Graph Theory

Part of Texas Counties.

We would like to visit each of the above counties, crossing each county only once, starting from Harris county. Is this possible? This problem can be modeled as a graph. We can represent each county as a vertex or dot and connect the counties that are adjacent with edges.
A graph model of the counties shown in the picture above.
If we start at vertex Ha, travel along edge e2 to vertex Ch, travel along edge e6 to Je, and so on and eventually arrive to vertex Or, is called a path.

Our problem can then be re-phrased as follows: Is there a path from vertex Ha to vertex Ha that traverses every vertex exactly once?

Definition: Graph
A graph (or undirected graph) G consists of a set V of vertices (or nodes) and a set E of edges (or arcs) such that each edge e belongs to E is associated with an unordered pair of vertices. If the edge is unique then we note e = (v, w)
If G is a graph with a set of vertices V and a set of edges E, we write: G = (V, E).
The set V and E are assumed to be finite and V is assumed to be non-empty.

Incidence:
An edge e in a graph that is associated with the pair of vertices v and w is said to be incident on v and w.
v and w are said to be incident on e and adjacent vertices.

Directed Graph:
A directed graph or digraph G consists of a set V of vertices (or nodes) and a set E of edges (or arcs) such that each edge e is associated with an ordered pair of vertices. If there is a unique edge e associated with the ordered pair (v, w), we can write e = (v, w)

Unless specified otherwise, the sets E and V are assumed to be finite and V is assumed to be nonempty.
In directed graphs, two distinct edges can be associated with the same pair of vertices. Such edges are called parallel edges. An edge incident on a single vertex is called a loop.

A graph that has neither loops nor parallel edges, is called a simple graph.

Weighted Graphs:  
Frequently in manufacturing, it is necessary to bore many holes in sheets of metal. Components can then be bolted to these sheets. A drill, controlled by a computer is used to drill the holes. To save time and money, the drill press should be moved as quickly as possible. We model the situation as a graph.  
The vertices of the graph are the holes to be drilled. Every pair is connected by an edge. We write in each edge the time to move the drill press between the corresponding holes.
A graph with numbers on the edges is called a weighted graph. If edge e is labeled k.
The weight of edge (c,e) is 5.
In a weighted graph, the length of a path is the sum of the weights of the edges in a path.

Suppose that in this problem, the path is required to begin at vertex a and end at vertex e. We can find the minimum-length path by listing all possible paths from a to e that pass through every vertex exactly one time and choose the shortest one.

<table>
<thead>
<tr>
<th>Path</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>a, b, c, d, e</td>
<td>21</td>
</tr>
<tr>
<td>a, b, d, c, e</td>
<td>28</td>
</tr>
<tr>
<td>a, c, b, d, e</td>
<td>24</td>
</tr>
<tr>
<td>a, c, d, b, e</td>
<td>26</td>
</tr>
<tr>
<td>a, d, b, c, e</td>
<td>27</td>
</tr>
<tr>
<td>a, d, c, b, e</td>
<td>22</td>
</tr>
</tbody>
</table>

We see that the path that visits the vertices a, b, c, d, e, in this order has the minimum length.

Listing all paths from vertex v to vertex w, as we did, is time consuming. Unfortunately, no one knows a method that is much more practical for arbitrary graphs. This problem is a version of the traveling salesperson problem.

However there are several efficient algorithms for finding a path in a graph that has a root and children, a tree.
Similarity Graphs:

Suppose that we want to design a program that finds students who cheat in their programming assignments by copying somebody’s else program. We want to group like program into classes based on certain properties of the program. Suppose we select:

1. The number of lines in the program
2. The number of return statements in the program
3. The number of function calls in the program.

<table>
<thead>
<tr>
<th>Program</th>
<th># of program lines</th>
<th># return statements</th>
<th># function calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>66</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>41</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>68</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>34</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>75</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

A similarity graph \( G \) is constructed as follows. The vertices correspond to programs. A vertex is denoted \((p_1, p_2, p_3)\), where \( p_i \) is the value of the property \( i \).

