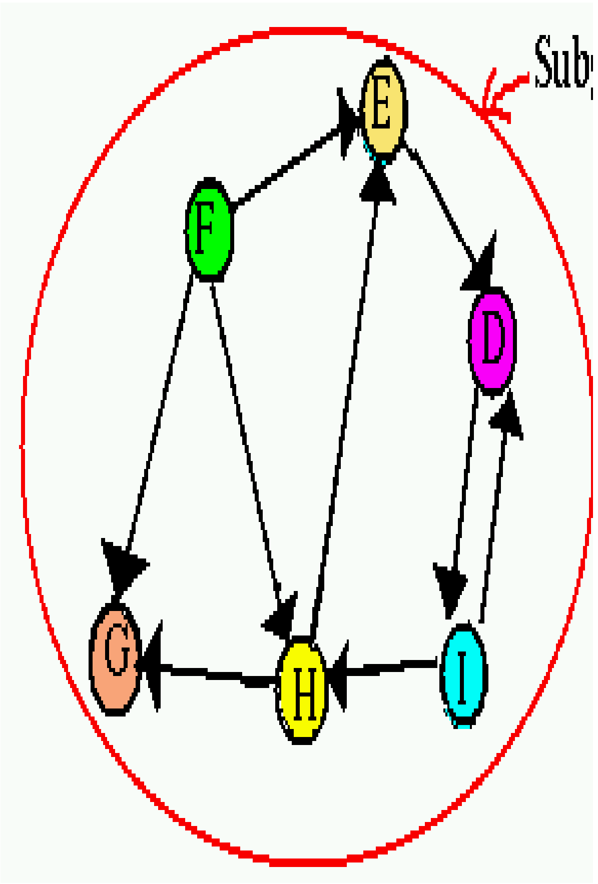
Labeled Graphs



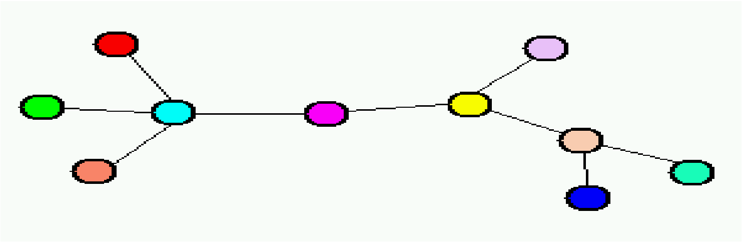
We may give edges and vertices labels. In our diagram all vertices have been labeled and two of the edges have been labeled.

Graphing applications often require the labeling of vertices and for some applications, such as in organic chemistry , the labeling of edges is also common, (to represent different types of bonds).

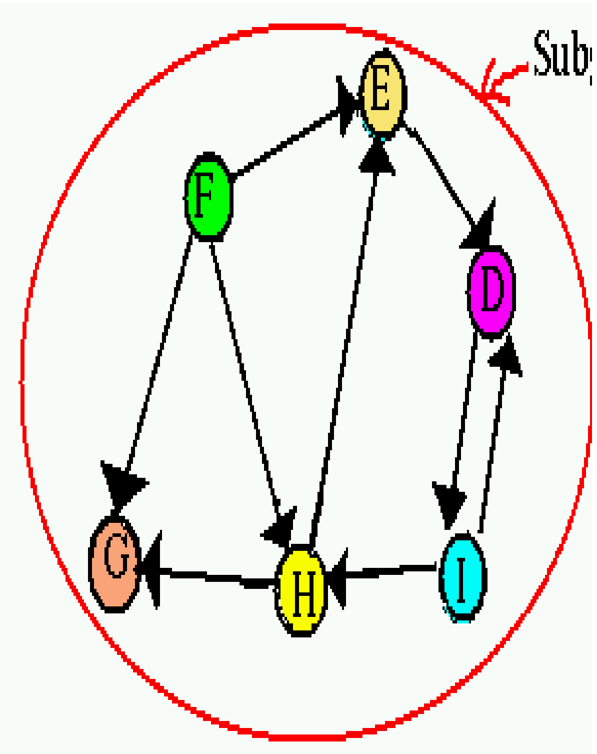
Definitions



- **Path**: Path can be defined as sequence of vertices(u1, u2, u3…..v) in such a way that there are (u1, u2),(u2, u3),…..(uk, v) edges in G(E).
- Path from E to G ??
- **Length**: the number of edges in the Path. Length of Path from E to G ?
- **Cycle**: a path that starts and ends at the same node
  - **Free Tree:** A free tree is a connected graph without a cycle.



