

Graph Theory

Representation of Graphs:

There are two different sequential representation of graph. They are

- Adjacency Matrix representation
- Path Matrix representation.

adjacency Matrix Representation.

Suppose G_1 is a simple directed graph with m nodes, and suppose the nodes of G_1 have been ordered and are called v_1, v_2, \dots, v_m , then the adjacency matrix $A = (a_{ij})$ of the graph G_1 is the $m \times m$ matrix defined as follows.

1 if v_i is adjacent to v_j . That is if there is an edge $e(v_i, v_j)$

$$a_{ij} = 0 \text{ otherwise}$$

Suppose G_1 is an undirected graph. Then the adjacency matrix A of G_1 will be a symmetric matrix. i.e., one in which $a_{ij} = a_{ji}$; for every i and j .

Drawbacks

1. It was may be difficult to insert and delete nodes in G_1 .
2. If the number of edges is $O(m)$ or $O(m \log m)$, then the matrix A will be sparse, hence a great deal of space will be wasted.

1/2

Path matrix Representation.

Let G_1 be a simple directed graph with m nodes, v_1, v_2, \dots, v_m . The path matrix of G_1 is the m -square matrix $P = (P_{ij})$ defined as follows.

$$P_{ij} = \begin{cases} 1 & \text{if there is a path from } v_i \text{ to } v_j \\ 0 & \text{otherwise.} \end{cases}$$

Graphs and Multigraphs.

A graph G_1 consists of two things:

1. A set V of elements called nodes (or points or vertices)
2. A set E of edges such that each edge e in E is identified with a unique.

unordered pair $[u, v]$ of nodes in V , denoted by $e = [u, v]$.
sometimes we indicate the parts of a graph by writing $G_1 = (V, E)$. Suppose $e = [u, v]$. Then the nodes u and v are called the endpoints of e . and u and v are said to be adjacent nodes or neighbors.
The degree of a node u , written $\deg(u)$, is the number of edges containing u ; if $\deg(u) = 0$ — that is, if u does not belong to any edge — then u is called an isolated node.

Path and cycle

A path P of length n from a node u to a node v is defined as a sequence of $n+1$ nodes. $P = (v_0, v_1, v_2, \dots, v_n)$ such that $u = v_0$; v_{i-1} is adjacent to v_i for $i = 1, 2, \dots, n$ and $v_n = v$.

Types of path

1. Simple path
2. Cycle path
- (i) simple path

Simple path is a path in which first and last vertex are different ($v_0 \neq v_n$)

(ii) cycle path.

Cycle path is a path in which first and last vertex are same ($v_0 = v_n$) is also called as closed path

connected graph

A graph G_1 is said to be connected if there is a path between any two of its nodes.

complete Graph

A graph G_1 is called to be complete if every node u in G_1 is adjacent to every other node v in G_1 .

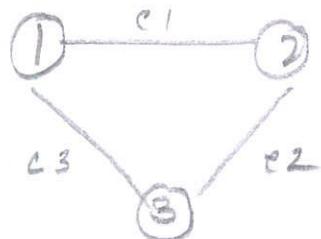
Tree

A connected graph T without any cycles is called a tree graph or free tree or, simply, a tree

Labeled or weighted graph

114

If the weight is assigned to each edge of the graph then it is called as weighted or labeled graph. St. Peter's 1



(a)



(b)

Weighted or Labeled graph:-

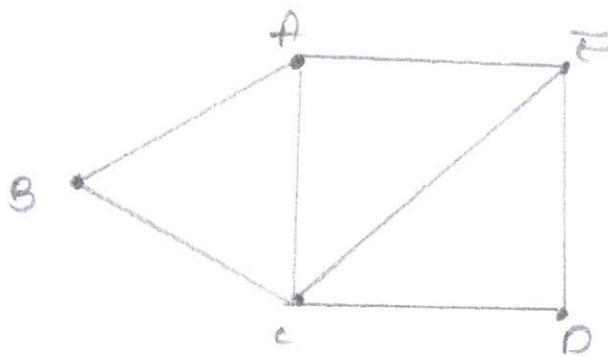
The definition of a graph may be generalized by permitting the following:

1. Multiple edges: Distinct edges e and e' are called multiple edges if they connect the same endpoints, that is, if $e = [u, v]$ and $e' = [u, v]$.

