11. Graphs and Trees 1

Aaron Tan

29-30 October 2015

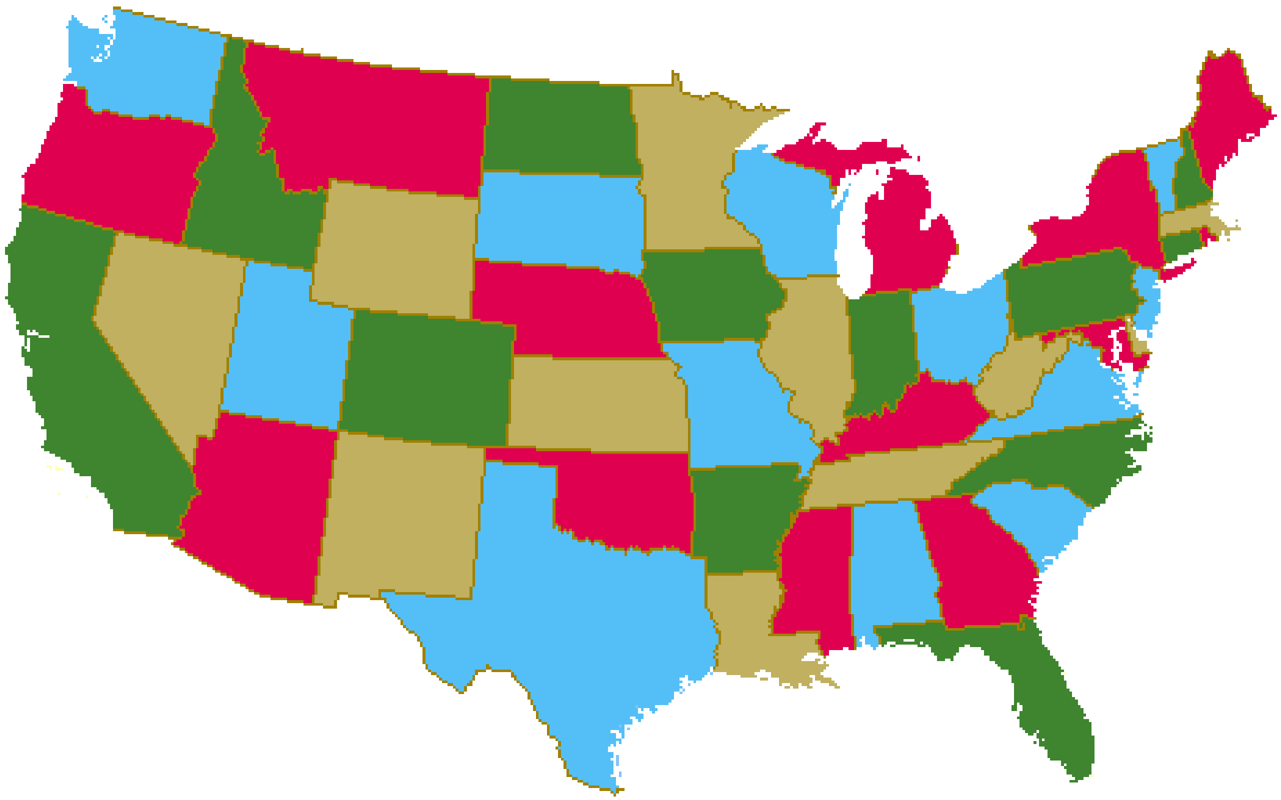
10.1 Graphs: Definitions and Basic Properties

Graphs: Introduction

Shall we add some colours to this map of the United States?

Graphs: Introduction

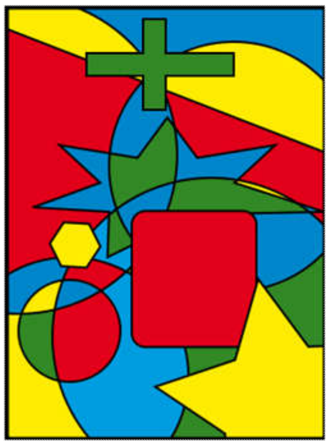
Shall we add some colours to this map of the United States?



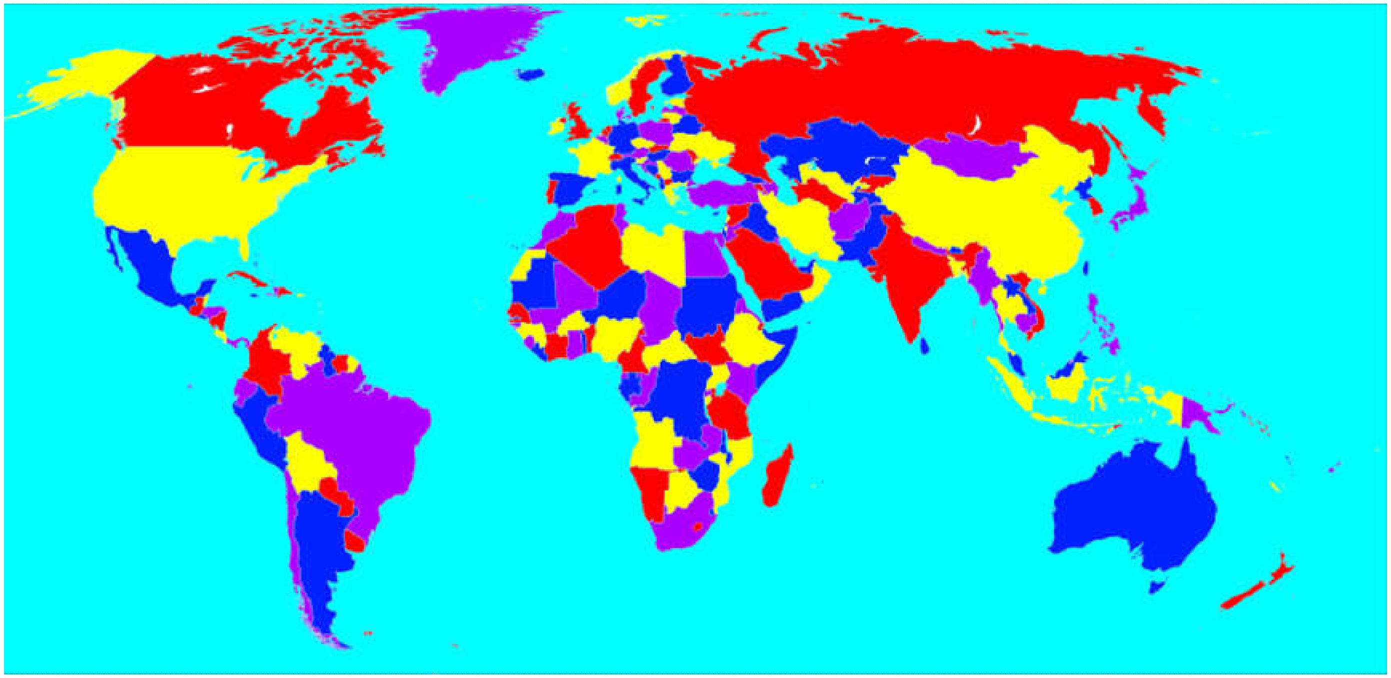
Map Colouring

- Four-Colour Conjecture
  - Proposed by Guthrie in 1852, who conjectured that…
  - Four colours are sufficient to colour any map in a plane, such that regions that share a common boundary do not share the same colour.
  - Many false proofs since then.
  - Finally proved by Appel and Haken in 1977, with the help of computer.
  - Robertson et al. provided another proof in 1996.

Map Colouring



Example of a 4-coloured map.



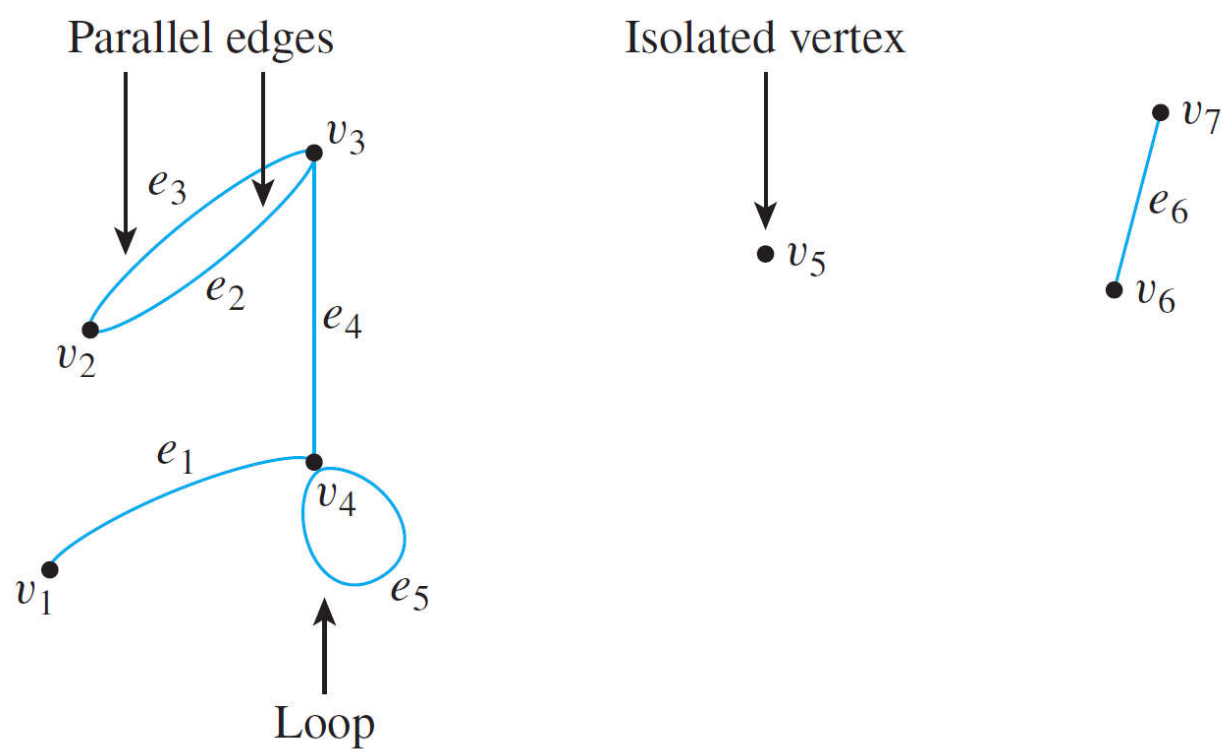
World map with 4 colours.

- But this is a map, not a graph!
- However, we can model it as a graph.
- But what is a graph?

Graphs: Definitions and Basic Properties

- A graph G consists of
  - a set of vertices $V(G)$, and
  - a set of edges $E(G)$.
- Sometimes, we write $G = \{V, E\}$.

An edge connects one vertex to another, or a vertex to itself.



Graphs: Definitions and Basic Properties

Definition: Graph

A **graph** G consists of 2 finite sets: a nonempty set $V(G)$ of **vertices** and a set $E(G)$ of **edges**, where each edge is associated with a set consisting of either one or two vertices called its **endpoints**.

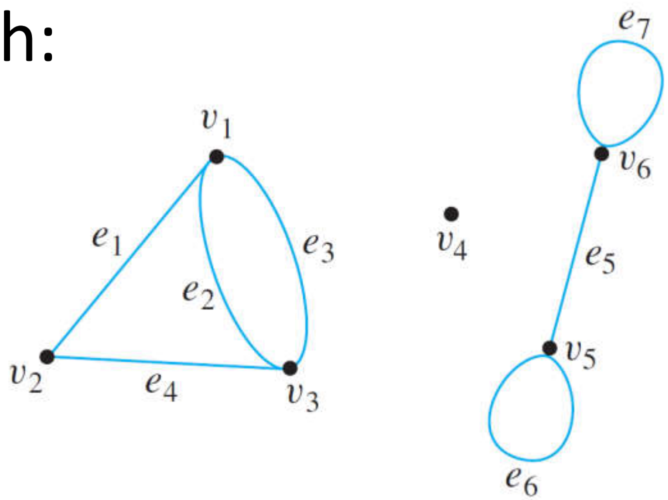
An edge is said to **connect** its endpoints; two vertices that are connected by an edge are called **adjacent vertices**; and a vertex that is an endpoint of a loop is said to be **adjacent to itself**.

An edge is said to be **incident on** each of its endpoints, and two edges incident on the same endpoint are called **adjacent edges**.

We write $e = \{v, w\}$ for an edge e incident on vertices v and w.

Graphs: Definitions and Basic Properties

Example: Consider the following graph:



a. Write the vertex set V and the edge set E, and give the list of edges with their end-points.

$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$

$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$

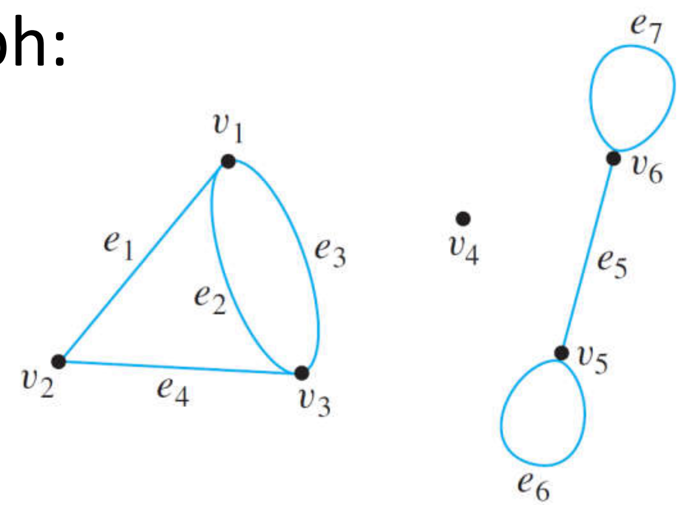
$e_1 = \{v_1, v_2\}$
$e_2 = \{v_1, v_3\}$
$e_3 = \{v_1, v_3\}$

$e_4 = \{v_2, v_3\}$
$e_5 = \{v_5, v_6\}$
$e_6 = \{v_5\}$
$e_7 = \{v_6\}$

Graphs: Definitions and Basic Properties

Example: Consider the following graph:

b. Find all edges that are incident on v_1, all vertices that are adjacent to v_1, all edges that are adjacent to e_1, all loops, all parallel edges, all vertices that are adjacent to themselves, and all isolated vertices.