Connected Graphs



- A ***connected* graph** is one where there is a path between any two vertices.
- In an **undirected graph** *G*, two vertices *u* and *v* are called **connected** if *G* contains a path from *u* to *v*.

  Otherwise, they are called **disconnected**.

  If the two vertices are additionally connected by a path of length 1, i.e. by a single edge, **the vertices are called adjacent.** A graph is said to be connected if every pair of vertices in the graph is connected.
- **Neighbors:** The neighbors of vertex V are all u such that u v, and (u, v) there is a direct edge E. In this case, we can also say that u and v are **adjacent.** In the graph, the adjacent of F are E, H and G.

- A graph is said to be **strongly connected** if a path exists between each of vertices of graph.

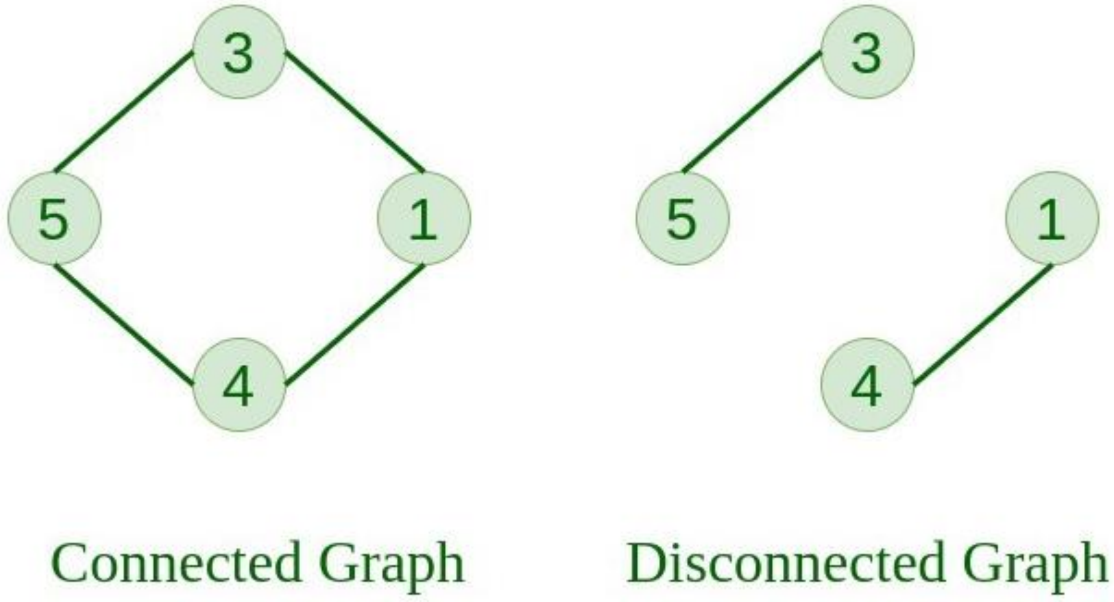
Connected and Disconnected Graphs

5. **Connected Graph**

- The graph in which from one node we can visit any other node in the graph is known as a connected graph.

6. **Disconnected Graph**

- The graph in which at least one node is not reachable from a node is known as a disconnected graph.



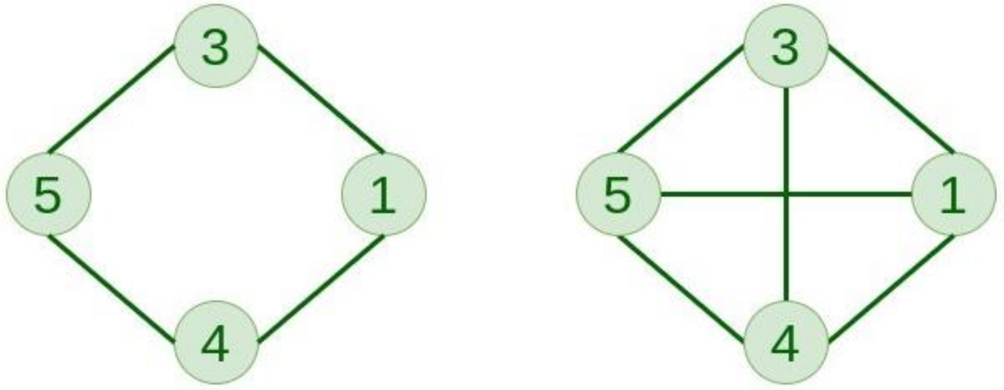
Regular and Complete Graphs

7. Regular Graph

- The graph in which the degree of every vertex is equal to K is called K regular graph.