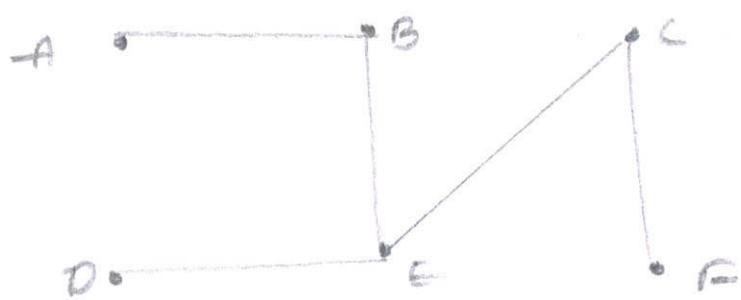
1. Multiple edges: Distinct edges e and e' are called multiple edges if they connect the same endpoints, that is, if $e = [u, v]$ and $e' = [u, v]$.

2. Loops: An edge e is called a loop if it has identical endpoints, that is, if $e = [u, u]$.

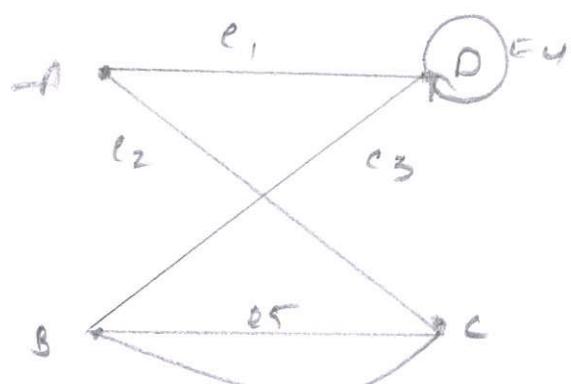
3. Finite graph: A multigraph M is said to be finite if it has a finite number of nodes and a finite number of edges.



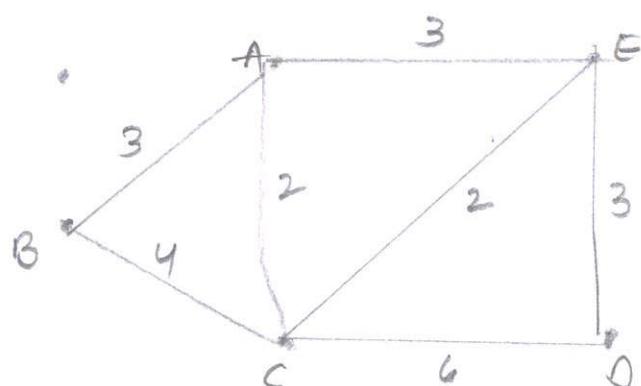
(a) Graph



(b) Tree



(c) multigraph.



(d) weighted graph.

Directed Graph

A directed graph G , also called a digraph or graph, is the same as a multigraph except that each edge e in G is assigned a direction, or in other words, each edge e is identified with an ordered pair (u, v) of nodes in G .

Outdegree and Indegree.

Indegree: The indegree of a vertex is the number of edges for which v is head.

Example



$$\text{indegree of } 1 = 1$$

$$\text{indegree of } 2 = 2.$$

Outdegree: The outdegree of a node or vertex is the number of edges for which v is tail.

Example



$$\text{outdegree of } 1 = 1$$

$$\text{outdegree of } 2 = 2$$

Simple Directed Graph

A directed graph G_1 is said to be simple if G_1 has no parallel edges. A simple graph G_1 may have loops, but it cannot have more than one loop at a given node.

Simple Directed Graph:-

A directed graph G_1 is said to be simple if G_1 has no parallel edges. A simple graph G_1 may have loops, but it cannot have more than one loop at a given node.

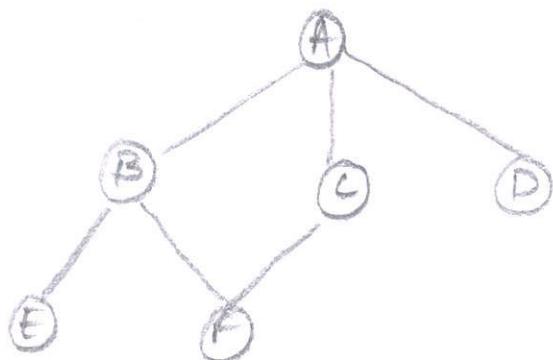
Graph Traversal

116

The breadth first search (BFS) and the depth first search (DFS) are the two algorithms used for traversing and searching a node in a graph. They can also be used to find out whether a node is reachable from a given node or not.

