Applications of Graph Theory

Graph Theory is used in vast area of science and technologies. Some of them are given below:

1. Computer Science

In **computer science** graph theory is used for the **study of algorithms** like:

- Dijkstra's Algorithm
- Prims's Algorithm
- Kruskal's Algorithm
- Graphs are used to define the **flow of computation**.
- Graphs are used to represent **networks of communication**.
- Graphs are used to represent **data organization**.
- Graph transformation systems work on rule-based in-memory manipulation of graphs. Graph databases ensure **transaction-safe, persistent storing and querying of graph structured data**.
- Graph theory is used to find **shortest path in road** or a network.
- In **Google Maps**, various locations are represented as vertices or nodes and the roads are represented as edges and graph theory is used to find the shortest path between two nodes.

2. Electrical Engineering

In **Electrical Engineering**, graph theory is used in **designing of circuit connections**. These circuit connections are named as topologies. Some topologies are series, bridge, star and parallel topologies.

3. Linguistics

- In **linguistics**, graphs are mostly used for **parsing of a language tree** and **grammar of a language tree**.
- Semantics networks are used within **lexical semantics**, especially as applied to computers, modeling word meaning is easier when a given word is understood in terms of related words.
- Methods in **phonology** (e.g. theory of optimality, which uses lattice graphs) and **morphology** (e.g. morphology of finite - state, using finite-state transducers) are common in the analysis of language as a graph.

4. Physics and Chemistry

- In physics and chemistry, graph theory is used to **study molecules**.
- The **3D structure of complicated simulated atomic structures** can be studied quantitatively by gathering statistics on graph-theoretic properties related to the topology of the atoms.
- **Statistical physics** also uses graphs. In this field graphs can represent local connections between interacting parts of a system, as well as the dynamics of

a physical process on such systems.

- Graphs are also used to express **the micro-scale channels** of porous media, in which the vertices represent the pores and the edges represent the smaller channels connecting the pores.
- Graph is also helpful in **constructing the molecular structure** as well as lattice of the molecule. It also helps us to show the bond relation in between atoms and molecules, also help in comparing structure of one molecule to other.

5. Computer Network

- In computer network, the **relationships among interconnected computers** within the network, follow the principles of graph theory.
- Graph theory is also used in **network security**.
- We can use the vertex coloring algorithm to find a proper **coloring of the map** with four colors.
- Vertex coloring algorithm may be used for assigning at most four different frequencies for any **GSM (Grouped Special Mobile) mobile phone networks**.

6. Social Sciences

- Graph theory is also used in sociology. For example, to explore rumor spreading, or to measure actors' prestige notably through the use of social network analysis software.
- Acquaintanceship and friendship graphs describe whether people know each other or not.
- In influence graphs model, certain people can influence the behavior of others.
- In collaboration graphs model to check whether two people work together in a particular way, such as acting in a movie together.

7. Biology

- Nodes in biological networks represent bimolecular such as genes, proteins or metabolites, and edges connecting these nodes indicate functional, physical or chemical interactions between the corresponding bimolecular.
- Graph theory is used in transcriptional regulation networks.
- It is also used in Metabolic networks.
- In PPI (Protein - Protein interaction) networks graph theory is also useful.
- Characterizing drug - drug target relationships.

8. Mathematics

In mathematics, operational research is the important field. Graph theory provides many useful applications in operational research. Like:

- Minimum cost path.
- A scheduling problem.

9. General

Graphs are used to represent the routes between the cities. With the help of tree that is a type of graph, we can create hierarchical ordered information such as **family tree**.

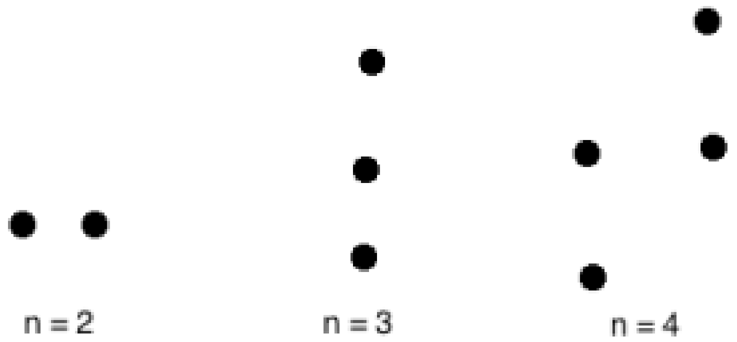
Types of Graphs

Though, there are a lot of different types of graphs depending upon the number of vertices, number of edges, interconnectivity, and their overall structure, some of such common types of graphs are as follows:

1. Null Graph

A **null graph** is a graph in which there are no edges between its vertices. A null graph is also called empty graph.

Example



A null graph with n vertices is denoted by Nn.

2. Trivial Graph

A **trivial graph** is the graph which has only one verte+x.

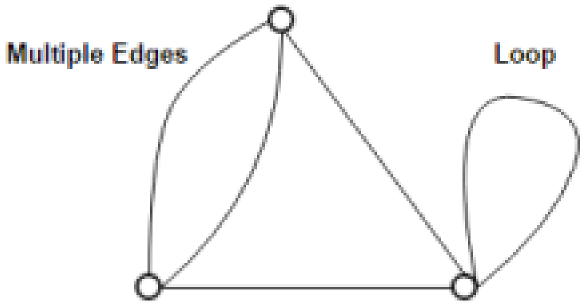
Example

In the above graph, there is only one vertex 'v' without any edge. Therefore, it is a trivial graph.

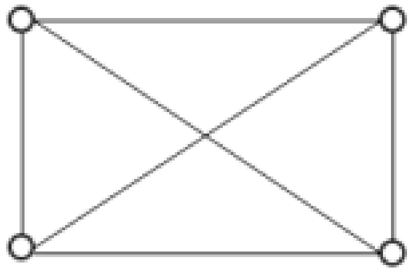
3. Simple Graph

A **simple graph** is the undirected graph with **no parallel edges** and **no loops**. A simple graph which has n vertices, the degree of every vertex is at most n -1.

Example



Not a Simple Graph



Simple Graph

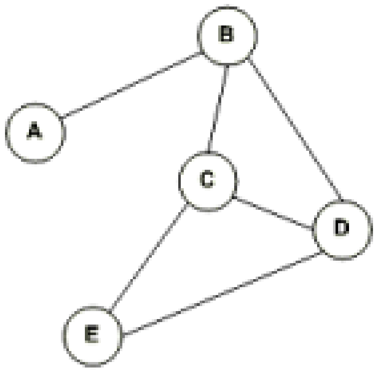
In the above example, First graph is not a simple graph because it has two edges between the vertices A and B and it also has a loop.

Second graph is a simple graph because it does not contain any loop and parallel edges.

4. Undirected Graph

An **undirected graph** is a graph whose edges are **not directed**.

Example



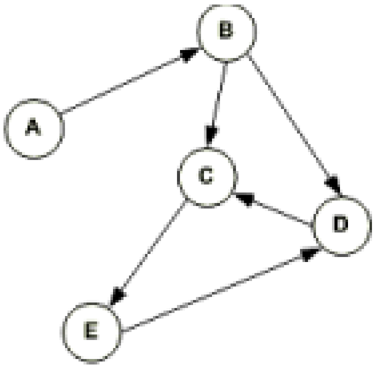
In the above graph since there is no directed edges, therefore it is an undirected graph.

5. Directed Graph

A **directed graph** is a graph in which the **edges are directed** by

arrows. Directed graph is also known as **digraphs**.

Example



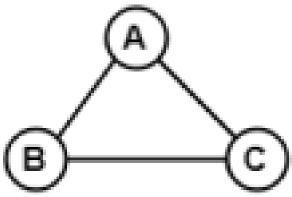
In the above graph, each edge is directed by the arrow. A directed edge has an arrow from A to B, means A is related to B, but B is not related to A.

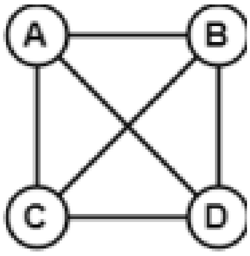
6. Complete Graph

A graph in which every pair of vertices is joined by exactly one edge is called **complete graph**. It contains all possible edges.

A complete graph with n vertices contains exactly nC2 edges and is represented by Kn.

Example



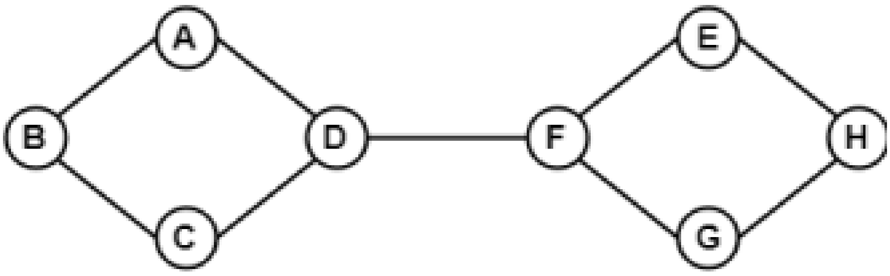


In the above example, since each vertex in the graph is connected with all the remaining vertices through exactly one edge therefore, both graphs are complete graph.