We define a dissimilarity function \( s \) as follows. For each pair of vertices \( v= (p_1, p_2, p_3) \) and \( w=(q_1, q_2, q_3) \), we set

\[
 s(v, w)= | p_1-q_1| + | p_2- q_2| + | p3- q3 |
\]

If \( v \) and \( w \) are two vertices corresponding to two programs, \( s(v, w) \) is a measure of how dissimilar the programs are. A large value \( s(v, w) \) indicates dissimilarity, while a small value indicates similarity.
For a fixed number $S$, we insert an edge between $v$ and $w$ if $s(v, w) < S$.
We say that $v$ and $w$ are in the same class if $v = w$ or there is a path from $v$ to $w$.

In this example, if $S = 25$, the programs are grouped in three classes: 
\{1, 3, 5\}, \{2\}, \{4\}.

Definition:
A complete graph on $n$ vertices, denoted $K_n$, is the simple graph with
$n$ vertices in which there is an edge between every pair of distinct
vertices.

Definition:
A graph $G = (V, E)$ is bipartite if there exists subsets $V_1$ and $V_2$ (either
possibly empty) of $V$ such that $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$, and
each edge in $E$ is incident on one vertex in $V_1$ and one vertex in $V_2$.

The graph above is bipartite since if we let $V_1 = \{v_1, v_2, v_3\}$ and
$V_2 = \{v_4, v_5\}$
each edge is incident one vertex in V1 and one vertex in V2.

The **complete** bipartite graph on m and n vertices denoted Km,n is the simple graph whose vertex set is partitioned into sets V1 with m vertices and V2 with n vertices in which the edge set consists of all edges of the form (v1, v2) with v1 in V1 and v2 in V2.

**Paths and Cycles**

**Definition:**
Let v0 and vn be vertices in a graph. A path from v0 to vn of length n is an alternating sequence of n+1 vertices and n edges beginning with vertex v0 and ending with vertex vn,

\[(v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n)\]

in which edge ei is incident on vertices vi-1 and vi for i=1, ..., n

A connected graph is a graph in which we can go from any vertex to any other vertex on a path.

A directed graph G is strongly connected if given any vertices v and w in G, there exists a path from v to w.

**Subgraphs:**

Let G = (V, E) be a graph. We call (V’, E’) a subgraph of G if

- V’ ⊆ V and E’ ⊆ E
- For every edge e’ ∈ E’, if e’ is incident on v’ and w’, then v’ and w’ ∈ V’

Let G be a graph and let v be a vertex in G. The subgraph G’ of G consisting of all edges and vertices in G that are contained in some path beginning at v is called a component of G containing v.
Let v and w be vertices in a graph G.

-A simple path from v to w is a path from v to w with no repeated vertices.

-A cycle (or circuit) is a path of nonzero length from v to v with no repeated edges.

-A simple cycle is a cycle from v to v, in which there are no repeated vertices, except for v.

See graph 8.2.1 and find a simple path ? a cycle, a simple cycle, a subgraph.

Königsberg Bridge Problem:
The first paper in graph theory was Leonhard Euler’s in 1736. The paper presented a general theory that included a solution to what is called Königsberg bridge problem.

A graph model of the bridges of Königsberg
A problem consists of starting from any location, walk over each bridge exactly once; then return to the starting location. This amounts to finding a cycle in the graph that includes all of the edges and all the vertices.

A cycle in a graph $G$ that includes all of the edges and all of the vertices is called an Euler cycle.

There is no Euler cycle in the Königsberg problem.

Suppose that there is a simple cycle going from $v$ to $v$, it means that there is a path from vertex $v$ to vertex $v$ that traverse every edge exactly once.

Suppose that there is a simple cycle going from $v$ to $v$ and consider the vertex $y$. Each time we arrive a $y$, we must leave $y$ using a different edge. Furthermore every edge that touches $y$ must be used. Thus the edges at $y$ occur in pairs. It follows that an even number of edges much touch $y$. Since three edges touch $y$, we have a contradiction. Therefore there is no path in $G$ that traverses every edge exactly once.

**Degree of a vertex:**

The degree of a vertex $v$, $\delta(v)$, is the number of edges incident on $v$. By definition, each loop on $v$ contributes 2 to the degree of $v$.

