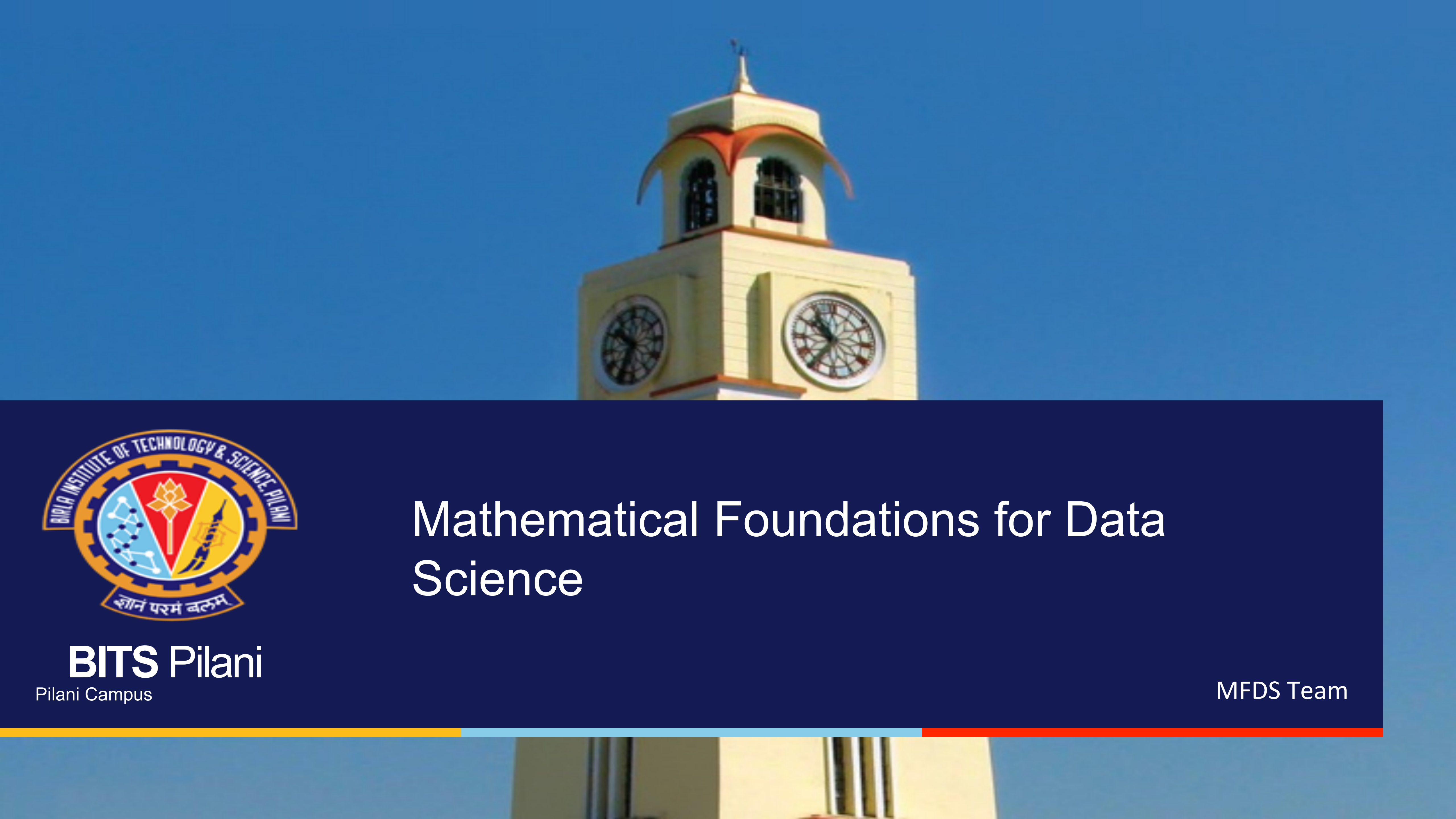


BITS Pilani
Pilani Campus

Mathematical Foundations for Data Science

MFDS Team

BITS Pilani

Pilani Campus

DSECL ZC416, MFDS

Lecture No.15

Agenda

- Isomorphism of Graphs
- Paths and Connected Components
- Euler Path and Circuit
- Hamilton Path and Circuit

Isomorphism of 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.*

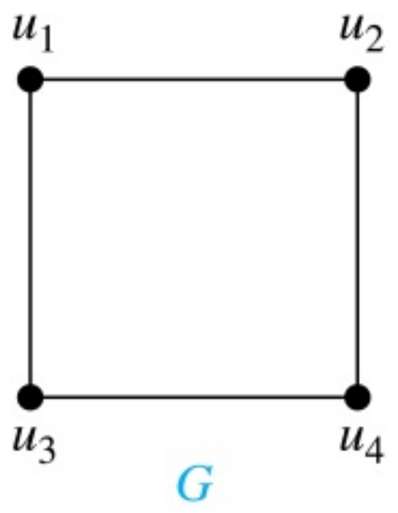
Two simple graphs that are not isomorphic are called *nonisomorphic*.

