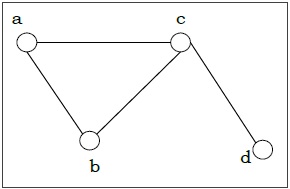
**UNIT-1**

**What is a Graph?**

**Definition** − A graph (denoted as G=(V,E)G=(V,E)) consists of a non-empty set of vertices or nodes V and a set of edges E.

**Example** − Let us consider, a Graph is G=(V,E)G=(V,E) where V={a,b,c,d}V={a,b,c,d} and E={{a,b},{a,c},{b,c},{c,d}}E={{a,b},{a,c},{b,c},{c,d}}

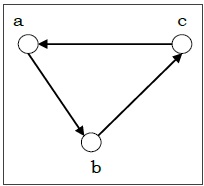
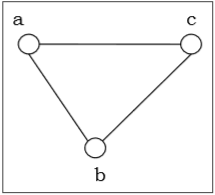


**Types of graph:-**

There are various types of graphs depending upon the number of vertices, number of edges, interconnectivity, and their overall structure.

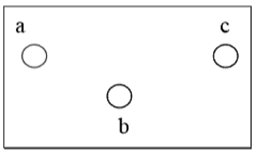
**Directed and Undirected Graph**

A graph G=(V,E)G=(V,E) is called a directed graph if the edge set is made of ordered vertex pair and a graph is called undirected if the edge set is made of unordered vertex pair.



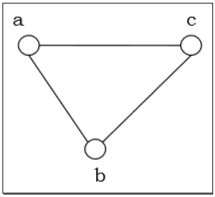
**Null Graph**

A null graph has no edges. The null graph of nn vertices is denoted by NnNn



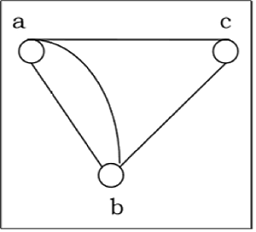
**Simple Graph**

A graph is called simple graph/strict graph if the graph is undirected and does not contain any loops or multiple edges.



**Multi-Graph**

If in a graph multiple edges between the same set of vertices are allowed, it is called Multigraph. In other words, it is a graph having at least one loop or multiple edges.



**Isomorphism**

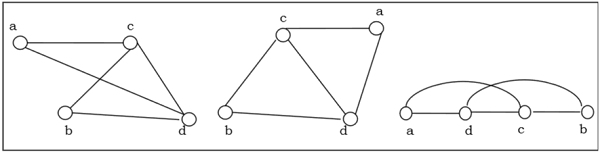
If two graphs G and H contain the same number of vertices connected in the same way, they are called isomorphic graphs (denoted by G≅HG≅H).

It is easier to check non-isomorphism than isomorphism. If any of these following conditions occurs, then two graphs are non-isomorphic −

* The number of connected components are different
* Vertex-set cardinalities are different
* Edge-set cardinalities are different
* Degree sequences are different

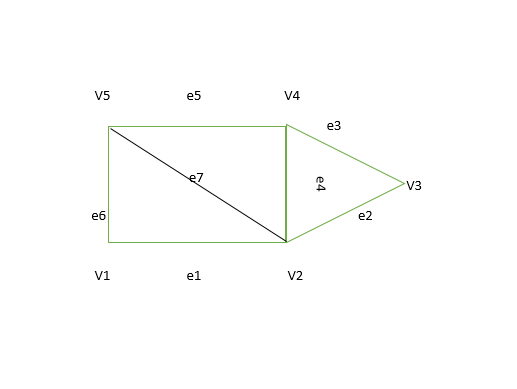
Example

The following graphs are isomorphic −

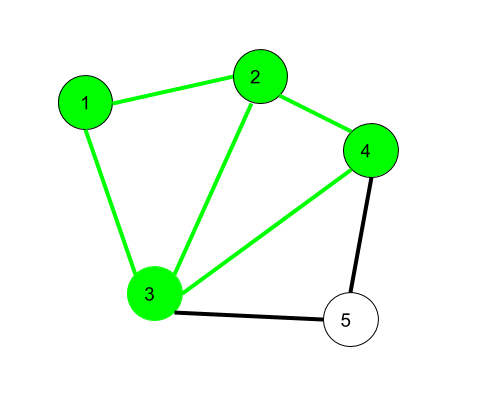


**Subgraph:**

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.