7. Connected Graph

A **connected graph** is a graph in which we can visit from any one vertex to any other vertex. In a connected graph, at least one edge or path exists between every pair of vertices.

Example

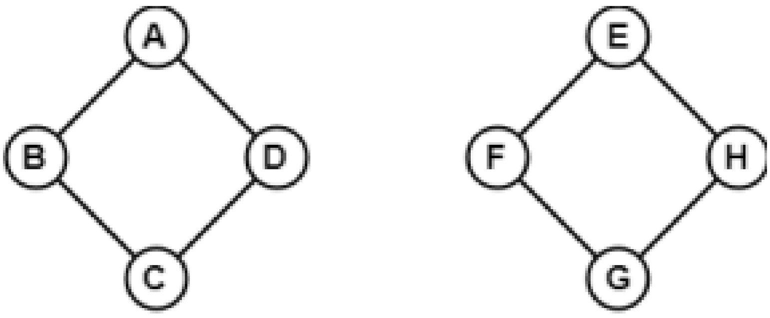


In the above example, we can traverse from any one vertex to any other vertex. It means there exists at least one path between every pair of vertices therefore, it a connected graph.

8. Disconnected Graph

A **disconnected graph** is a graph in which any path does not exist between every pair of vertices.

Example



The above graph consists of two independent components which are disconnected. Since it is not possible to visit from the vertices of one component to the vertices of other components therefore, it is a disconnected graph.

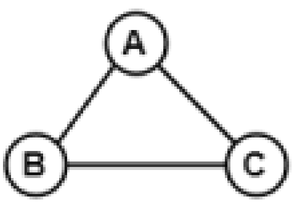
9. Regular Graph

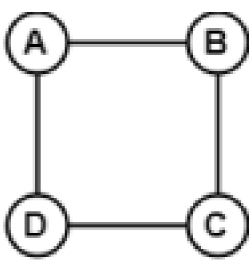
A **Regular graph** is a graph in which degree of all the vertices is

same. If the degree of all the vertices is k, then it is called k-regular

graph.

Example





In the above example, all the vertices have degree 2. Therefore they are called

2- **Regular graph**.

10. Cyclic Graph

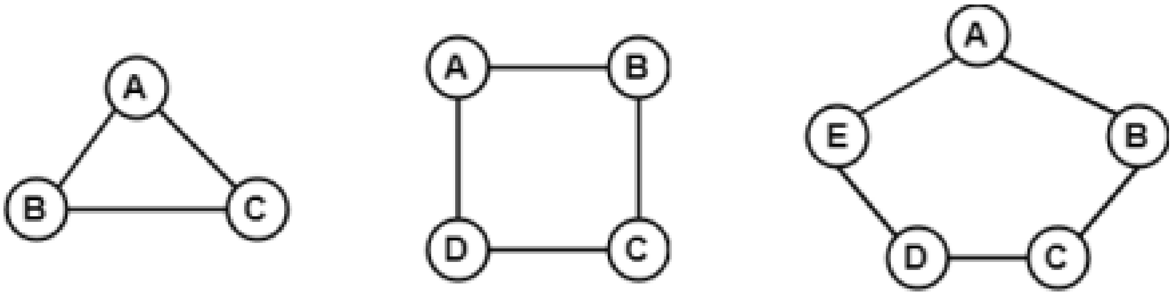
A graph with 'n' vertices (where, n>=3) and 'n' edges forming a cycle of 'n' with all its edges is known as **cycle graph**.

A graph containing at least one cycle in it is known as a **cyclic**

graph. In the cycle graph, degree of each vertex is 2.

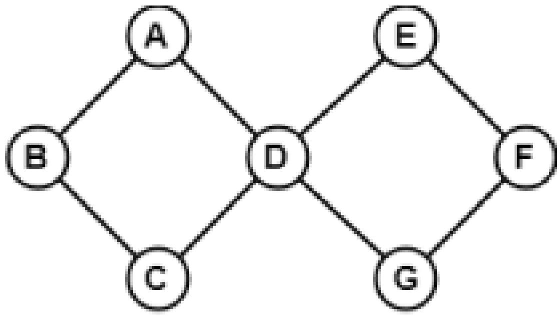
The cycle graph which has n vertices is denoted by Cn.

Example 1



In the above example, all the vertices have degree 2. Therefore they all are cyclic graphs.

Example 2

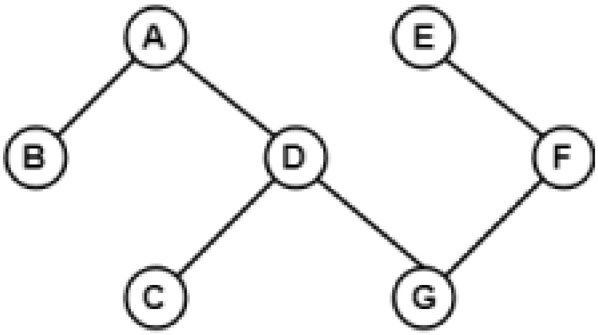


Since, the above graph contains two cycles in it therefore, it is a cyclic graph.

11. Acyclic Graph

A graph which does not contain any cycle in it is called as an **acyclic graph**.

Example



Since, the above graph does not contain any cycle in it therefore, it is an acyclic graph.

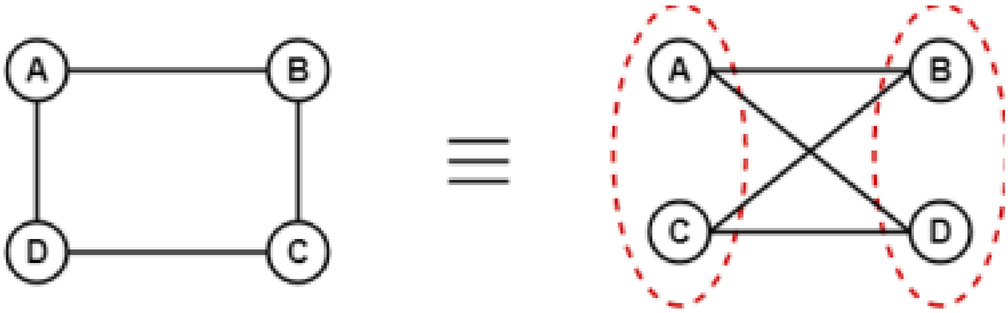
12. Bipartite Graph

A **bipartite graph** is a graph in which the vertex set can be partitioned into two sets such that edges only go between sets, not within them.

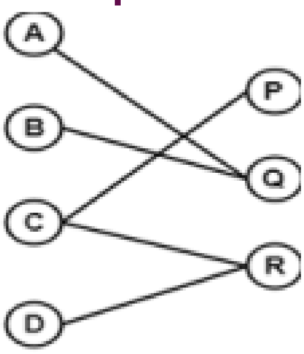
A graph G (V, E) is called bipartite graph if its vertex-set V(G) can be decomposed into two non-empty disjoint subsets V1(G) and V2(G) in such a way that each edge e ∈ E(G) has its one last joint in V1(G) and other last point in V2(G).

The partition V = V1 ∪ V2 is known as bipartition of G.

Example 1



Example 2



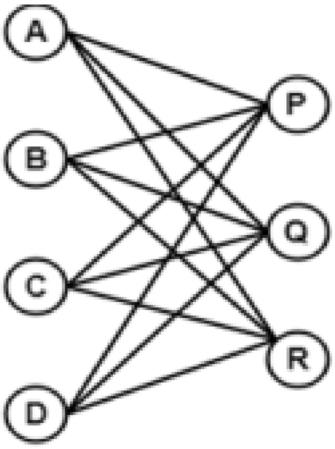
13. Complete Bipartite Graph

A **complete bipartite graph** is a bipartite graph in which each vertex in the first set is joined to each vertex in the second set by exactly one edge.

A complete bipartite graph is a bipartite graph which is complete.

1. Complete Bipartite graph = Bipartite graph + Complete graph

Example



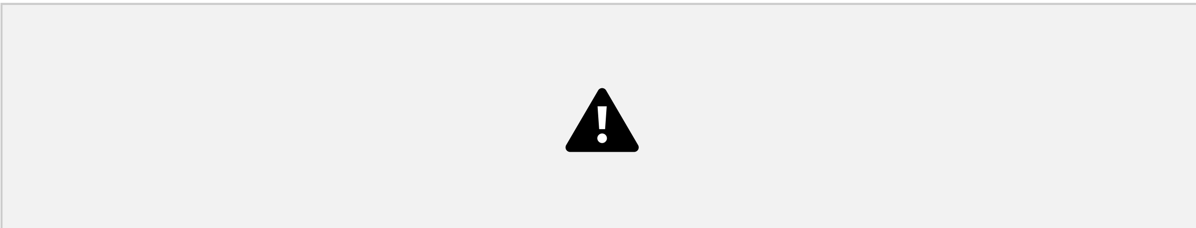
The above graph is known as $K_{4,3}$.

14. Star Graph

A star graph is a complete bipartite graph in which n-1 vertices have degree 1 and a single vertex have degree (n -1). This exactly looks like a star where (n - 1) vertices are connected to a single central vertex.

A star graph with n vertices is denoted by S_n.