Depth first search (DFS)

The aim of DFS algorithm is to traverse the graph in such a way that it tries to go far from the root node stack is used in the implementation of the depth first search. Let's see how depth first search works with respect to the following graph:



As started before, in DFS, nodes are visited by going through the depth of the tree from the starting node. If we do the depth first traversal of the above graph and print the visited node, it will be "ABEFCID". DFS visits the root node and then its children nodes until it reaches the end node. i.e. E and F nodes, then moves up to the parent nodes.

Algorithmic steps

- 1, step 1: push the root node in the stack.
- 2, step 2: loop until stack is empty.
- 3, step 3: peek the node of the stack.
- 4, step 4: If the node has unvisited child nodes, get the unvisited child node, mark it as traversed and push it on stack.
- 5, step 5: If the node does not have any unvisited child nodes, pop the node from the stack.

```

public void DFS()
{
    // DFS uses stack datastructure.
    Stack s = new Stack();
    s.push(this, rootNode);
    rootNode.visited = true;
    printNode(rootNode);
    while(!s.isEmpty())
    {
        Node n = (Node)s.peek();
        Node child = getChildNode(n);
        if (child != null)
        {
            child.visited = true;
            printNode(child);
            s.push(child);
        }
        else
    }
}

```

```

    {
        s.pop();
    }
}

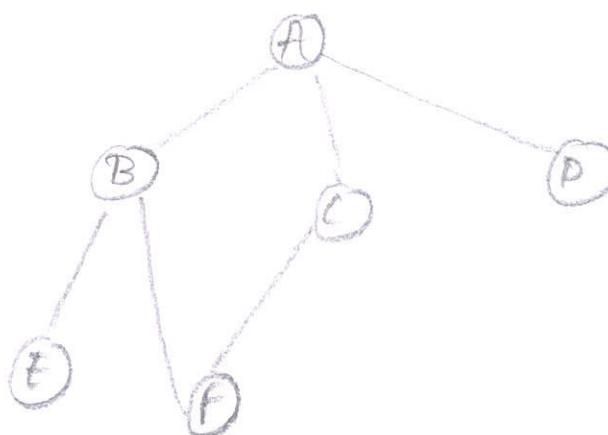
```

// clear visited property of nodes
 clearNodes();

}

Breadth First Search(BFS)

This is a very different approach for traversing the graph nodes. The aim of BFS algorithm is to traverse the graph as close as possible to the root node, Queue is used in the implementation of the breadth first search. Let's see how BFS traversal works with respect to the following graph.



If we do the breadth first traversal of the above graph and print the visited node as the output. it will print the following output. "AB(CDEF)"
 The BFS visits the nodes level by level, so it will start with level 0 which is the root node, and then it moves to the next levels which are B, C and D. then the levels which are E and F.

Algorithmic steps

step 1: push the root node in the queue.

step 2: Loop until the queue is empty.

step 3: Remove the node from the queue.

step 4: if the removed node has unvisited child nodes mark them as visited and insert the unvisited children in the queue.

`public void bfs()`

{

// BFS uses Queue data structure

Queue q = new linkedlist();

q.add(this, rootNode);

printNode(this, rootNode);

rootNode.visited = true;

while (!q.isEmpty())

{

Node n = (Node) q.remove();

Node child = null;

while ((child = getChildNode(n)) != null)

{

child.visited = true;

printNode(child);

q.add(child);

}

// clear visited property of nodes

clearNodes();

}

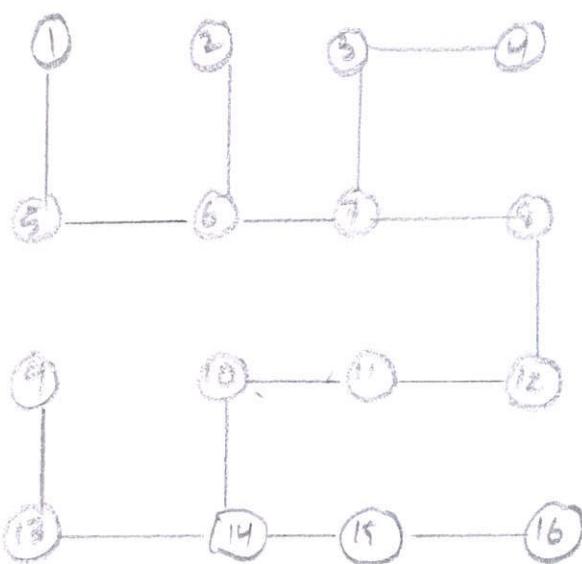
spanning trees;

170

In the mathematical field of graph theory a spanning tree T of a connected undirected graph G is a tree composed of all the vertices and some of the edges of G . Informally, a spanning tree of G is a selection of edges of G that form a tree spanning every vertex. That is, every vertex lies in the tree but no cycles are formed. On the other hand, every bridge of G must belong to T .

