Discrete Mathematics

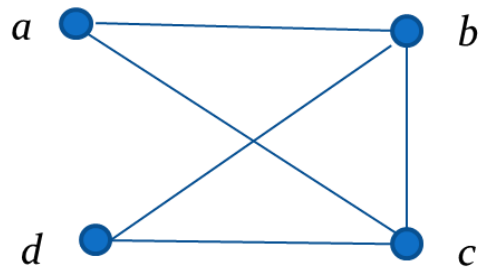
Graph

Shin Hong

Graphs

Definition: A *graph G* = (*V, E*) consists of a nonempty set *V* of *vertices* (or *nodes*) and a set *E* of *edges.* Each edge has either one or two vertices associated with it, called its *endpoints*. An edge is said to *connect* its endpoints.

Example:
This is a graph with four vertices and five edges.



Remarks:

- The graphs we study here are unrelated to graphs of functions studied in Chapter 2.
- We have a lot of freedom when we draw a picture of a graph. All that matters is the connections made by the edges, not the particular geometry depicted. For example, the lengths of edges, whether edges cross, how vertices are depicted, and so on, do not matter
- A graph with an infinite vertex set is called an *infinite graph.* A graph with a finite vertex set is called a *finite graph*. We (following the text) restrict our attention to finite graphs.

Some Terminology 1

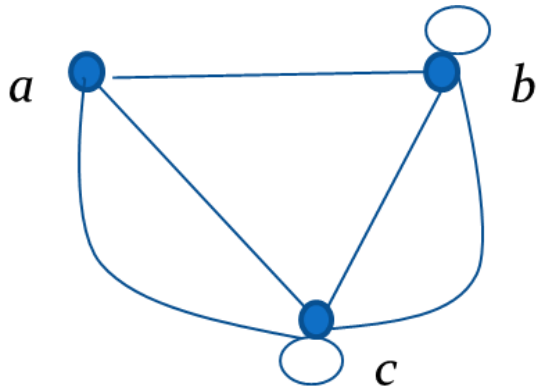
In a *simple graph* each edge connects two different vertices and no two edges connect the same pair of vertices.

Multigraphs may have multiple edges connecting the same two vertices. When *m* different edges connect the vertices *u* and *v*, we say that {*u*,*v*} is an edge of *multiplicity m*.

An edge that connects a vertex to itself is called a *loop*.

A *pseudograph* may include loops, as well as multiple edges connecting the same pair of vertices.

Example:
This pseudograph has both multiple edges and a loop.



Remark: There is no standard terminology for graph theory. So, it is crucial that you understand the terminology being used whenever you read material about graphs.

Directed Graphs[1]

Definition: An *directed graph* (or *digraph*) $G = (V, E)$ consists of a nonempty set V of *vertices* (or *nodes*) and a set E of *directed edges* (or *arcs*). Each edge is associated with an ordered pair of vertices. The directed edge associated with the ordered pair (u,v) is said to *start at* u and *end at* v.

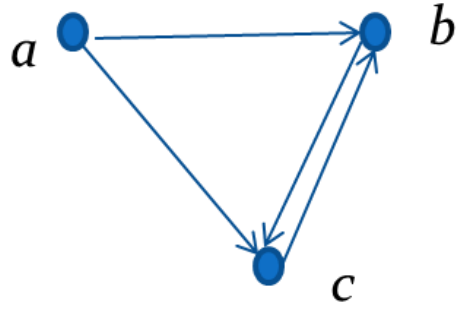
Remark:

- Graphs where the end points of an edge are not ordered are said to be *undirected graphs*.

Some Terminology 2

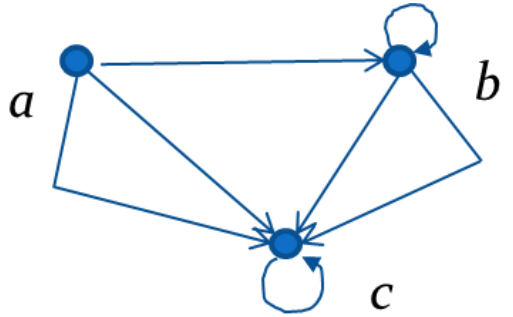
A *simple directed graph* has no loops and no multiple edges.

Example:
This is a directed graph with
three vertices and four edges.



A *directed multigraph* may have multiple directed edges. When there are *m* directed edges from the vertex *u* to the vertex *v*, we say that (*u,v*) is an edge of *multiplicity m*.

Example:
In this directed multigraph the
multiplicity of (*a,b*) is **1 and the**
multiplicity of (*b,c*) is 2.



Graph Models: Social Networks[1]

Graphs can be used to model social structures based on different kinds of relationships between people or groups.

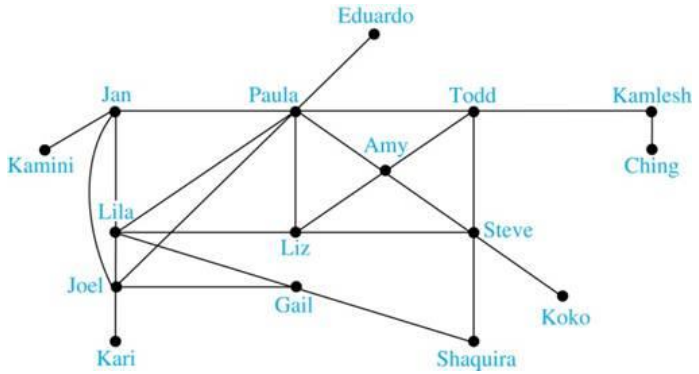
In a *social network*, vertices represent individuals or organizations and edges represent relationships between them.