**Walk –**   
A walk is a sequence of vertices and edges of a graph i.e. if we traverse a graph then we get a walk.   
Note: Vertices and Edges can be repeated.



Here, 1->2->3->4->2->1->3 is a walk.

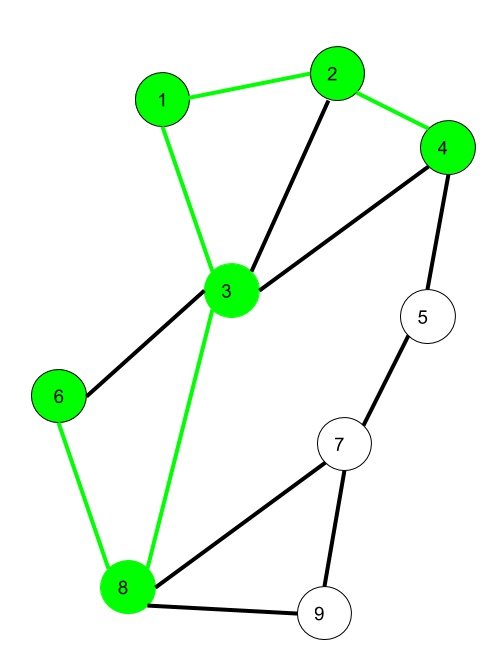
Walk can be open or closed.

**Open walk-**A walk is said to be an open walk if the starting and ending vertices are different i.e. the origin vertex and terminal vertex are different.   
**Closed walk-**A walk is said to be a closed walk if the starting and ending vertices are identical i.e. if a walk starts and ends at the same vertex, then it is said to be a closed walk.

In the above diagram:   
1->2->3->4->5->3 is an open walk.   
1->2->3->4->5->3->1 is a closed walk.

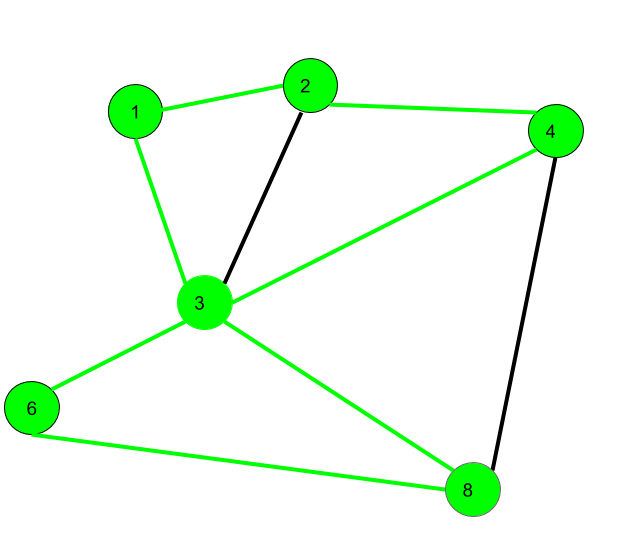
**Path –**   
It is a trail in which neither vertices nor edges are repeated i.e. if we traverse a graph such that we do not repeat a vertex and nor we repeat an edge. As path is also a trail, thus it is also an open walk.

Vertex not repeated  … Edge not repeated



Here 6->8->3->1->2->4 is a Path

**Circuit –**   
Traversing a graph such that not an edge is repeated but vertex can be repeated and it is closed also i.e. it is a closed trail.   
 Vertex can be repeated. And Edge can not be repeated.

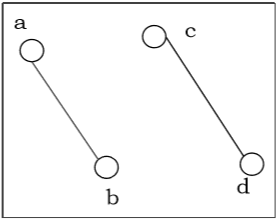
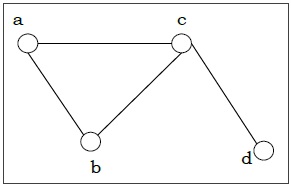


Here 1->2->4->3->6->8->3->1 is a circuit.

Circuit is a closed trail. These can have repeated vertices only.

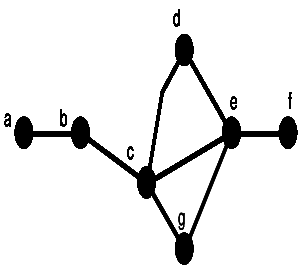
**Connected and Disconnected Graph**

A graph is connected if any two vertices of the graph are connected by a path; while a graph is disconnected if at least two vertices of the graph are not connected by a path. If a graph G is disconnected, then every maximal connected subgraph of GG is called a connected component of the graph GG.