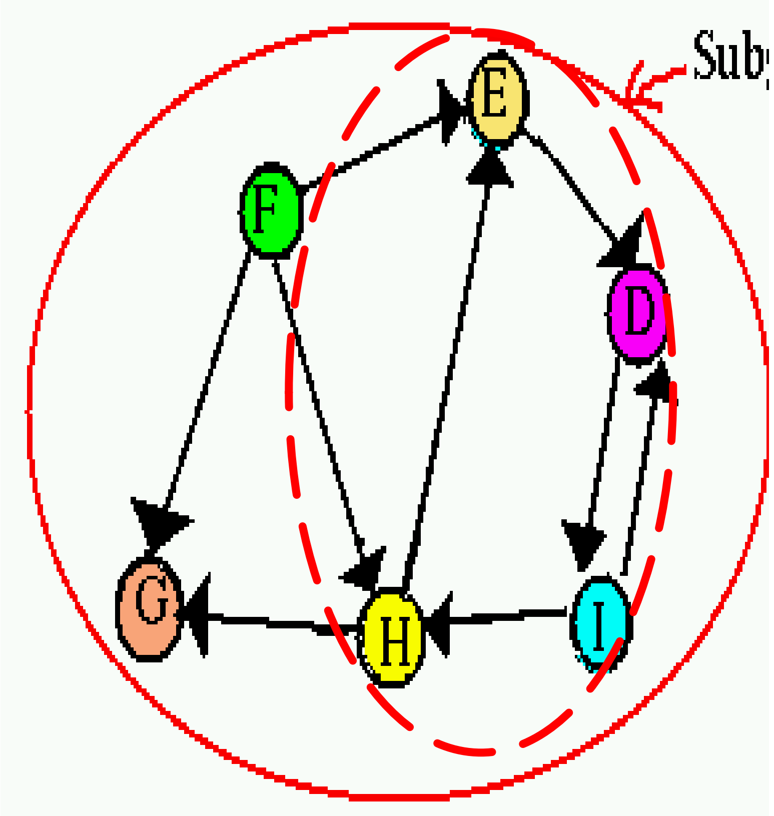
8. Complete Graph

- The graph in which from each node there is an edge to each other node.



Regular Complete Graph

cyclic graph



Directed Cyclic Graph

A **cyclic graph** or **cycle graph** or **circular graph** is a graph that consists of a single **cycle**, or in other words, some number of vertices connected in a closed chain.

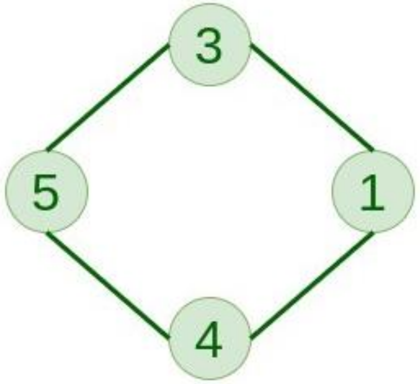
Cycle and Cyclic Graph

9. **Cycle Graph**

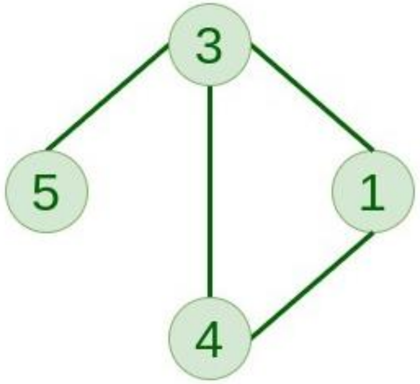
- The graph in which the graph is a cycle in itself, the degree of each vertex is 2.

10. **Cyclic Graph**

- A graph containing at least one cycle is known as a Cyclic graph.