Useful graph models of social networks include:

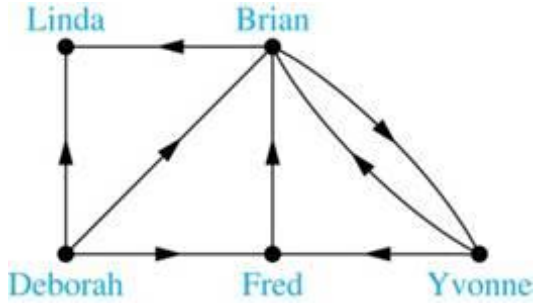
- *friendship graphs* - undirected graphs where two people are connected if they are friends (in the real world, on Facebook, or in a particular virtual world, and so on.)
- *collaboration graphs* - undirected graphs where two people are connected if they collaborate in a specific way
- *influence graphs* - directed graphs where there is an edge from one person to another if the first person can influence the second person

Graph Models: Social Networks 2

Example: A friendship graph where two people are connected if they are Facebook friends.



Example: An influence graph



Jump to long description

Examples of Collaboration Graphs

The *Hollywood graph* models the collaboration of actors in films.

- We represent actors by vertices and we connect two vertices if the actors they represent have appeared in the same movie.
- We will study the Hollywood Graph in Section 10.4 when we discuss Kevin Bacon numbers.

An *academic collaboration graph* models the collaboration of researchers who have jointly written a paper in a particular subject.

- We represent researchers in a particular academic discipline using vertices.
- We connect the vertices representing two researchers in this discipline if they are coauthors of a paper.
- We will study the academic collaboration graph for mathematicians when we discuss *Erdős numbers* in Section 10.4.

Transportation Graphs

Graph models are extensively used in the study of transportation networks.

Airline networks can be modeled using directed multigraphs where

- airports are represented by vertices
- each flight is represented by a directed edge from the vertex representing the departure airport to the vertex representing the destination airport

Road networks can be modeled using graphs where

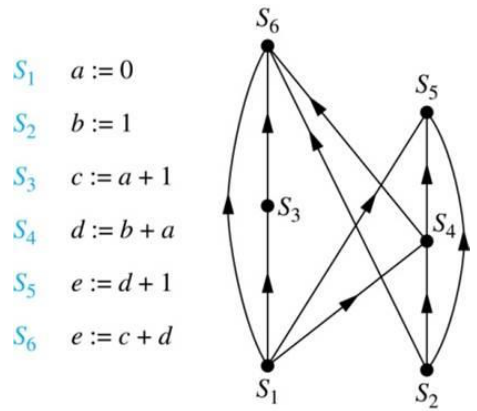
- vertices represent intersections and edges represent roads.
- undirected edges represent two-way roads and directed edges represent one-way roads.

Software Design Applications 2

We can use a directed graph called a *precedence graph* to represent which statements must have already been executed before we execute each statement.

- Vertices represent statements in a computer program
- There is a directed edge from a vertex to a second vertex if the second vertex cannot be executed before the first

Example: This precedence graph shows which statements must already have been executed before we can execute each of the six statements in the program.



Jump to long description

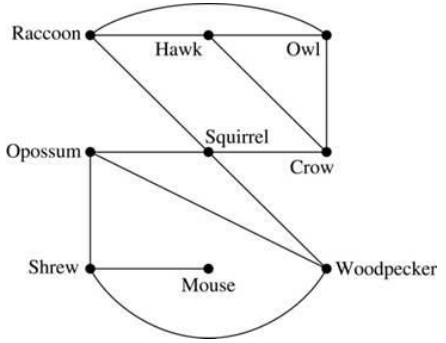
Biological Applications[1]

Graph models are used extensively in many areas of the biological science. We will describe two such models, one to ecology and the other to molecular biology.

Niche overlap graphs model competition between species in an ecosystem

- Vertices represent species and an edge connects two vertices when they represent species who compete for food resources.

Example: This is the niche overlap graph for a forest ecosystem with nine species.



Biological Applications 2

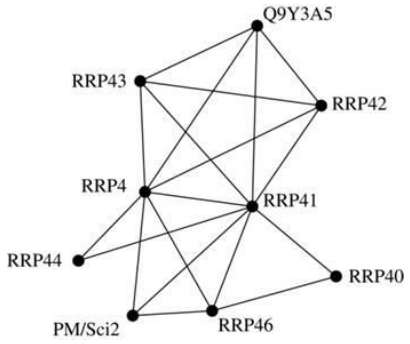
We can model the interaction of proteins in a cell using a *protein interaction network.*

In a *protein interaction graph*, vertices represent proteins and vertices are connected by an edge if the proteins they represent interact.

Protein interaction graphs can be huge and can contain more than 100,000 vertices, each representing a different protein, and more than 1,000,000 edges, each representing an interaction between proteins

Protein interaction graphs are often split into smaller graphs, called *modules*, which represent the interactions between proteins involved in a particular function.

Example: This is a module of the protein interaction graph of proteins that degrade RNA in a human cell.



Jump to long description

Basic Terminology

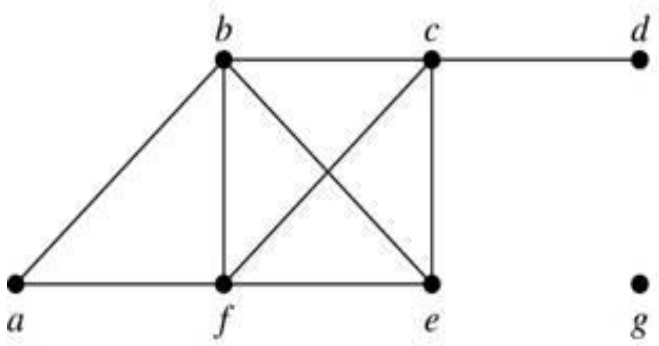
- Two vertices *u* and *v* in an undirected graph *G* are *adjacent* (*neighbors*) if there is an edge *e* between *u* and *v*.
  - Such an edge *e* is called *incident with* the vertices *u*

- The set of all neighbors of a vertex *v* of $G = (V, E)$, denoted by $N(v)$, is called the *neighborhood* of *v*.
  - If *A* is a subset of *V*, we denote by $N(A)$ the set of all vertices in *G* that are adjacent to at least one vertex in *A*: $N(A) = \bigcup_{v \in A} N(v)$.