A spanning tree of a connected graph G can be defined as a maximal set of edges of G that contains no cycle, or as a minimal set of edges that connect all vertices.

Example.



Spanning forests

A spanning forest is a type of subgraph that generalises the concept of spanning tree. However, there are two definitions in common use. One is that a spanning forest is a subgraph that consists of a spanning tree in each connected component of a graph.

counting spanning trees

121

The number $t(G)$ of spanning trees of a connected graph is an important invariant. In some cases, it is easy to calculate $t(G)$ directly. It is also widely used in data structures in different computer languages. For example, if G is itself a tree, then $t(G)=1$, while G is the cycle graph C_n with n vertices, then $t(G)=n$. For any graph G , the number $t(G)$ can be calculated using Kirchhoff's matrix-tree theorem.

Cayley's formula is a formula for the number of spanning trees in the complete graph K_n with n vertices. The formula states that $t(K_n)=n^{n-2}$. Another way of stating Cayley's formula is that there are exactly n^{n-2} labelled trees with n vertices. Cayley's formula can be proved using Kirchhoff's matrix-tree theorem or via the Prüfer code.

If G is the complete bipartite graph $K_{p,q}$, then $t(G)=p^{q-1}q^{p-1}$. While if G is the n -dimensional hypercube graph Q_n , then

$$t(G) = 2^{2^n - n - 1} \prod_{k=2}^n k^{\binom{n}{k}}$$

These formulae are also consequences of the matrix-tree theorem.

Uniform spanning trees

122

A spanning tree chosen randomly from among all the spanning trees with equal probability is called a uniform spanning tree (UST). This model has been extensively researched in probability and mathematical physics.

Algorithms.

The classic spanning tree algorithm, depth-first search (DFS), is due to Robert Tarjan. Another important algorithm is based on breadth-first search (BFS).

Planar Graphs :-

In graph theory, a planar graph is a graph that can be embedded in the plane, i.e., it can be drawn on the plane in such a way that its edges intersect only at their endpoints.

A planar graph already drawn in the plane without edge intersections is called a plane graph or planar embedding of the graph. A plane graph can be defined as a planar graph with a mapping from every node to a point in 2D space, and from every edge to a plane curve, such that the extreme points of each curve are the points mapped from its end nodes, and all curves are disjoint except on their extreme points. Plane graphs can be encoded by combinatorial maps.

It is easily seen that a graph that can be drawn on the plane can be drawn on the sphere as well, and vice versa.

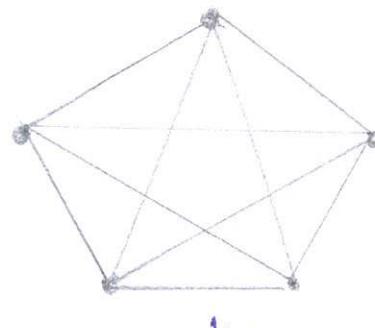
The equivalence class of topologically equivalent drawings on the sphere is called a planar map. Although a plane graph has an external or unbounded face, none of the faces of a planar map have a particular status. Applications:-

- Telecommunications - e.g. spanning trees
- vehicle routing - e.g. planning routes on roads without underpasses.
- VLSI - e.g. laying out circuits on computer chip.
- The puzzle game planarity requires the player to "untangle" a planar graph so that none of its edges intersect.