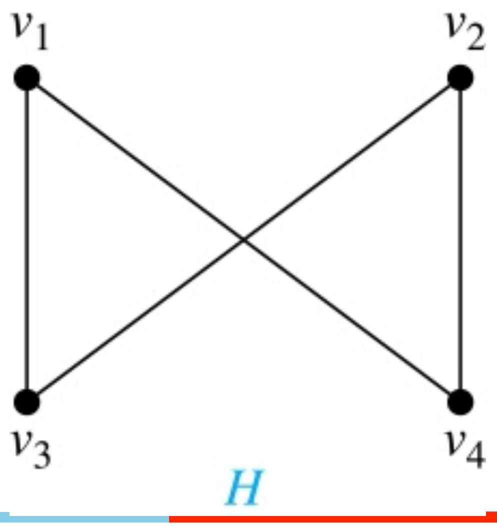
Isomorphism of Graphs

- The function f with $f(u_1) = v_1, f(u_2) = v_4$, $f(u_3) = v_3$, and $f(u_4) = v_2$ is a one-to-one correspondence between V and W.

- The adjacent vertices in G are u_1 and u_2, u_1 and u_3, u_2 and u_4, and u_3 and u_4.

- Each of the pairs $f(u_1) = v_1$ and $f(u_2) = v_4$, $f(u_1) = v_1$ and $f(u_3) = v_3$, $f(u_2) = v_4$ and $f(u_4) = v_2$, and $f(u_3) = v_3$ and $f(u_4) = v_2$ consists of two adjacent vertices in H.





Isomorphism of Graphs

- It is difficult to determine whether two simple graphs are isomorphic using brute force because there are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with n vertices.

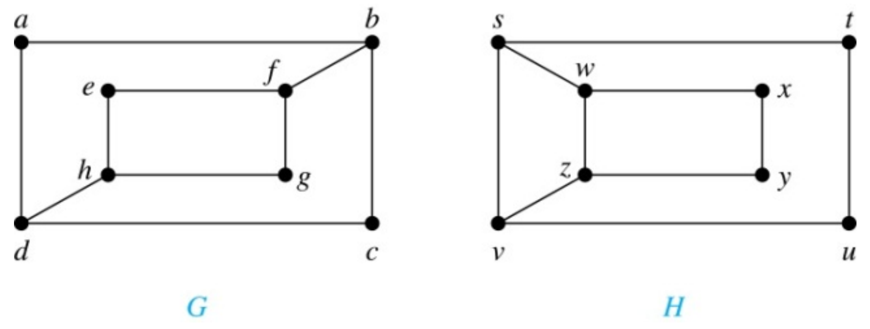
- The best algorithms for determining weather two graphs are isomorphic have exponential worst case complexity in terms of the number of vertices of the graphs.

- Sometimes it is not hard to show that two graphs are not isomorphic. We can do so by finding a property, preserved by isomorphism, that only one of the two graphs has. Such a property is called *graph invariant*.

- There are many different useful graph invariants that can be used to distinguish nonisomorphic graphs, such as the number of vertices, number of edges, and degree sequence (list of the degrees of the vertices in nonincreasing order). We will encounter others in later sections of this chapter.

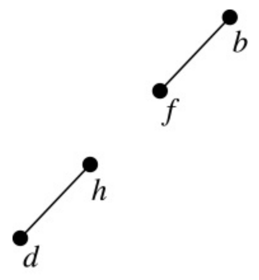
Isomorphism of Graphs

Example: Determine whether these two graphs are isomorphic.

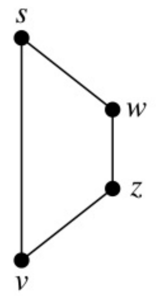


Solution: Both graphs have eight vertices and ten edges. They also both have four vertices of degree two and four of degree three.

However, *G* and *H* are not isomorphic. Note that since *deg*(*a*) = 2 in *G*, *a* must correspond to *t*, *u*, *x*, or *y* in H, because these are the vertices of degree 2. But each of these vertices is adjacent to another vertex of degree two in *H*, which is not true for *a* in *G*.

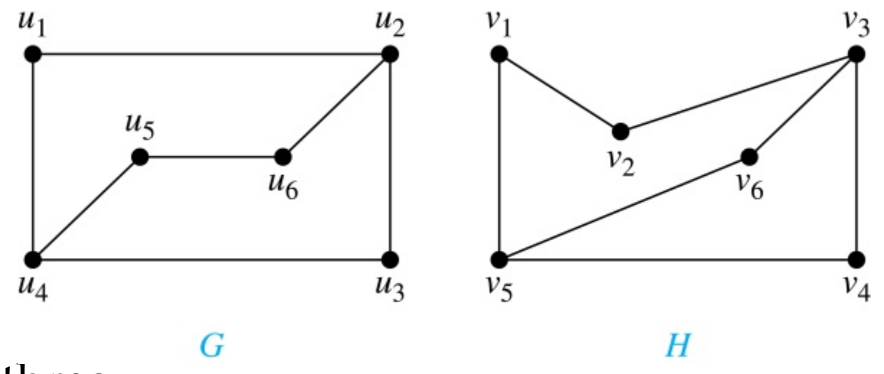


Alternatively, note that the subgraphs of *G* and *H* made up of vertices of degree three and the edges connecting them must be isomorphic. But the subgraphs, as shown at the right, are not isomorphic.