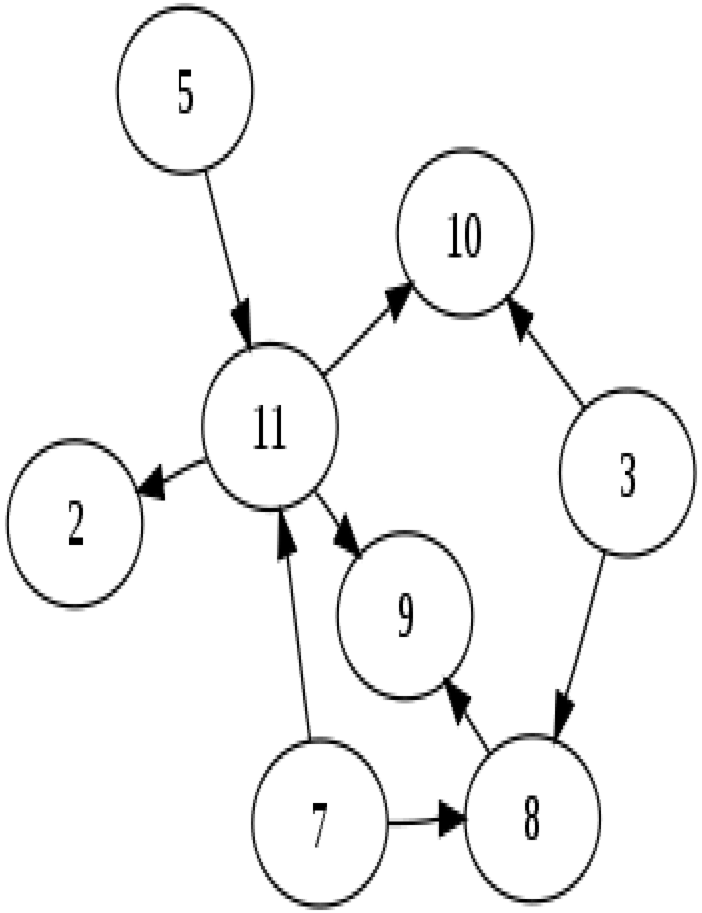


Cycle Graph



Cyclic Graph

Directed Acyclic Graph



A **directed acyclic graph** (**DAG**), is a **directed graph** with no **directed cycles**.

That is, it is formed by a collection of **vertices** and **directed edges**, each edge connecting one vertex to another, such that there is no way to start at some vertex *v* and follow a sequence of edges that eventually loops back to *v* again.

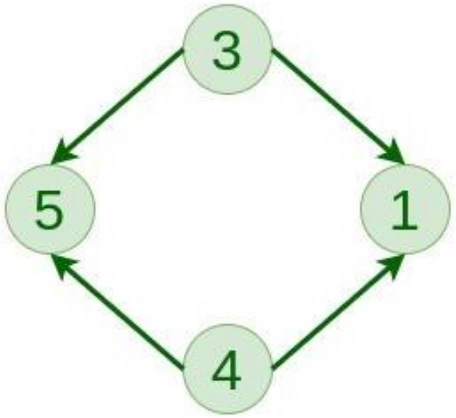
Directed Acyclic and Bipartite Graph

11. Directed Acyclic Graph

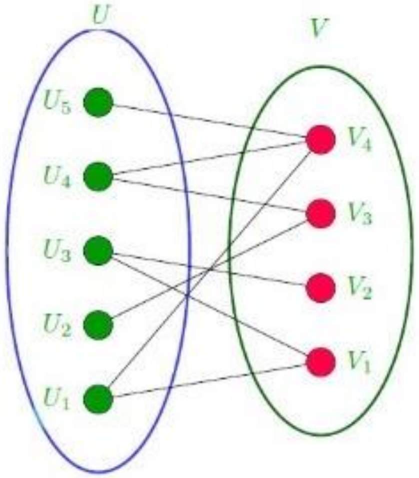
- A Directed Graph that does not contain any cycle.

12. Bipartite Graph

- A graph in which vertex can be divided into two sets such that vertex in each set does not contain any edge between them.

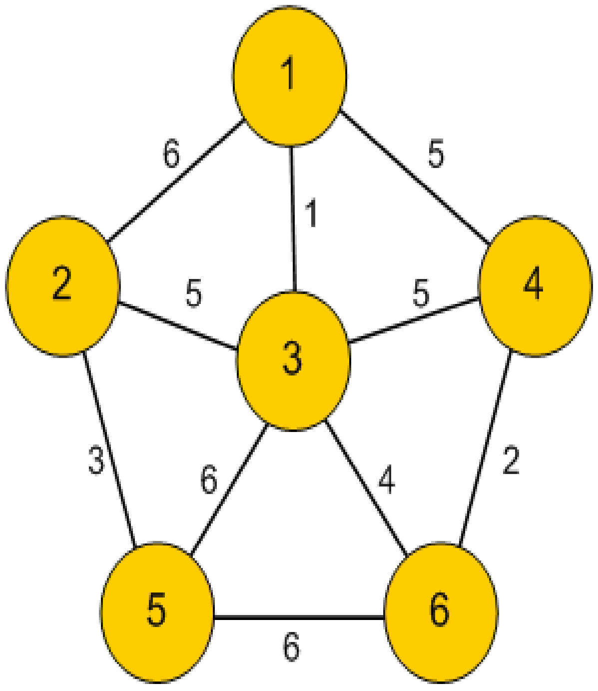


Directed Acyclic Graph



Bipartite Graph

Weighted Graphs



Edges might also be numerically labeled.