**The Hand Shaking Theorem:**

Let $G = (V, E)$ be an undirected graph with $e$ edges. Then

$$2e = \sum_{v \in V} \text{deg}(v)$$

How many edges are there in a graph with 10 vertices each of degree of 6?

Solution: $e = 30$
Theorem 1:

If a graph $G$ has a Euler cycle, then $G$ is connected and every vertex has an even degree.

Proof: Suppose that $G$ has an Euler cycle. We argued that every vertex in $G$ has even degree. If $v$ and $w$ are vertices in $G$, the portion of the Euler cycle that takes us from $v$ to $w$ serves as a path from $v$ to $w$. Therefore $G$ is connected.

Theorem 2:
If $G$ is a connected graph and every vertex has even degree, then $G$ has an Euler cycle.

Proof is by induction.

Theorem 3:
If a graph $G$ contains a cycle from $v$ to $v$, $G$ contains a simple cycle from $v$ to $v$.

Necessary and Sufficient Conditions for Euler cycle (Theorem):
A connected graph with at least two vertices has an Euler circuit if each of its vertices has an even degree.

Algorithm for Constructing Euler Circuits:
procedure Euler($G$: connected graph with all vertices of even degree) circuit:= a circuit in $G$ beginning at an arbitrary chosen vertex with edges successively added to form a path that returns to this vertex
**Hamiltonian Cycle**

**Definition:**
A cycle in a graph G that contains each vertex in G exactly once, except for the starting and ending vertex that appears twice, is a Hamiltonian cycle.

The above graph has a Hamiltonian cycle (a, b, c, d, e, f, g, a), however it does not have an Euler cycle.

**Theorem:**
If G is a simple graph with n vertices with n ≥ 3 such that the degree of every vertex is at least n/2, then G has a Hamiltonian cycle.

**Theorem:**
If G is a simple graph with n vertices with n ≥ 3 such that: \[ \text{deg}(u) + \text{deg}(v) \geq n \] for every pair of non-adjacent vertices u and v, then G has a Hamiltonian cycle.

**Shortest-Path Problems:**
Many problems can be modeled using graphs with weights assigned to their edges.
Problems involving distances can be modeled by assigning distances between cities to the edges. Graphs that have a number assigned to each edge are called weighted graphs. Weighted graphs are used to model computer networks. Communication cost, response time or distance between computers can all be studied using weighted graphs.

Determining a path of least length between two vertices in a network is one problem that arises often.

**A shortest-Path Algorithm:**

There are several different algorithms that find a shortest path between two vertices in a weighted graph. The most famous one is an algorithm discovered by Dijkstra in 1959.

Procedure Dijkstra( G: weighted connected simple graph with all weights positive)
{G has vertices a= v0, v1, ….vn and weights w(vi, vj)
 where w(vi, vj) = ∞ if \{vi, vj\} is not an edge in G}
for i:=1 to n
   L(vi):= ∞
   L(a):=0
S:= ∅
{the labels are now initialized so that the label of a is 0 and all other labels are ∞, and S is the empty set}
while z ∉ S
begin
   u:= a vertex not in S with L(u) minimal
   S= S ∪ \{u\}
for all vertices v not in S
   if L(u) + w(u,v) < L(v) then  L(v):= L(u)+w(u,v)
   {this adds a vertex to S with minimal label and updates the labels of vertices not in S}
end { L(z)= length of shortest path from a to z}
**Theorem:**
Dijkstra’s algorithm finds the length of a shortest path between two vertices in a connected simple undirected weighted graph.

**Computational Complexity of Dijkstra Algorithm:**
We count the number of additions and comparisons. The algorithm uses no more than n-1 iterations for a graph of n vertices.

**Number of operations for each iteration:**
We can identify the vertex not in S with the smallest label using no more than n-1 comparisons. Then 2 operations consisting of an addition and a comparison. It follows that we have 2(n-1) operations in each iteration. Since we have (n-1) iterations, using no more than 2(n-1) operations we have that the algorithm is O(n²).