Edges incident on v_1: e_1, e_2 and e_3.
Vertices adjacent to v_1: v_2 and v_3.
Edges adjacent to e_1: e_2, e_3 and e_4.
Loops: e_6 and e_7.
e_2 and e_3 are parallel.

v_5 and v_6 are adjacent to themselves.
Isolated vertex: v_4.

Graphs: Definitions and Basic Properties

Definition: Directed Graph

A **directed graph**, or **digraph**, *G*, consists of 2 finite sets: a nonempty set *V*(*G*) of **vertices** and a set *D*(*G*) of **directed edges**, where each edge is associated with an ordered pair of vertices called its **endpoints**.

If edge *e* is associated with the pair (*v*, *w*) of vertices, then *e* is said to be the (**directed**) **edge** from *v* to *w*. We write *e* = (*v*, *w*).

Modelling Graph Problems

Map Colouring Problem

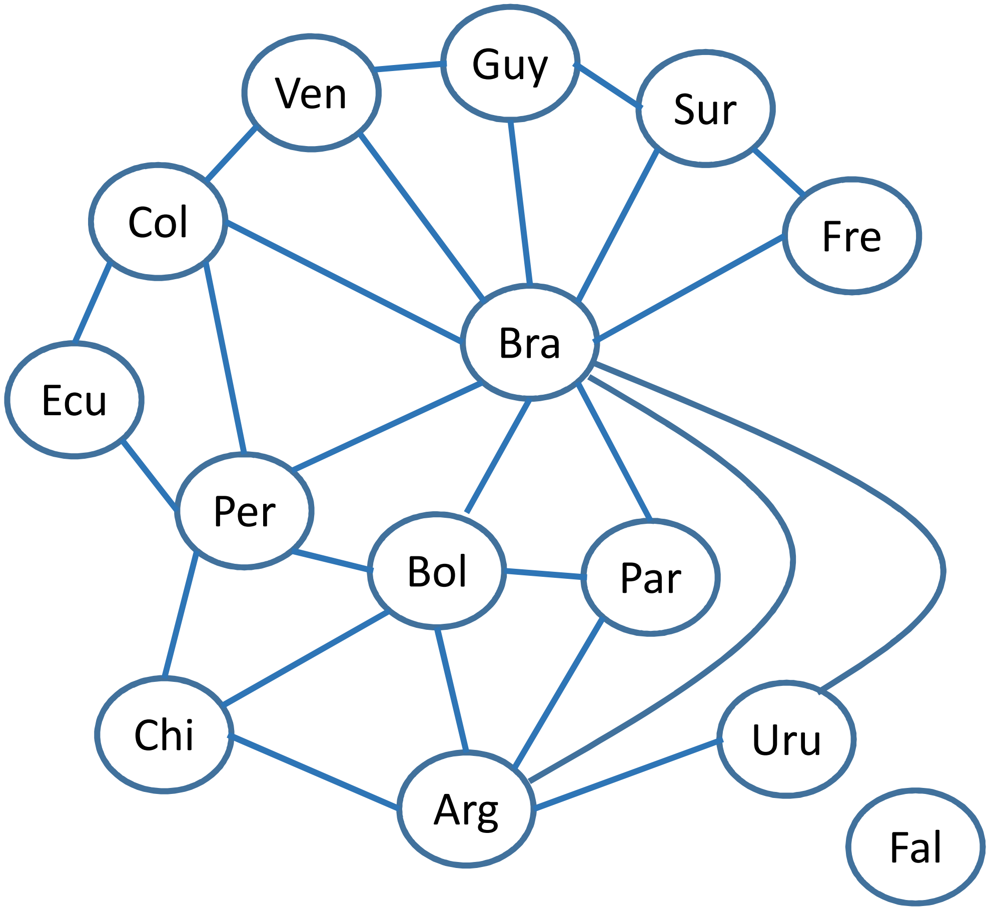
Solve it as a graph problem.

Draw a graph in which the vertices represent the states, with every edge joining two vertices represents the states sharing a common border.

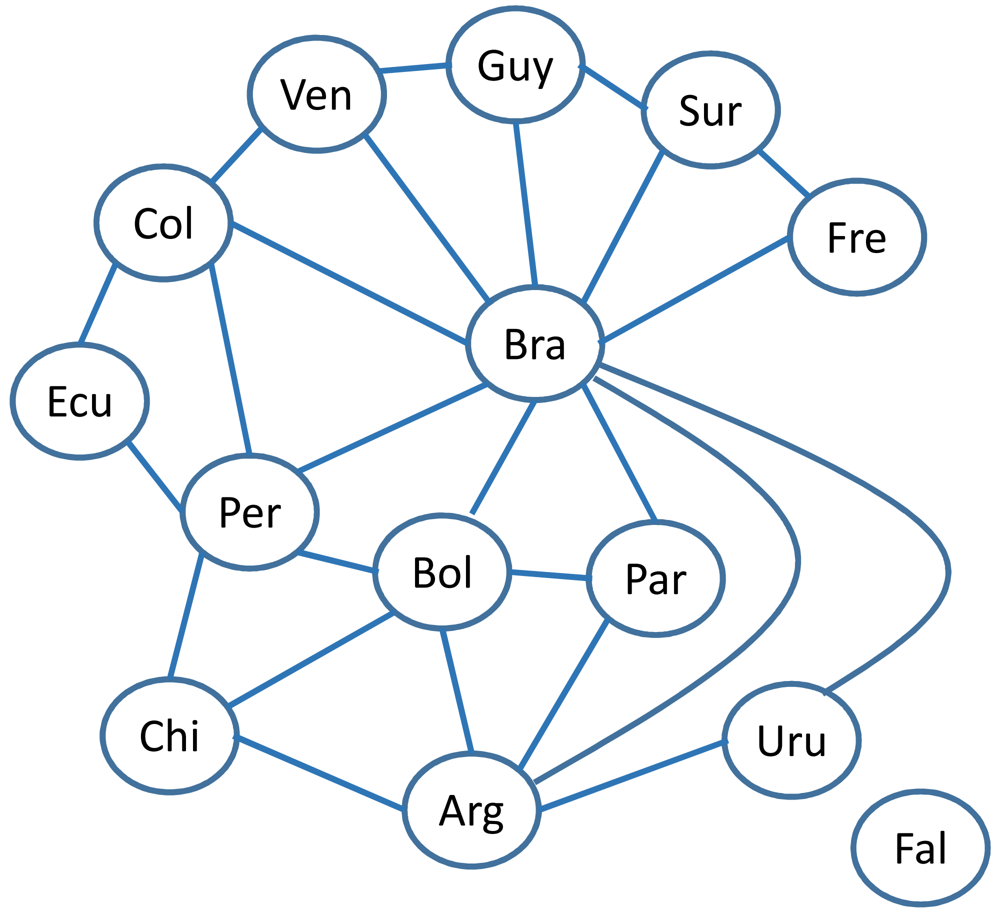
Such two vertices cannot be coloured with the same colour.

A **vertex colouring** of a graph is an assignment of colours to vertices so that no two adjacent vertices have the same colour.

Modelling Graph Problems



Modelling Graph Problems



Wedding Planner

You are your best friend's wedding planner and you need to plan the seating arrangement for his 16 guests attending his wedding dinner. However, some of the guests cannot get along with some others.

- *A* doesn't get along with *F*, *G* or *H*.
- *B* doesn't get along with *C*, *D* or *H*.
- *C* doesn't get along with *B*, *D*, *E*, *G* or *H*.
- *D* doesn't get along with *B*, *C* or *E*.
- *E* doesn't get along with *C*, *D*, *F*, or *G*.
- *F* doesn't get along with *A*, *E* or *G*.
- *G* doesn't get along with *A*, *C*, *E* or *F*.
- *H* doesn't get along with *A*, *B* or *C*.

You don't want to put guests who cannot get along with each other at the same table!

How many tables do you need?

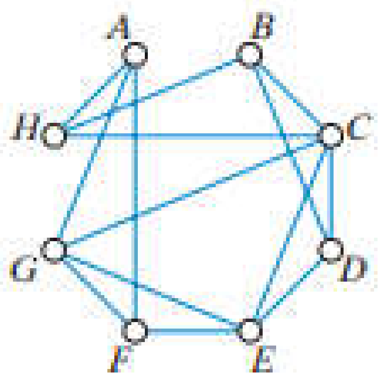
Acknowledgement: http://www.math.uri.edu/~eaton/0131873814_MEb.pdf

Wedding Planner

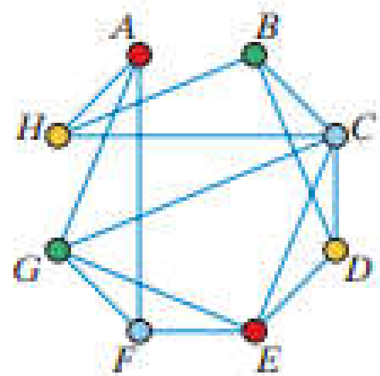
- *A* doesn't get along with *F*, *G* or *H*.
- *B* doesn't get along with *C*, *D* or *H*.
- *C* doesn't get along with *B*, *D*, *E*, *G* or *H*.
- *D* doesn't get along with *B*, *C* or *E*.
- *E* doesn't get along with *C*, *D*, *F*, or *G*.
- *F* doesn't get along with *A*, *E* or *G*.
- *G* doesn't get along with *A*, *C*, *E* or *F*.
- *H* doesn't get along with *A*, *B* or *C*.

Graph with vertices representing the guests, and an edge is drawn between two guests who don't get along.

Vertex colouring problem.
4 colours (4 tables)?



(a)



(b)

Acknowledgement: http://www.math.uri.edu/~eaton/0131873814_MEb.pdf

Other Vertex Colouring Problems

	If the vertices represent…	And two vertices are adjacent if ….	Then a vertex colouring can be used to…
1.	classes,	the corresponding classes have students in common,	schedule classes.
2.	radio stations,	the stations are close enough to interfere with each other,	assign non-interfering frequencies to the stations.
3.	traffic signals at an intersection,	the corresponding signals cannot be green at the same time,	designate sets of signals that can be green at the same time.

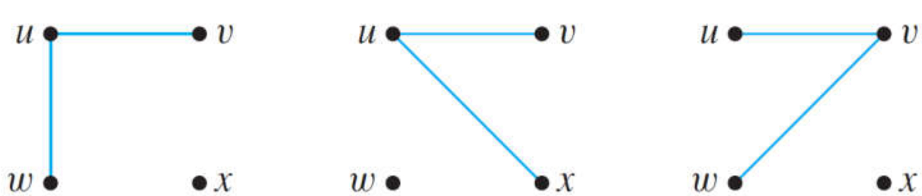
Simple Graphs

Definition: Simple Graph

A **simple graph** is an undirected graph that does not have any loops or parallel edges.

Example: Draw all simple graphs with the 4 vertices {*u*, *v*, *w*, *x*} and two edges, one of which is {*u*, *v*}.

Each possible edge of a simple graph corresponds to a subset of two vertices. Hence there are $\binom{4}{2}$ = 6 such subsets. One is given.



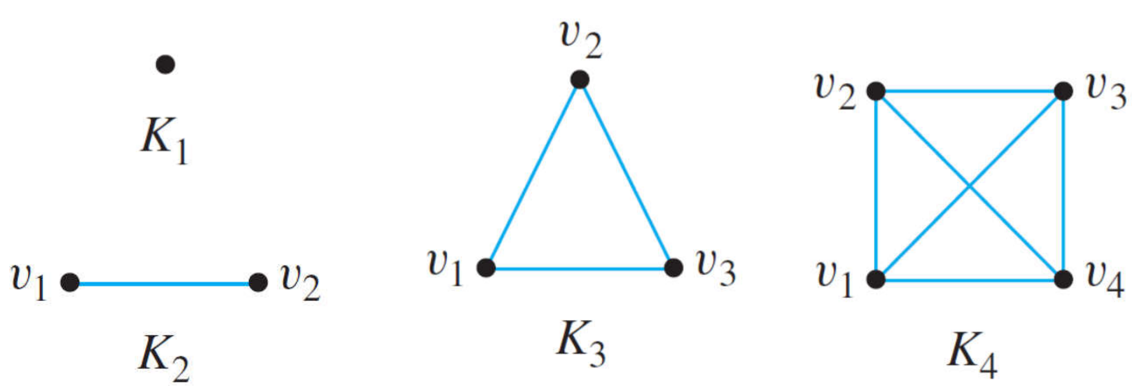
Draw the other two.

Complete Graphs

Definition: Complete Graph

A **complete graph** on ***n* vertices**, $n > 0$, denoted $\boldsymbol{K_n}$, is a simple graph with n vertices and exactly one edge connecting each pair of distinct vertices .

Example: The complete graphs K_1, K_2, K_3 and K_4.