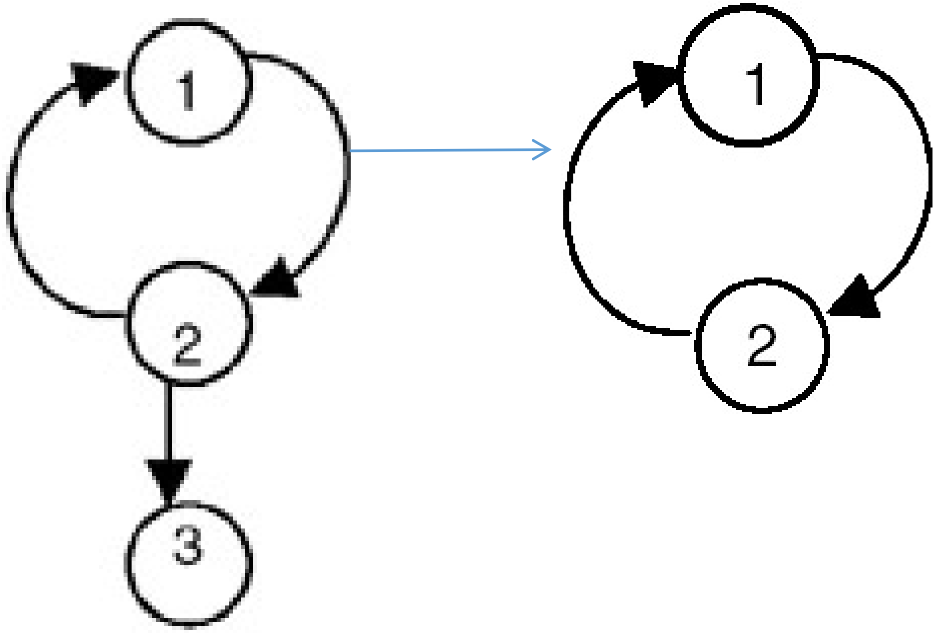
For instance if the vertices represent cities, the edges might be labeled to represent distances.

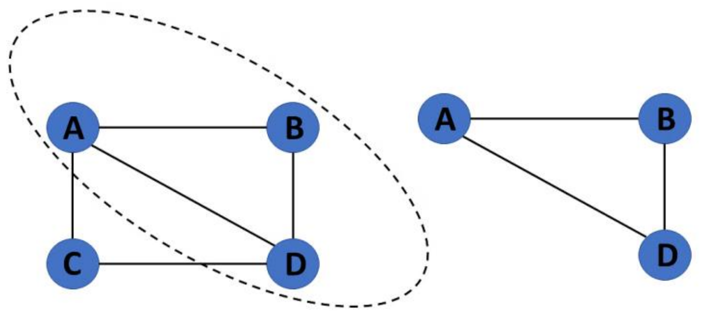
In particular, G is said to be **weighted** if each edge e in G is assigned a nonnegative numerical value w(e) called the **weight or length** of e.

In such a case, each path P in G is assigned a weight or length which is the sum of the weights of the edges along the path P.

If we are given no other information about weights, we may assume any graph G to be weighted by assigning the weight w(e) = 1 to each edge e in G.

Sub Graphs

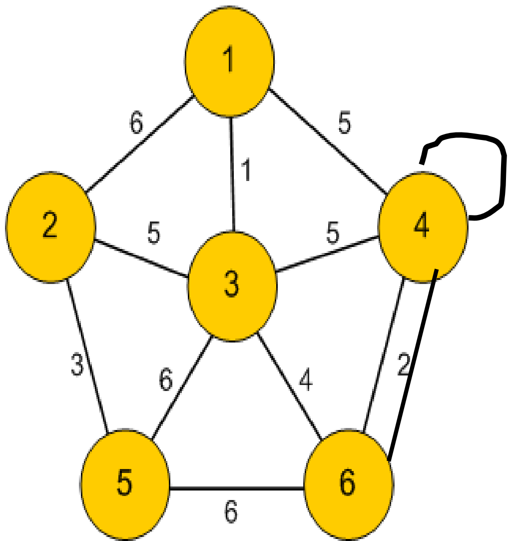




- *The vertices and edges of a graph that are subsets of another graph are known as a subgraph.*
- A graph G1 = (V1, E1) is called a subgraph of a graph G(V, E) if V1(G) is a subset of V(G) and E1(G) is a subset of E(G) such that each edge of G1 has same end vertices as in G.

