Graph Theory

Zahra Norozi :91100413

DISCRETE MATHEMATICS WITH APPLICATIONS (chapter 10) // **SUSANNA S. EPP** DePaul University

Learning Objectives

Explore how graphs are represented in computer memory
Learn about Euler and Hamilton circuits
Learn about isomorphism of graphs
Explore various graph algorithms
Examine planar graphs and graph coloring

Matrix Representation of a Graph

Adjacency Matrices

Let G be a graph with n vertices, where $n > 0$. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$. The **adjacency matrix** A_G with respect to the particular listing, $v_1, v_2, \ldots, v_n$, of n vertices of G is an $n \times n$ matrix $[a_{ij}]$ such that the (i, j)th entry a_{ij} of A_G is the number of edges from v_i to v_j. That is,

$$a_{ij} = \text{ the number of edges from } v_i \text{ to } v_j.$$

Because a_{ij} is the number of edges from v_i to v_j, the adjacency matrix A_G is a square matrix over the set of nonnegative integers.

Consider the graph in Figure 10.42.

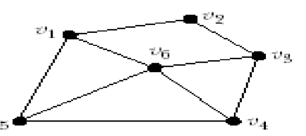

FIGURE 10.42 A graph

The vertices of the graph are listed as v_1, v_2, v_3, v_4, v_5, and v_6. The adjacency matrix of this graph with respect to this ordering of vertices is

$$A_G = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \qquad (10.1)$$

The vertices of the graph are listed as v_1, v_2, v_3, and v_4. The adjacency matrix of this graph with respect to this ordering of vertices is

$$A_G = \begin{bmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Sometimes for convenience we label the columns and rows by $v_1, v_2, v_3, v_4, \ldots$ as follows:

$$A_G = \begin{array}{c c} & \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} & \begin{bmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array} \qquad (10.2)$$

Notice that the matrix A_G is a symmetric matrix, i.e., $a_{ij} = a_{ji}$.

The adjacency matrix A_G of a graph has the following properties.

1. If G does not contain any loops and parallel edges, then each element of A_G is either 0 or 1.
2. If G does not contain any loops, then all of the diagonal elements of A_G are 0.

Theorem 10.3.5: Let G be a graph with n vertices, $v_1, v_2, \ldots, v_n$, and let $A = [a_{ij}]$ denote the adjacency matrix with respect to this ordering of the vertices of G. For any positive integer k, $A^k = [b_{ij}]$ denotes the matrix multiplication of k copies of A. Then b_{ij} denotes the number of distinct walks of length k from vertex v_i to vertex v_j in graph G.

Let G be a graph with n vertices, $v_1, v_2, \ldots, v_n$, where $n > 0$ and m edges $e_1, e_2, \ldots, e_m$. The **incidence matrix** I_G with respect to the ordering $v_1, v_2, \ldots, v_n$ of n vertices and m edges $e_1, e_2, \ldots, e_m$ is an $n \times m$ matrix $[a_{ij}]$ such that

$$a_{ij} = \begin{cases} 0 & \text{if } v_i \text{ is not an end vertex of } e_j, \\ 1 & \text{if } v_i \text{ is an end vertex of } e_j, \text{ but } e_j \text{ is not a loop,} \\ 2 & \text{if } e_j \text{ is a loop at } v_i. \end{cases}$$

DEFINITION The vertices of this graph G are v_1, v_2, v_3, and v_4 and the edges are e_1, e_2, e_3, e_4, and e_5. For incidence matrices we consider this ordering of vertices and edges. For incidence matrices we label the rows by v_1, v_2, v_3, and v_4 and the columns by e_1, e_2, e_3, e_4, and e_5. Then the incidence matrix I_G with respect to the above ordering of vertices and edges is the following 4×5 matrix.

$$\begin{array}{c c} & \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} & \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Notice that the sum of the ith row is the degree of v_i. This is true for any incidence matrix.