Isomorphism of Graphs

Example: Determine whether these two graphs isomorphic.



Solution: Both graphs have six vertices and seven edges.
They also both have four vertices of degree two and two of degree three.
The subgraphs of G and H consisting of all the vertices of degree two and the edges connecting them are isomorphic. So, it is reasonable to try to find an isomorphism f.

We define an injection f from the vertices of G to the vertices of H that preserves the degree of vertices. We will determine whether it is an isomorphism.

The function f with $f(u_1) = v_6$, $f(u_2) = v_3$, $f(u_3) = v_4$, and $f(u_4) = v_5$, $f(u_5) = v_1$, and $f(u_6) = v_2$ is a one-to-one correspondence between G and H. Showing that this correspondence preserves edges is straightforward, so we will omit the details here. Because f is an isomorphism, it follows that G and H are isomorphic graphs.

See the text for an illustration of how adjacency matrices can be used for this verification.

Algorithms for Graph Isomorphism

- The best algorithms known for determining whether two graphs are isomorphic have exponential worst-case time complexity (in the number of vertices of the graphs).
- However, there are algorithms with linear average-case time complexity.

- Graph isomorphism is a problem of special interest because it is one of a few NP problems not known to be either tractable or NP-complete

Applications of Graph Isomorphism

The question whether graphs are isomorphic plays an important role in applications of graph theory. For example,

– chemists use molecular graphs to model chemical compounds. Vertices represent atoms and edges represent chemical bonds. When a new compound is synthesized, a database of molecular graphs is checked to determine whether the graph representing the new compound is isomorphic to the graph of a compound that this already known.

– Electronic circuits are modeled as graphs in which the vertices represent components and the edges represent connections between them. Graph isomorphism is the basis for

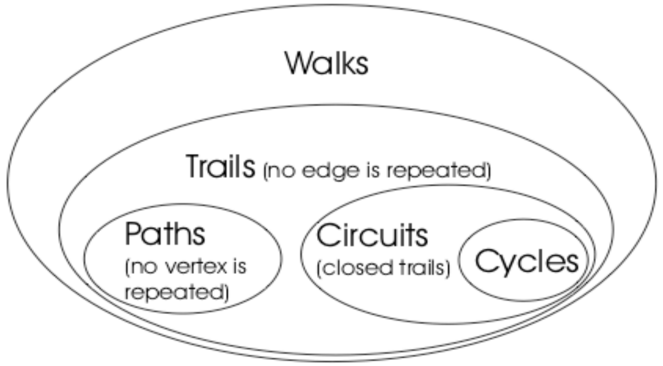
- the verification that a particular layout of a circuit corresponds to the design's original schematics.
- determining whether a chip from one vendor includes the intellectual property of another vendor.

Paths

Informal Definition: A *path* is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph. As the path travels along its edges, it visits the vertices along this path, that is, the endpoints of these.

Applications: Numerous problems can be modeled with paths formed by traveling along edges of graphs such as:

- determining whether a message can be sent between two computers.
- efficiently planning routes for mail delivery.



Paths

Definition: Let n be a nonnegative integer and G an undirected graph. A *path* of *length* n from u to v in G is a sequence of n edges $e_1, \ldots , e_n$ of G for which there exists a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices such that e_i has, for $i = 1, \ldots, n$, the endpoints x_{i-1} and x_i.

- When the graph is simple, we denote this path by its vertex sequence $x_0, x_1, \ldots , x_n$(since listing the vertices uniquely determines the path).
- The path is a *circuit* if it begins and ends at the same vertex ($u = v$) and has length greater than zero.
- The path or circuit is said to *pass through* the vertices $x_1, x_2, \ldots , x_{n-1}$ and *traverse* the edges $e_1, \ldots , e_n$.
- A path or circuit is *simple* if it does not contain the same edge more than once.

This terminology is readily extended to directed graphs. (*see text*)

Paths

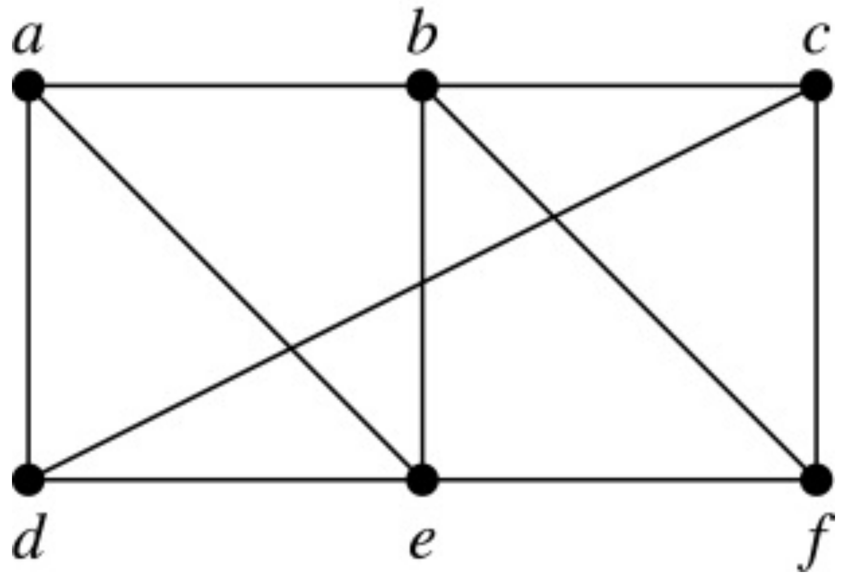
In the simple graph here:

– *a*, *d*, *c*, *f*, *e* is a simple path of length 4.