Definitions

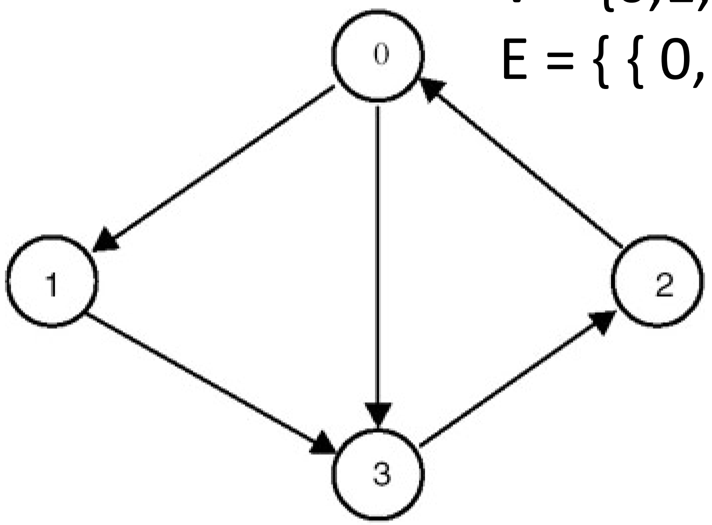
- **Cost**: the sum of weights on each edge in the path
- A ***self-loop*** is an edge whose two end-points are actually the same vertex.
- Two vertices *u* and *v* are connected by ***multiple edges*** or **Parallel Edges,** if there is more than one edge of the form (*u*,*v*) in the graph.



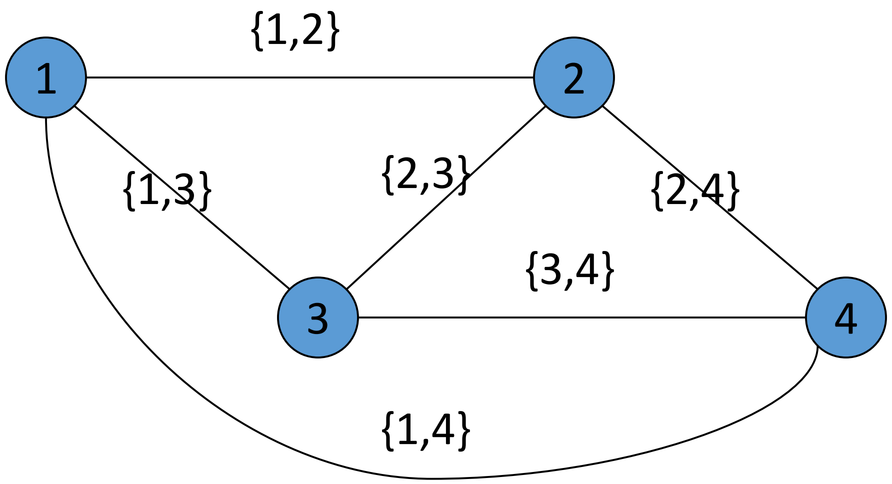
Simple Graphs

DEF: A ***simple graph*** *G* = (*V,E*) consists of a non-empty set *V* of ***vertices*** (or ***nodes***) and a set *E* (possibly empty) of ***edges*** and does not have any self-loop or parallel edges.

V = {0,1,2,3}
E = { { 0,1} , {0,3} , {1,3}, {3,2}, {2,0}}



Simple Graphs



- Vertices are labeled to associate with particular computers
- Each edge can be viewed as the set of its two endpoints