Special Circuits

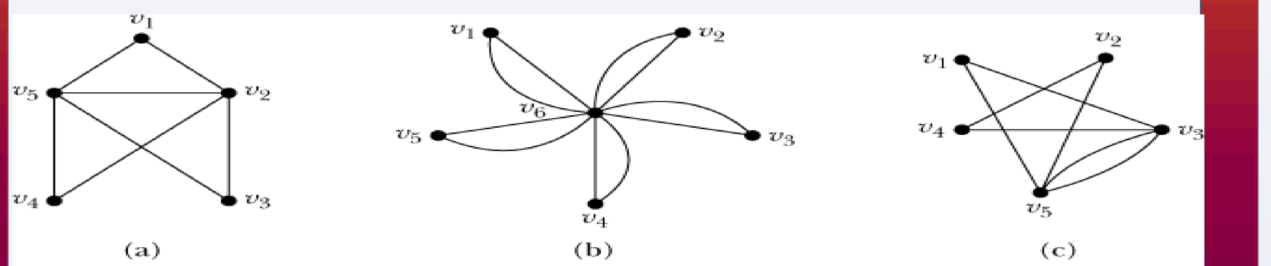

FIGURE 10.59 Eulerian graphs

DEFINITION A circuit in a graph that includes all the edges of the graph is called an **Euler circuit**.

A graph G is said to be **Eulerian** if either G is a trivial graph or G has an Euler circuit.

Theorem 10.4.4: If a connected graph G is Eulerian, then every vertex of G has even degree.

Lemma 10.4.5: Let G be a connected graph with one or two vertices. If every vertex of G is of even degree, then G has an Euler circuit.

Theorem 10.4.6: Let G be a connected graph such that every vertex of G is of even degree. Then G has an Euler circuit.

Since 1736, two additional bridges have been constructed on the Pregel River, one between regions B and C, another between regions A and D. The graph with the additional two bridges is shown in Figure 10.60.

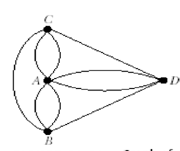

FIGURE 10.60 Graph of the Königsberg bridge problem with two additional bridges

This is a connected graph with each vertex of even degree. Hence, this graph has an Euler circuit.

Theorem 10.4.16: Let $G = (V, E)$ be a simple graph with n vertices such that G contains a Hamiltonian cycle. Let $v_1, v_2, \ldots, v_t \in V$, where $t < n$. Then the number of components in the subgraph $G_1 = G - \{v_1, v_2, \ldots, v_t\}$ is less than or equal to t.

Theorem 10.4.17: Let G be a simple graph with $n > 2$ vertices. If each vertex has degree at least $\frac{n}{2}$, then G has a Hamiltonian cycle.

Theorem 10.4.18: Let G be a simple connected graph with n vertices, where $n > 2$. If for any two vertices u and v of G, such that u and v are not adjacent, $\deg(u) + \deg(v) \geq n$, then G has a Hamiltonian cycle.

REMARK 1 The directed Hamiltonian cycle and the directed Hamiltonian path of a directed graph are a directed path and a directed circuit of the graph, respectively, containing each vertex of the graph.

A cycle in a graph G is called a **Hamiltonian cycle** if it contains each vertex of G.

From Definition 10.4.12, it follows that a Hamiltonian cycle is a closed trail that contains each vertex of the graph exactly once.

If a graph G has a Hamiltonian cycle, then G is called a **Hamiltonian graph**. A path in a graph G is called a **Hamiltonian path** if it contains each vertex of G.

Definition :An open trail in a graph is called an **Euler trail** if it contains all the edges and all the vertices

This diagram in Figure 10.63 is a connected graph with 20 vertices. Each Vertex represents a famous city. It follows that the game is equivalent to finding a cycle in the graph in Figure 10.63 that contains each vertex exactly once except for the starting and ending vertices, which appear twice, making a Hamiltonian Cycle.