Complete Bipartite Graphs

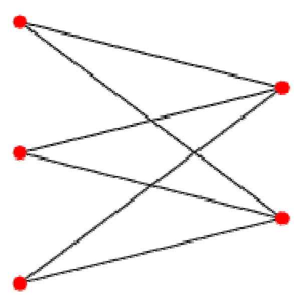
Definition: Complete Bipartite Graph

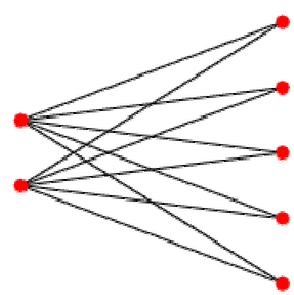
A **complete bipartite graph** on **(m, n) vertices**, where m, $n > 0$, denoted $\boldsymbol{K_{m,n}}$, is a simple graph with distinct vertices $v_1, v_2, \ldots, v_m$, and $w_1, w_2, \ldots, w_n$ that satisfies the following properties:

For all i, $k = 1, 2, \ldots, m$ and for all j, $l = 1, 2, \ldots, n$,

1. There is an edge from each vertex v_i to each vertex w_j.
2. There is no edge from any vertex v_i to any other vertex v_k.
3. There is no edge from any vertex w_j to any other vertex w_l.

$K_{3,2}$





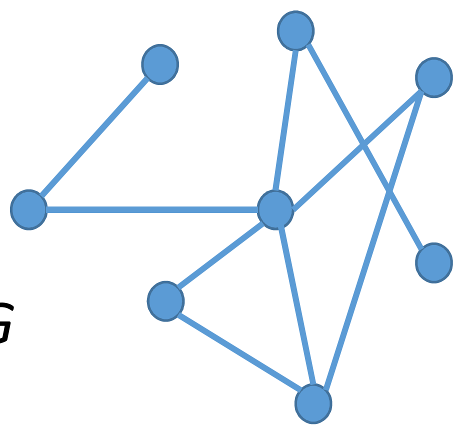
$K_{2,5}$

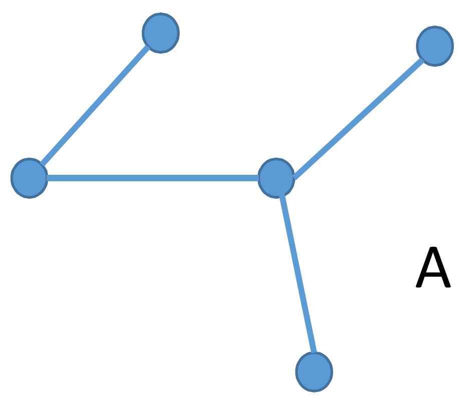
Subgraph of a Graph

Definition: Subgraph of a Graph

A graph *H* is said to be a **subgraph** of graph *G* if, and only if, every vertex in *H* is also a vertex in *H*, every edge in *H* is also an edge in *G*, and every edge in *H* has the same endpoints as it has in *G*.

A graph *G*





A subgraph of *G*

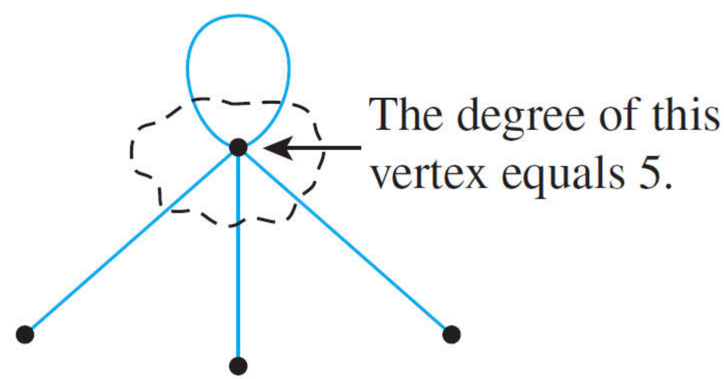
Degree of a Vertex and Total Degree of a Graph

Definition: Degree of a Vertex and Total Degree of a Graph

Let G be a graph and v a vertex of G. The **degree** of v, denoted **deg(v)**, equals the number of edges that are incident on v, with an edge that is a loop counted twice.

The **total degree of G** is the sum of the degrees of all the vertices of G.

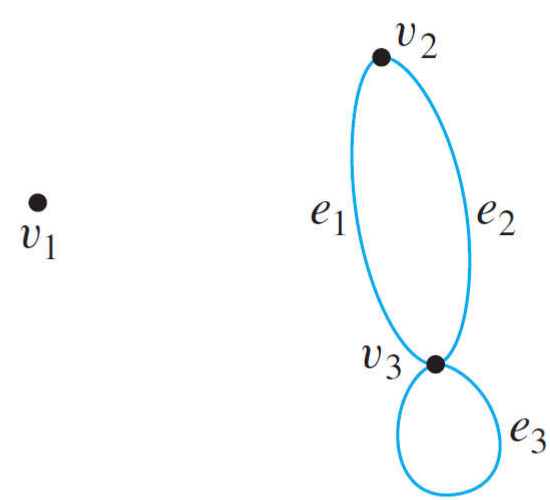
The degree of a vertex can be obtained from the drawing of a graph by counting how many end segments of edges are incident on the vertex.



Degree of a Vertex and Total Degree of a Graph

Example: Find the degree of each vertex of the graph G shown below. Then find the total degree of G.

$\deg(v_1) = 0$
$\deg(v_2) = 2$
$\deg(v_3) = 4$
Total degree of $G = 6$



Theorem 10.1.1 The Handshake Theorem

If G is any graph, then the sum of the degrees of all the vertices of G equals twice the number of edges of G. Specifically, if the vertices of G are $v_1, v_2, \ldots, v_n$, where $n \geq 0$, then

$$\begin{aligned} \text{The total degree of } G &= \deg(v_1) + \deg(v_2) + \ldots + \deg(v_n) \\ &= 2\cdot(\text{the number of edges of } G). \end{aligned}$$

Corollary 10.1.2

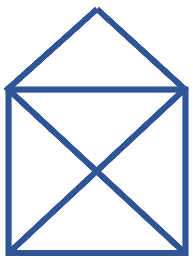
The total degree of a graph is even.

Proposition 10.1.3

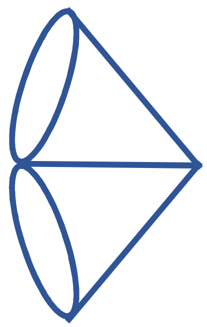
In any graph there are an even number of vertices of odd degree.

10.2 Trails, Paths, and Circuits

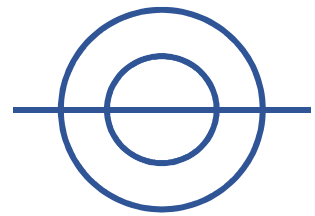
Let’s Have Some Fun



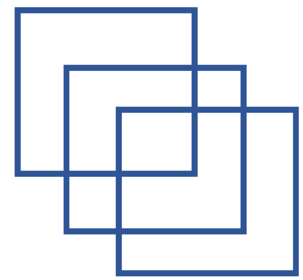
(1)



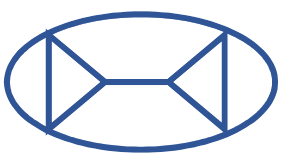
(2)



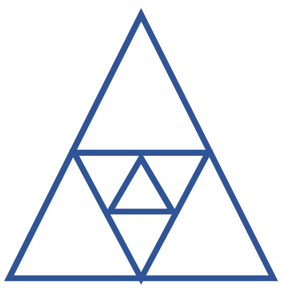
(3)



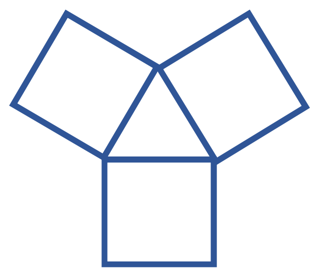
(4)



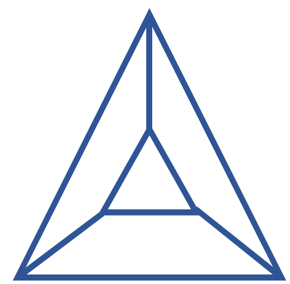
(5)



(6)



(7)



(8)

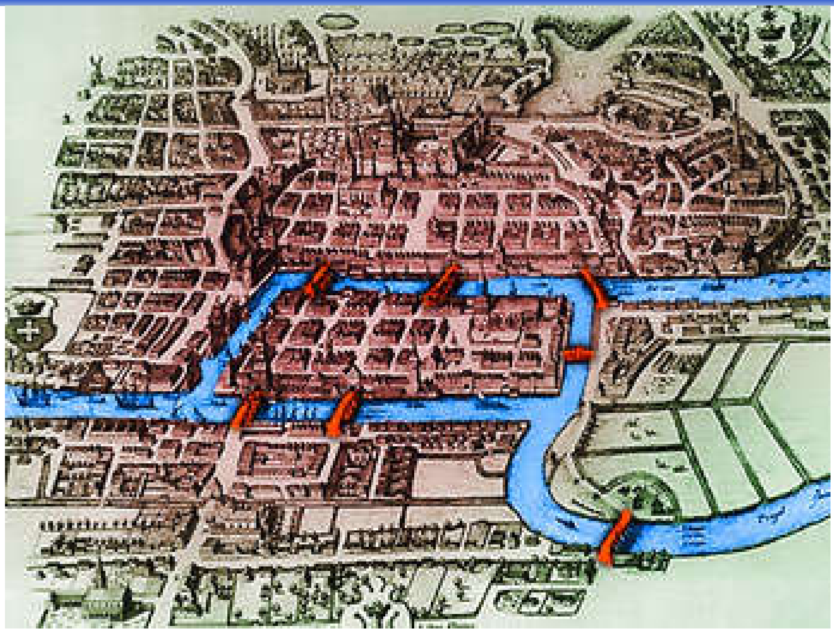
Königsberg bridges

The subject of graph theory began in the year 1736 when the great mathematician Leonhard Euler published a paper giving the solution to the following puzzle:

The town of Königsberg in Prussia (now Kaliningrad in Russia) was built at a point where two branches of the Pregel River came together. It consisted of an island and some land along the river banks.

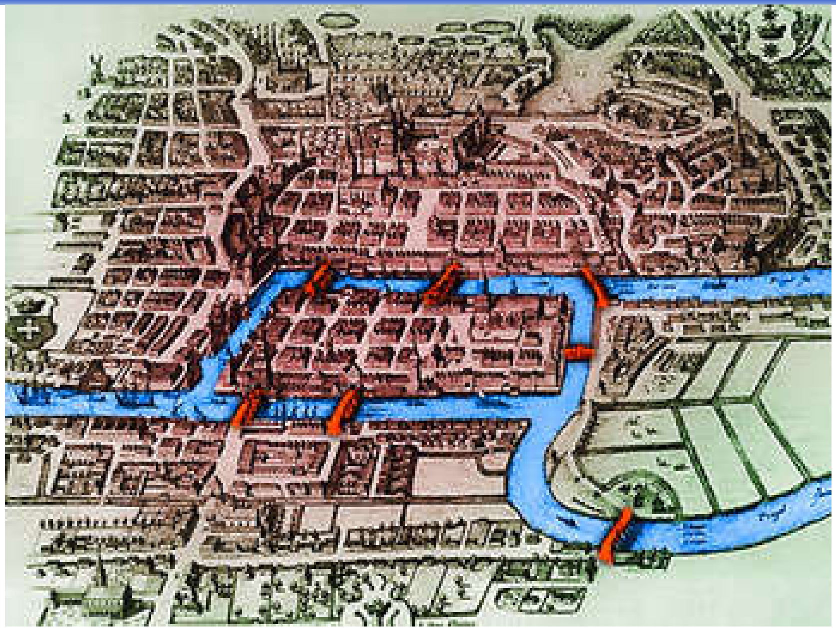
These were connected by 7 bridges.

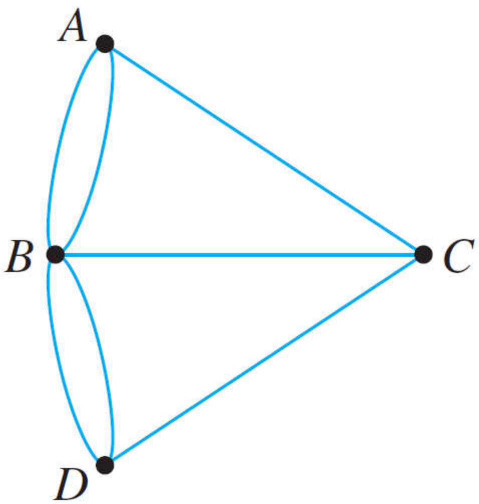
Königsberg bridges



Question: Is it possible to take a walk around town, starting and ending at the same location and crossing each of the 7 bridges exactly once?

Königsberg bridges





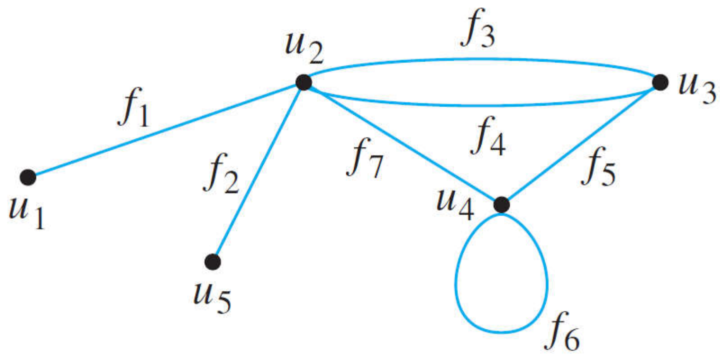
In terms of this graph, the question is:

Is it possible to find a route through the graph that starts and ends at some vertex, one of *A*, *B*, *C*, or *D*, and traverses each edge exactly once?