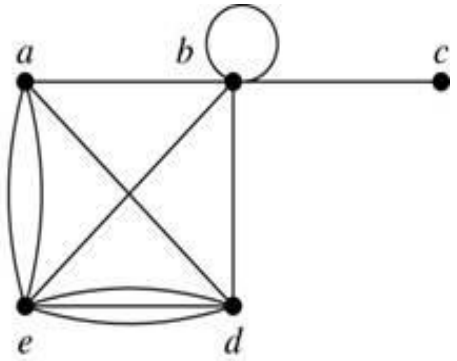
- The *degree of a vertex in a undirected graph* is the number of edges incident with it, except that a loop at a vertex.
  - The degree of the vertex *v* is denoted by $\deg(v)$.

Degrees and Neighborhoods of Vertices

Example: What are the degrees and neighborhoods of the vertices in the graphs *G* and *H*?



G



H

Solution:

G : deg(*a*) = 2, deg(*b*) = deg(*c*) = deg(*f*) = 4,
deg(*d*) = 1, deg(e) = 3, deg(*g*) = 0.

$N(a) = \{b, f\}, N(b) = \{a, c, e, f\}, N(c) = \{b, d, e, f\},$
$N(d) = \{c\}, N(e) = \{b, c, f\}, N(f) = \{a, b, c, e\},$
$N(g) = \varnothing$

H: deg(*a*) = 4, deg(*b*) = deg(e) = 6,
deg(*c*) = 1, deg(*d*) = 5.

$N(a) = \{b, d, e\}, N(b) = \{a, b, c, d, e\},$
$N(c) = \{b\}, N(d) = \{a, b, e\},$
$N(e) = \{a, b, d\}.$

Degrees of Vertices

Theorem 1 (*Handshaking Theorem*): If $G = (V,E)$ is an undirected graph with m edges, then

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

Proof:

Each edge contributes twice to the degree count of all vertices. Hence, both the left-hand and right-hand sides of this equation equal twice the number of edges.

Think about the graph where vertices represent the people at a party and an edge connects two people who have shaken hands.

Degree of Vertices

Theorem 2: An undirected graph has an even number of vertices of odd degree.

Proof: Let V_1 be the vertices of even degree and V_2 be the vertices of odd degree in an undirected graph $G = (V, E)$ with m edges. Then

$$\text{even} \rightarrow \quad 2m = \sum_{v \in V} \deg(v) = \sum_{v \in V_1} \deg(v) + \sum_{v \in V\,2} \deg(v).$$

must be even since deg(v) is even for each $v \in V_1$

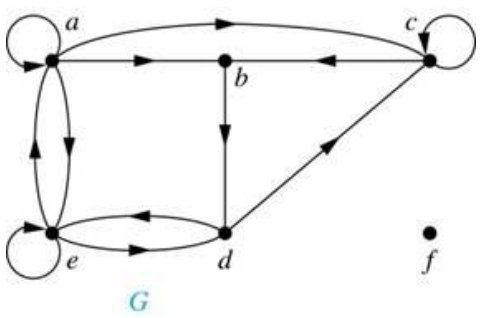
This sum must be even because $2m$ is even and the sum of the degrees of the vertices of even degrees is also even. Because this is the sum of the degrees of all vertices of odd degree in the graph, there must be an even number of such vertices.

Directed Graphs 2

- At a *directed graph G = (V, E),* each edge is an ordered pair of vertices. The directed edge (*u*,*v*) is said to start at *u* and end at *v*.
  - *u* is the *initial vertex*
  - *v* is the *terminal* (or *end*) *vertex*
  - *u* is adjacent to *v* ; *v* is adjacent from *u*
  - The initial and terminal vertices of a loop are the same.

Directed Graphs 3

- The *in-degree of a vertex v*, denoted $deg^-(v)$, is the number of edges which terminate at *v*.
- The *out-degree of v*, denoted $deg^+(v)$, is the number of edges with *v* as their initial vertex.
  - Note that a loop at a vertex contributes 1 to both the in-degree and the out-degree of the vertex.
- **Example:** In the graph *G* we have



$$\deg^-(a)=2, \deg^-(b)=2, \deg^-(c)=3, \deg^-(d)=2,$$
$$\deg^-(e)=3, \deg^-(f)=0.$$

$$\deg^+(a)=4, \deg^+(b)=1, \deg^+(c)=2, \deg^+(d)=2,$$
$$\deg^+(e)=3, \deg^+(f)=0.$$