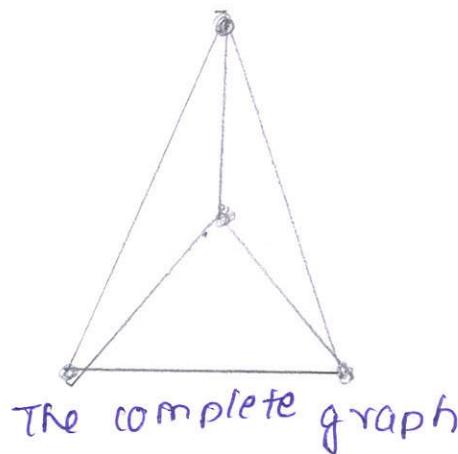
Example graphs



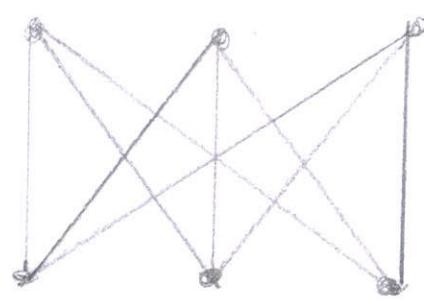
Butterfly graph



K_5



The complete graph



$K_{3,3}$
unit-5, pg - 14/24

Graph theory and applications.

124

Graphs are among the most ubiquitous models of both natural and human-made structures. They can be used to model many types of relations and process dynamics in physical, biological and social systems. Many problems of practical interest can be represented by graphs.

In computer science, graph are used to represent networks of communication, data organization, computational devices, the flow of computation, etc. One practical example. The link structure of a website could be represented by a directed graph. The vertices are the web pages available at the website and a directed edge from page A to page B exists if and only if A contains a link to B. A similar approach can be taken to problems in travel, biology, computer chip design, and many other fields. The development of algorithms to handle graphs is therefore of major interest in computer science. There, the transformation of graphs is often formalized and represented by graph rewrite systems. They are either directly used or properties of the rewrite systems are studied, complementary to graph transformation systems focussing on rule-based in-memory manipulation of graphs are graph databases geared towards transaction-safe, persist storing and querying of graph-structured data.

Graph-theoretic methods, in various forms, have proven particularly useful in linguistics, since natural language often lends itself well to discrete structure. Traditionally, syntax and compositional semantics follow tree-based structures, whose expressive power lies in the principle of compositionality, modeled in a hierarchical graph. Within lexical semantics, especially as applied to computers, modeling word meaning is easier when a given word is understood in terms of related words. Semantic networks are therefore important in computational linguistics. Still other methods in phonology and morphology are common in the analysis of language as a graph. Indeed, the usefulness of this area of mathematics to linguistics has borne organizations such as TextGraphs as well as various 'Net' projects, such as coordNet, verbNet and others.

Graph theory is also used to study molecules in both chemistry and physics. In condensed matter physics, the three dimensional structure of complicated simulated atomic structures can be studied quantitatively by gathering statistics on graph-theoretic properties related to the topology of the atoms; for example, Franzblau's shortest-path rings. In chemistry a graph makes a natural model for a molecule, where vertices structures, ranging from chemical editors to database searching.

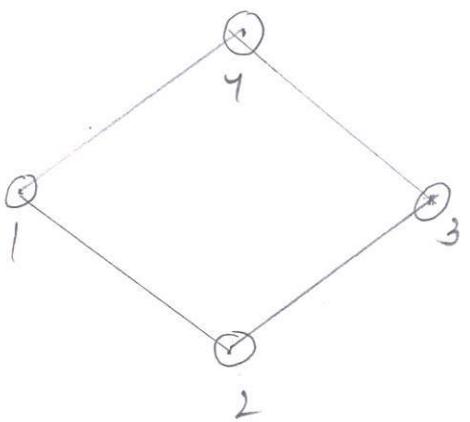
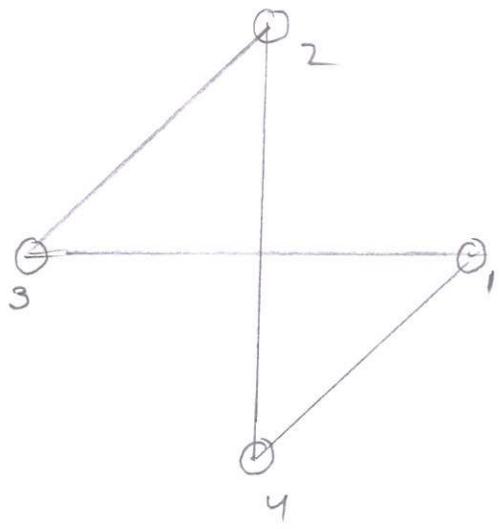
In mathematics, graphs are useful in geometry and certain parts of topology, e.g knot theory. Algebraic graph theory has close links with group theory.