Definitions

Travel in a graph is accomplished by moving from one vertex to another along a sequence of adjacent edges.

In the graph below, for instance, you can go from u_1 to u_4 by taking f_1 to u_2 and then f_7 to u_4. This is represented by writing $u_1 f_1 u_2 f_7 u_4$.



Or, you could take a roundabout route:

$$u_1 f_1 u_2 f_3 u_3 f_4 u_2 f_3 u_3 f_5 u_4 f_6 u_4 f_7 u_2 f_3 u_3 f_5 u_4.$$

Walk, Trail, Path, Closed Walk, Circuit, Simple Circuit

Definitions

Let G be a graph, and let v and w be vertices of G.

A **walk from *v* to *w*** is a finite alternating sequence of adjacent vertices and edges of G. Thus a walk has the form

$$v_0\, e_1\, v_1\, e_2 \ldots v_{n-1}\, e_n\, v_n\,,$$

where the v's represent vertices, the e's represent edges, $v_0=v$, $v_n=w$, and for all $i \in \{1, 2, \ldots, n\}$, v_{i-1} and v_i are the endpoints of e_i.

The **trivial walk** from v to v consist of the single vertex v.

A **trail from *v* to *w*** is a walk from v to w that does not contain a repeated edge.

A **path from *v* to *w*** is a trail that does not contain a repeated vertex.

Walk, Trail, Path, Closed Walk, Circuit, Simple Circuit

Definitions

A **closed walk** is a walk that starts and ends at the same vertex.

A **circuit** (or **cycle**) is a closed walk that contains at least one edge and does not contain a repeated edge.

A **simple circuit** (or **simple cycle**) is a circuit that does not have any other repeated vertex except the first and last.

Walk, Trail, Path, Closed Walk, Circuit, Simple Circuit

	Repeated Edge?	Repeated Vertex?	Starts and Ends at Same Point?	Must Contain at Least One Edge?
Walk	allowed	allowed	allowed	no
Trail	no	allowed	allowed	no
Path	no	no	no	no
Closed walk	allowed	allowed	yes	no
Circuit	no	allowed	yes	yes
Simple circuit	no	first and last only	yes	yes

Often a walk can be specified unambiguously by giving either a sequence of edges or a sequence of vertices.

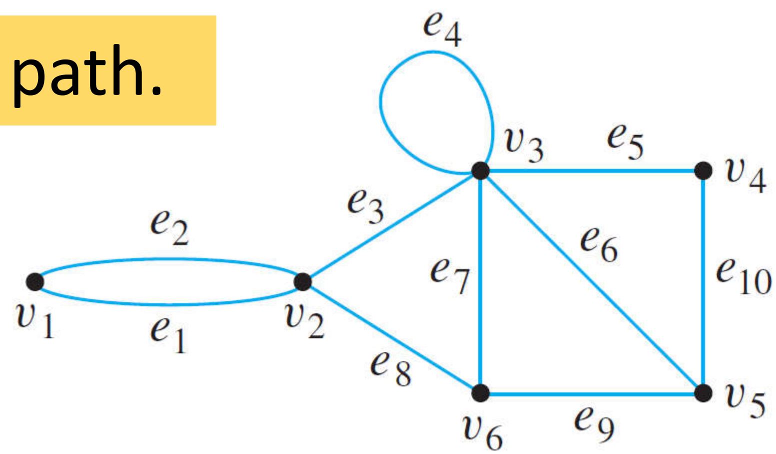
Walk, Trail, Path, Closed Walk, Circuit, Simple Circuit

In this graph, determine which of the following walks are trails, paths, circuits, or simple circuits.

a. $v_1 e_1 v_2 e_3 v_3 e_4 v_3 e_5 v_4$ Trail; not a path.

b. $e_1 e_3 e_5 e_5 e_6$ Walk; not a trail.

c. $v_2 v_3 v_4 v_5 v_3 v_6 v_2$ Circuit; not a simple circuit.



d. $v_2 v_3 v_4 v_5 v_6 v_2$

e. $v_1 e_1 v_2 e_1 v_1$

f. v_1

Notes

Because most of the major developments in graph theory have happened relatively recently and in a variety of different contexts, the terms used in the subject have not been standardized.

For example, what this book calls a *graph* is sometimes called a *multigraph*, what this book calls a *simple graph* is sometimes called a *graph*, what this book calls a *vertex* is sometimes called a *node*, and what this book calls an *edge* is sometimes called an *arc.*

Notes

Similarly, instead of the word *trail*, the word *path* is sometimes used; instead of the word *path*, the words simple path are sometimes used; and instead of the words *simple circuit*, the word *cycle* is sometimes used.

The terminology in this book is among the most common, but if you consult other sources, be sure to check their definitions.

Connectedness

A graph is connected if it is possible to travel from any vertex to any other vertex along a sequence of adjacent edges of the graph.

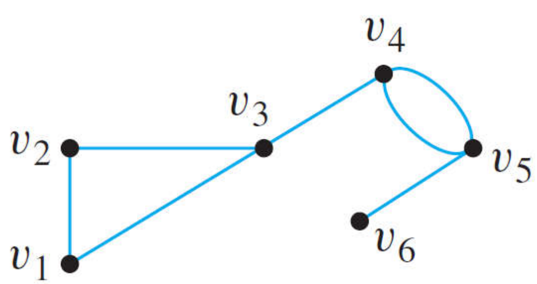
Definition: Connectedness

Two vertices v and w of a graph G are **connected** if, and only if, there is a walk from v to w.

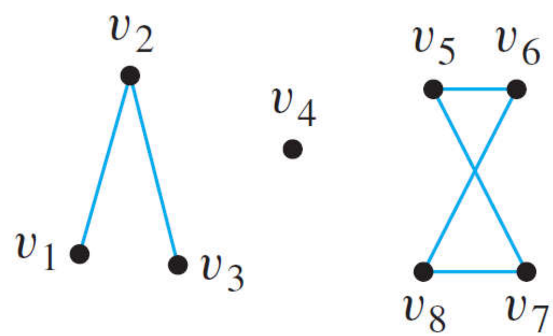
The graph *G* is connected if, and only if, given *any* two vertices v and w in G, there is a walk from v to w. Symbolically,

G is connected iff $\forall$ vertices $v, w \in V(G)$, $\exists$ a walk from v to w.

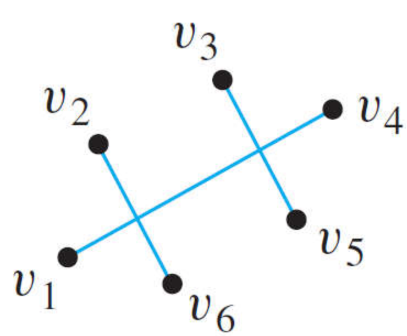
Example: Which of the following graphs are connected?



(a)



(b)



(c)

Some useful facts relating circuits and connectedness are collected in the following lemma.

Lemma 10.2.1

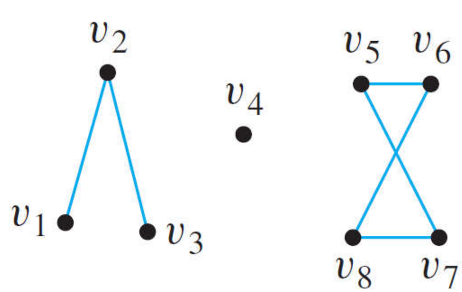
Let G be a graph.

a. If G is connected, then any two distinct vertices of G can be connected by a path.
b. If vertices v and w are part of a circuit in G and one edge is removed from the circuit, then there still exists a trail from v to w in G.
c. If G is connected and G contains a circuit, then an edge of the circuit can be removed without disconnecting G.

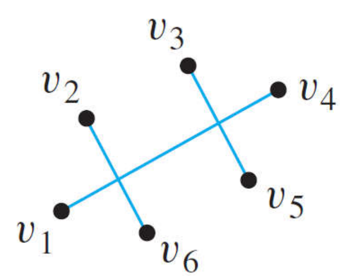
Connected Component

The graphs in (*b*) and (*c*) are both made up of three pieces, each of which is itself a connected graph.

A *connected component* of a graph is a connected subgraph of largest possible size.



(b)



(c)

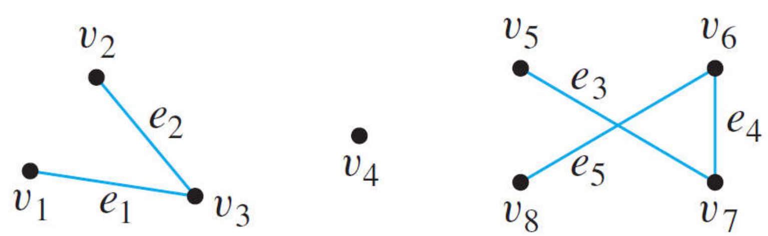
Definition: Connected Component

A graph *H* is a **connected component** of a graph *G* if, and only if,

1. The graph *H* is a subgraph of *G*;
2. The graph *H* is connected; and
3. No connected subgraph of *G* has *H* has a subgraph and contains vertices or edges that are not in *H*.

The fact is that any graph is a kind of union of its connected components.

Find all connected components of the following graph G.



G has 3 connected components H_1, H_2 and H_3 with vertex sets V_1, V_2 and V_3 and edge sets E_1, E_2 and E_3 , where

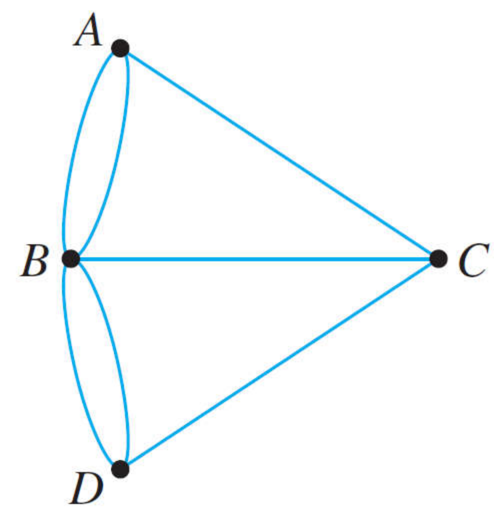
$$V_1 = \{v_1, v_2, v_3\}, \qquad E_1 = \{e_1, e_2\}$$

$$V_2 = \{v_4\}, \qquad E_2 = \emptyset$$

$$V_3 = \{v_5, v_6, v_7, v_8\}, \quad E_3 = \{e_3, e_4, e_5\}$$

Euler Circuits

Now, let's go back to the puzzle of the Königsberg bridges.



Is it possible to find a route through the graph that starts and ends at some vertex, one of *A*, *B*, *C*, or *D*, and traverses each edge exactly once?

Definition: Euler Circuit

Let *G* be a graph. An **Euler circuit** for *G* is a circuit that contains every vertex and every edge of *G*.

That is, an Euler circuit for *G* is a sequence of adjacent vertices and edges in *G* that has at least one edge, starts and ends at the same vertex, uses every vertex of *G* at least once, and uses every edge of *G* exactly once.

Theorem 10.2.2

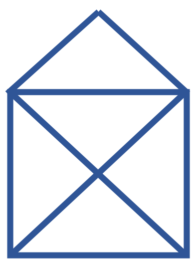
If a graph has an Euler circuit, then every vertex of the graph has positive even degree.

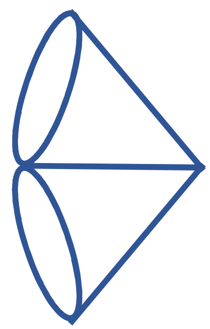
Contrapositive Version of Theorem 10.2.2

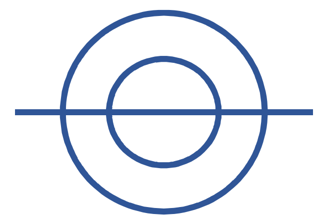
If some vertex of a graph has odd degree, then the graph does not have an Euler circuit.

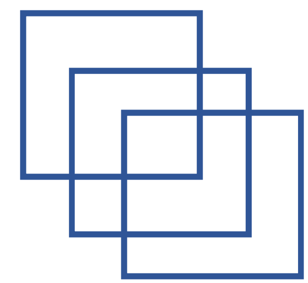
Euler Circuits

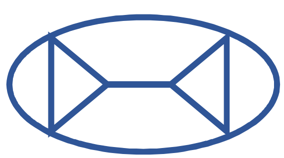
Does each of the following graphs have an Euler circuit?

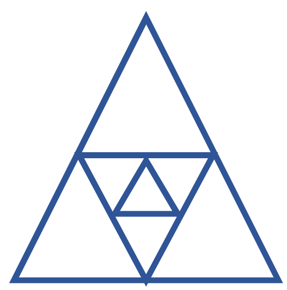

(1)

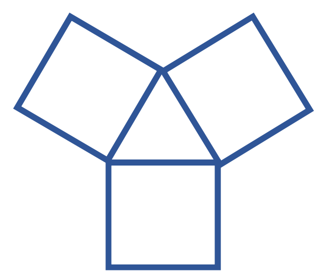

(2)

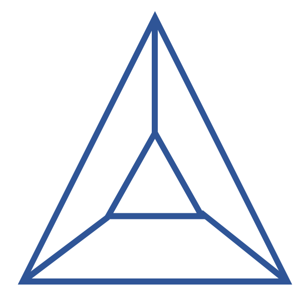

(3)


(4)


(5)


(6)


(7)


(8)

Is the converse of Theorem 10.2.2 true?