Example



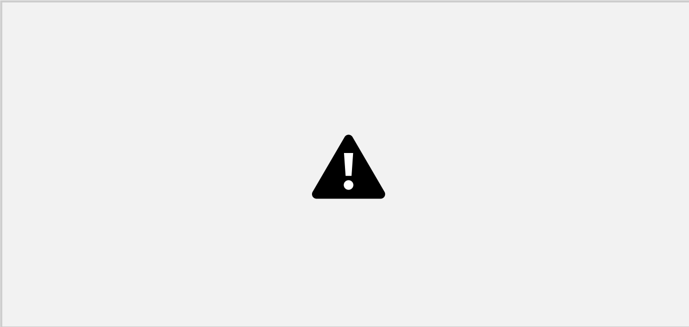
In the above example, out of n vertices, all the (n-1) vertices are connected to a single vertex. Hence, it is a star graph.

15 Weighted Graph

A weighted graph is a graph whose edges have been labeled with some weights or numbers.

The length of a path in a weighted graph is the sum of the weights of all the edges in the path.

Example

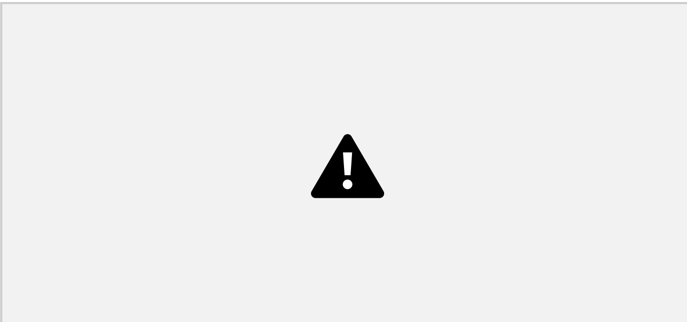


In the above graph, if path is a -> b -> c -> d -> e -> g then the length of the path is 5 + 4 + 5 + 6 + 5 = 25.

16. Multi-graph

A graph in which there are multiple edges between any pair of vertices or there are edges from a vertex to itself (loop) is called a **multi - graph**.

Example

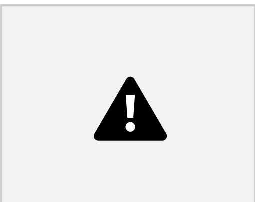


In the above graph, vertex-set B and C are connected with two edges. Similarly, vertex sets E and F are connected with 3 edges. Therefore, it is a multi graph.

17. Planar Graph

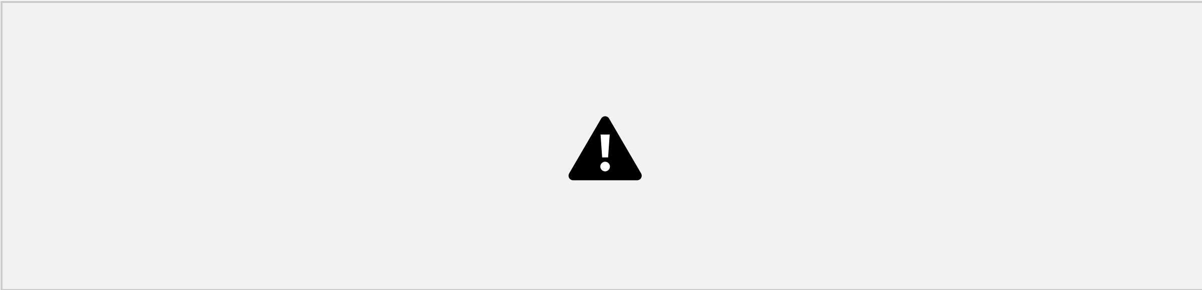
A **planar graph** is a graph that we can draw in a plane in such a way that no two edges of it cross each other except at a vertex to which they are incident.

Example



The above graph may not seem to be planar because it has edges crossing each other. But we can redraw the above graph.

The three plane drawings of the above graph are:

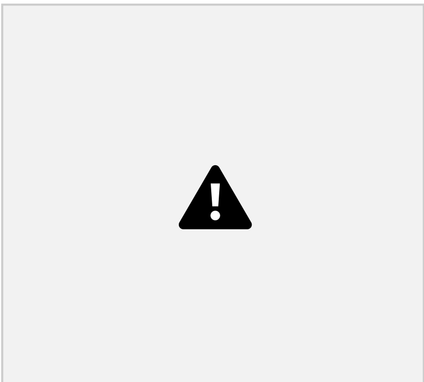


The above three graphs do not consist of two edges crossing each other and therefore, all the above graphs are planar.

18. Non - Planar Graph

A graph that is not a planar graph is called a non-planar graph. In other words, a graph that cannot be drawn without at least on pair of its crossing edges is known as non-planar graph.

Example



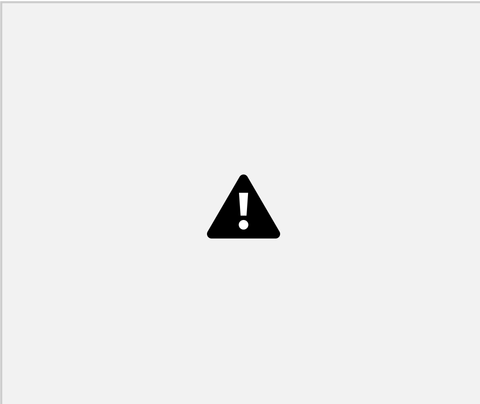
The above graph is a non - planar graph.

Properties of graph theory are basically used for characterization of graphs depending on the structures of the graph. Following are some basic properties of graph theory:

1 Distance between two vertices

Distance is basically the number of edges in a shortest path between vertex X and vertex Y. If there are many paths connecting two vertices, then the shortest path is considered as the distance between the two vertices. Distance between two vertices is denoted by **d(X, Y)**.

Example



Suppose, we want to find the distance between vertex B and D, then first of all we have to find the shortest path between vertex B and D.

There are many paths from vertex B to vertex D:

- B -> C -> A -> D, length = 3
- B -> D, length = 1(Shortest Path)
- B -> A -> D, length = 2
- B -> C -> D, length = 2
- B -> C -> A -> D, length = 3

Hence, the minimum **distance between vertex B and vertex D is 1**.

2. Eccentricity of a vertex

Eccentricity of a vertex is the maximum distance between a vertex to all other vertices. It is denoted by **e(V)**.

To count the eccentricity of vertex, we have to find the distance from a vertex to all other vertices and **the highest distance is the eccentricity** of that particular vertex.

Example

In the above example, if we want to find the maximum eccentricity of vertex 'a' then:

- The distance from vertex a to b is 1 (i.e. ab)
- The distance from vertex a to c is 1 (i.e. ac)
- The distance from vertex a to f is 2 (i.e. ac -> cf or ad -> df)
- The distance from vertex a to d is 1 (i.e. ad)
- The distance from vertex a to e is 2 (i.e. ab -> be or ad -> de) ○ The distance from vertex a to g is 3 (i.e. ab -> be -> eg or ac -> cf -> fg etc.)

Hence, the **maximum eccentricity of vertex** 'a' is 3, which is a maximum distance from vertex ?a? to all other vertices.

Similarly, maximum eccentricities of other vertices of the given graph are:

- e(b) = 3
- e(c) = 3
- e(d) = 2
- e(e) = 3
- e(f) = 3
- e(g) = 3

3. Radius of connected Graph

The **radius** of a connected graph is the **minimum eccentricity** from all the vertices. In other words, the minimum among all the distances between a vertex to all other vertices is called as the radius of the graph. It is denoted by **r(G)**.

Example

From the example of 5.2, r(G) = 2, which is the minimum eccentricity for the vertex 'd'.

4. Diameter of a Graph

Diameter of a graph is the **maximum eccentricity** from all the vertices. In other words, the maximum among all the distances between a vertex to all other vertices is considered as the diameter of the graph G. It is denoted by **d(G)**.

Example

From the above example, if we see all the eccentricities of the vertices in a graph, we will see that the diameter of the graph is the maximum of all those eccentricities.

Diameter of graph **d(G) = 3**, which is the maximum eccentricity.

5. Central point

If the eccentricity of the graph is equal to its radius, then it is known as **central point** of the graph.

Or

If **r(V) = e(V), then V is the central point** of the graph G.

Example

From the above example, 'd' is the central point of the graph. i.e.

1. e(d) = r(d) = 2

6. Centre

The set of all the central point of the graph is known as centre of the

graph. **Example**

From the example of 5.2, {**'d'**} is the centre of the graph.

7. Circumference

The total number of edges in the longest cycle of graph G is known as the circumference of G.

Example

In the above example, the **circumference is 6**, which is derived from the longest path a -> c -> f -> g -> e -> b -> a or a -> c -> f -> d -> e -> b -> a.

8. Girth

The total number of edges in the shortest cycle of graph G is known as **girth**. It is denoted by g(G).

Example

In the above example, the **girth of the graph is 4**, which is derived from the shortest cycle a -> c -> f -> d -> a, d -> f -> g -> e -> d or a -> b -> e -> d -> a.