**Components**

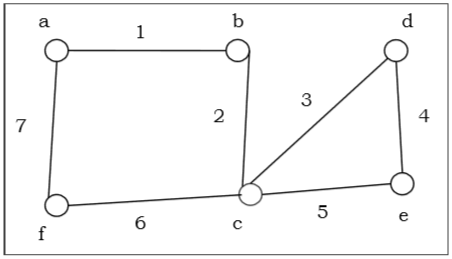
A component of a graph is defined as a maximal subgraph in which a path exists from every node to every other (i.e., they are mutually reachable). The size of a component is defined as the number of nodes it contains. A connected graph has only one component.



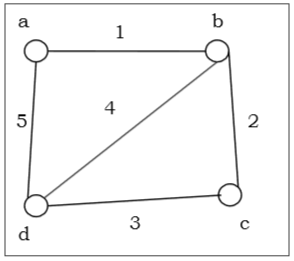
**Euler Graphs**

A connected graph GG is called an Euler graph, if there is a closed trail which includes every edge of the graph GG. An Euler path is a path that uses every edge of a graph exactly once. An Euler path starts and ends at different vertices.

An Euler circuit is a circuit that uses every edge of a graph exactly once. An Euler circuit always starts and ends at the same vertex. A connected graph GG is an Euler graph if and only if all vertices of GG are of even degree, and a connected graph GG is Eulerian if and only if its edge set can be decomposed into cycles.



The above graph is an Euler graph as “a1b2c3d4e5c6f7g”“a1b2c3d4e5c6f7g” covers all the edges of the graph.



**Hamiltonian paths and circuits :**

**Hamiltonian Path –** A simple path in a graph that passes through **every vertex exactly once** is called a Hamiltonian path.

**Hamiltonian Circuit –** A simple circuit in a graph that passes through every vertex exactly once is called a Hamiltonian circuit.

Unlike Euler paths and circuits, there is no simple necessary and sufficient criteria to determine if there are any Hamiltonian paths or circuits in a graph. But there are certain criteria which rule out the existence of a Hamiltonian circuit in a graph, such as- if there is a vertex of degree one in a graph then it is impossible for it to have a Hamiltonian circuit.   
There are certain theorems which give sufficient but not necessary conditions for the existence of Hamiltonian graphs.

**Tree**

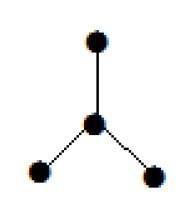
A **connected acyclic graph** is called a tree. In other words, a connected graph with no cycles is called a tree.

The edges of a tree are known as **branches**. Elements of trees are called their nodes. The nodes without child nodes are called **leaf nodes**.

A tree with ‘n’ vertices has ‘n-1’ edges. If it has one more edge extra than ‘n-1’, then the extra edge should obviously has to pair up with two vertices which leads to form a cycle. Then, it becomes a cyclic graph which is a violation for the tree graph.

**Example 1**

The graph shown here is a tree because it has no cycles and it is connected. It has four vertices and three edges, i.e., for ‘n’ vertices ‘n-1’ edges as mentioned in the definition.



**Note** − Every tree has at least two vertices of degree one.

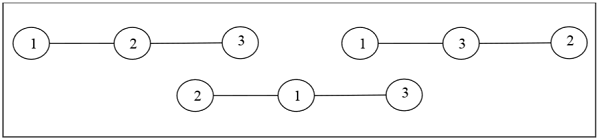
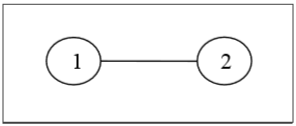
* **Centers and Bi-Centers of a Tree**

The center of a tree is a vertex with minimal eccentricity. The eccentricity of a vertex X in a tree G is the maximum distance between the vertex X and any other vertex of the tree. The maximum eccentricity is the tree diameter. If a tree has only one center, it is called Central Tree and if a tree has only more than one centers, it is called Bi-central Tree. Every tree is either central or bi-central.