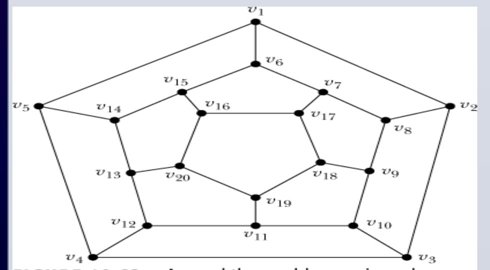

FIGURE 10.63 Around the world game in a plane

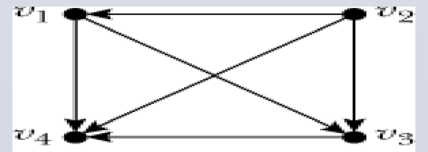

FIGURE 10.70 A digraph representing the outcome of the tournament

During a certain soccer tournament with 4 teams, each team has played against all the others exactly once and there were no ties. All the teams can be listed in order so that each has defeated the team next on the list.

Let the teams be denoted by v_1, v_2, v_3, and v_4 and let the matches correspond to the vertices and the arcs of a directed graph, respectively, in such a way that the initial and terminal vertices of an arc correspond to the winner and loser, respectively, of the corresponding match.

$v_2 \rightarrow v_1 \rightarrow v_3 \rightarrow v_4$ is a Hamiltonian directed path.

Isomorphism

DEFINITIO Let $G_1 = (V_1, E_1, g_1)$ and $G_2 = (V_2, E_2, g_2)$ be two graphs. G_1 is said to be **isomorphic** to G_2 if there exist a one-to-one correspondence $f : V_1 \rightarrow V_2$ and a one-to-one correspondence $h : E_1 \rightarrow E_2$ in such a way that for any edge $e_k \in E_1$, $g_1(e_k) = \{v_i, v_j\}$ in G_1 if and only if $g_2(h(e_k)) = \{f(v_i), f(v_j)\}$ in G_2.

REMARK Definition 10.5.1 can also be stated as follows:

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. G_1 is said to be *isomorphic* to G_2 if there exist a one-to-one correspondence $f : V_1 \rightarrow V_2$ and a one-to-one correspondence $h : E_1 \rightarrow E_2$ such that for any edge e_k in E_1, vertices v_i, v_j are end vertices of e_k in G_1 if and only if $f(v_i), f(v_j)$ are end vertices of $h(e_k)$ in G_2.

When we say that two graphs are the same we mean they are isomorphic to each other.

Theorem 10.5.5: Let G, G_1, G_2, and G_3 be graphs. Then the following assertions hold.

(i) G is isomorphic to itself.
(ii) If G_1 is isomorphic to G_2, then G_2 is isomorphic to G_1.
(iii) If G_1 is isomorphic to G_2 and G_2 is isomorphic to G_3, then G_1 is isomorphic to G_3.

Two graphs G_1 and G_2 are said to be *isomorphic*, written $G_1 \simeq G_2$, if G_1 is isomorphic to G_2.

Two graphs G_1 and G_2 are said to be **different** if G_1 is not isomorphic to G_2.

Theorem 10.5.8: Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two simple graphs. G_1 is isomorphic to G_2 if there exists a one-to-one correspondence $f : V_1 \rightarrow V_2$ such that vertices v_i, v_j are adjacent vertices in G_1 if and only if $f(v_i), f(v_j)$ are adjacent vertices in G_2.

Theorem 10.5.9: Let G_1 and G_2 be two graphs such that G_1 is isomorphic to G_2. Then G_1 has a vertex of degree k if and only if G_2 has a vertex of degree k.

Theorem 10.5.12: Let G_1 and G_2 be two graphs such that G_1 is isomorphic to G_2. Then G_1 has a cycle of length k if and only if G_2 has a cycle of length k.

Theorem 10.5.13: Two simple graphs are isomorphic if and only if their vertices can be labeled in such a way that the corresponding adjacency matrices are equal.