9. Sum of degrees of vertices Theorem

For non-directed graph G = (V,E) where, Vertex set V = {V1, V2, ... Vn}

then, 

In other words, for any graph, the sum of degrees of vertices equals twice the number of edges.

Corollary 1

For directed graph G = (V, E) where, Vertex Set V = {V1, V2, ... Vn}

then, 

Corollary 2

The number of vertices in any non- directed graph with odd degree is

even. **Example**

It is impossible to make a graph with v (number of vertices) = 6 where the vertices have degrees 1, 2, 2, 3, 3, 4. This is because the sum of the degrees deg(V) is,

1. deg(V) = 1 + 2 + 2 + 3 + 3 + 4 = 15</p>

2. deg(V) is always an even number but 15 is odd!

Corollary 3

In an non-directed graph, if the degree of each vertex is k, then

1. $k|V| = 2|E|$

Corollary 4

If the degree of each vertex in a non-directed graph is at least k, then

1. $k|V| \leq 2|E|$

Corollary 5

If the degree of each vertex in a non- directed graph is at most k,

then 1. $k|V| \geq 2|E|$

Graph Representations

In graph theory, a graph representation is a technique to store graph into the memory of computer.

To represent a graph, we just need the set of vertices, and for each vertex the neighbors of the vertex (vertices which is directly connected to it by an edge). If it is a weighted graph, then the weight will be associated with each edge.

There are different ways to optimally represent a graph, depending on the density of its edges, type of operations to be performed and ease of use.

1. Adjacency Matrix

- Adjacency matrix is a sequential representation.

- It is used to represent which nodes are adjacent to each other. i.e. is there any edge connecting nodes to a graph.
- In this representation, we have to construct a nXn matrix A. If there is any edge from a vertex i to vertex j, then the corresponding element of A, $a^{i,j}$ = 1, otherwise $a^{i,j}$= 0.

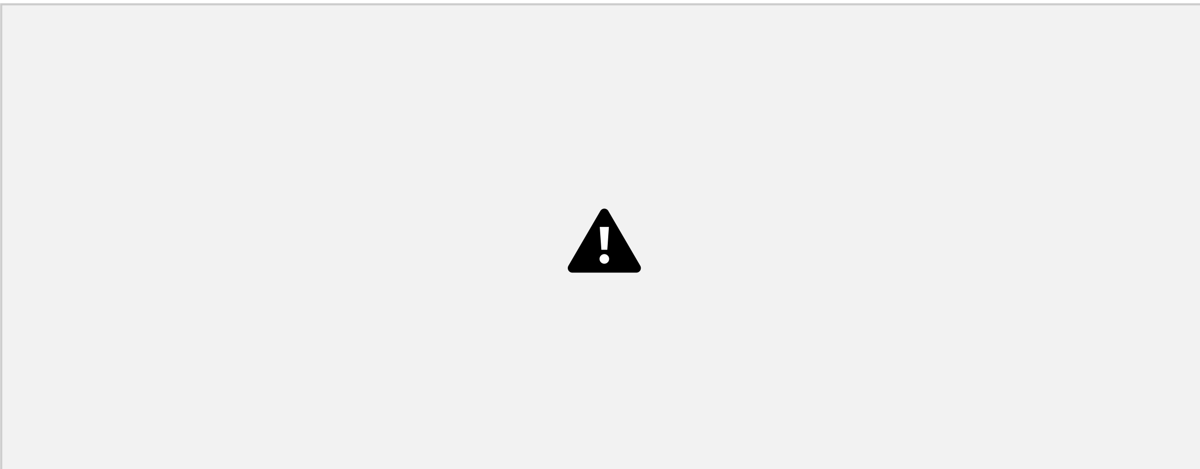
Note, even if the graph on 100 vertices contains only 1 edge, we still have to have a 100x100 matrix with lots of zeroes.

- If there is any weighted graph then instead of 1s and 0s, we can store the weight of the edge.

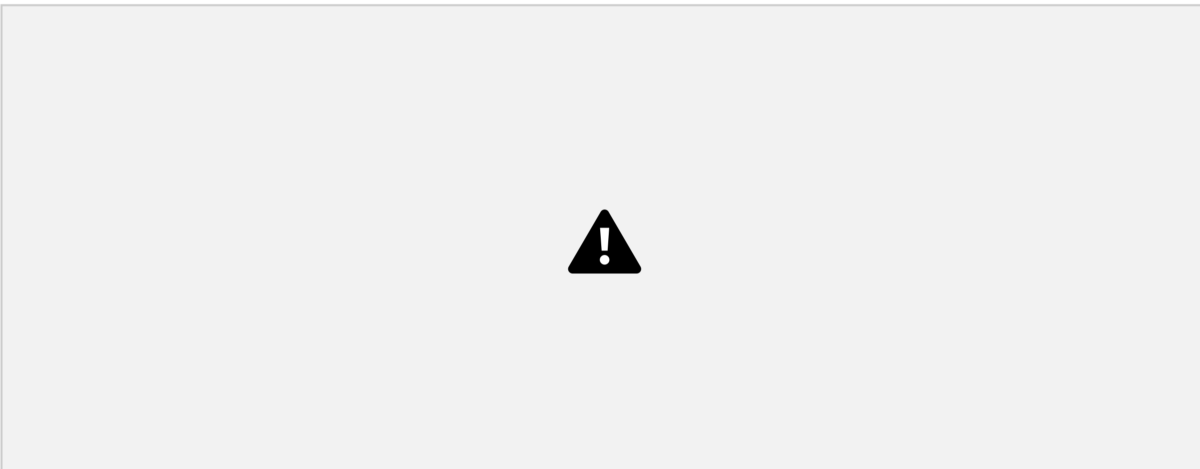
Example

Consider the following **undirected graph representation**:

Undirected graph representation



Directed graph represenation
See the directed graph representation:



In the above examples, 1 represents an edge from row vertex to column vertex, and 0 represents no edge from row vertex to column vertex.

Undirected weighted graph represenation

Pros: Representation is easier to implement and follow.

Cons: It takes a lot of space and time to visit all the neighbors of a vertex, we have to traverse all the vertices in the graph, which takes quite some time.

2. Incidence Matrix

In **Incidence matrix representation**, graph can be represented using a matrix of size:

Total number of vertices by total number of edges.

It means if a graph has 4 vertices and 6 edges, then it can be represented using a matrix of 4X6 class. In this matrix, columns represent edges and rows represent vertices.

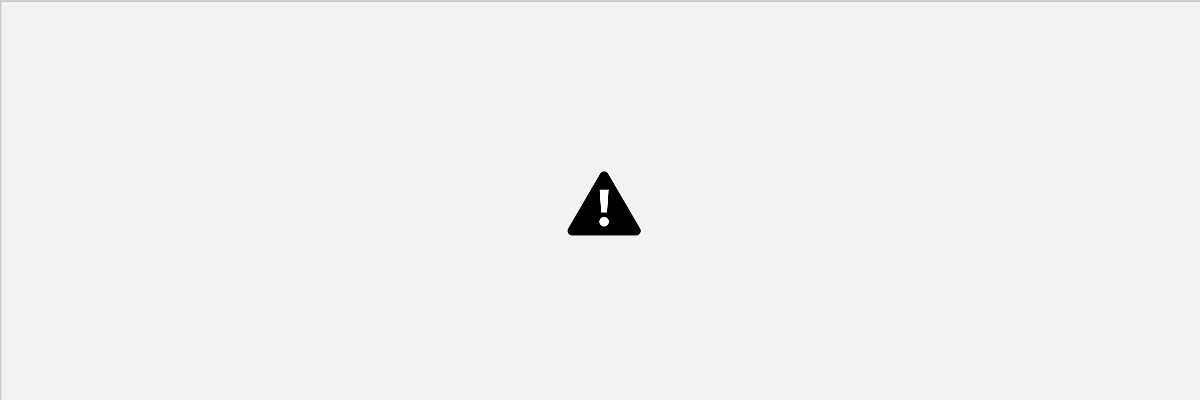
This matrix is filled with either **0 or 1** or -1. Where,

- 0 is used to represent row edge which is not connected to column vertex.

- 1 is used to represent row edge which is connected as outgoing edge to column vertex.
- -1 is used to represent row edge which is connected as incoming edge to column vertex.

Example

Consider the following directed graph representation.