– *d*, *e*, *c*, *a* is not a path because *e* is not connected to *c*.

– *b*, *c*, *f*, *e*, *b* is a circuit of length 4.

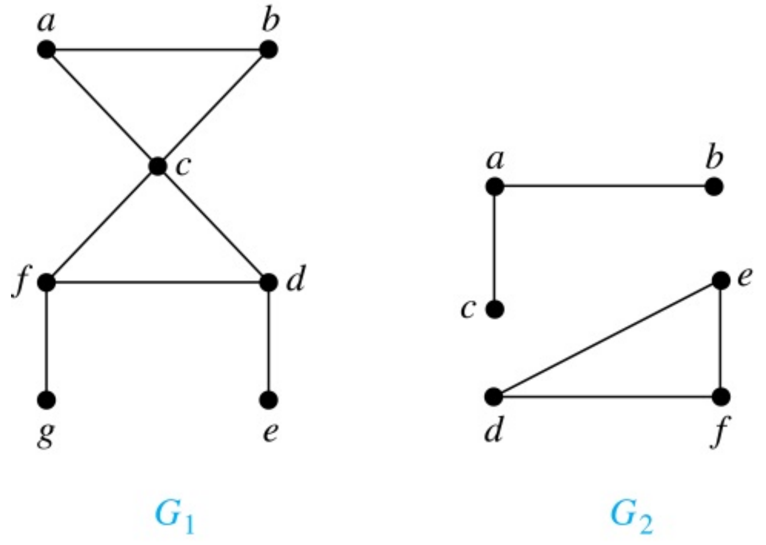
– *a*, *b*, *e*, *d*, *a*, *b* is a path of length 5, but it is not a simple path.



Connectedness in Undirected Graphs

Definition: An undirected graph is called *connected* if there is a path between every pair of vertices. An undirected graph that is not *connected* is called *disconnected*. We say that we *disconnect* a graph when we remove vertices or edges, or both, to produce a disconnected subgraph.

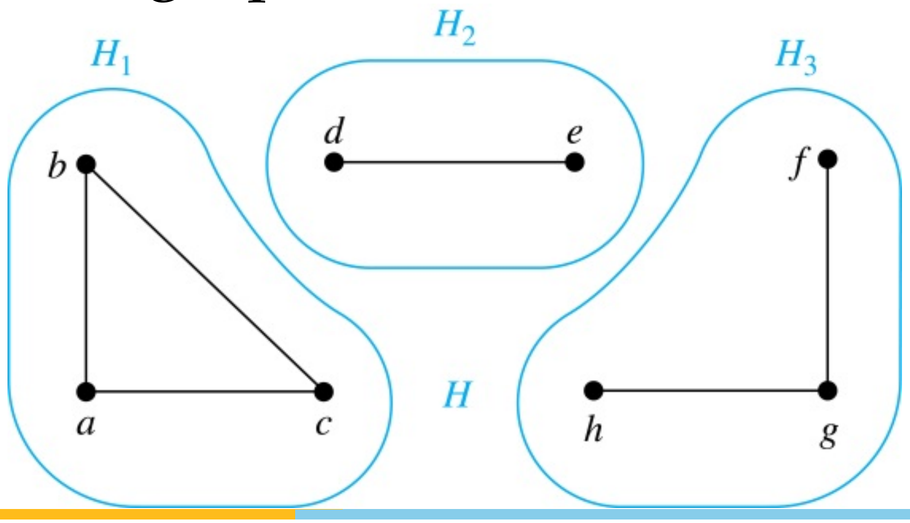
Example: G_1 is connected because there is a path between any pair of its vertices, as can be easily seen. However G_2 is not connected because there is no path between vertices a and f, for example.



Connected Components

Definition: A *connected component* of a graph G is a connected subgraph of G that is not a proper subgraph of another connected subgraph of G. A graph G that is not connected has two or more connected components that are disjoint and have G as their union.

Example: The graph H is the union of three disjoint subgraphs H_1, H_2, and H_3, none of which are proper subgraphs of a larger connected subgraph of G. These three subgraphs are the connected components of H.



Connectedness in Directed Graphs

Definition: A directed graph is *strongly connected* if there is a path from *a* to *b* and a path from *b* to *a* whenever *a* and *b* are vertices in the graph.