If every vertex of a graph has even degree, then the graph has an Euler circuit.

Theorem 10.2.3

If a graph *G* is connected and the degree of every vertex of *G* is a positive even integer, then *G* has an Euler circuit.

The proof of Theorem 10.2.3 is constructive: It contains an algorithm to find an Euler circuit for any connected graph in which every vertex has even degree.

Theorem 10.2.4

A graph *G* has an Euler circuit if, and only if, *G* is connected and every vertex of *G* has positive even degree.

A corollary to Theorem 10.2.4 gives a criterion for determining when it is possible to find a walk from one vertex of a graph to another, passing through every vertex of the graph at least once and every edge of the graph exactly once.

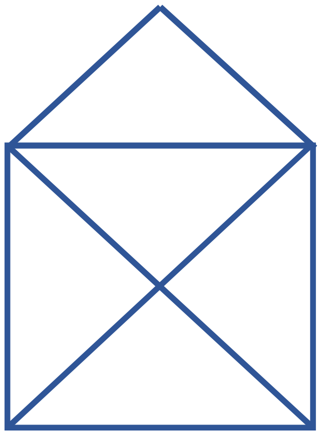
Definition: Euler Trail

Let G be a graph, and let v and w be two distinct vertices of G. An **Euler trail/path from *v* to *w*** is a sequence of adjacent edges and vertices that starts at v, ends at w, passes through every vertex of G at least once, and traverses every edge of G exactly once.

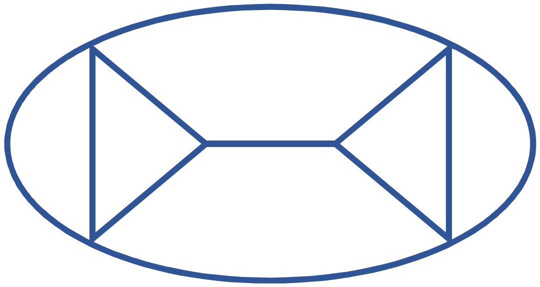
Corollary 10.2.5

Let G be a graph, and let v and w be two distinct vertices of G. There is an Euler trail from v to w if, and only if, G is connected, v and w have odd degree, and all other vertices of G have positive even degree.

The following graphs do not have an Euler circuit.
Do they have an Euler trail/path?



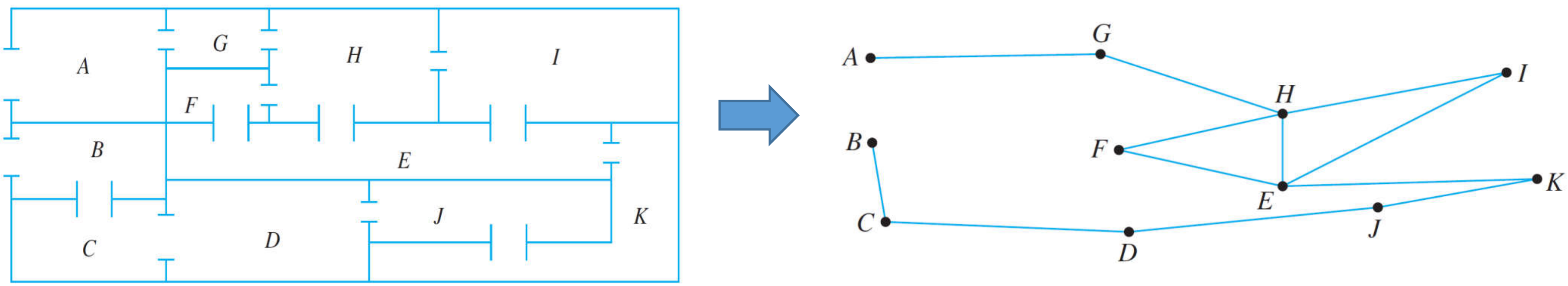
(1)



(5)

Euler Circuits

The floor plan shown below is for a house that is open for public viewing. Is it possible to find a trail that starts in room *A*, ends in room *B*, and passes through every interior doorway of the house exactly once? If so, find such a trail.



Each vertex of this graph has even degree except for *A* and *B*, each of which has degree 1.

Hence by Corollary 10.2.5, there is an Euler path from *A* to *B*. One such trail is *AGHFEIHEKJDCB.*

Hamiltonian Circuits

Recall Theorem 10.2.4:

Theorem 10.2.4

A graph G has an Euler circuit if, and only if, G is connected and every vertex of G has positive even degree.

A related question:

Given a graph G, is it possible to find a circuit for G in which all the *vertices* of G (except the first and the last) appear exactly once?

In 1859 the Irish mathematician Sir William Rowan Hamilton introduced a puzzle in the shape of a dodecahedron (DOH-dek-a-HEE-dron). (Figure 10.2.6 contains a drawing of a dodecahedron, which is a solid figure with 12 identical pentagonal faces.)

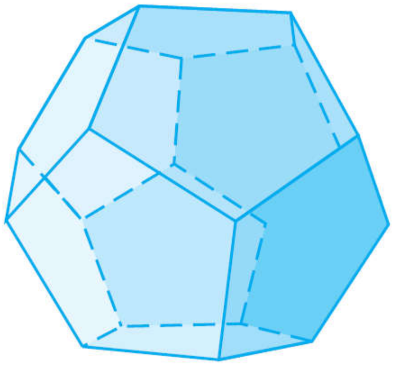
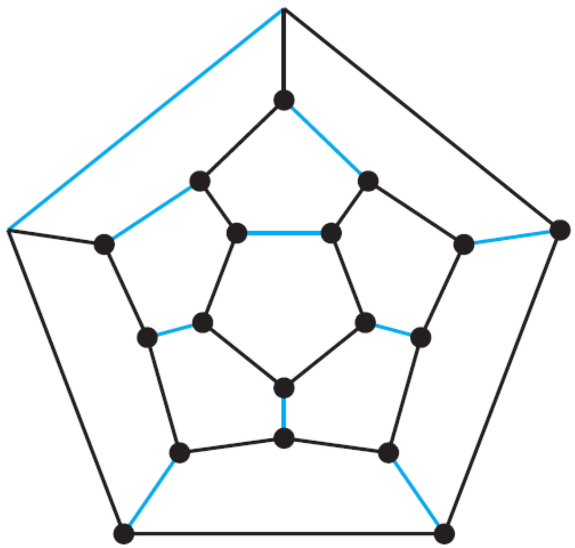


Figure 10.2.6 Dodecahedron

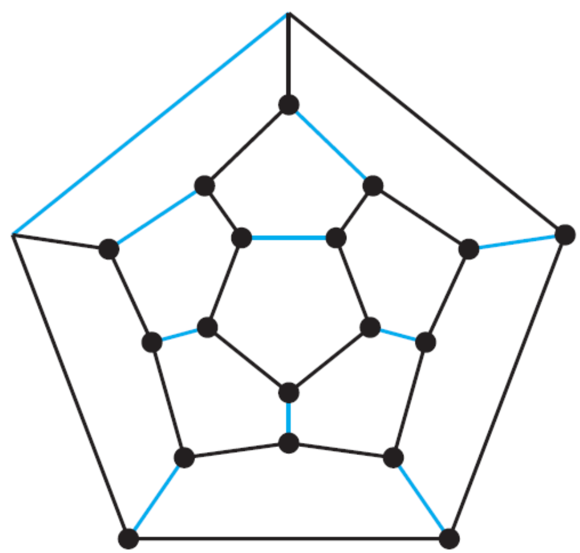
Each vertex was labeled with the name of a city — London, Paris, Hong Kong, New York, and so on.

The problem Hamilton posed was to start at one city and tour the world by visiting each other city exactly once and returning to the starting city.

One way to solve the puzzle is to imagine the surface of the dodecahedron stretched out and laid flat in the plane, as follows:



The circuit denoted with black lines is one solution. Note that although every city is visited, many edges are omitted from the circuit. (More difficult versions of the puzzle required that certain cities be visited in a certain order.)



Definition: Hamiltonian Circuit

Given a graph G, a **Hamiltonian circuit** for G is a simple circuit that includes every vertex of G.

That is, a Hamiltonian circuit for G is a sequence of adjacent vertices and distinct edges in which every vertex of G appears exactly once, except for the first and the last, which are the same.

Note that although an Euler circuit for a graph *G* must include every vertex of *G*, it may visit some vertices more than once and hence may not be a Hamiltonian circuit.

On the other hand, a Hamiltonian circuit for *G* does not need to include all the edges of *G* and hence may not be an Euler circuit.

Despite the analogous-sounding definitions of Euler and Hamiltonian circuits, the mathematics of the two are <u>very different</u>.

Theorem 10.2.4 gives a simple criterion for determining whether a given graph has an Euler circuit.

Unfortunately, there is no analogous criterion for determining whether a given graph has a Hamiltonian circuit, nor is there even an efficient algorithm for finding such a circuit.

There is, however, a simple technique that can be used in many cases to show that a graph does *not* have a Hamiltonian circuit.

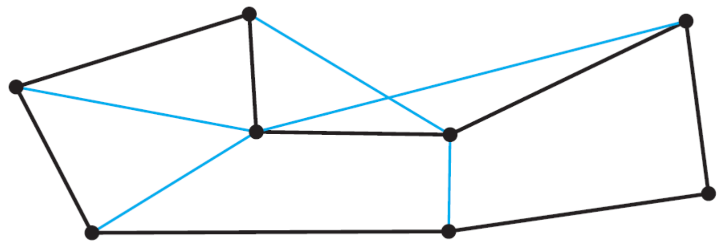
This follows from the following considerations:

Suppose a graph *G* with at least two vertices has a Hamiltonian circuit *C* given concretely as

$$C\colon v_0 e_1 v_1 e_2 \cdots v_{n-1} e_n v_n.$$

Since *C* is a simple circuit, all the e_i are distinct and all the v_j are distinct except that $v_0 = v_n$. Let *H* be the subgraph of *G* that is formed using the vertices and edges of *C*.

An example of such an *H* is shown below.



H is indicated by the black lines.

Note that *H* has the same number of edges as it has vertices since all its *n* edges are distinct and so are its *n* vertices $v_1, v_2, \ldots, v_n$.

Also, by definition of Hamiltonian circuit, every vertex of *G* is a vertex of *H*, and *H* is connected since any two of its vertices lie on a circuit. In addition, every vertex of *H* has degree 2.

$C: v_0 e_1 v_1 e_2 \cdots v_{n-1} e_n v_n.$

The reason for this is that there are exactly two edges incident on any vertex. These are e_i and e_{i+1} for any vertex v_i except $v_0=v_n$, and they are e_1 and e_n for v_0 ($=v_n$).

These observations have established the truth of the following proposition in all cases where G has at least two vertices.

Proposition 10.2.6

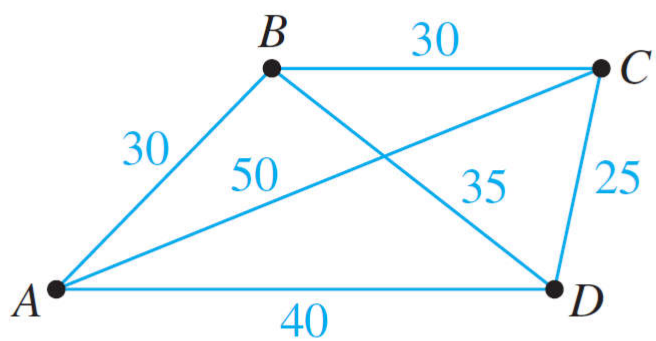
If a graph G has a Hamiltonian circuit, then G has a subgraph H with the following properties:

1. H contains every vertex of G.
2. H is connected.
3. H has the same number of edges as vertices.
4. Every vertex of H has degree 2.

The contrapositive of Proposition 10.2.6 says that if a graph G does *not* have a subgraph H with properties (1)–(4), then G does *not* have a Hamiltonian circuit.

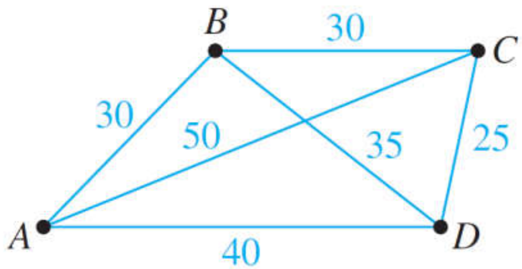
Travelling Salesman Problem

Imagine that the drawing below is a map showing four cities and the distances in kilometers between them.



Suppose that a salesman must travel to each city exactly once, starting and ending in city *A*. Which route from city to city will minimize the total distance that must be travelled?

This problem can be solved by writing all possible Hamiltonian circuits starting and ending at *A* and calculating the total distance travelled for each.



Route	Total Distance (In Kilometers)	
$ABCDA$	$30 + 30 + 25 + 40 = 125$	
$ABDCA$	$30 + 35 + 25 + 50 = 140$	
$ACBDA$	$50 + 30 + 35 + 40 = 155$	
$ACDBA$	140	[$ABDCA$ backwards]
$ADBCA$	155	[$ACBDA$ backwards]
$ADCBA$	125	[$ABCDA$ backwards]

Thus either route *ABCDA* or *ADCBA* gives a minimum total distance of 125 km.

The general travelling salesman problem involves finding a Hamiltonian circuit to minimize the total distance travelled for an arbitrary graph with *n* vertices in which each edge is marked with a distance.

One way to solve the general problem is to use the previous method: Write down all Hamiltonian circuits starting and ending at a particular vertex, compute the total distance for each, and pick one for which this total is minimal.