In Figure 10.77, *v*1 and *v*2 are the end vertices of edge *e*1 in *G*1 and *f(v*1*)* = *u*1 and *f(v*2*)* = *u*3 are end vertices of edge *h(e*1*)* = *f*2 in *G*2. Also, *f(v*3*)* = *u*4 and *f(v*1*)* = *u*1 are the end vertices of edge *f*4 = *h(e*2*)* in *G*2 and *v*3 and *v*1 are end vertices of edge *e*2 in *G*1.
Similarly, for other vertices, any two vertices vi and v j are end vertices of some edge e k in G1 if and only if f(vi) and f(v j) are end vertices of edge h(e k) in G2.
G1 is therefore isomorphic to G2

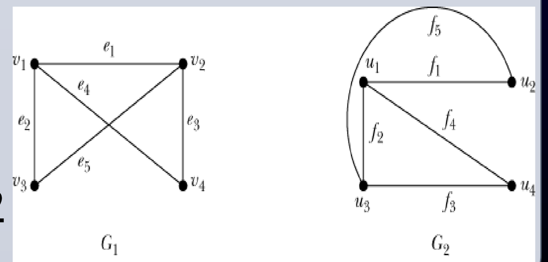

FIGURE 10.77 Graphs G_1 and G_2

Graph Algorithms

Graphs can be used to show how different chemicals are related or to show airline routes. They can also be used to show the highway structure of a city, state, or country. The edges connecting two vertices can be assigned a nonnegative real number, called the **weight** of the edge . If the graph represents a highway structure, the weight can represent the distance between two places, or the travel time from one place to another. Such graphs are called **weighted graphs**.

Let G be a graph with n vertices, where $n > 0$. Let $V = \{v_1, v_2, \ldots, v_n\}$ be the vertex set of G. We list the vertices of G as $v_1, v_2, \ldots, v_n$. Let W be an $n \times n$ matrix such that its (i, j)th entry, for $i \neq j$, $W[i, j]$ is given by

$$W[i,j] = \begin{cases} w_{ij} & \text{if } v_i - v_j \text{ is an edge in } G \text{ and } w_{ij} \text{ is the weight of the edge } v_i - v_j, \\ \infty & \text{if there is no edge from } v_i \text{ to } v_j. \end{cases}$$

Also, $W[i, i] = 0$ for all i. The matrix W is called the **weight matrix** of graph G.

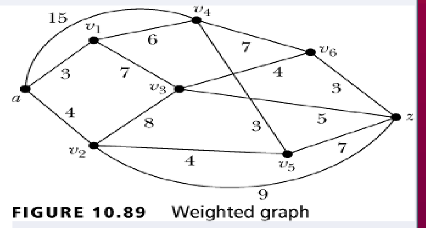

FIGURE 10.89 Weighted graph

Shortest Path Algorithm

Let G be a weighted graph. Let u and v be two vertices in G, and let P be a path in G from u to v. The **length** of path P, written $l(P)$, is the sum of the weights of all the edges on path P, which is also called the **length** of v from u via P.

Consider graph G of Figure 10.89. Let $V = \{a = v_0, v_1, v_2, v_3, v_4, v_5, v_6, z\}$ be the vertex set of G. Suppose we need to find a path from a to z. Let $P_1 : a - v_1 - v_4 - v_5 - z$, $P_2 : a - v_2 - z$, and $P_3 : a - v_1 - v_3 - z$. Then $l(P_1) = 19$, $l(P_2) = 13$, and $l(P_3) = 15$. Now P_1, P_2, and P_3 are paths from a to z. However, among these paths, the length of P_2 is the shortest. If the vertices in graph G represent cities and the weights of the edges represent the travel time between cities, then among paths P_1, P_2, and P_3 traveling from a to z via path P_2 is the fastest.

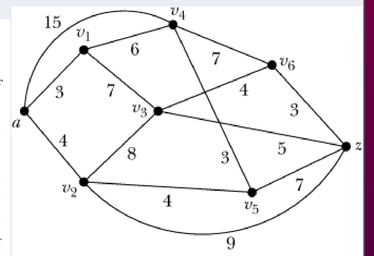

FIGURE 10.89 Weighted graph

Dijkstra's Shortest Path Algorithm

Dijkstra's algorithm iteratively constructs the set S that consists of all the vertices of G for which the length of a shortest path has been determined. Initially, $S = \emptyset$. Let $N = V - S$, where V is the set of all vertices of G. It follows that initially $N = V$. Moreover, $V = S \cup N$.