Directed Graphs 4

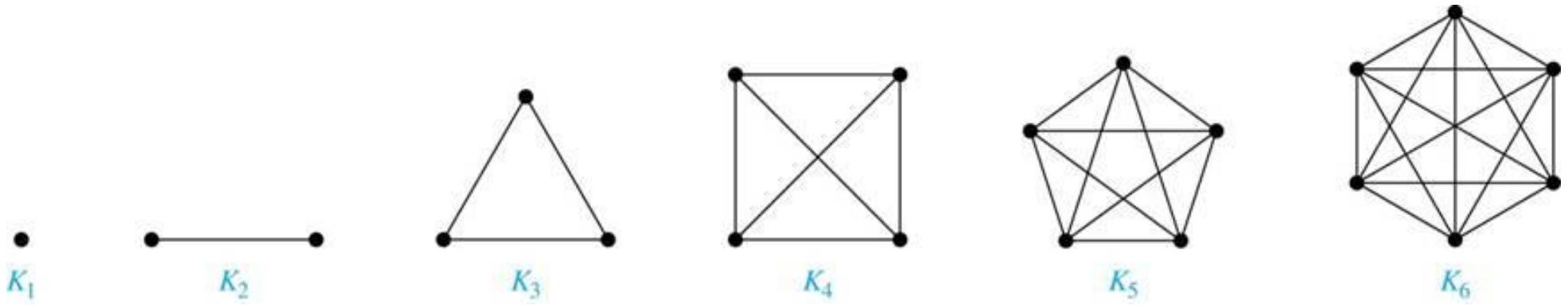
Theorem 3: Let $G = (V, E)$ be a graph with directed edges. Then:

$$|E| = \sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v).$$

Proof: The first sum counts the number of outgoing edges over all vertices and the second sum counts the number of incoming edges over all vertices. It follows that both sums equal the number of edges in the graph.

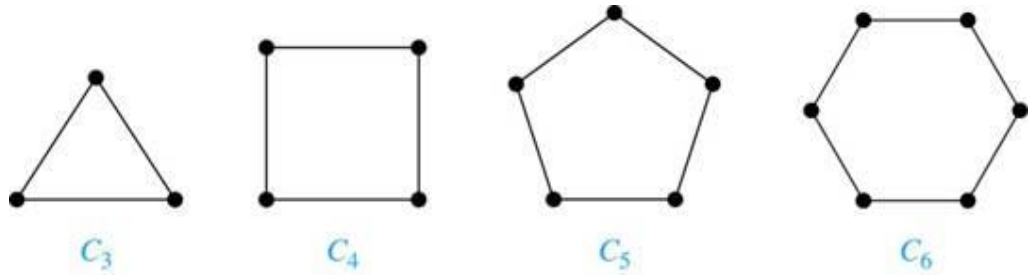
Special Types of Simple Graphs: Complete Graphs

A *complete graph on n vertices*, denoted by K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices.

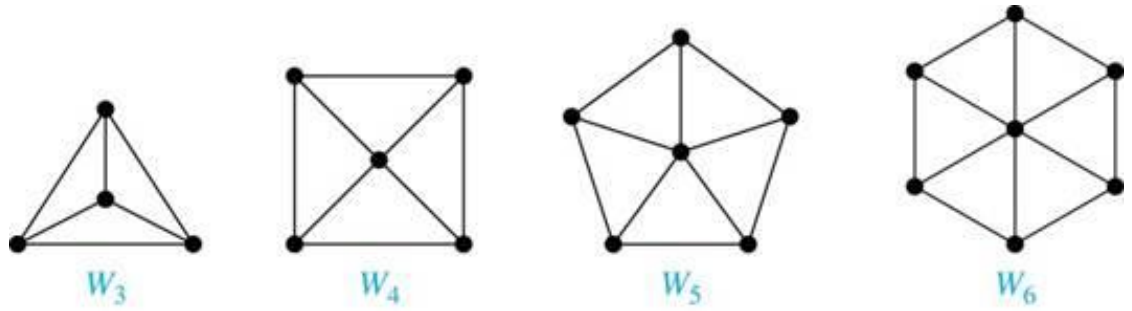


Special Types of Simple Graphs: Cycles and Wheels

A *cycle* C_n for $n \geq 3$ consists of n vertices $v_1, v_2, \cdots, v_n$, and edges $\{v_1, v_2\}$, $\{v_2, v_3\}, \cdots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$.

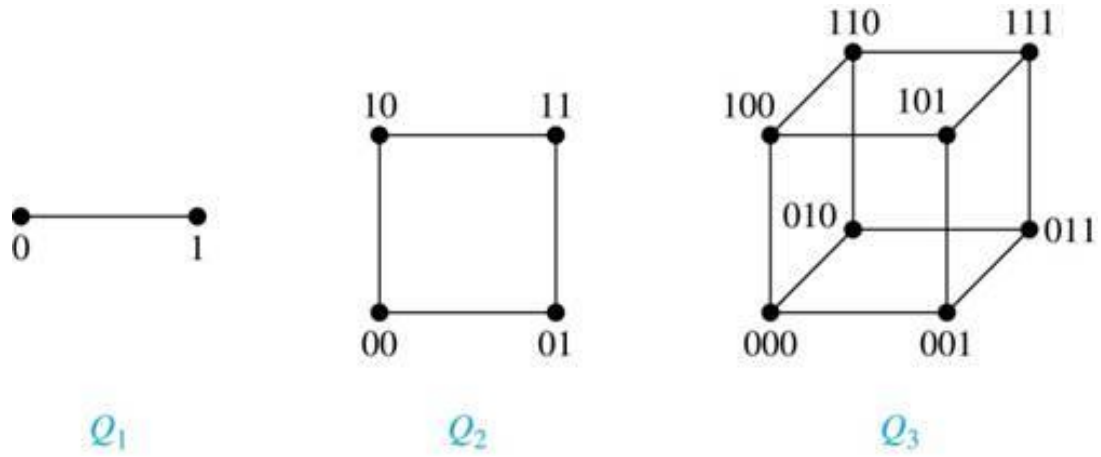


A *wheel* W_n is obtained by adding an additional vertex to a cycle C_n for $n \geq 3$ and connecting this new vertex to each of the n vertices in C_n by new edges.