However, this is impractical for even medium-sized values of *n*. For $n = 30$ vertices, there would be $(29!)/2 \approx 4.42\times10^{30}$ Hamiltonian circuits starting and ending at a particular vertex to check. If each circuit could be found and its total distance computed in just one nanosecond, it would take approximately 1.4×10^{14} years to compute!

At present, there is no known algorithm for solving the general travelling salesman problem that is more efficient.

However, there are efficient algorithms that find "pretty good" solutions—that is, circuits that, while not necessarily having the least possible total distances, have smaller total distances than most other Hamiltonian circuits.

10.3 Matrix Representations of Graphs

Matrices

Definition: Matrix

An $m \times n$ (read "m by n") **matrix** A over a set S is a rectangular array of elements of S arranged into m row and n columns.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix}$$

← *i*th row of **A**

↑ *j*th column of **A**

We write **A** = (a_{ij}).

If **A** and **B** are matrices, then **A** = **B** if, and only if, **A** and **B** have the same size and the corresponding entries of **A** and **B** are all equal; that is,

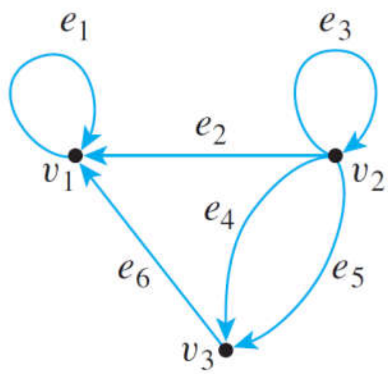
$$a_{ij} = b_{ij} \text{ for all } i = 1, 2, \cdots, m \text{ and } j = 1, 2, \cdots, n.$$

A matrix for which the numbers of rows and columns are equal is called a square matrix.

If **A** is a square matrix of size $n \times n$, then the main diagonal of **A** consists of all the entries $a_{11}, a_{22}, \cdots a_{nn}$.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1i} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2i} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ii} & \cdots & a_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{ni} & \cdots & a_{nn} \end{bmatrix} \leftarrow \text{main diagonal of } \mathbf{A}$$

Matrices and Directed Graphs



Directed Graph G

(a)

$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 2 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

Adjacency Matrix

(b)

Figure 10.3.1 A Directed Graph and Its Adjacency Matrix

This graph G is represented by the matrix **A** = (a_{ij}) for which a_{ij} = number of arrows from v_i to v_j for all i = 1, 2, 3 and j = 1, 2, 3.

A is called the adjacency matrix of G.

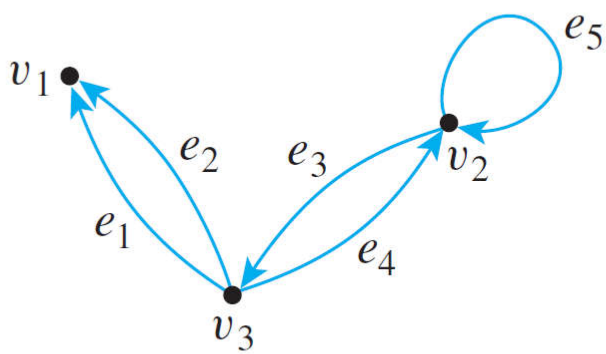
Definition: Adjacency Matrix of a Directed Graph

Let *G* be a directed graph with ordered vertices v_1, v_2, … v_n. The **adjacency matrix of *G*** is the $n \times n$ matrix $\mathbf{A} = (a_{ij})$ over the set of non-negative integers such that

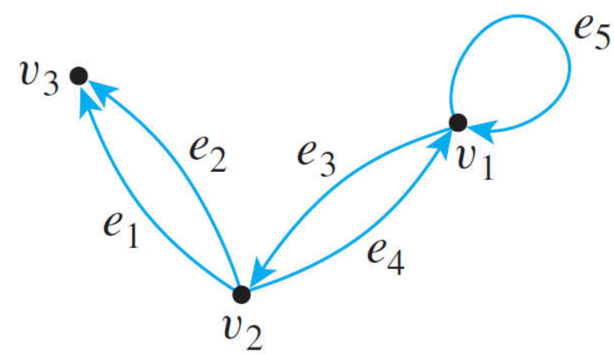
$$a_{ij} = \text{the number of arrows from } v_i \text{ to } v_j$$

for all $i, j = 1, 2, \ldots, n$.

Example: Find the adjacency matrices of the two directed graphs below.



(a)



(b)

$$
\begin{array}{c c}
 & \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\
\begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 2 & 1 & 0 \end{bmatrix}
\end{array}
$$

(a)

Matrices and Undirected Graphs

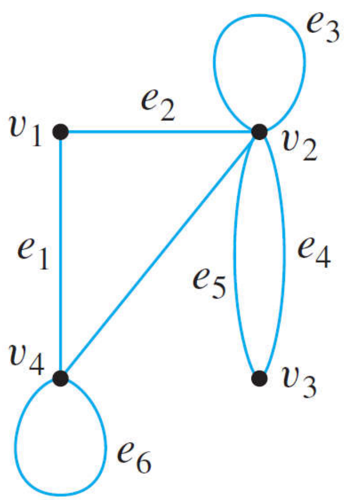
Definition: Adjacency Matrix of an Undirected Graph

Let G be an undirected graph with ordered vertices v_1, v_2, … v_n. The **adjacency matrix of *G*** is the $n \times n$ matrix $\mathbf{A} = (a_{ij})$ over the set of non-negative integers such that

$$a_{ij} = \text{the number of edges connecting } v_i \text{ and } v_j$$

for all $i, j = 1, 2, \ldots, n$.

Example: Find the adjacency matrix for the graph *G* shown below.



$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 2 & 1 \\ 0 & 2 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Note that the matrix is symmetric.

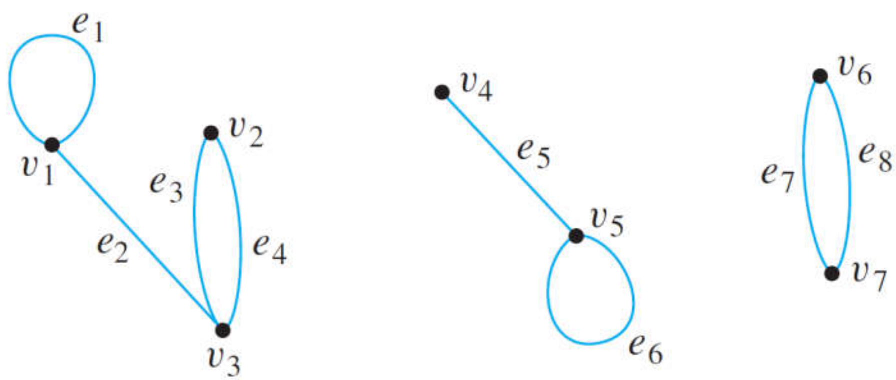
Definition: Symmetric Matrix

An $n \times n$ square matrix A = (a_{ij}) is called **symmetric** if, and only if, for all *i*, *j* = 1, 2, …, *n*,

$$a_{ij} = a_{ji}.$$

Matrices and Connected Components

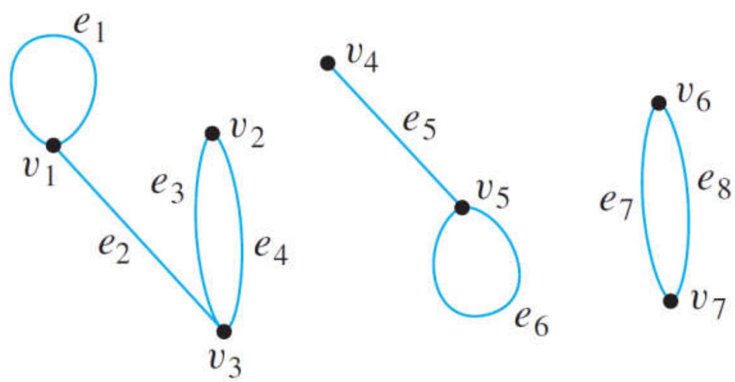
Consider a graph *G*, as shown below, that consists of several connected components.



Adjacency matrix of *G*:

$$\mathbf{A} = \left[\begin{array}{ccc|cc|cc} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{array}\right]$$

As you can see, **A** consists of square matrix blocks (of different sizes) down its diagonal and blocks of 0’s everywhere else.



The reason is that vertices in each connected component share no edges with vertices in other connected components.

$$\mathbf{A} = \left[\begin{array}{ccc:cc:cc} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{array}\right]$$

For instance, since v_1, v_2, and v_3 share no edges with v_4, v_5, v_6, or v_7, all entries in the top three rows to the right of the third column are 0 and all entries in the left three columns below the third row are also 0.

Matrices and Connected Components

Sometimes matrices whose entries are all 0's are themselves denoted 0. If this convention is followed here, **A** is written as:

$$
\mathbf{A} = \left[\begin{array}{ccc|cc|cc}
1 & 0 & 1 & & & & \\
0 & 0 & 2 & \multicolumn{2}{c|}{\mathbf{0}} & \multicolumn{2}{c}{\mathbf{0}} \\
1 & 2 & 0 & & & & \\
\hline
\multicolumn{3}{c|}{\mathbf{0}} & 0 & 1 & \multicolumn{2}{c}{\mathbf{0}} \\
 & & & 1 & 1 & & \\
\hline
\multicolumn{3}{c|}{\mathbf{0}} & \multicolumn{2}{c|}{\mathbf{0}} & 0 & 2 \\
 & & & & & 2 & 0
\end{array}\right]
$$

The previous reasoning can be generalized to prove the following theorem:

Theorem 10.3.1

Let G be a graph with connected components $G_1, G_1, \ldots, G_k$. If there are n_i vertices in each connected component G_i and these vertices are numbered consecutively, then the adjacency matrix of G has the form:

$$\begin{bmatrix} A_1 & O & O & & O & O \\ O & A_2 & O & \cdots & O & O \\ O & O & A_3 & & O & O \\ & \vdots & & & \vdots & \\ O & O & O & \cdots & O & A_k \end{bmatrix}$$

where each A_i is $n_i \times n_i$ adjacency matrix of G_i, for all $i = 1, 2, \ldots, k$, and the O's represent matrices whose entries are all 0s.

Matrix Multiplication

Definition: Scalar Product

Suppose that all entries in matrices **A** and **B** are real numbers. If the number of elements, *n*, in the *i*th row of **A** equals the number of elements in the *j*th column of **B**, then the **scalar product** or **dot product** of the *i*th row of **A** and the *j*th column of **B** is the real number obtained as follows:

$$\begin{bmatrix} a_{i1} & a_{i2} & \cdots & a_{in} \end{bmatrix} \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{nj} \end{bmatrix} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj}.$$

Definition: Matrix Multiplication

Let **A** = (a_{ij}) be an $m \times k$ matrix and **B** = (b_{ij}) an $k \times n$ matrix with real entries. The (matrix) product of **A** times **B**, denoted **AB**, is that matrix (c_{ij}) defined as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & & \vdots \\ \boxed{a_{i1} & a_{i2} & \cdots & a_{ik}} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mk} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1j} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2j} & \cdots & b_{2n} \\ & \cdot & & \cdot & & \cdot \\ & \cdot & & \cdot & & \cdot \\ & \cdot & & \cdot & & \cdot \\ b_{k1} & b_{k2} & \cdots & b_{kj} & \cdots & b_{kn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1j} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2j} & \cdots & c_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ c_{i1} & c_{i2} & \cdots & \boxed{c_{ij}} & \cdots & c_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mj} & \cdots & c_{mn} \end{bmatrix}$$

where

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{ik}b_{kj} = \textstyle\sum_{r=1}^{k} a_{ir}b_{rj}.$$

for all i = 1, 2, …, m and j = 1, 2, …, n.

Example – Computing a Matrix Product

Let $\mathbf{A} = \begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}$. Compute $\mathbf{AB}$.

Solution:

$$\begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ c_{21} & c_{22} \end{bmatrix},$$

where

$$c_{11} = 2\cdot 4 + 0\cdot 2 + 3\cdot(-2) = 2 \qquad \begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}$$

$$c_{12} = 2\cdot 3 + 0\cdot 2 + 3\cdot(-1) = 3 \qquad \begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}$$

Example – Computing a Matrix Product

Let $\mathbf{A} = \begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}$. Compute $\mathbf{AB}$.

Solution:

$$\begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ -2 & -1 \end{bmatrix},$$

where

$$c_{21} = (-1) \cdot 4 + 1 \cdot 2 + 0 \cdot (-2) = -2 \qquad \begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}$$

$$c_{22} = (-1) \cdot 3 + 0 \cdot 2 + 3 \cdot (-1) = -1 \qquad \begin{bmatrix} 2 & 0 & 3 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 2 & 2 \\ -2 & -1 \end{bmatrix}.$$

Multiplication of real numbers is commutative, but matrix multiplication is not.

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 0 & 1 \end{bmatrix}, \quad \text{but} \quad \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

On the other hand, both real number and matrix multiplications are associative ($(ab)c = a(bc)$, for all elements a, b, and c for which the products are defined).

Identity Matrix

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}.$$

These computations show that $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ acts as an identity on the left side for multiplication with 2×3 matrices and that $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ acts as an identity on the right side for multiplication with 3×3 matrices.

Definition: Identity Matrix

For each positive integer n, the $\boldsymbol{n \times n}$ **identity matrix**, denoted $I_n = (\delta_{ij})$ or just **I** (if the size of the matrix is obvious from context), is the $n \times n$ matrix in which all the entries in the main diagonal are 1's and all other entries are 0's. In other words,

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \qquad \text{for all } i, j = 1, 2, \ldots, n.$$

The German mathematician Leopold Kronecker introduced the symbol δ_{ij} to make matrix computations more convenient. In his honour, this symbol is called the *Kronecker delta*.