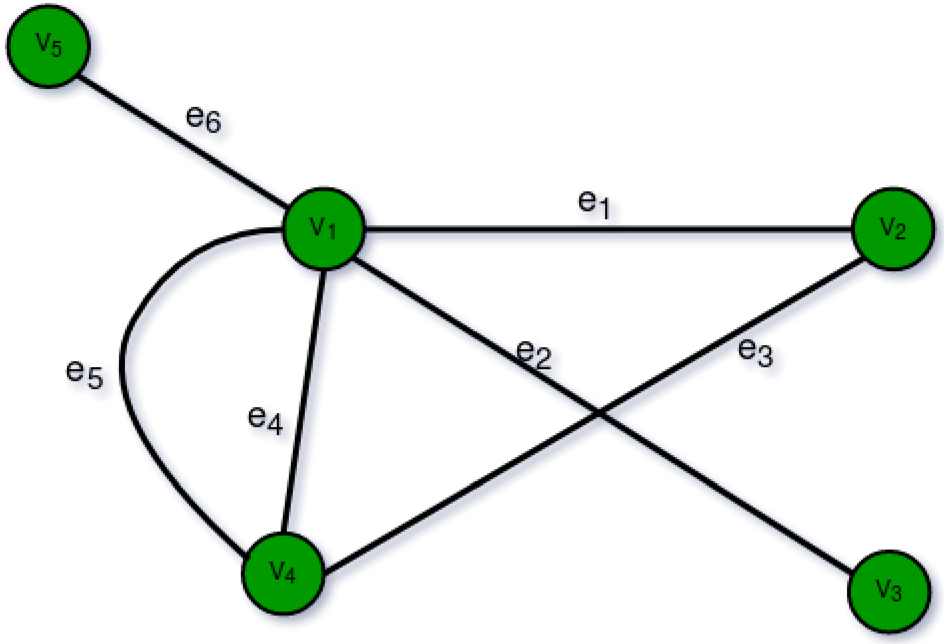
Simple Graphs

Different purposes require different types of graphs.

For Example: Suppose a local computer network

- Is bidirectional (undirected)
- Has no loops (no “self-communication”)
- Has unique connections between computers

Multigraphs



If computers are connected via internet instead of directly, there may be several routes to choose from for each connection. Depending on traffic, one route could be better than another. Makes sense to allow multiple edges, but still no self-loops:

Multi Graphs: Any graph which contains some parallel edges but doesn't contain any self-loop is called a multigraph. For example a Road Map.

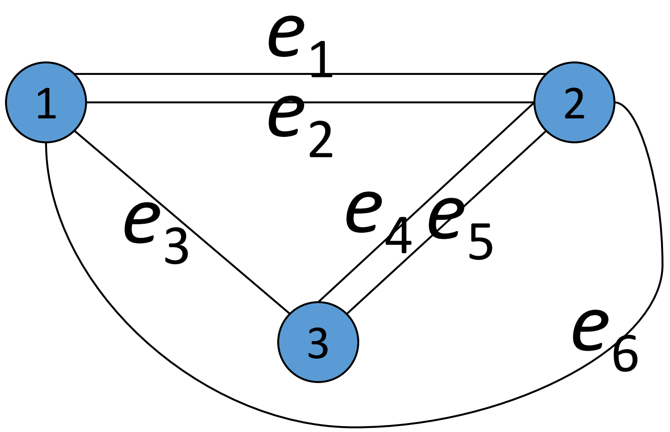
Multigraphs

DEF: A multi-graph or pseudograph is a graph which is permitted to have multiple edges, (also called "parallel edges"),

Two Types:

1. **Undirected multigraph (edges without own identity)**
2. **Directed multigraph (edges without own identity)**

Multigraphs

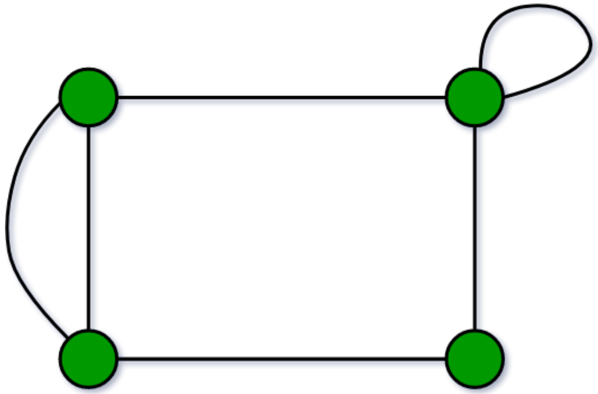


Edge-labels distinguish between edges sharing same endpoints. Labeling can be thought of as function:

$e_1 \rightarrow \{1,2\}$, $e_2 \rightarrow \{1,2\}$, $e_3 \rightarrow \{1,3\}$,

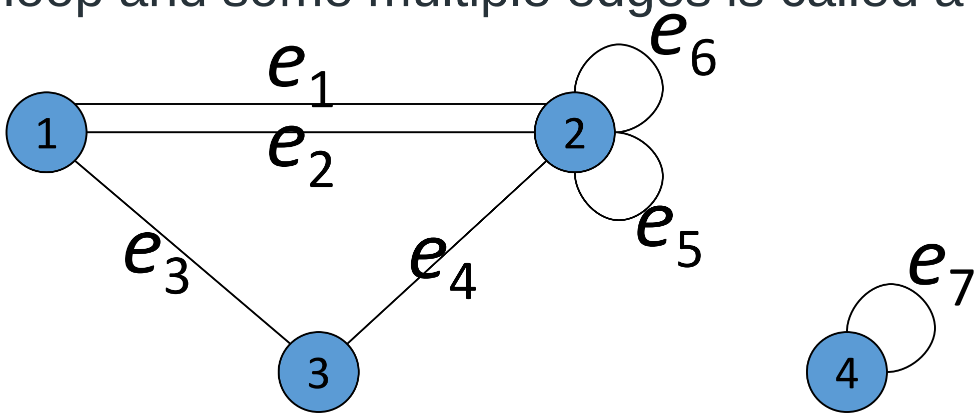
$e_4 \rightarrow \{2,3\}$, $e_5 \rightarrow \{2,3\}$, $e_6 \rightarrow \{1,2\}$

Pseudographs



If self-loops are allowed we get a pseudograph:

A graph G with a self-loop and some multiple edges is called a pseudo graph.



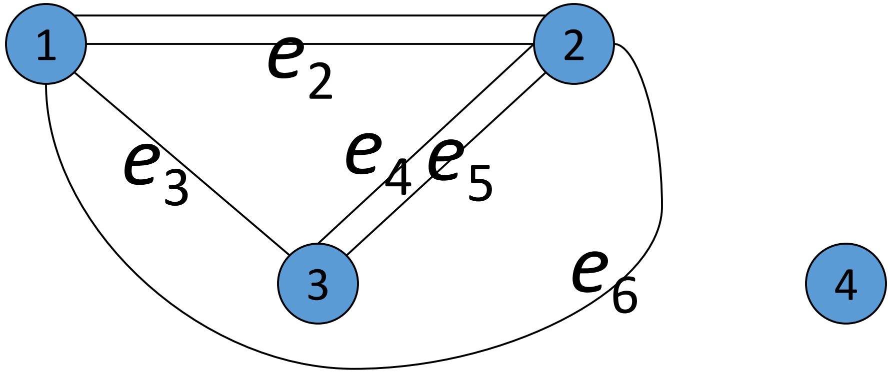
Now edges may be associated with a single vertex, when the edge is a loop

$e_1 \rightarrow \{1,2\}$, $e_2 \rightarrow \{1,2\}$, $e_3 \rightarrow \{1,3\}$,

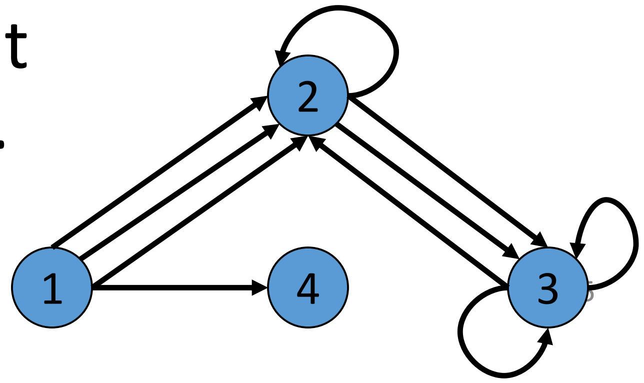
$e_4 \rightarrow \{2,3\}$, $e_5 \rightarrow \{2\}$, $e_6 \rightarrow \{2\}$, $e_7 \rightarrow \{4\}$

Definitions

- **Isolated Vertex**: A vertex V is called an isolated vertex if there is no edge connected from any other vertex to the vertex V_i



- **Pendent Vertex:** A vertex V is called an pendent vertex if its Indegree = 1 and its OutDegree = 0 .



Graphs

Advantages of graphs:

- Graphs can be used to model and analyze complex systems and relationships.
- They are useful for visualizing and understanding data.
- Graph algorithms are widely used in computer science and other fields, such as social network analysis, logistics, and transportation.
- Graphs can be used to represent a wide range of data types, including social networks, road networks, and the internet.

Disadvantages of graphs:

- Large graphs can be difficult to visualize and analyze.
- Graph algorithms can be computationally expensive, especially for large graphs.
- The interpretation of graph results can be subjective and may require domain-specific knowledge.
- Graphs can be susceptible to noise and outliers, which can impact the accuracy of analysis results.