Special Types of Simple Graphs: *n*-Cubes

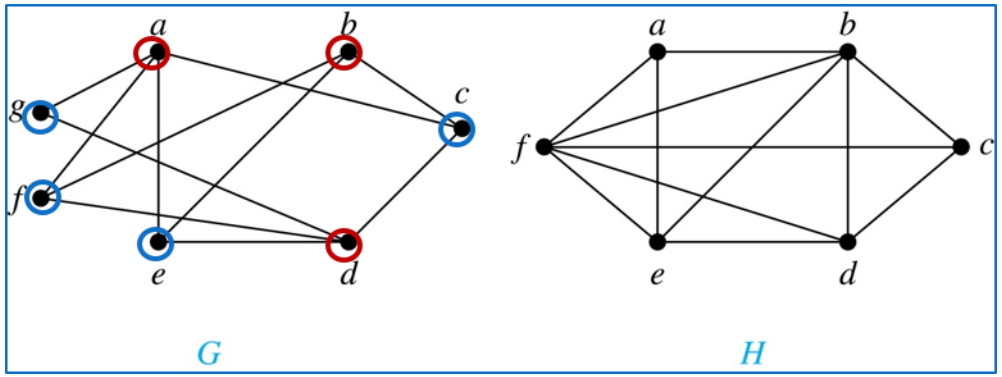
An *n-dimensional hypercube*, or *n-cube,* Q_n, is a graph with 2^n vertices representing all bit strings of length *n*, where there is an edge between two vertices that differ in exactly one bit position.



Bipartite Graphs[1]

- A simple graph G is *bipartite* if V can be partitioned into two disjoint subsets V_1 and V_2 such that every edge connects a vertex in V_1 and a vertex in V_2. In other words, there are no edges which connect two vertices in V_1 or in V_2.
  - a bipartite graph is a graph where it is possible to color the vertices red or blue so that no two adjacent vertices are the same color.

G is bipartite

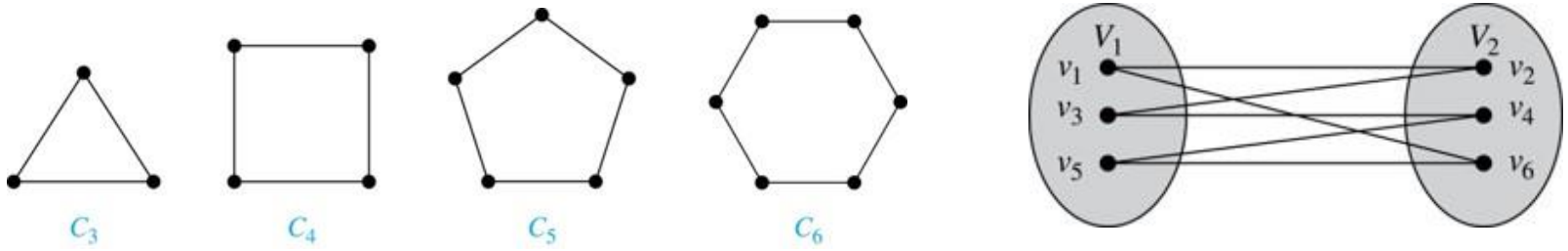


H is not bipartite since if we color a red, then the adjacent vertices f and b must both be blue.

Bipartite Graphs 2

Example: Show that C_6 is bipartite.

Solution: We can partition the vertex set into $V_1 = \{v_1, v_3, v_5\}$ and $V_2 = \{v_2, v_4, v_6\}$ so that every edge of C_6 connects a vertex in V_1 and V_2 .



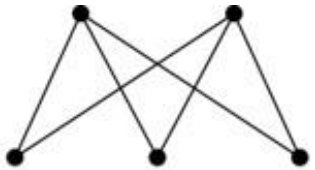
Example: Show that C_3 is not bipartite.

Solution: If we divide the vertex set of C_3 into two nonempty sets, one of the two must contain two vertices. But in C_3 every vertex is connected to every other vertex. Therefore, the two vertices in the same partition are connected. Hence, C_3 is not bipartite.

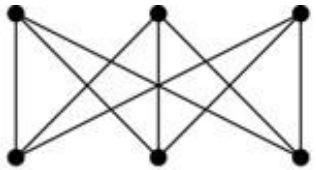
Complete Bipartite Graphs

Definition: A *complete bipartite graph* $K_{m,n}$ is a graph that has its vertex set partitioned into two subsets V_1 of size m and V_2 of size n such that there is an edge from every vertex in V_1 to every vertex in V_2.

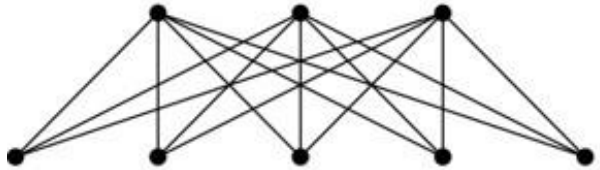
Example: We display four complete bipartite graphs here.