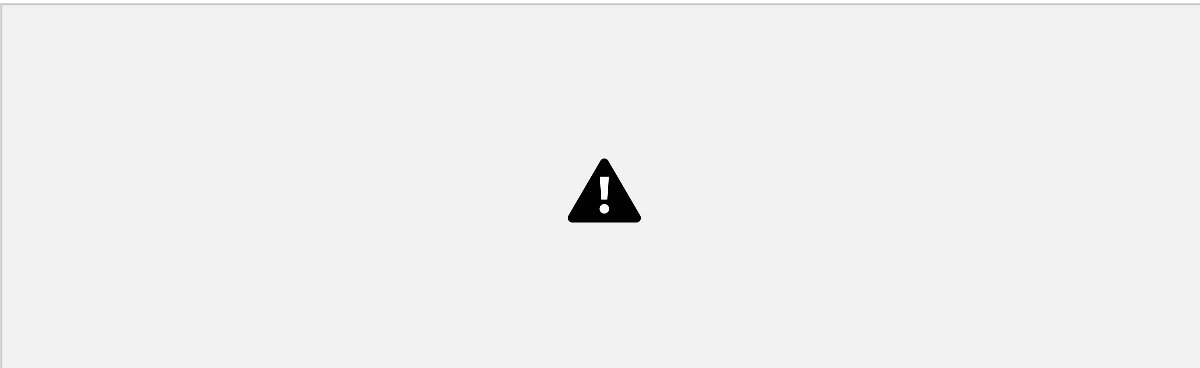


3. Adjacency List

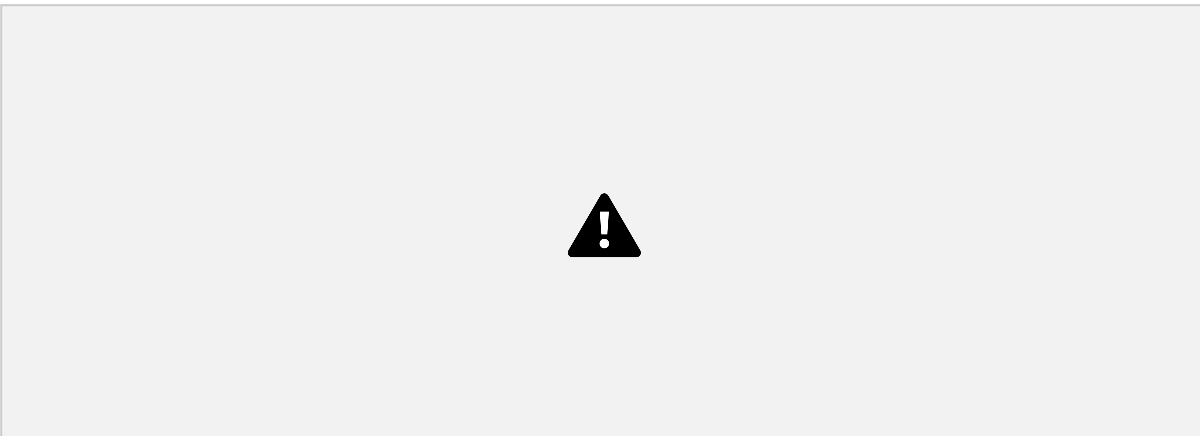
- Adjacency list is a linked representation.
- In this representation, for each vertex in the graph, we maintain the list of its neighbors. It means, every vertex of the graph contains list of its adjacent vertices.
- We have an array of vertices which is indexed by the vertex number and for each vertex v, the corresponding array element points to a **singly linked list** of neighbors of v.

Example

Let's see the following directed graph representation implemented using linked list:



We can als0000000o implement this representation using array as follows:



Pros:

- Adjacency list saves lot of space.
- We can easily insert or delete as we use linked list.
- Such kind of representation is easy to follow and clearly shows the adjacent nodes of node.

Cons:

- The adjacency list allows testing whether two vertices are adjacent to each other but it is slower to support this operation.

Tree and Forest

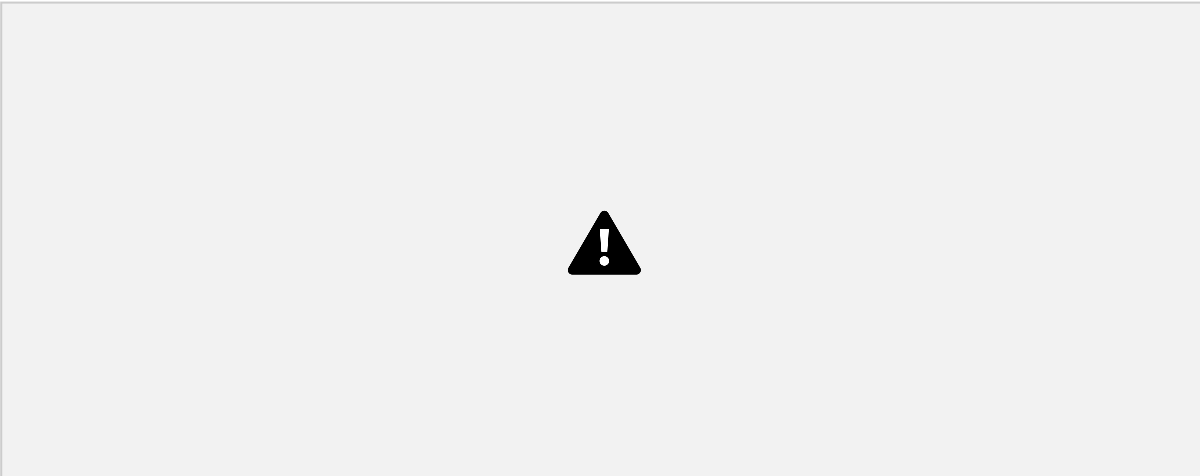
1. What is Tree and Forest?

Tree

- In graph theory, a **tree** is an **undirected, connected and acyclic graph**. In other words, a connected graph that does not contain even a single cycle is called a tree.
- A tree represents hierarchical structure in a graphical form.

- The elements of trees are called their nodes and the edges of the tree are called branches.
- A tree with n vertices has (n-1) edges.
- Trees provide many useful applications as simple as a family tree to as complex as trees in data structures of computer science.
- A **leaf** in a tree is a vertex of degree 1 or any vertex having no children is called a leaf.

Example

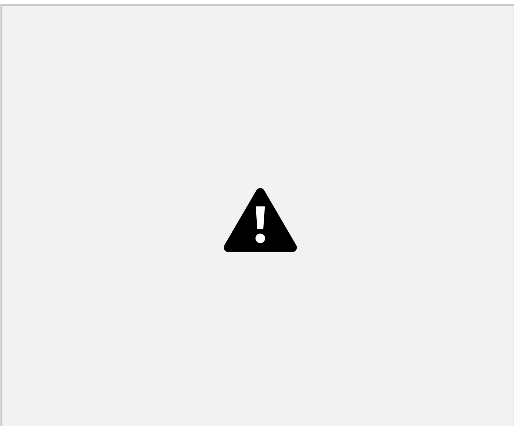


In the above example, all are trees with fewer than 6 vertices.

Forest

In graph theory, a **forest** is **an undirected, disconnected, acyclic graph**. In other words, a disjoint collection of trees is known as forest. Each component of a forest is tree.

Example



The above graph looks like a two sub-graphs but it is a single disconnected graph. There are no cycles in the above graph. Therefore it is a forest.

2. Properties of Trees

1. Every tree which has at least two vertices should have at least two leaves.
2. Trees have many characterizations:

   Let T be a graph with n vertices, then the following statements are equivalent:

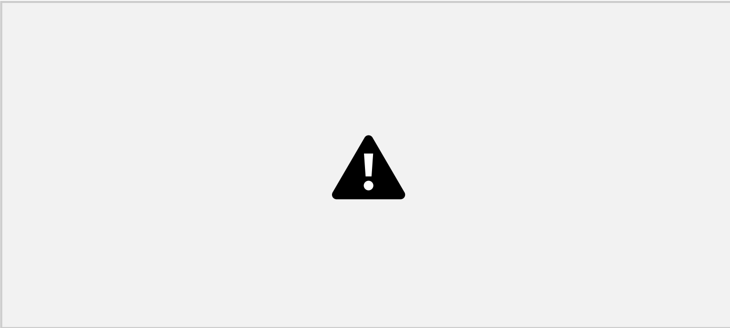
   - T is a tree.
   - T contains no cycles and has n-1 edges.
   - T is connected and has (n -1) edge.
   - T is connected graph, and every edge is a cut-edge.
   - Any two vertices of graph T are connected by exactly one path.
   - T contains no cycles, and for any new edge e, the graph T+ e has exactly one cycle.
3. Every edge of a tree is cut -edge.
4. Adding one edge to a tree defines exactly one cycle.
5. Every connected graph contains a spanning tree.
6. Every tree has at least two vertices of degree two.

3. Spanning Tree

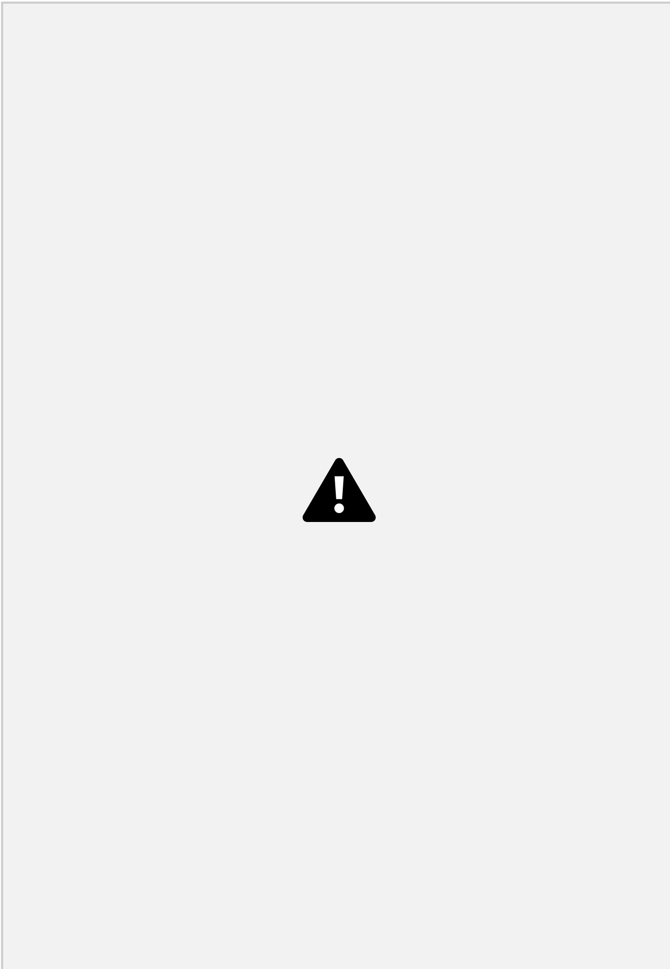
A **spanning tree** in a connected graph G is a sub-graph H of G that includes all the vertices of G and is also a tree.