* **Labeled Trees**

**Definition** − A labeled tree is a tree the vertices of which are assigned unique numbers from 1 to n. We can count such trees for small values of n by hand so as to conjecture a general formula. The number of labeled trees of n number of vertices is nn-2. Two labeled trees are isomorphic if their graphs are isomorphic and the corresponding points of the two trees have the same labels.

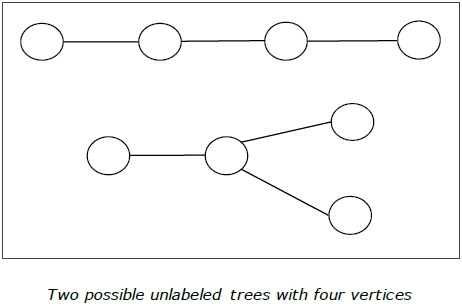
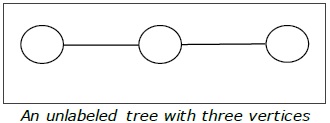
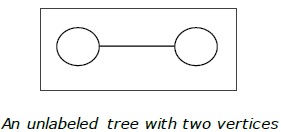
Example



* **Unlabeled Trees**

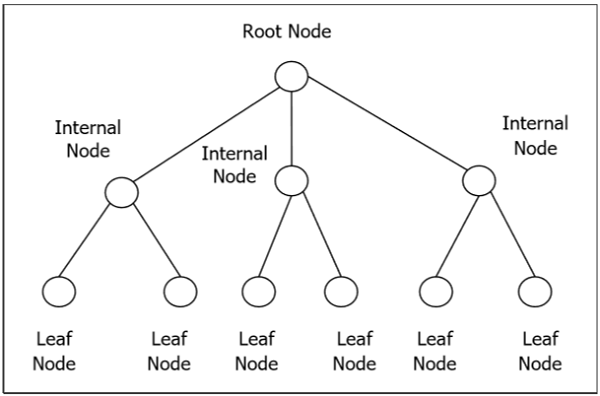
**Definition** − An unlabeled tree is a tree the vertices of which are not assigned any numbers. The number of labeled trees of n number of vertices is (2n)!(n+1)!n!(2n)!(n+1)!n! (nth Catalan number)

Example



* **Rooted Tree**

A rooted tree G is a connected acyclic graph with a special node that is called the root of the tree and every edge directly or indirectly originates from the root. An ordered rooted tree is a rooted tree where the children of each internal vertex are ordered. If every internal vertex of a rooted tree has not more than m children, it is called an m-ary tree. If every internal vertex of a rooted tree has exactly m children, it is called a full m-ary tree. If m = 2, the rooted tree is called a binary tree.



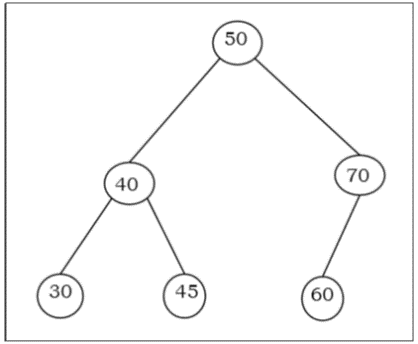
* **Binary Search Tree**

Binary Search tree is a binary tree which satisfies the following property −

* X in left sub-tree of vertex V, Value(X) ≤ Value (V)
* Y in right sub-tree of vertex V, Value(Y) ≥ Value (V)

So, the value of all the vertices of the left sub-tree of an internal node V are less than or equal to V and the value of all the vertices of the right sub-tree of the internal node V are greater than or equal to V. The number of links from the root node to the deepest node is the height of the Binary Search Tree.

Example

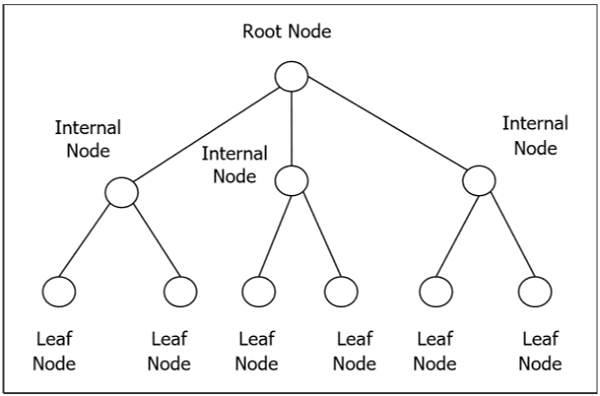


**Distance and Centers in Trees**

The center of a tree is a vertex with minimal eccentricity. The eccentricity of a vertex X in a tree G is the maximum distance between the vertex X and any other vertex of the tree. The maximum eccentricity is the tree diameter. If a tree has only one center, it is called Central Tree and if a tree has only more than one centers, it is called Bi-central Tree. Every tree is either central or bi-central.