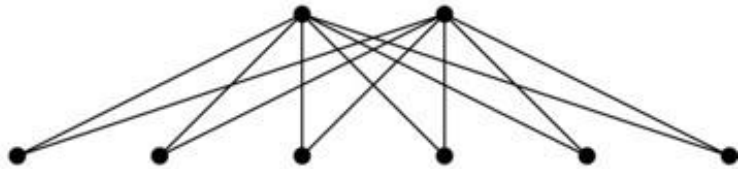
$K_{2,3}$



$K_{3,3}$



$K_{3,5}$

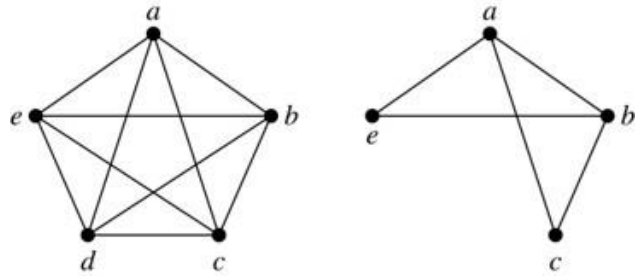


$K_{2,6}$

New Graphs from Old 1

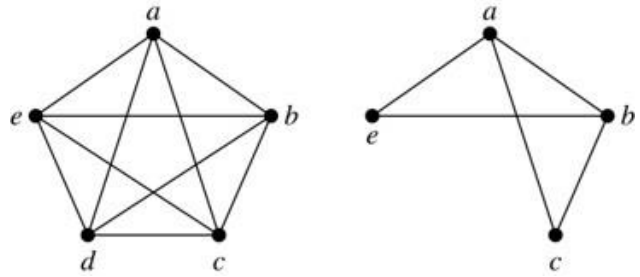
Definition: A *subgraph of a graph* $G = (V,E)$ is a graph (W,F), where $W \subset V$ and $F \subset E$. A subgraph H of G is a proper subgraph of G if $H \neq G$.

Example: Here we show K_5 and one of its subgraphs.



Definition: Let $G = (V, E)$ be a simple graph. The *subgraph induced* by a subset W of the vertex set V is the graph (W,F), where the edge set F contains an edge in E if and only if both endpoints are in W.

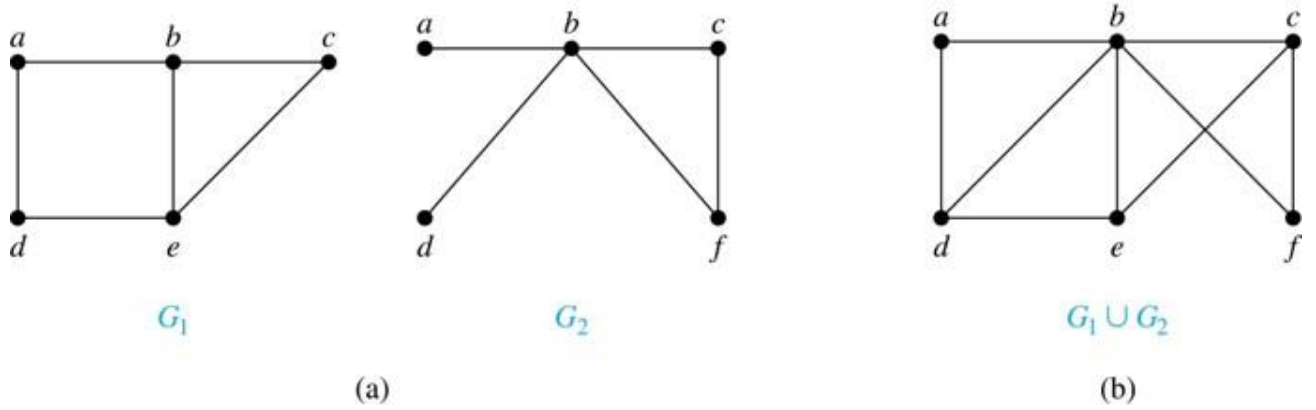
Example: Here we show K_5 and the subgraph induced by $W = \{a,b,c,e\}$.



New Graphs from Old 2

Definition: The *union* of two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$. The union of G_1 and G_2 is denoted by $G_1 \cup G_2$.

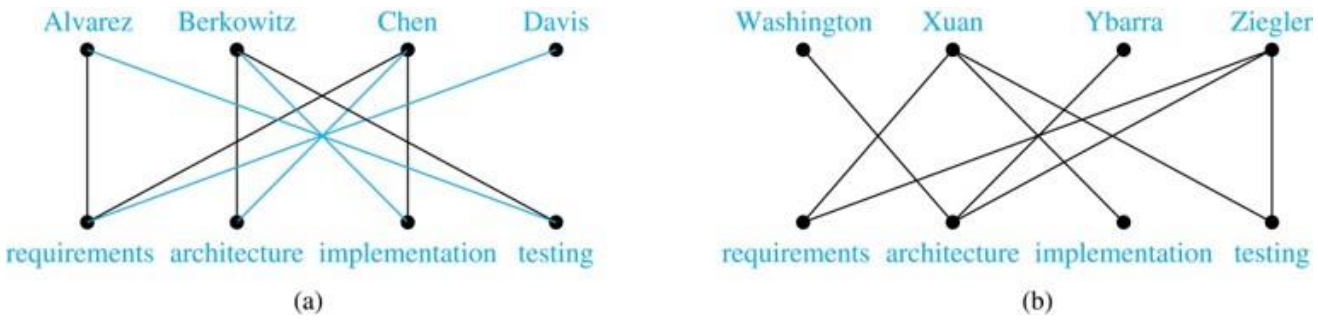
Example:



Jump to long description

Bipartite Graphs and Matchings

- Bipartite graphs are used to model applications that involve matching the elements of one set to elements in another
  - *Job assignments*
  - *Marriage*



- **Hall's Marriage Theorem**. The bipartite graph G with bipartition (V_1, V_2) has a complete matching from V_1 to V_2 if and only if $|N(A)| \geq |A|$ for all subset of A of V_1.

Representing Graphs: Adjacency Lists

Definition: An *adjacency list* can be used to represent a graph with no multiple edges by specifying the vertices that are adjacent to each vertex of the graph.