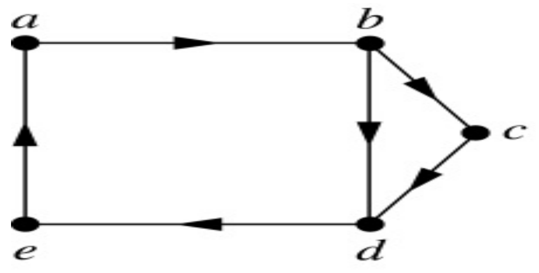
Definition: A directed graph is *weakly connected* if there is a path between every two vertices in the underlying undirected graph, which is the undirected graph obtained by ignoring the directions of the edges of the directed graph.

Connectedness in Directed Graphs

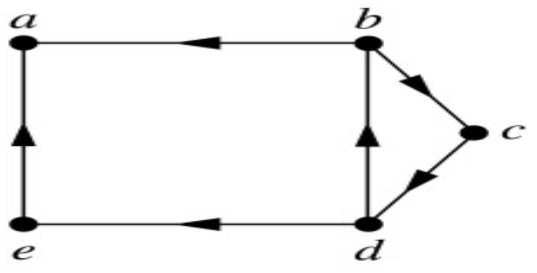
Example: *G* is strongly connected because there is a path between any two vertices in the directed graph. Hence, *G* is also weakly connected. The graph *H* is not strongly connected, since there is no directed path from *a* to *b*, but it is weakly connected.

Definition: The subgraphs of a directed graph *G* that are strongly connected but not contained in larger strongly connected subgraphs, that is, the maximal strongly connected subgraphs, are called the *strongly connected components* or *strong components* of *G*.

Example (*continued*): The graph *H* has three strongly connected components, consisting of the vertex *a*; the vertex *e;* and the subgraph consisting of the vertices *b*, *c*, *d* and edges (*b*,*c*), (*c*,*d*), and (*d*,*b*).



G



H

Counting Paths between Vertices

We can use the adjacency matrix of a graph to find the number of paths between two vertices in the graph.

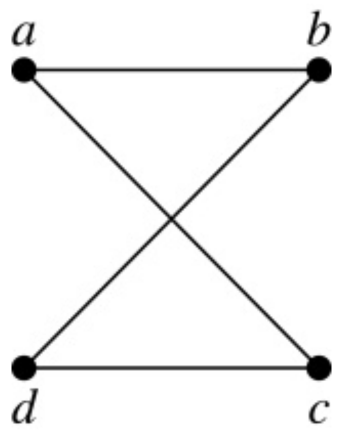
Theorem: Let G be a graph with adjacency matrix **A** with respect to the ordering $v_1, \ldots, v_n$ of vertices (with directed or undirected edges, multiple edges and loops allowed). The number of different paths of length r from v_i to v_j, where $r > 0$ is a positive integer, equals the (i,j)th entry of $\mathbf{A}^r$

Counting Paths between Vertices

Example: How many paths of length four are there from *a* to *d* in the graph G.

The adjacency matrix of *G* (ordering the vertices as *a, b, c, d*) is given above. Hence the number of paths of length four from *a* to *d* is the (1, 4)th entry of $\mathbf{A}^4$. The eight paths are as:

G



adjacency matrix of G

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

a, b, a, b, d	*a, b, a, c, d*
a, b, d, b, d	*a, b, d, c, d*
a, c, a, b, d	*a, c, a, c, d*
a, c, d, b, d	*a, c, d, c, d*

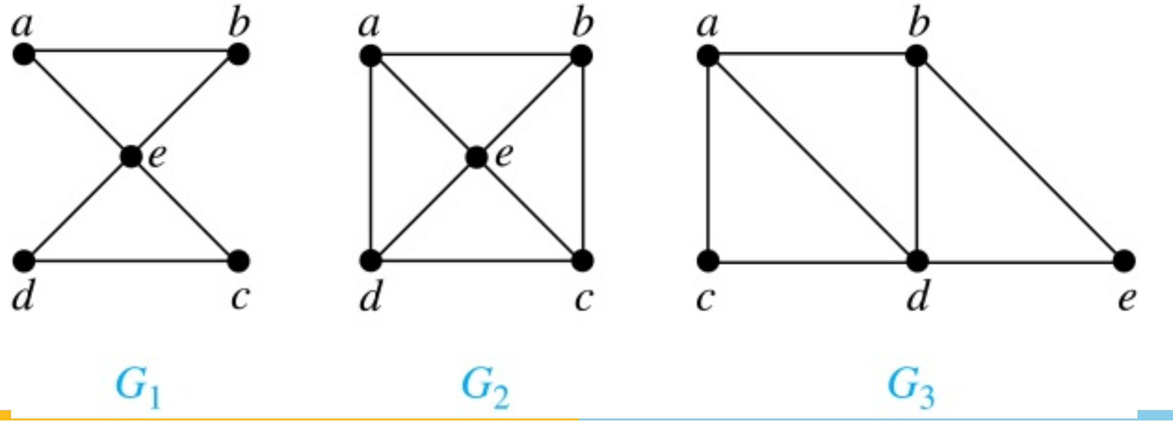
$$\mathbf{A}^4 = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

Euler Paths and Circuits (*continued*)

Definition: An *Euler circuit* in a graph G is a simple circuit containing every edge of G. An *Euler path* in G is a simple path containing every edge of G.