A graph structure can be extended by assigning a weight to each edge of the graph. Graphs with weights, or weighted graphs, are used to represent structure in which pairwise connections have some numerical values. For example if a graph represents a road network, the weights could represent the length of each road.

Basic Concepts Isomorphism,

Let G_1 and G_2 be two graphs and let f be a function from the vertex set of G_1 to the vertex set of G_2 . Suppose that f is one-to-one and onto & $f(v)$ is adjacent to $f(w)$ in G_2 if and only if v is adjacent to w in G_1 .

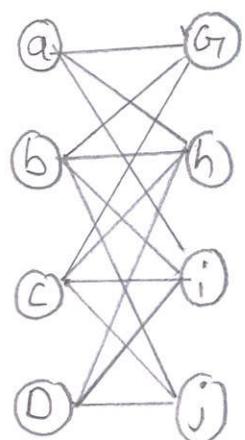
Then we say that the function f is an isomorphism and that the two graphs G_1 and G_2 are isomorphic. So two graphs G_1 and G_2 are isomorphic if there is a one-to-one correspondence between vertices of G_1 and those of G_2 with the property that if two vertices of G_1 are adjacent then so are their images in G_2 . If two graphs are isomorphic then as far as we are concerned they are the same graph though the location of the vertices may be different. To show you how the program can be used to explore isomorphisms draw the graph in figure 4 with program.



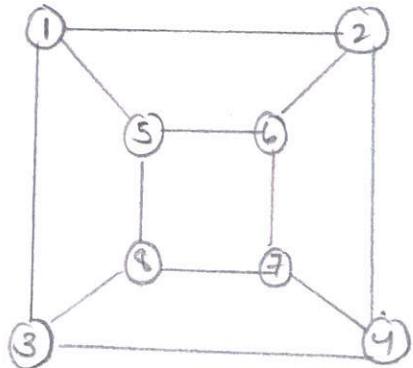
Example:

The two graphs shown below are isomorphic, despite their different looking drawings

graph G



graph H



An isomorphism between G and H

$$f(b) = 6$$

$$f(c) = 4$$

$$f(d) = 3$$

$$f(g) = 5$$

$$f(h) = 2$$

$$f(i) = 4$$

$$f(j) = 7$$

Subgraphs:

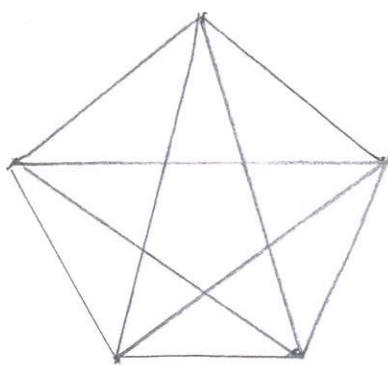
① A subgraph of a graph G_1 is a graph whose vertex set is a subset of that of G_1 , and whose adjacency relation is a subset of that of G_1 restricted to this subset. In the other direction, a supergraph of a graph G_1 is a graph of which G_1 is a subgraph, we say a graph G_1 contains another graph H if some subgraph of G_1 is H or is isomorphic to H .

② A subgraph H is a spanning subgraph, or factor of a graph G_1 if it has the same vertex set as G_1 . We say H spans G_1 .

③ A subgraph H of a graph G_1 is said to be induced if, for any pair of vertices x and y of H , xy is an edge of H if and only if xy is an edge of G_1 . In other words, H is an induced subgraph of G_1 if it has all the edges that appear in G_1 over the same vertex set. If the vertex set of H is the subset S of $V(G_1)$, then H can be written as $G_1[S]$ and is said to be induced by S .

④ A graph that does not contain H as an induced subgraph is said to be H -free.

⑤ A universal graph in a class k of graphs is a simple graph in which every element in k can be embedded as a subgraph.



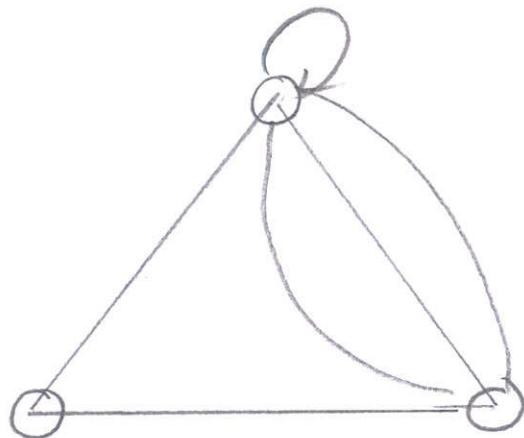
K_5 . a complete graph. if a subgraph looks like this, the vertices in that subgraph form a clique of size 5.