Example:

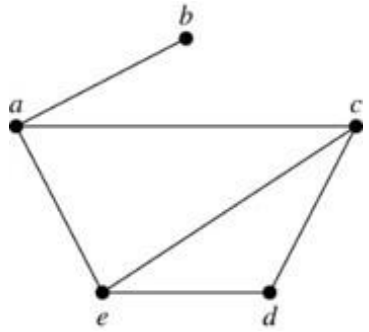


TABLE 1 An Adjacency List for a Simple Graph.

Vertex	*Adjacent Vertices*
a	*b, c, e*
b	*a*
c	*a, d, e*
d	*c, e*
e	*a, c, d*

Example:

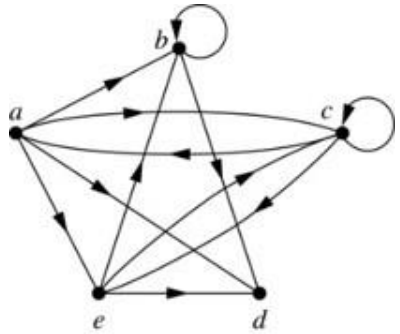


TABLE 2 An Adjacency List for a Directed Graph.

Initial Vertex	*Terminal Vertices*
a	*b, c, d, e*
b	*b, d*
c	*a, c, e*
d	
e	*b, c, d*

Jump to long description

Representation of Graphs: Adjacency Matrices

Definition: Suppose that $G = (V, E)$ is a simple graph where $|V| = n$. Arbitrarily list the vertices of G as $v_1, v_2, \ldots, v_n$. The *adjacency matrix* $\mathbf{A}_G$ of G, with respect to the 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 the graphs adjacency matrix is $\mathbf{A}_G = [a_{ij}]$, then

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

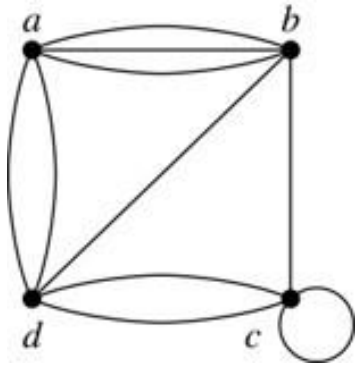
Representation of Graphs: Adjacency Matrices

Adjacency matrices can also be used to represent graphs with loops and multiple edges.

A loop at the vertex v_i is represented by a 1 at the (i, j)th position of the matrix.

When multiple edges connect the same pair of vertices v_i and v_j, (or if multiple loops are present at the same vertex), the (i, j)th entry equals the number of edges connecting the pair of vertices.

Example: We give the adjacency matrix of the pseudograph shown here using the ordering of vertices a, b, c, d.



$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

Jump to long description

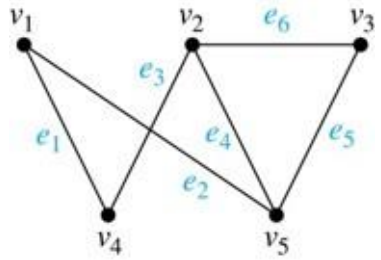
Representation of Graphs: Incidence Matrices 1

Definition: Let $G = (V, E)$ be an undirected graph with vertices where $v_1, v_2, \ldots v_n$ and edges $e_1, e_2, \ldots e_m$. The incidence matrix with respect to the ordering of V and E is the $n \times m$ matrix $\mathbf{M} = [m_{ij}]$, where

$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

Representation of Graphs: Incidence Matrices 2

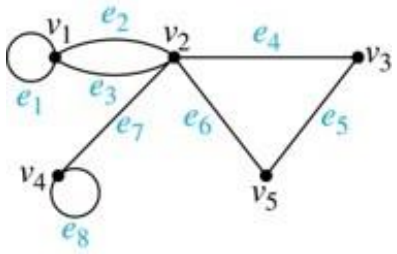
Example: Simple Graph and Incidence Matrix



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

The rows going from top to bottom represent v_1 through v_5 and the columns going from left to right represent e_1 through e_6.

Example: Pseudograph and Incidence Matrix



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

The rows going from top to bottom represent v_1 through v_5 and the columns going from left to right represent e_1 through e_8.

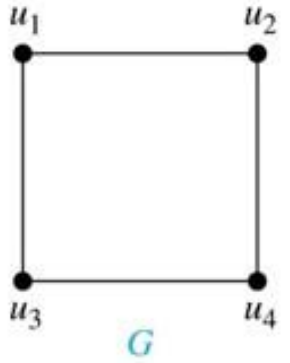
Jump to long description

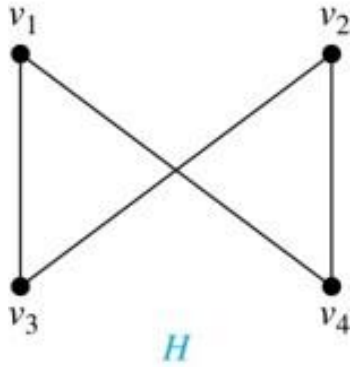
Isomorphism of Graphs[1]

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 2

Example: Show that the graphs $G = (V, E)$ and $H = (W, F)$ are isomorphic.





Solution: 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. Note that 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.

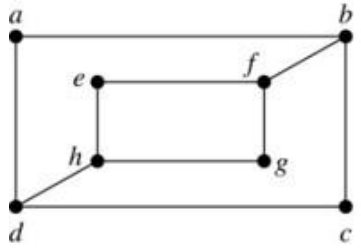
Jump to long description

Isomorphism of Graphs 3

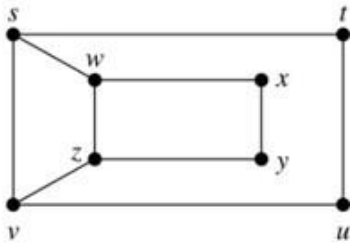
- It is difficult to determine whether two simple graphs are isomorphic using brute force because there are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with n vertices.
  - The best algorithms for determining weather two graphs are isomorphic have exponential 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*.
  - e.g. the number of vertices, number of edges, and degree sequence (list of the degrees of the vertices in nonincreasing order).