Example

Consider the following graph G:



From the above graph G we can implement following three spanning trees

H: 

Methods to find the spanning Tree

We can find the spanning tree systematically by using either of two methods:

1. Cutting- down Method
2. Building- up Method

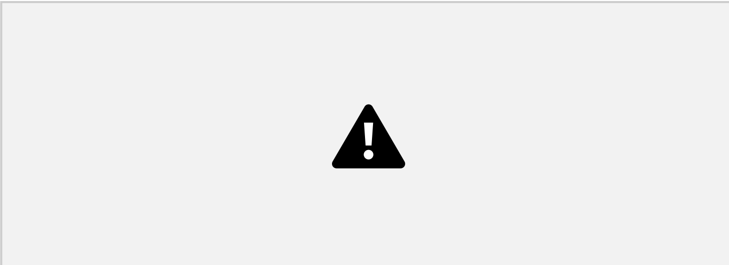
1. Cutting- down method

- Start choosing any cycle in Graph G.

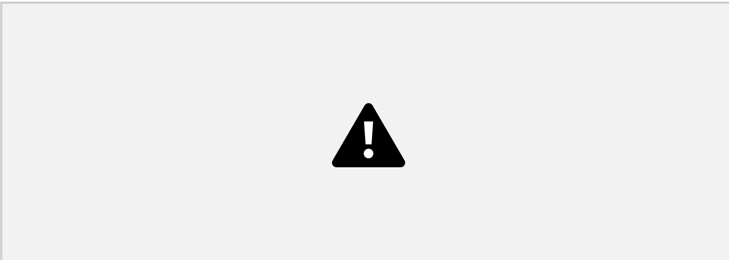
- Remove one of cycle's edges.
- Repeat this process until there are no cycles left.

Example

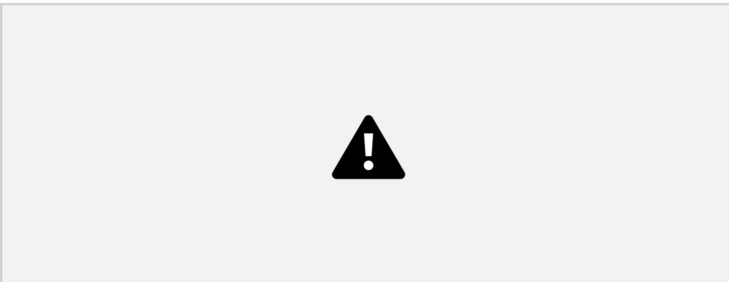
- Consider a graph G,



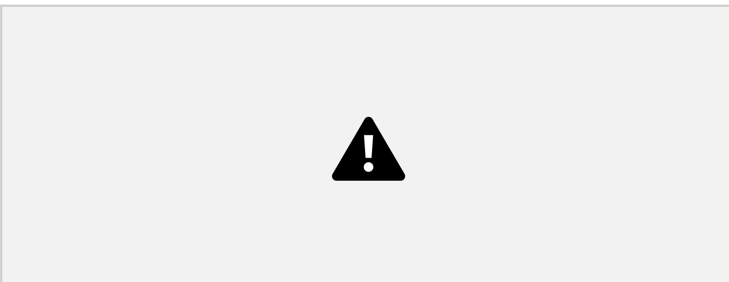
- If we remove the edge ac which destroy the cycle a-d-c-a in the above graph then we get the following graph:



- Remove the edge cb, which destroy the cycle a-d-c-b-a from the above graph then we get the following graph:



- If we remove the edge ec, which destroy the cycle d-e-c-d from the above graph then we get the following spanning tree:



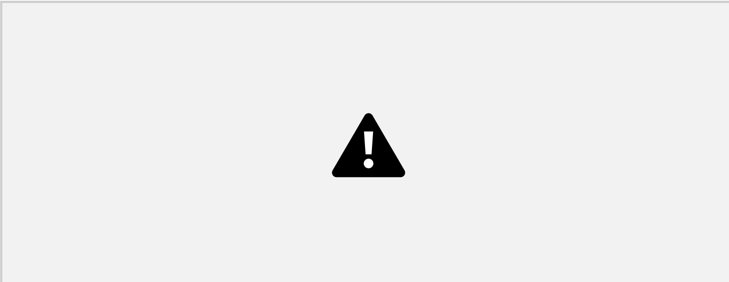
2. Building - up method

- Select edges of graph G one at a time. In such a way that there are no cycles are created.

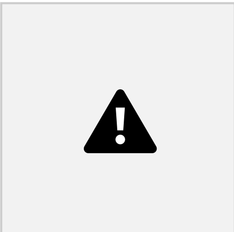
- Repeat this process until all the vertices are included.

Example

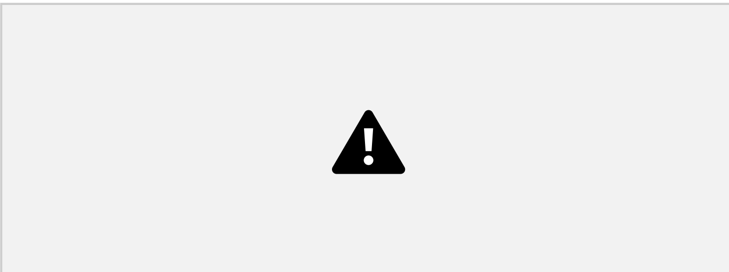
Consider the following graph G,



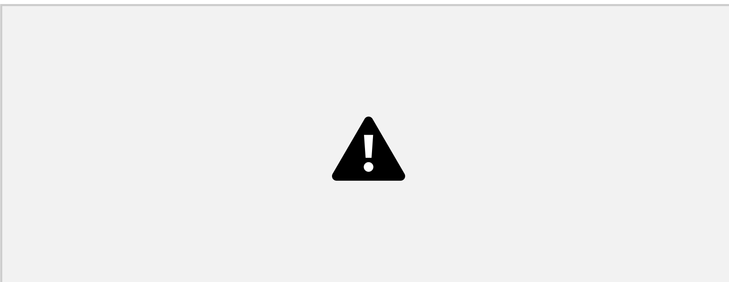
- Choose the edge **ab**,



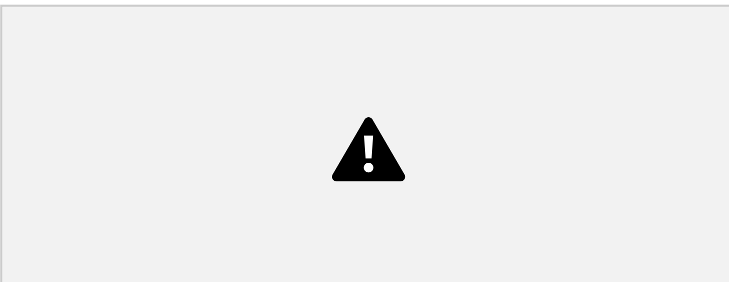
- Choose the edge **de**,



- After that , choose the edge **ec**,



- Next, choose the edge **cb**, then finally we get the following spanning tree:

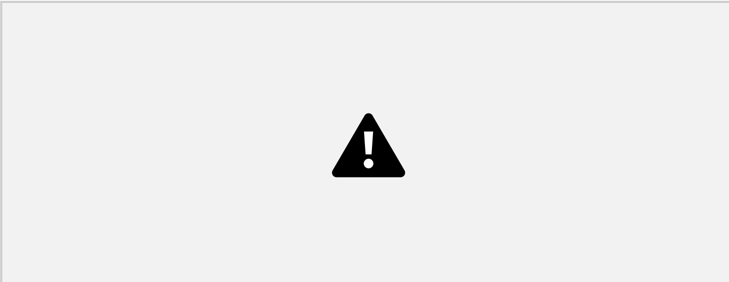


Circuit Rank

The number of edges we need to delete from G in order to get a spanning tree. **Spanning tree G = m- (n-1)**, which is called the **circuit rank** of G.