For each vertex in $v \in V$, we assign the label $L(v)$ as follows:

a. Initially, $L(a) = 0$ and $L(v) = \infty$ for all other vertices of V.
b. If $v \in S$, then $L(v)$ gives the length of a shortest path from a to v.
c. After each iteration of the algorithm, the value of $L(v)$ for certain vertices of V, as described below, is updated.

At the termination of the algorithm, $z \in S$ and $L(z)$ gives the length of a shortest path from a to z.

In Dijkstra's shortest path algorithm, at each iteration of the algorithm, we choose a vertex $v \in N$ such that

$$L(v) = \min\{L(u) \mid u \in N\}.$$

Next the vertex v is added to S, removed from N, and for all vertices $w \in N$ that are adjacent to v, we check whether the path from a to w via v (using the current shortest path from a to v) is shorter than the current path from a to w. This is done by checking whether

$$L(w) > L(v) + W[v, u].$$

If this is true, then the value of $L(w)$ is updated as explained in the following steps.

1. $S := \emptyset$
2. $N := V$
3. For all vertices $u \in V$, $u \neq a$, $L(u) := \infty$
4. $L(a) := 0$
5. while $z \notin S$ do
 5.a Let $v \in N$ be such that $L(v) = \min\{L(u) \mid u \in N\}$
 5.b $S := S \cup \{v\}$
 5.c $N := N - \{v\}$
 5.d For all $w \in N$ such that there is an edge from v to w
  5.d.1. if $L(v) + W[v, u] < L(w)$ then $L(w) = L(v) + W[v, u]$.

ALGORITHM 10.1: Dijkstra's shortest path algorithm.

Input: G—graph
n—number of vertices in G
W—weight matrix
a—source vertex
z—destination vertex

Output: $L(z)$—the length of a shortest path from a to z

```
function DijkstraSP(G,W,a,z,n)
begin
  S := ∅;
  N := vertices in G;
  for all u ∈ N do
    L[u] := ∞;
  L[a] := 0;
  while z ∉ S do
  begin
    min := ∞;
    for all u ∈ N do
      if L[u] < min then
      begin
        min := L[u];
        v := u;
      end
    S := S ∪ {v};
    N := N − {v};
    for all w ∈ N do
      if (v,w) is an edge in G
          and L[v] + W[v,w] < L[w] then
        L[w] := L[v] + W[v,w];
  end
    return L[z];
end
```

Theorem 10.6.2: Let G be a weighted graph with n vertices, $n > 0$. Let a and z be two vertices in G. Dijkstra's algorithm correctly finds the length of a shortest path from vertex a to vertex z.

Theorem 10.6.4: In the worst case, Dijkstra's shortest path algorithm is $\Theta(n^2)$.

Let G be a directed graph and u and v be two vertices on G. If there is a path from u to v, then we say that u is a **predecessor** of v and v is a **successor** of u. If there is an edge from u to v, then we say that u is an **immediate predecessor** of v and v is an **immediate successor** of u.

Let G be a directed graph with the vertex set $V = \{v_1, v_2, \ldots, v_n\}$, where $n \geq 0$. A **topological ordering** of V is a linear ordering $v_{i1}, v_{i2}, \ldots, v_{in}$ of the vertices such that, if v_{ij} is a predecessor of v_{ik}, then v_{ij} precedes v_{ik}; that is, $j < k$ in this linear ordering $1 \leq j \leq n, 1 \leq k \leq n$.

Planar Graphs and Graph Coloring

DEFINITI A graph G is called a **planar graph** if it can be drawn in the plane such that no two edges intersect except at the vertices, which may be the common end vertices of the edges.

A graph drawn in the plane (on paper or a chalkboard) is called a **plane graph** if no two edges meet at any point except the common vertex, if they meet at all.

The set of edges that bound a region is called its **boundary**. Of course, there exists a region of infinite area in any plane graph G. This is the part of the plane that lies outside the planar representation of G. This region is called the **exterior face**. A face that is not exterior is called an **interior face**. We illustrate these concepts by the following planar representations of some planar graphs.

Theorem 10.7.7: Euler. Let G be a connected planar graph with n_v vertices, n_e edges, and n_f faces. Then $n_v - n_e + n_f = 2$.