There are also similarities and differences between real numbers and matrices with respect to the computation of powers.

Any number can be raised to a non-negative integer power, but a matrix can be multiplied by itself only if it has the same number of rows as columns.

As for real numbers, however, the definition of matrix powers is recursive.

Just as any number to the zero power is defined to be 1, so any $n \times n$ matrix to the zero power is defined to be the $n \times n$ identity matrix.

n^{th} Power of a Matrix

Definition: Identity Matrix

For any $n \times n$ matrix **A**, the **powers of A** are defined as follows:

$\mathbf{A}^0 = \mathbf{I}$ where **I** is the $n \times n$ identity matrix

$\mathbf{A}^n = \mathbf{A}\,\mathbf{A}^{n-1}$ for all integers $n \geq 1$

Example – Power of a Matrix

Let $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}$. Compute $\mathbf{A}^0$, $\mathbf{A}^1$, $\mathbf{A}^2$, and $\mathbf{A}^3$.

Solution:

$$\mathbf{A}^0 = \text{the } 2 \times 2 \text{ identity matrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{A}^1 = \mathbf{A}\mathbf{A}^0 = \mathbf{A}\mathbf{I} = \mathbf{A}$$

$$\mathbf{A}^2 = \mathbf{A}\mathbf{A}^1 = \mathbf{A}\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 2 \\ 2 & 4 \end{bmatrix}$$

$$\mathbf{A}^3 = \mathbf{A}\mathbf{A}^2 = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}\begin{bmatrix} 5 & 2 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 9 & 10 \\ 10 & 4 \end{bmatrix}$$

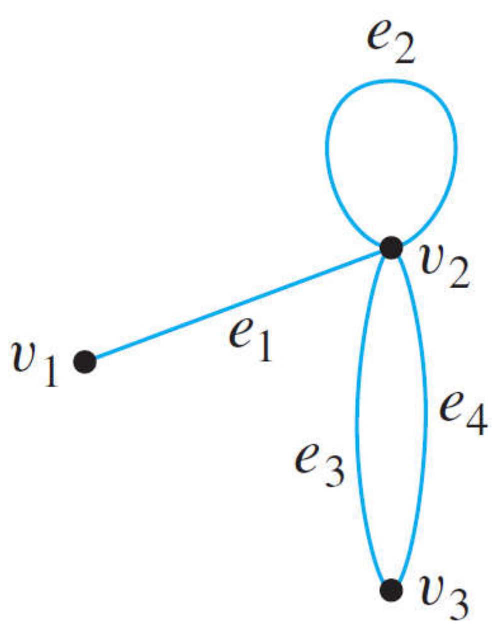
Counting Walks of Length *N*

A walk in a graph consists of an alternating sequence of vertices and edges.

If repeated edges are counted each time they occur, then the number of edges in the sequence is called the **length** of the walk.

For instance, the walk $v_2 e_3 v_3 e_4 v_2 e_2 v_2 e_3 v_3$ has length 4 (counting e_3 twice).

Example: Consider the following graph G.
How many distinct walks of length 2 connect v_2 and v_2?



One walk of length 2 from v_2 to v_1:
$v_2e_1v_1e_1v_2$.

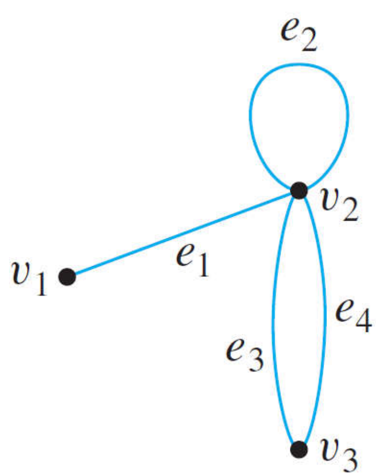
One walk of length 2 from v_2 to v_2:
$v_2e_2v_2e_2v_2$.

Four walks of length 2 from v_2 to v_3:
$v_2e_3v_3e_4v_2$,
$v_2e_4v_3e_3v_2$,
$v_2e_3v_3e_3v_2$,
$v_2e_4v_3e_4v_2$.

Total = 6

The general question of finding the number of walks that have a given length and connect two particular vertices of a graph can easily be answered using matrix multiplication.

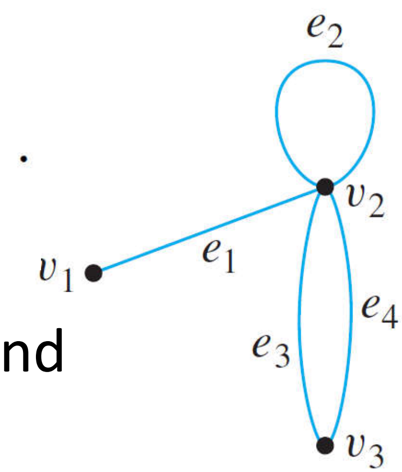
Consider the adjacency matrix **A** of the graph *G*.



$$\mathbf{A} = \begin{array}{c c} & \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array} & \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 2 \\ 0 & 2 & 0 \end{bmatrix} \end{array}.$$

Counting Walks of Length N

Compute $\mathbf{A}^2$: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 2 \\ 0 & 2 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 2 \\ 0 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 6 & 2 \\ 2 & 2 & 4 \end{bmatrix}.$



Note that the entry in the second row and the second column is 6, which equals the number of walks of length 2 from v_2 to v_2.

Reason: To compute a_{22}, you multiply the second row of **A** times the second column of **A** to obtain a sum of three terms:

$$\begin{bmatrix} 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = 1 \cdot 1 + 1 \cdot 1 + 2 \cdot 2.$$

More generally, if **A** is the adjacency matrix of a graph G, the $i\,j$-th entry of $\mathbf{A}^2$ equals the number of walks of length 2 connecting the i-th vertex to the j-th vertex of G.

Even more generally, if n is any positive integer, the $i\,j$-th entry of $\mathbf{A}^n$ equals the number of walks of length n connecting the i-th and the j-th vertices of G.

Theorem 10.3.2

If G is a graph with vertices $v_1, v_1, \ldots, v_m$ and **A** is the adjacency matrix of G, then for each positive integer n and for all integers $i, j = 1, 2, \ldots, m$,
the ij-th entry of $\mathbf{A}^n$ = the number of walks of length n from v_i to v_j.

10.4 Isomorphims of Graphs

Introduction

The two drawings shown in Figure 10.4.1 both represent the same graph: Their vertex and edge sets are identical, and their edge-endpoint functions are the same. Call this graph *G*.

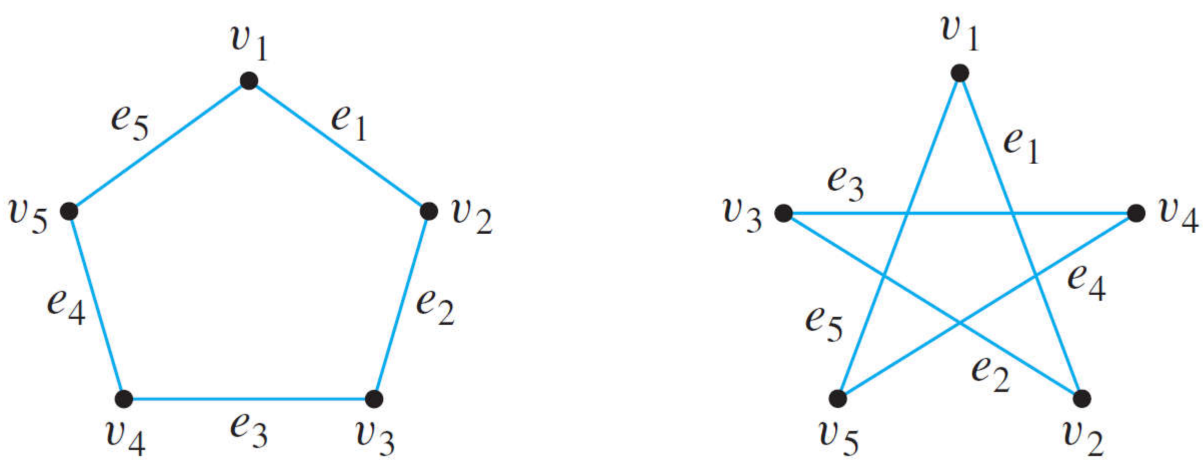


Figure 10.4.1

Now consider the graph G' represented in Figure 10.4.2.

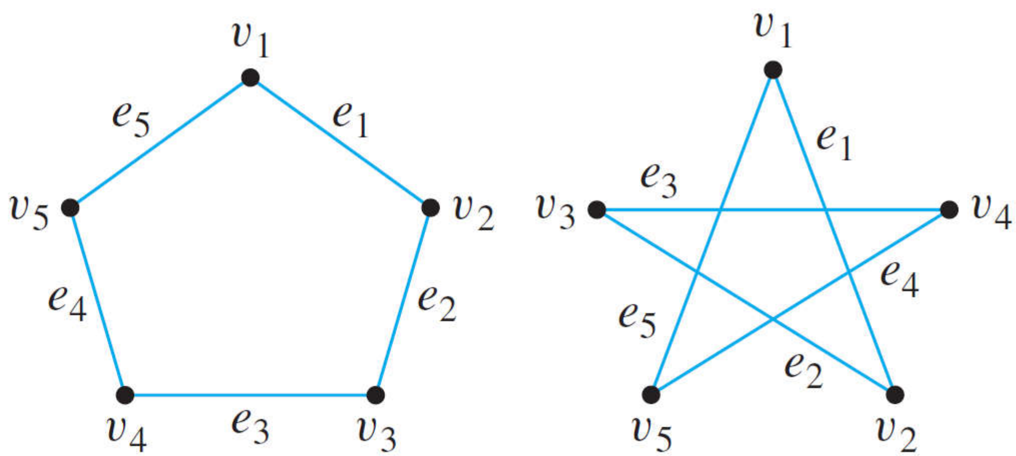


Figure 10.4.1

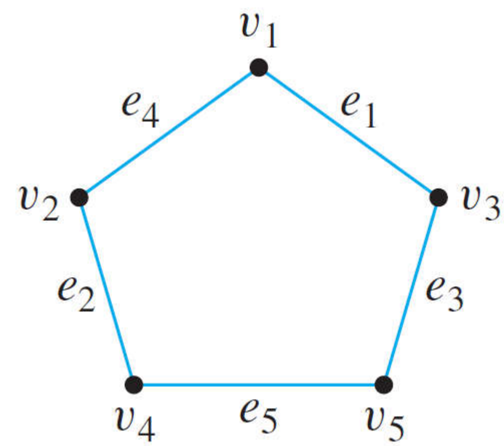


Figure 10.4.2

Observe that G' is a different graph from G (for instance, in G the endpoints of e_1 are v_1 and v_2, whereas in G' the endpoints of e_1 are v_1 and v_3).

Yet G' is certainly very similar to G. In fact, if the vertices and edges of G' are relabeled by the functions shown in Figure 10.4.3, then G' becomes the same as G.

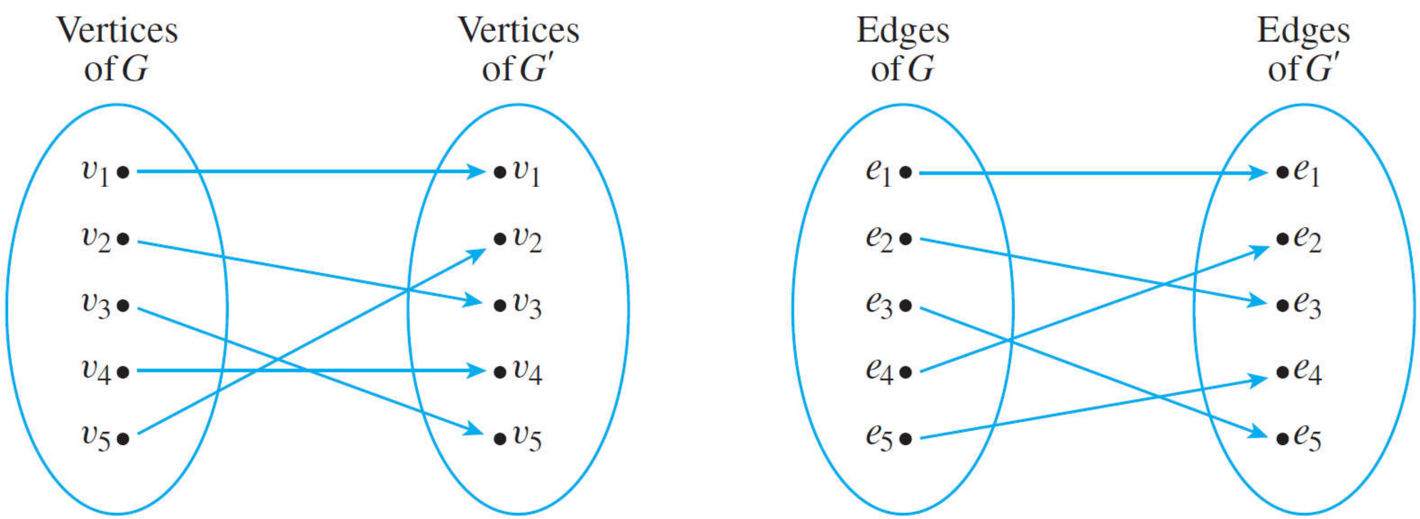


Figure 10.4.3

Note that these relabeling functions are **one-to-one** and **onto**.