1. Where, m = No. of edges.
2. n = No. of vertices.

Example



In the above graph, edges m = 7 and vertices n = 5

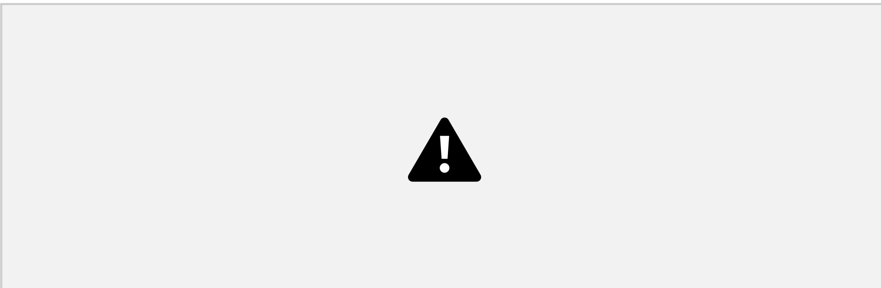
Then the circuit rank is,

1. G = m - (n - 1)
2. = 7 - (5 - 1)
3. = 3

Connectivity

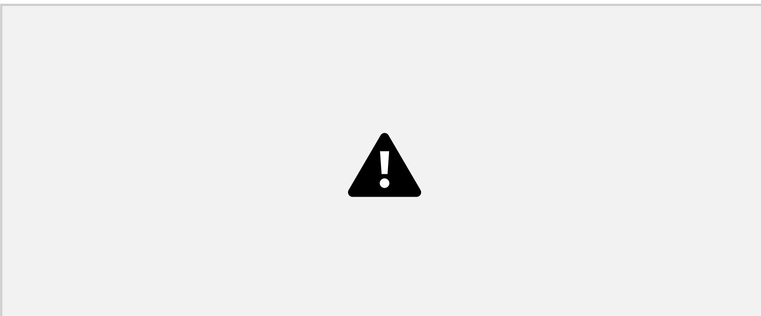
- **Connectivity** is a basic concept of graph theory. It defines whether a graph is connected or disconnected. Without connectivity, it is not possible to traverse a graph from one vertex to another vertex.
- A graph is said to be **connected graph if there is a path between every pair of vertex**. From every vertex to any other vertex there must be some path to traverse. This is called the connectivity of a graph.
- A graph is said to be **disconnected, if there exists multiple disconnected vertices and edges**.
- Graph connectivity theories are essential in network applications, routing transportation networks, network tolerance etc.

Example



In the above example, it is possible to travel from one vertex to another vertex. Here, we can traverse from vertex B to H using the path B -> A -> D -> F -> E -> H. Hence it is a connected graph.

Example



In the above example, it is not possible to traverse from vertex B to H because there is no path between them directly or indirectly. Hence, it is a disconnected graph.

Let's see some basic concepts of Connectivity.

1. Cut Vertex

A single vertex whose removal disconnects a graph is called a cut-vertex.

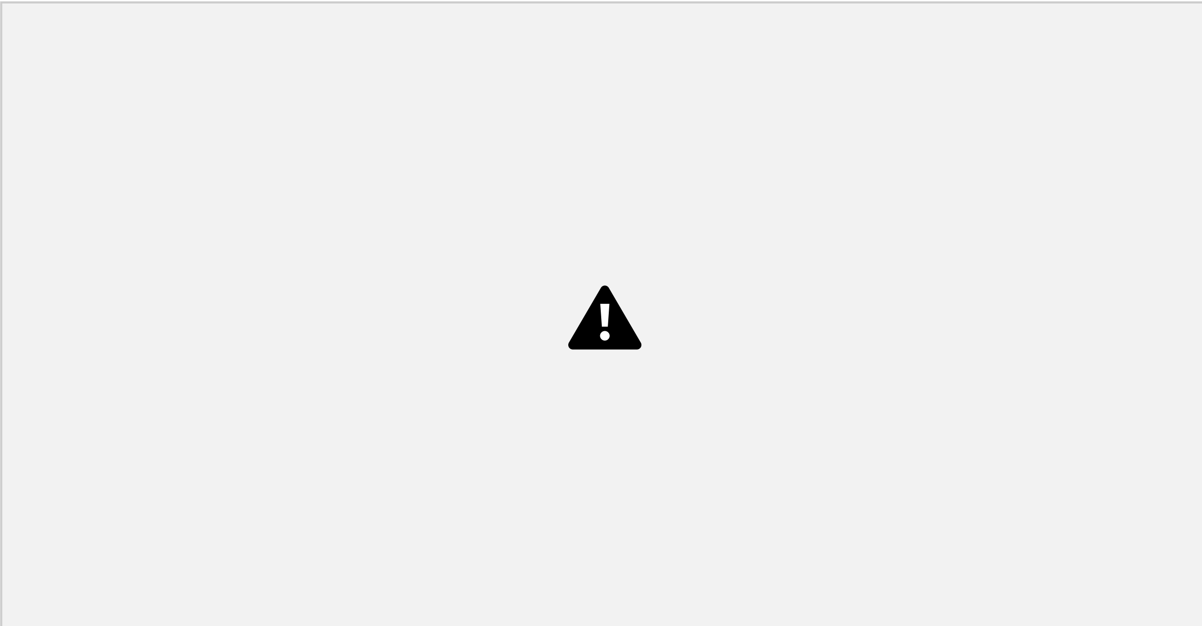
Let G be a connected graph. A vertex v of G is called a cut vertex of G, if G-v (Remove v from G) results a disconnected graph.

When we remove a vertex from a graph then graph will break into two or more graphs. This vertex is called a cut vertex.

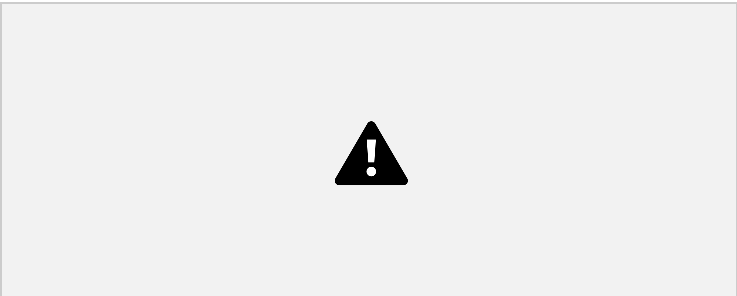
Note: Let G be a graph with n vertices:

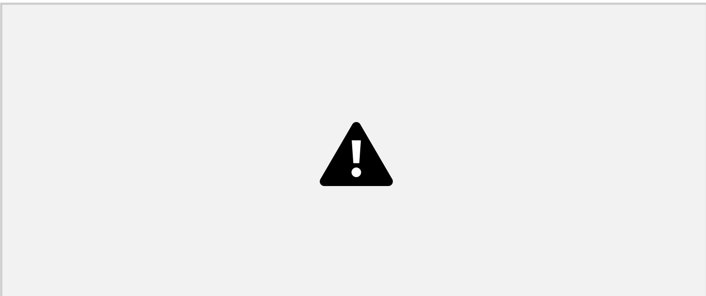
- A connected graph G may have maximum (n-2) cut vertices.
- Removing a cut vertex may leave a graph disconnected.
- Removing a vertex may increase the number of components in a graph by at least one.
- Every non-pendant vertex of a tree is a cut vertex.

Example 1



Example 2





In the above graph, vertex 'e' is a cut-vertex. After removing vertex 'e' from the above graph the graph will become a disconnected graph.

2. Cut Edge (Bridge)

A **cut- Edge or bridge** is a single edge whose removal disconnects a graph.

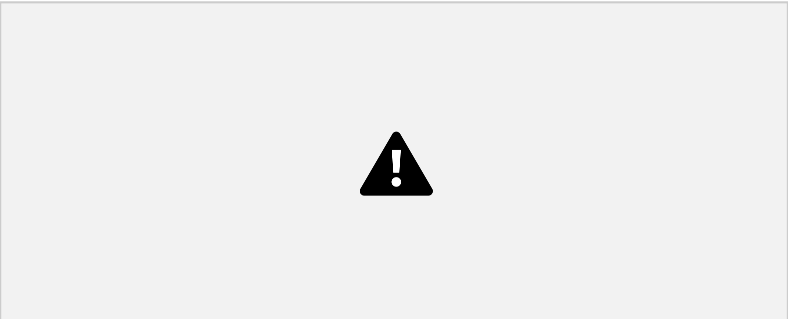
Let G be a connected graph. An edge e of G is called a cut edge of G, if G-e (Remove e from G) results a disconnected graph.

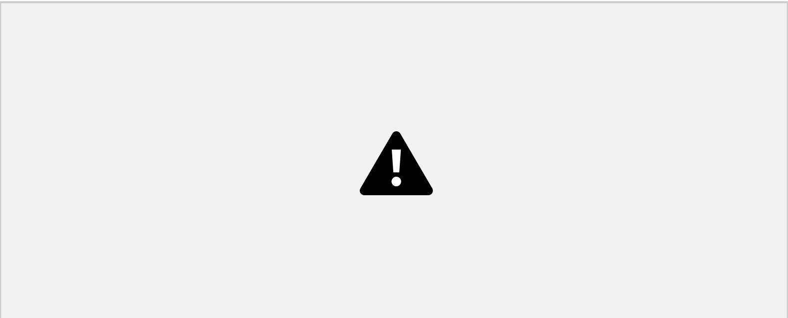
When we remove an edge from a graph then graph will break into two or more graphs. This removal edge is called a cut edge or bridge.