**Rooted Tree**

A rooted tree ***G*** is a connected acyclic graph with a special node that is called the root of the tree and every edge directly or indirectly originates from the root. An ordered rooted tree is a rooted tree where the children of each internal vertex are ordered. If every internal vertex of a rooted tree has not more than m children, it is called an m-ary tree. If every internal vertex of a rooted tree has exactly m children, it is called a full m-ary tree. If ***m = 2***, the rooted tree is called a binary tree.



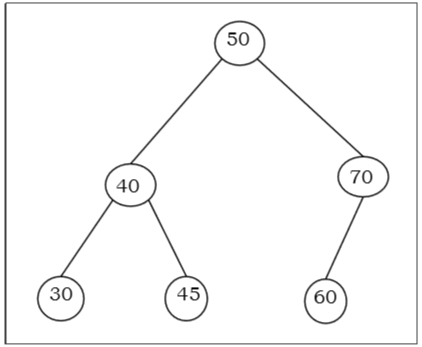
**Binary Search Tree**

The binary Search tree is a binary tree which satisfies the following property −

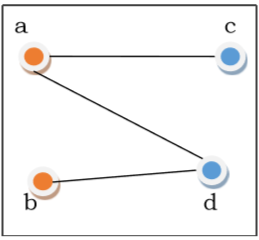
* X in left sub-tree of vertex V, Value(X) ≤ Value (V)
* Y in right sub-tree of vertex V, Value(Y) ≥ Value (V)

So, the value of all the vertices of the left sub-tree of an internal node ***V*** are less than or equal to ***V*** and the value of all the vertices of the right sub-tree of the internal node ***V*** are greater than or equal to ***V***. The number of links from the root node to the deepest node is the height of the Binary Search Tree.

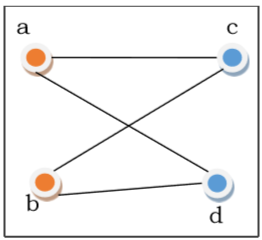
Example



**Bipartite Graph** - If the vertex-set of a graph G can be split into two disjoint sets, V1 and V2 , in such a way that each edge in the graph joins a vertex in V1 to a vertex in V2 , and there are no edges in G that connect two vertices in V1 or two vertices in V2 , then the graph G is called a bipartite graph.



**Complete Bipartite Graph** - A complete bipartite graph is a bipartite graph in which each vertex in the first set is joined to every single vertex in the second set. The complete bipartite graph is denoted by Kx,y where the graph G contains x vertices in the first set and y vertices in the second set.



**What Is a Line Graph?**

A line graph—also known as a line plot or a line chart—is a graph that uses lines to connect individual data points. A line graph displays quantitative values over a specified time interval. In finance, line graphs are commonly used to depict the historical price action of an asset or security.

Line graphs can be compared with other visualizations of data including [bar charts](https://www.investopedia.com/terms/b/barchart.asp), pie charts, and (in trading) [candlestick charts](https://www.investopedia.com/terms/c/candlestick.asp), among others.

**Chordal Graph**

**chordal graph** is one in which all [cycles](https://en.wikipedia.org/wiki/Cycle_(graph_theory)) of four or more [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) have a *chord*, which is an [edge](https://en.wikipedia.org/wiki/Edge_(graph_theory)) that is not part of the cycle but connects two vertices of the cycle. Equivalently, every [induced cycle](https://en.wikipedia.org/wiki/Induced_cycle) in the graph should have exactly three vertices. The chordal graphs may also be characterized as the graphs that have perfect elimination orderings, as the graphs in which each minimal separator is a clique, and as the [intersection graphs](https://en.wikipedia.org/wiki/Intersection_graph) of subtrees of a tree. They are sometimes also called **rigid circuit graphs**[[1]](https://en.wikipedia.org/wiki/Chordal_graph#cite_note-dirac-1) or **triangulated graphs.**