Corollary 10.7.8: The graph $K_{3,3}$ is not a planar graph.

Theorem 10.7.10: Let G be a connected simple planar graph with $n_v \geq 3$ vertices and n_e edges. Then

$$n_e \leq 3n_v - 6.$$

Corollary 10.7.11: The graph K_5 is not a planar graph.

A graph H is said to be a **subdivision of a graph** G if there exist graphs $H_1, H_2, \ldots, H_{n-1}, H_n$, such that $H_0 = G$, $H_n = H$, and H_i is obtained from H_{i-1} by a one-step subdivision of an edge of H_{i-1} for $i = 1, 2, \ldots, n$.

If a graph H is a subdivision of a graph G, then we say that H is obtained from G by subdividing edges of G.

Two graphs G and H are said to be **homeomorphic** graphs if there is an isomorphism from a subdivision of G to a subdivision of H.

Theorem 10.7.16: Kuratowski. A simple graph is planar if and only if it does not contain a subgraph homeomorphic to K_5 or $K_{3,3}$.

Graph Coloring

Let $G = (V, E)$ be a simple graph and $C = \{c_1, c_2, \ldots, c_n\}$ be a set of n colors.

(i) A **vertex coloring** of G using the colors of C is a function $f : V \rightarrow C$.
(ii) Let $f : V \rightarrow C$ be a vertex coloring of G. If for every adjacent vertices $u, v \in V$, $f(u) \neq f(v)$, then f is called a **proper vertex coloring**.

For each vertex v, its image $f(v)$ is called the *color* of v.

It follows that a vertex coloring of a graph G is an assignment of the colors $c_1, c_2, \ldots, c_n$ to the vertices of graph G. Similarly, a proper vertex coloring of G is an assignment of the colors $c_1, c_2, \ldots, c_n$ to the vertices of graph G such that adjacent vertices have different colors.

Let $G = (V, E)$ be a simple graph and $C = \{c_1, c_2, \ldots, c_n\}$ be a set of n colors.

(i) An **edge coloring** of G using the colors of C is a function $f : E \rightarrow C$.
(ii) Let $f : E \rightarrow C$ be an edge coloring of G. If for every two edges e_1 and e_2 meeting at a common vertex $f(e_1) \neq f(e_2)$, then f is called a **proper edge coloring**.

For each edge e, its image $f(e)$ is called the *color* of e.

It follows that a proper edge coloring of a graph G is an assignment of the colors $c_1, c_2, \ldots, c_n$ to the edges of graph G such that any two edges meeting at a common vertex have different colors.

The smallest number of colors needed to make a proper vertex coloring of a simple graph G is called the **chromatic number** of G. The chromatic number of G is denoted by $\chi(G)$.

Theorem 10.7.20: Let G be a nontrivial simple graph. Then $\chi(G) = 2$ if and only if G is a bipartite graph.

Let G be a graph with vertices $v_1, v_2, \ldots, v_{n-1}, v_n$. The maximum of the integers $\deg(v_i)$, $i = 1, 2, \ldots, n$ is denoted by $\Delta(G)$. That is,

$$\Delta(G) = \max\{\deg(v_i) \mid i = 1, 2, \ldots, n\}.$$

The smallest number of colors needed to make a proper coloring of edges of a simple graph G is called the **chromatic index** of G. The chromatic index of G is denoted by $\chi'(G)$.

Theorem 10.7.27: For any simple graph G, $\chi'(G) = \Delta(G)$ or $\chi'(G) = \Delta(G) + 1$.

Theorem 10.7.23: For any simple graph G, $\chi(G) \leq \Delta(G) + 1$.

Conclusions : Graph theory is an exceptionally rich area for programmers and designers. Graphs can be used to solve some very complex problems, such as least cost routing, mapping, program analysis, and so on . Network devices, such as routers and switches, use graphs to calculate optimal routing for traffic . Far from being difficult to understand, graphs lend themselves well to Java implementation. As usual with Java and as I've often remarked in the past, you get a lot of power from a really small amount of code.