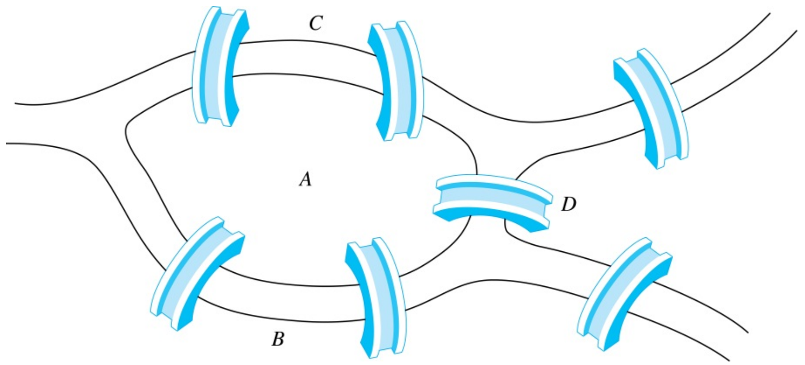
Example: Which of the undirected graphs G_1, G_2, and G_3 has a Euler circuit? Of those that do not, which has an Euler path?

Solution: The graph G_1 has an Euler circuit (e.g., a, e, c, d, e, b, a). But, as can easily be verified by inspection, neither G_2 nor G_3 has an Euler circuit. Note that G_3 has an Euler path (e.g., a, c, d, e, b, d, a, b), but there is no Euler path in G_2, which can be verified by inspection.

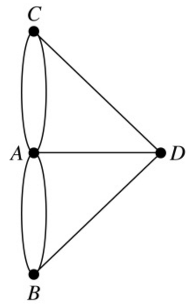


Euler Paths and Circuits

The town of Königsberg, Prussia (now Kalingrad, Russia) was divided into four sections by the branches of the Pregel river. In the 18th century seven bridges connected these regions.
People wondered whether whether it was possible to follow a path that crosses each bridge exactly once and returns to the starting point.
The Swiss mathematician Leonard Euler proved that no such path exist. This result is often considered to be the first theorem ever proved in graph theory.



The 7 Bridges of Königsberg



Multigraph Model of the Bridges of Königsberg

Necessary Conditions for Euler Circuits and Paths

An Euler circuit begins with a vertex *a* and continues with an edge incident with *a*, say {*a*, *b*}. The edge {*a*, *b*} contributes one to deg(*a*).

Each time the circuit passes through a vertex it contributes two to the vertex's degree.

Finally, the circuit terminates where it started, contributing one to deg(*a*). Therefore deg(*a*) must be even.

We conclude that the degree of every other vertex must also be even.

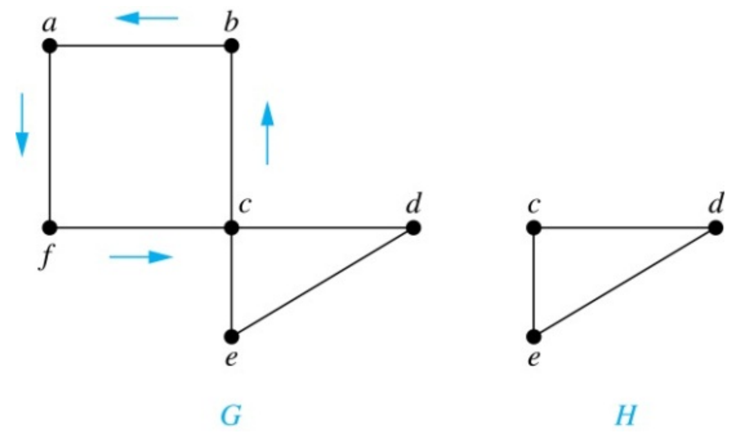
By the same reasoning, we see that the initial vertex and the final vertex of an Euler path have odd degree, while every other vertex has even degree. So, a graph with an Euler path has exactly two vertices of odd degree.

In the next slide we will show that these necessary conditions are also sufficient conditions.

Sufficient Conditions for Euler Circuits and Paths

Suppose that G is a connected multigraph with ≥ 2 vertices, all of even degree. Let $x_0 = a$ be a vertex of even degree. Choose an edge $\{x_0, x_1\}$ incident with a and proceed to build a simple path $\{x_0, x_1\}, \{x_1, x_2\}, \ldots, \{x_{n-1}, x_n\}$ by adding edges one by one until another edge can not be added.

We illustrate this idea in the graph G here. We begin at *a* and choose the edges $\{a, f\}$, $\{f, c\}$, $\{c, b\}$, and $\{b, a\}$ in succession.



The path begins at a with an edge of the form $\{a, x\}$; we show that it must terminate at a with an edge of the form $\{y, a\}$. Since each vertex has an even degree, there must be an even number of edges incident with this vertex. Hence, every time we enter a vertex other than a, we can leave it. Therefore, the path can only end at a.