Note: Let G be a graph with n vertices:

- A connected graph G may have at most (n-1) cut edges.
- Removing a cut edge may leave a graph disconnected.
- Removal of an edge may increase the number of components in a graph by at most one.
- A cut edge 'e' must not be the part of any cycle in G.
- If a cut edge exists, then a cut vertex must also exist because at least one vertex of a cut edge is a cut vertex.
- If a cut vertex exists, then the existence of any cut edge is not necessary.

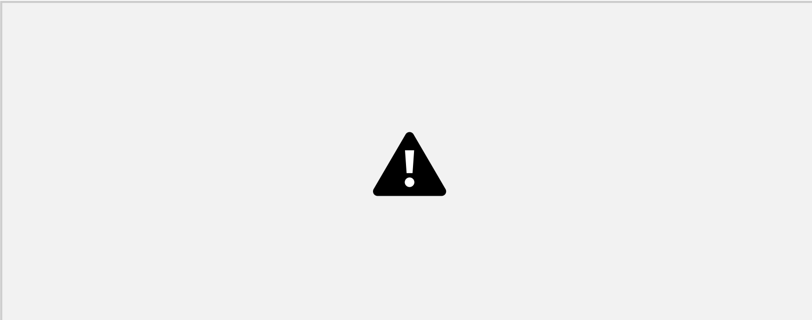
Example 1

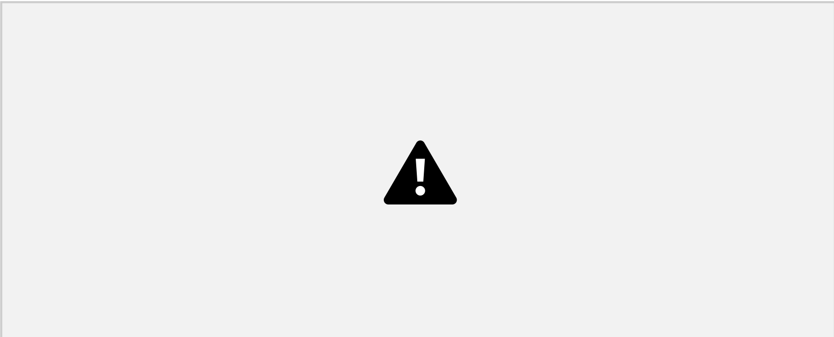




In the above graph, edge (c, e) is a cut-edge. After removing this edge from the above graph the graph will become a disconnected graph.

Example 2





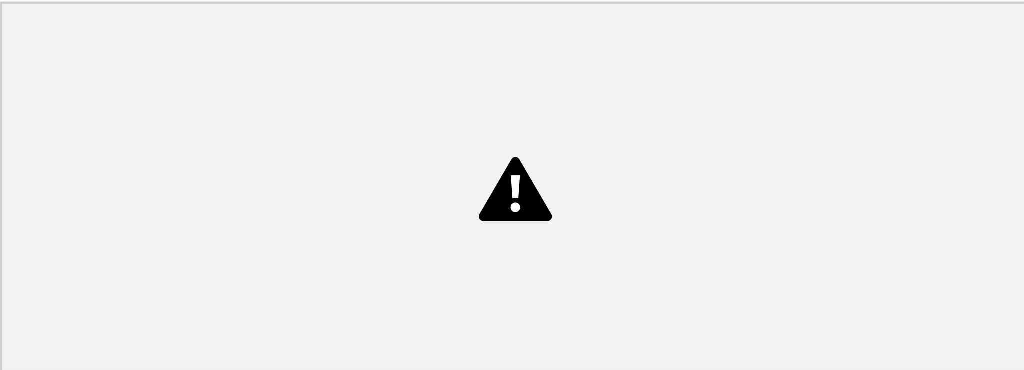
In the above graph, edge (c, e) is a cut-edge. After removing this edge from the above graph the graph will become a disconnected graph.

3. Cut Set

In a connected graph G, a cut set is a set S of edges with the following properties:

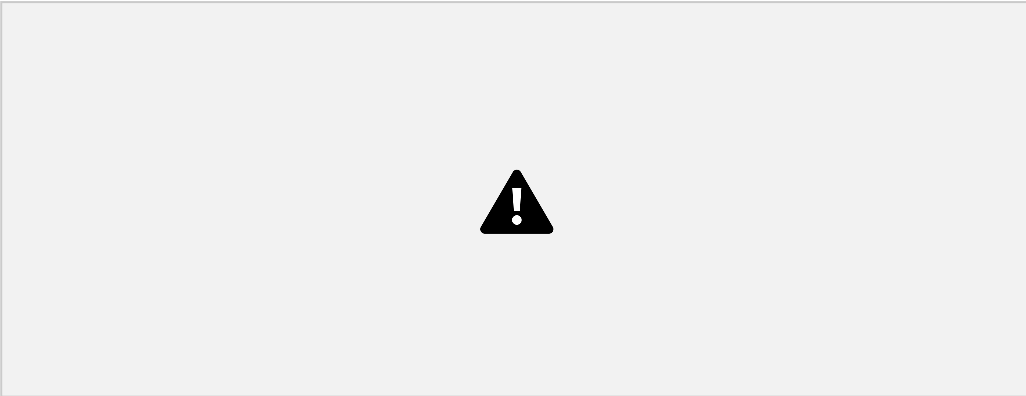
- The removal of all the edges in S disconnects G.
- The removal of some of edges (but not all) in S does not disconnect G.

Example 1



To disconnect the above graph G, we have to remove the three edges. i.e. bd, be and ce. We cannot disconnect it by removing just two of three edges. Hence, {bd, be, ce} is a cut set.

After removing the cut set from the above graph, it would look like as follows:



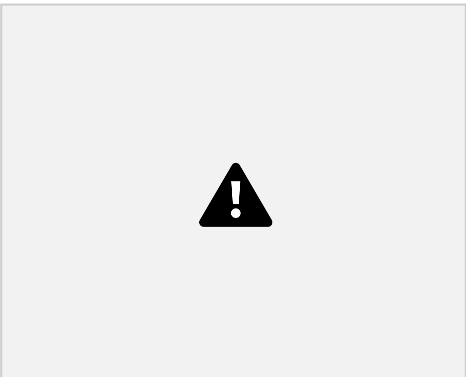
4. Edge Connectivity

The **edge connectivity** of a connected graph G is the minimum number of edges whose removal makes G disconnected. It is denoted by **λ(G)**.

When λ(G) ≥ k, then graph G is said to be **k-edge-connected**.

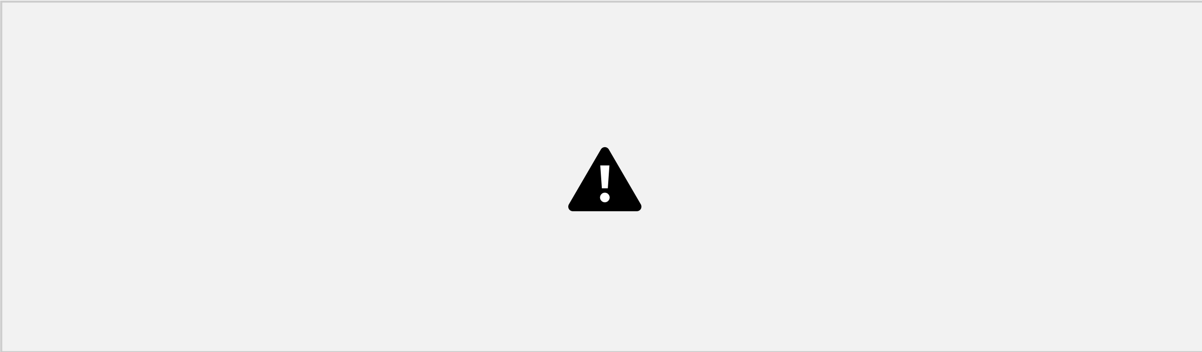
Example

Let's see an example,



From the above graph, by removing two minimum edges, the connected graph becomes disconnected graph. Hence, its edge connectivity is 2. Therefore the above graph is a **2-edge-connected graph**.

Here are the following four ways to disconnect the graph by removing two edges:



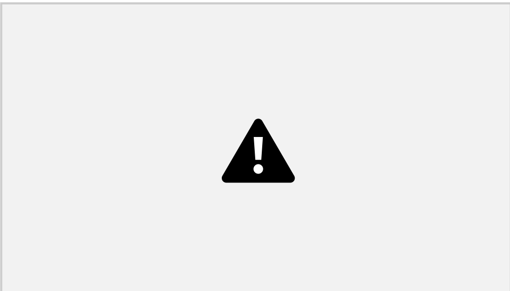
5. Vertex Connectivity

The connectivity (or vertex connectivity) of a connected graph G is the minimum number of vertices whose removal makes G disconnects or reduces to a trivial graph. It is denoted by K(G).

The graph is said to be k- connected or k-vertex connected when K(G) ≥ k. To remove a vertex we must also remove the edges incident to it.

Example

Let's see an example:



The above graph G can be disconnected by removal of the single vertex either 'c' or 'd'. Hence, its vertex connectivity is 1. Therefore, it is a 1-connected graph.