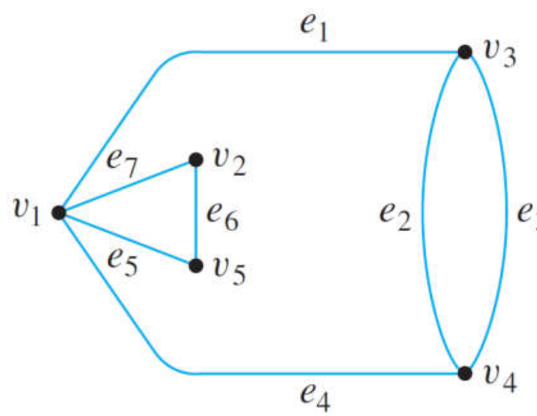
Two graphs that are the same except for the labeling of their vertices and edges are called *isomorphic*.

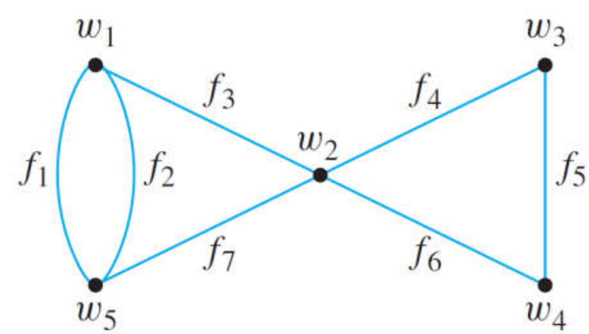
Definition: Isomorphic Graph

Let G and G' be graphs with vertex sets $V(G)$ and $V(G')$ and edge sets $E(G)$ and $E(G')$ respectively. **G is isomorphic to G'** if, and only if, there exist one-to-one correspondences g: $V(G) \rightarrow V(G')$ and h: $E(G) \rightarrow E(G')$ that preserve the edge-endpoint functions of G and G' in the sense that for all $v \in V(G)$ and $e \in E(G)$,

$$v \text{ is an endpoint of } e \Leftrightarrow g(v) \text{ is an endpoint of } h(e).$$

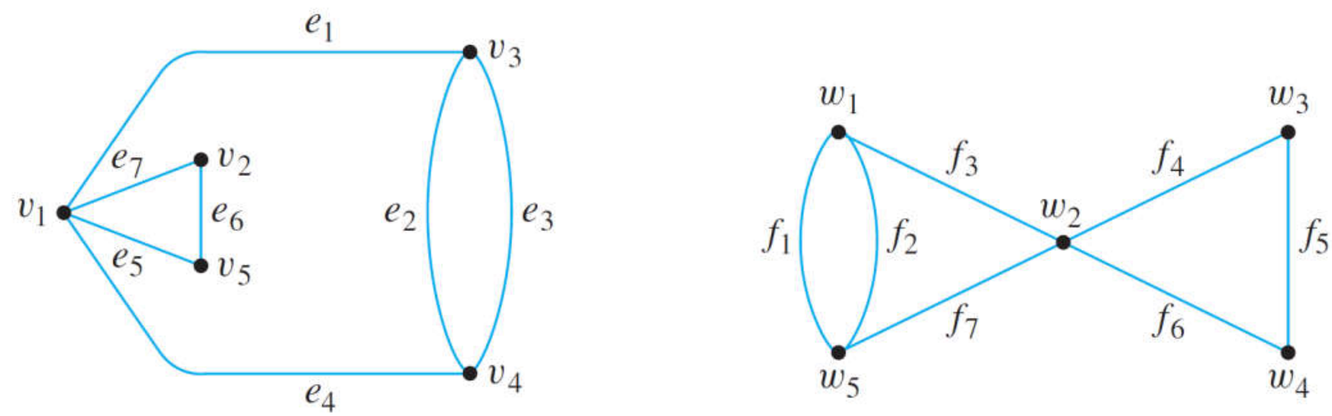
Example: Show that the following two graphs are isomorphic.

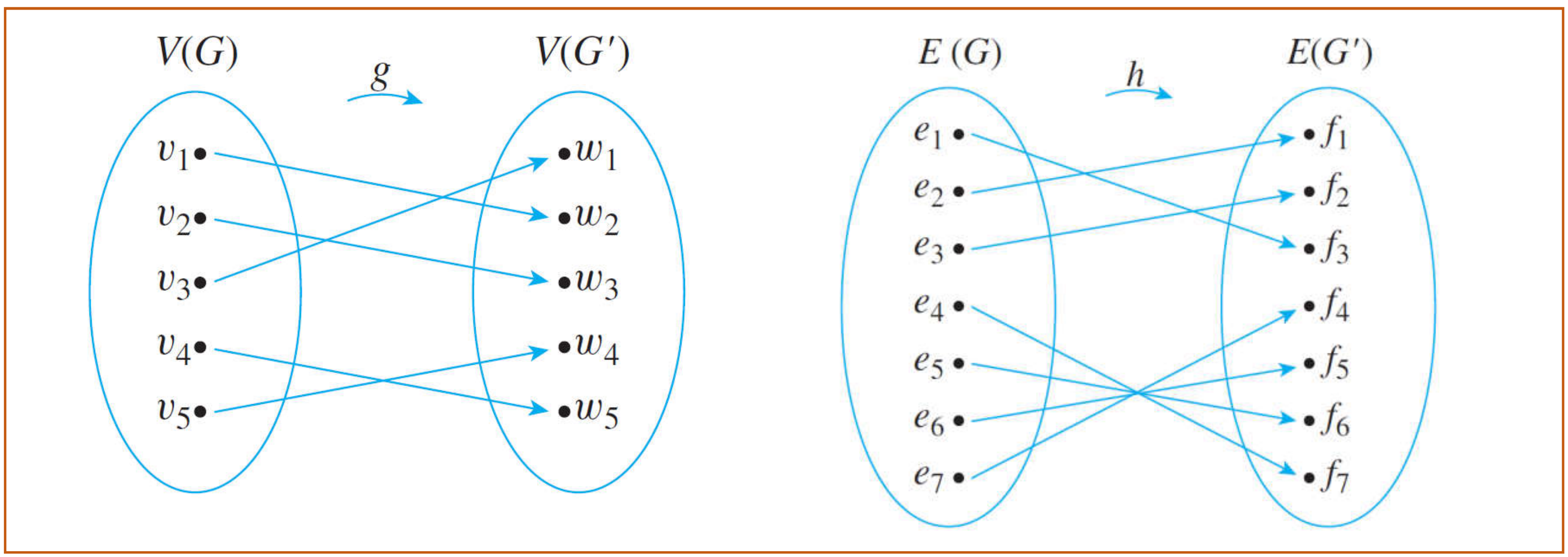




To solve this, you find functions g: $V(G) \rightarrow V(G')$ and h: $E(G) \rightarrow E(G')$ such that for all for all $v \in V(G)$ and $e \in E(G)$, v is an endpoint of e if, and only if, $g(v)$ is an endpoint of $h(e)$.

Example: Show that the following two graphs are isomorphic





It is not hard to show that graph isomorphism is an equivalence relation on a set of graphs; in other words, it is reflexive, symmetric, and transitive.

Theorem 10.4.1 Graph Isomorphism is an Equivalence Relation

Let S be a set of graphs and let R be the relation of graph isomorphism on S. Then R is an equivalence relation on S.

"Is there a general method (algorithm) to determine if two graphs G and G' are isomorphic?

There is such an algorithm, but it takes an unreasonably long time to perform.

If G and G' each has n vertices and m edges, the number of one-to-one correspondences from vertices to vertices is $n!$ and the number of one-to-one correspondences from edges to edges is $m!$, so the total number of pairs of functions to check is $n!m!$.

For example, if $m = n = 20$, there would be $20!20! \approx 5.9 \times 10^{36}$ pairs to check. If each check takes 1 ns, the total time would be approximately 1.9×10^{20} years!

Though there is no more efficient algorithm known for checking if two graphs are isomorphic, there are some simple tests that can be used to show that certain pairs of graphs are *not* isomorphic.

Since two isomorphic graphs must have the same number of vertices and edges, we know they are not isomorphic if they do not have the same number of vertices or edges.

A property that is preserved by graph isomorphism is called an isomorphic invariant. The number of vertices and edges are two such invariants.

Invariant for Graph Isomorphism

Definition: Invariant for Graph Isomorphism

A property P is called an **invariant for graph isomorphism** if, and only if, given any graphs G and G', if G has property P and G' is isomorphic to G, then G' has property P.

Invariant for Graph Isomorphism

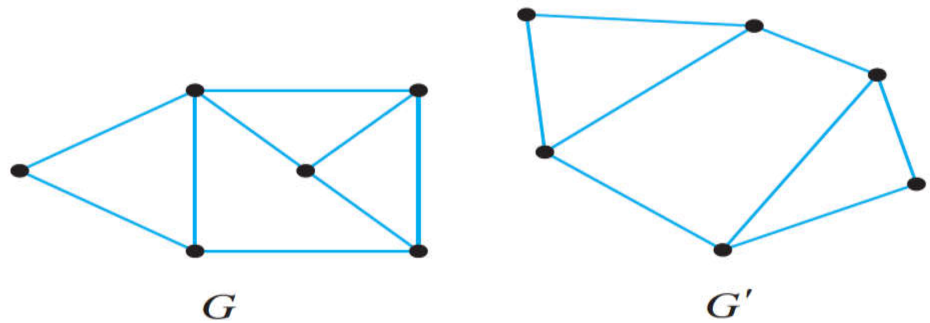
Theorem 10.4.2 Invariants for Graph Isomorphism

Each of the following properties is an invariant for graph isomorphism, where *n*, *m*, and *k* are all non-negative integers.

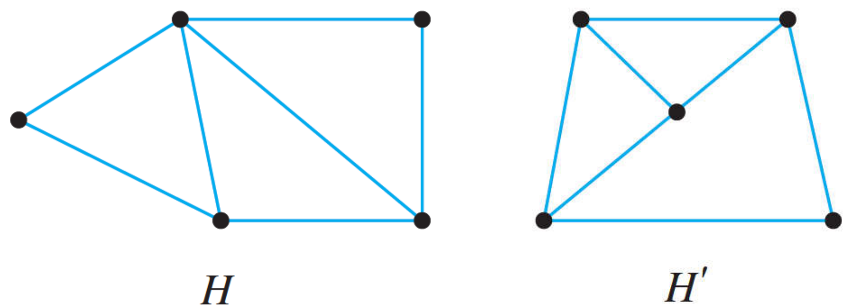
1. has *n* vertices;
2. has *m* edges;
3. has a vertex of degree *k*;
4. has *m* vertices of degree *k*;
5. has a circuit of length *k*;
6. has a simple circuit of length *k*;
7. has *m* simple circuits of length *k*;
8. is connected;
9. has an Euler circuit;
10. has a Hamiltonian circuit.

Show that the following pairs of graphs are not isomorphic by finding an isomorphic invariant that they do not share.

a.



b.



Graph Isomorphism for Simple Graphs

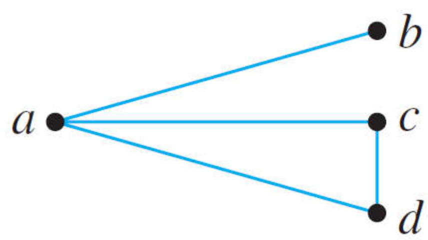
When graphs G and G' are both simple, the definition of G being isomorphic to G' can be written without referring to the correspondence between the edges of G and the edges of G'.

Definition

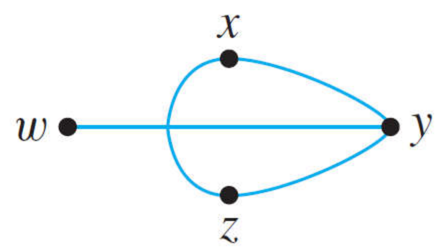
If G and G' are simple graphs, then **G is isomorphic to G'** if, and only if, there exists a one-to-one correspondence g from the vertex set $V(G)$ of G to the vertex set $V(G')$ of G' that preserves the edge-endpoint functions of G and G' in the sense that for all vertices u and v of G,

$$\{u, v\} \text{ is an edge in } G \Leftrightarrow \{g(u), g(v)\} \text{ is an edge in } G'.$$

Example: Are the two graphs shown below isomorphic? If so, define an isomorphism.



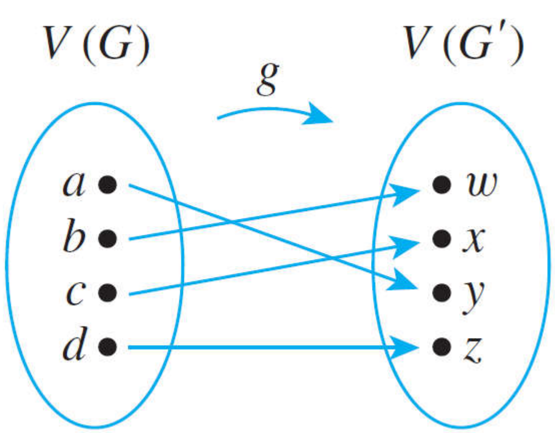
G



G'

Yes.

Define g: $V(G) \to V(G')$ by the arrow diagram shown below. Then g is one-to-one and onto by inspection.



Edges of G	Edges of G'
$\{a, b\}$	$\{y, w\} = \{g(a), g(b)\}$
$\{a, c\}$	$\{y, x\} = \{g(a), g(c)\}$
$\{a, d\}$	$\{y, z\} = \{g(a), g(d)\}$
$\{c, d\}$	$\{x, z\} = \{g(c), g(d)\}$

Next week's lectures

Trees

END OF FILE