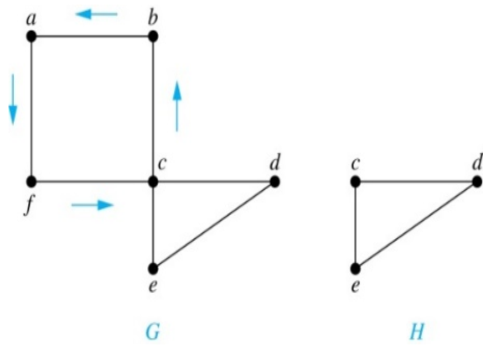
If all of the edges have been used, an Euler circuit has been constructed. Otherwise, consider the subgraph H obtained from G by deleting the edges already used.

In the example *H* consists of the vertices *c*, *d*, *e*.

Sufficient Conditions for Euler Circuits and Paths (*continued*)

Because G is connected, H must have at least one vertex in common with the circuit that has been deleted.

In the example, the vertex is *c*.



Every vertex in H must have even degree because all the vertices in *G* have even degree and for each vertex, pairs of edges incident with this vertex have been deleted. Beginning with the shared vertex construct a path ending in the same vertex (as was done before). Then splice this new circuit into the original circuit.

In the example, we end up with the circuit *a, f, c, d, e, c, b, a*.

Continue this process until all edges have been used. This produces an Euler circuit. Since every edge is included and no edge is included more than once.
Similar reasoning can be used to show that a graph with exactly two vertices of odd degree must have an Euler path connecting these two vertices of odd degree

Algorithm for Constructing an Euler Circuits

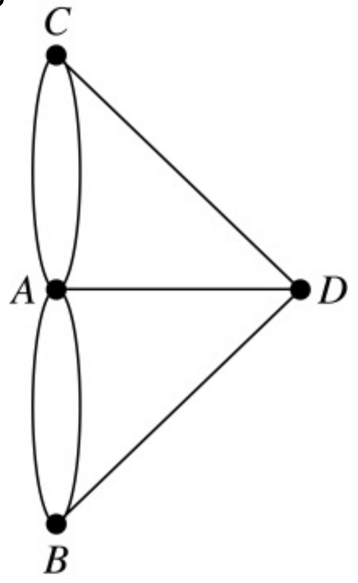
In our proof we developed this algorithms for constructing a Euler circuit in a graph with no vertices of odd degree.

```
procedure Euler(G: connected multigraph with all vertices of even degree)
circuit := a circuit in G beginning at an arbitrarily chosen vertex with edges
        successively added to form a path that returns to this vertex.
H := G with the edges of this circuit removed
while H has edges
    subciruit := a circuit in H beginning at a vertex in H that also is
            an endpoint of an edge in circuit.
    H := H with edges of subciruit and all isolated vertices removed
    circuit := circuit with subcircuit inserted at the appropriate vertex.
return circuit{circuit is an Euler circuit}
```

Necessary and Sufficient Conditions for Euler Circuits and Paths

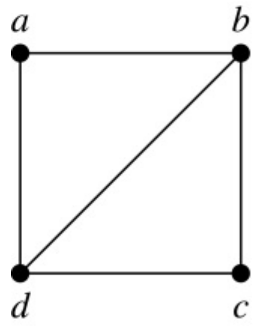
Theorem: A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has an even degree and it has an Euler path if and only if it has exactly two vertices of odd degree.

Example: Two of the vertices in the multigraph model of the Königsberg bridge problem have odd degree. Hence, there is no Euler circuit in this multigraph and it is impossible to start at a given point, cross each bridge exactly once, and return to the starting point.

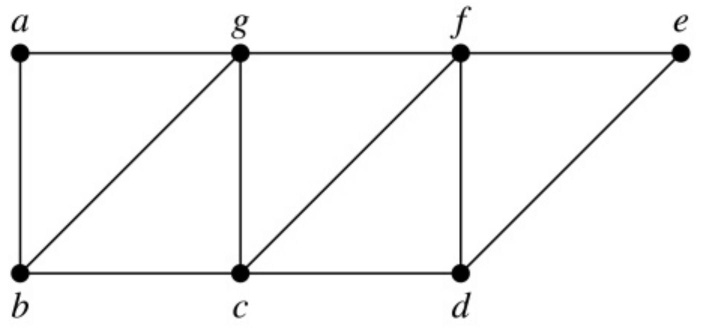


Euler Circuits and Paths

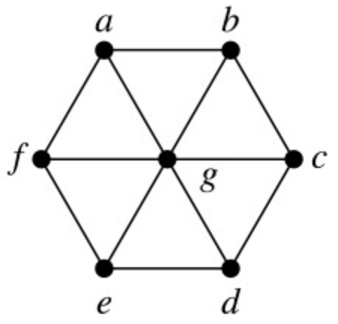
Example:



G_1



G_2



G_3

G_1 contains exactly two vertices of odd degree (b and d). Hence it has an Euler path, e.g., d, a, b, c, d, b.

G_2 has exactly two vertices of odd degree (b and d). Hence it has an Euler path, e.g., $b, a, g, f, e, d, c, g, b, c, f, d$.

G_3 has six vertices of odd degree. Hence, it does not have an Euler path.

Applications of Euler Paths and Circuits

Euler paths and circuits can be used to solve many practical problems such as finding a path or circuit that traverses each

- street in a neighborhood,
- road in a transportation network,
- connection in a utility grid,
- link in a communications network.