Multi graphs:

In mathematics, a multigraph or pseudograph is a graph which is permitted to have multiple edges. that is that have the same end nodes, thus two vertices may be connected by more than one edge. formally, a multigraph G is an ordered pair $G := (V, E)$ with.

- V a set of vertices or nodes,
- E a multiset of unordered pairs of vertices called edges or lines.

multigraphs might be used to model the possible flight connections offered by an airline. In this case the multigraph would be a directed graph with pairs of directed parallel edges connecting cities to show that it is possible to fly both to and from these locations.



A multigraph with multiple edges and a loop

Euler circuits:

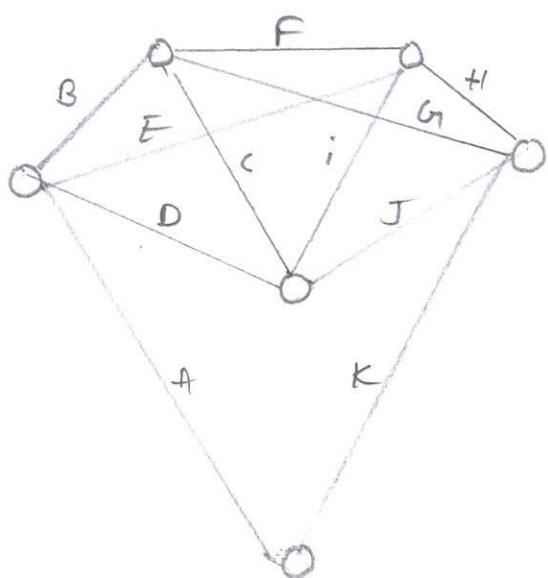
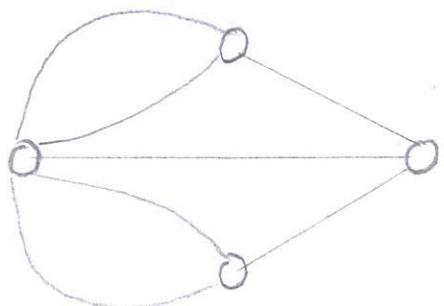
In graph theory an Eulerian trail is a trail in a graph which visits every edge exactly once. Similarly an Eulerian circuit is an Eulerian trail which starts and ends on the same vertex. They were first discussed by Leonhard Euler while solving the famous Seven Bridges of Königsberg problem in 1736. Mathematically the problem can be stated like this.

Given the graph on the right, is it possible to construct a path which visits each edge exactly once?

Euler proved that a necessary condition for the existence of Eulerian circuits is that all vertices in the graph have an even degree, and stated without proof that connected graphs with all vertices of even degree have an Eulerian circuit. The first complete proof of this latter claim was published in 1873 by Carl Hierholzer.

An Eulerian trail, Eulerian trail or Euler walk in an undirected graph is a path that uses each edge exactly once. If such a path exists, the graph is called traversable or semi-eulerian.

An Eulerian cycle, Eulerian circuit or Euler tour in an undirected graph is a cycle that uses each edge exactly once. If such a cycle exists, the graph is called unicursal. While such graphs are Eulerian graphs, not every Eulerian graph possesses an Eulerian cycle.



Hamiltonian graphs:

A Hamiltonian path or traceable path is a path that visits each vertex exactly once. A graph that contains a Hamiltonian path is called a traceable graph. A graph is Hamilton-connected if for every pair of vertices there is a Hamiltonian path between the two vertices.

A Hamiltonian cycle, Hamiltonian circuit, vertex tour or graph cycle is a cycle that visits each vertex exactly once. A graph that contains a Hamiltonian cycle is called a hamiltonian graph.

A Hamiltonian decomposition is an edge decomposition of a graph into Hamiltonian circuits.

Examples:

- a complete graph with more than two vertices is Hamiltonian.
- every cycle graph is Hamiltonian.
- every tournament has an odd number of Hamiltonian paths.
- every platonic solid, considered as a graph is Hamiltonian .

Chromatic Numbers :-

133

In graph theory, graph coloring is a special case of graph labeling: it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color; this is called a vertex coloring, and a face coloring of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color.