**Theorem 2:**
Dijkstra’s algorithm uses O(n²) operations (additions and comparisons) to find the length of a shortest path between two vertices in a connected simple undirected weighted graph with n vertices.

The traveling salesperson problem is related to the problem of finding a Hamiltonian cycle in a graph.

**Representing Graphs in Programs:**
There are many useful ways to represent graphs. One way is to represent a graph without multiple edges is to list all the edges of this graph.

**Adjacency Matrices:**
To simplify computation, graphs can be represented using matrices. Suppose that G = (V, E) is a simple graph where |V| = n. Suppose that the vertices of G are listed arbitrarily as v1, v2, … vn.
The adjacency matrix $A$ (or $A_g$) of $G$, with respect to this listing of vertices, is the $n \times n$ zero-one matrix with $1$ as its $(i,j)$ th entry when $v_i$ and $v_j$ are adjacent, and $0$ as its $(i,j)$th entry when they are not adjacent. In other words, if its adjacency matrix is $A = [a_{ij}]$, then

$$a_{ij} = \begin{cases} 
1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G \\
0 & \text{otherwise}
\end{cases}$$

We use zero-one matrices to represent directed graphs

**Isomorphism in Graphs**

**Definition:**
The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there is a one-to-one and onto function $f$ from $V_1$ to $V_2$, with the property that $a$ and $b$ are adjacent in $G_1$ if and only if $f(a)$ and $f(b)$ are adjacent in $G_2$, for all $a$ and $b$ in $V_1$.

Such a function $f$ is called an isomorphism.

In other words, when two simple graphs are isomorphic, there is a one-to-one correspondence between vertices of the two graphs that preserves the adjacency relationship.

Show that the graphs $G=(V, E)$ and $H (W, F)$ are isomorphic
**Graph Invariant:** A property preserved by isomorphism is called a graph invariant. Some properties that are preserved by isomorphism is the number of vertices and the number of edges.

- Isomorphic simple graphs must have the same number of vertices.
- Isomorphic simple graphs must have the same number of edges.

The degrees of the vertices in isomorphic simple graphs must be the same. That is, a vertex \( v \) of degree \( d \) in \( G \) must correspond to a vertex \( f(v) \) with degree \( d \) in \( H \), because a vertex \( w \) in \( G \) is adjacent to \( v \) if and only if \( f(v) \) and \( f(w) \) are adjacent.

To show that a function \( f \) from the vertex set of a graph \( G \) to the vertex set of a graph \( H \) is an isomorphism, we need to show that \( f \) preserves the presence and absence of edges.

**Adjacency Matrices and Isomorphic graphs:**
The function \( f \) is an isomorphism if the adjacency matrix of \( G \) is the same as the adjacency matrix of \( H \), when rows and columns are labeled to correspond to the image under \( f \) of the vertices in \( G \) that are the labels of these rows and columns in the adjacency matrix of \( G \).

**Planar Graphs:**
Three cities \( C_1, C_2, C_3 \) are to be directly connected by expressways to each of 3 other cities, \( C_4, C_5, C_6 \). Can this road system be designed so that the expressways do not cross?
The system above has its roads crossing.

**Definition:**
A graph is planar if it can be drawn without its edges crossing.

The problem of planarity occurs in city planning, printed circuits design and others.

If a connected, planar graph is drawn in the plane, the plane is divided into regions called faces. A face is characterized by the cycle that forms its boundaries.

A is bounded by the cycle(5,4,6,1,5)
The outer face D is considered to be bounded by the cycle (1, 2, 3, 4, 6, 1)

The graph above has $f = 4$ faces and $e= 8$ edges and $v = 6$ vertices. $f$, $e$ and $v$ satisfy the equation:

\[ f = e - v + 2 \]

In 1752, Euler proved that this equation holds for any connected planar graph.

Show that $K_{3, 3}$, is not planar ? Proof by contradiction:
-Suppose $K_3, 3$ is planar. Since every cycle has at least 4 edges, each face is bounded by at least 4 edges. In a planar graph, each edge belongs to at most 2 bounding cycles, therefore $2e \geq 4f$.