Other applications are found in the

- layout of circuits,
- network multicasting,
- molecular biology, where Euler paths are used in the sequencing of DNA.

Hamilton Paths and Circuits

Definition: A simple path in a graph G that passes through every vertex exactly once is called a *Hamilton path*, and a simple circuit in a graph G that passes through every vertex exactly once is called a *Hamilton circuit*.

That is, a simple path $x_0, x_1, \ldots, x_{n-1}, x_n$ in the graph $G = (V, E)$ is called a Hamilton path if $V = \{x_0, x_1, \ldots, x_{n-1}, x_n\}$ and $x_i \neq x_j$ for $0 \leq i < j \leq n$, and the simple circuit $x_0, x_1, \ldots, x_{n-1}, x_n, x_0$ (with $n > 0$) is a Hamilton circuit if $x_0, x_1, \ldots, x_{n-1}, x_n$ is a Hamilton path.

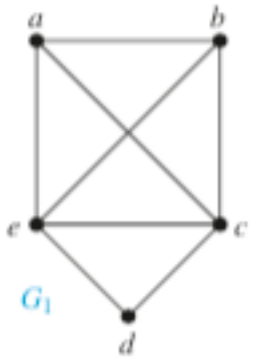
Hamilton Paths and Circuits (*continued*)

Example: Which of these simple graphs has a Hamilton circuit or, if not, a Hamilton path?

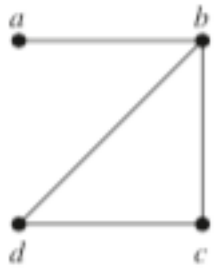
Solution: G_1 has a Hamilton circuit: a, b, c, d, e, a.

G_2 does not have a Hamilton circuit (Why?), but does have a Hamilton path : a, b, c, d.

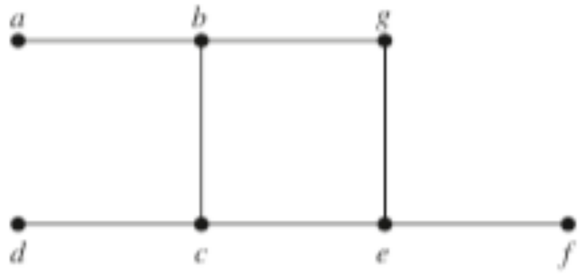
G_3 does not have a Hamilton circuit, or a Hamilton path. Why?



G_1



G_2



G_3

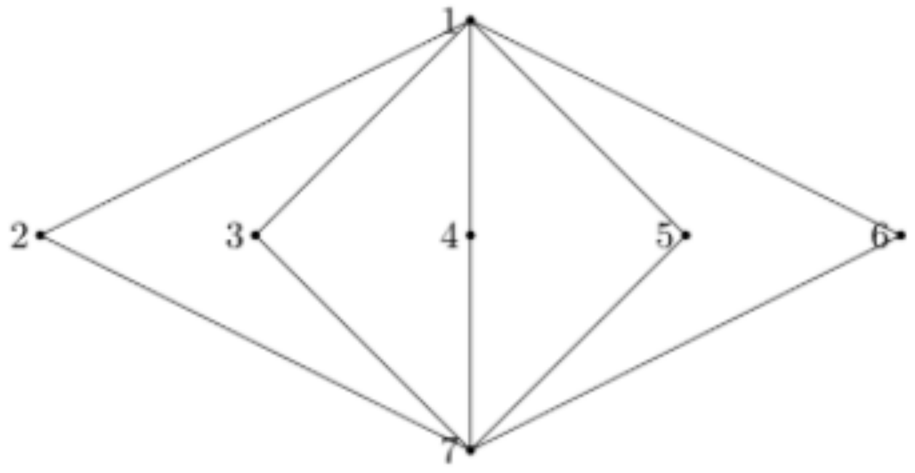
Sufficient Conditions for Hamilton Circuits

Unlike for an Euler circuit, no simple necessary and sufficient conditions are known for the existence of a Hamilton circuit.
However, there are some useful sufficient conditions. We describe two of these now.

Dirac's Theorem: If G is a simple graph with $n \geq 3$ vertices such that the degree of every vertex in G is $\geq n/2$, then G has a Hamilton circuit.

Ore's Theorem: If G is a simple graph with $n \geq 3$ vertices such that $\deg(u) + \deg(v) \geq n$ for every pair of nonadjacent vertices, then G has a Hamilton circuit.

Examples



Graph has 7 vertices and for every vertex
deg(v1) = 5 > n/2 (3.5)
deg(v4) = 2 $\not>$ n/2(3.5)
Dirac's Theorem condition is not satisfied

deg(v2) + deg(v6) < n
Hence Ore's Theorem condition is not satisfied

No conclusion can be drawn about this graph

Applications of Hamilton Paths and Circuits

Applications that ask for a path or a circuit that visits each intersection of a city, each place pipelines intersect in a utility grid, or each node in a communications network exactly once, can be solved by finding a Hamilton path in the appropriate graph.

The famous *traveling salesperson problem* (*TSP*) asks for the shortest route a traveling salesperson should take to visit a set of cities. This problem reduces to finding a Hamilton circuit such that the total sum of the weights of its edges is as small as possible.