Isomorphism of Graphs 4



G

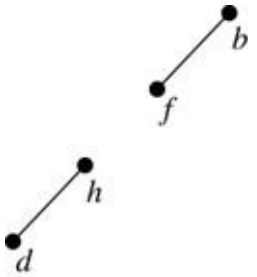


H

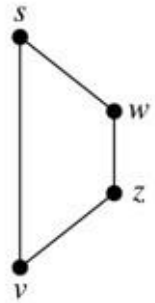
Example: Determine whether these two graphs are isomorphic.

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*.

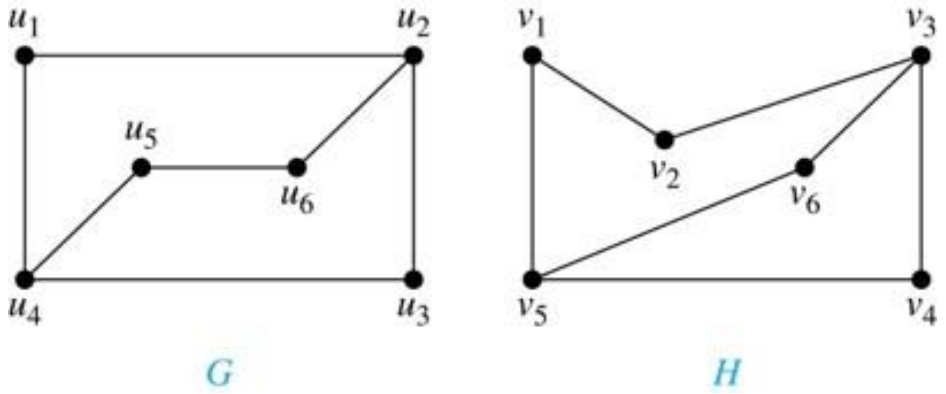


Alternatively, note that the subgraphs of *G* and *H* made up of vertices of degree three and the edges connecting them must be isomorphic. But the subgraphs, as shown at the right, are not isomorphic.



Jump to long description

Isomorphism of Graphs 5



Example: Determine whether these two graphs are 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.

Jump to long description

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.

You can use a public domain program called NAUTY to determine in less than a second whether two graphs with as many as 100 vertices are isomorphic.

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 (see Section 3.3).

Applications of Graph Isomorphism

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

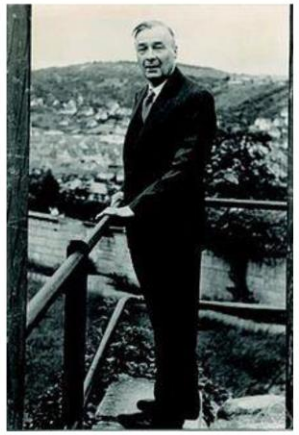
- chemists use molecular graphs to model chemical compounds. Vertices represent atoms and edges represent chemical bonds. When a new compound is synthesized, a database of molecular graphs is checked to determine whether the graph representing the new compound is isomorphic to the graph of a compound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

Hall's Marriage Theorem

- A matching of a bipartite graph is one of its subgraphs where a vertex is not incident with two edges

- **Theorem.** Let $G = (V_1 \cup V_2, E)$ be a bipartite graph. There exists a matching of size $|V_1|$ in G if and only if for every $A \subset V_1$, we have $|A| \leq |N(A)|$.

- We say that G *satisfies Hall's condition* if for every $A \subset V_1$, we have $|A| \leq |N(A)|$.



Philip Hall

http://www.math.caltech.edu/~2016-17/2term/ma006b/

Proof

- **Easy direction.** If there exists a subset $A \subset V_1$ such that $|A| > |N(A)|$, then there is no matching of size $|V_1|$ in G.
- **It remains to prove.** If every subset $A \subset V_1$ satisfies $|A| \leq |N(A)|$, then there is a matching of size $|V_1|$ in G.

Proof by Induction (1/2)

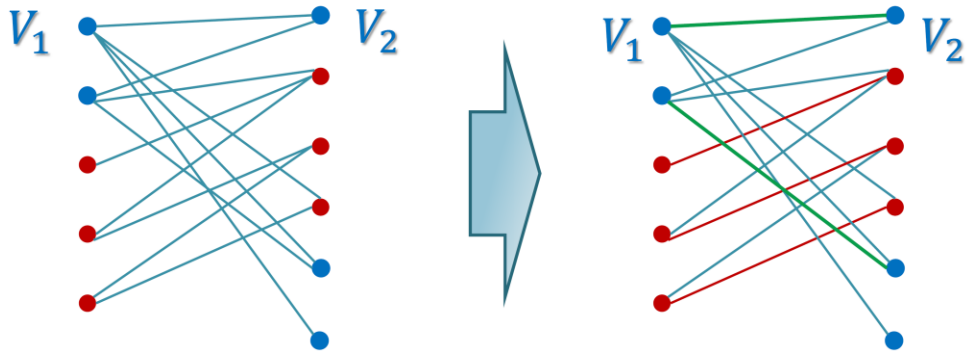
- We prove by induction on $|V_1|$ that
  - If every subset $A \subset V_1$ satisfies $|A| \leq |N(A)|$, then there is a matching of size $|V_1|$ in G.
- **Induction basis.** When $|V_1| = 1$ the claim obviously holds.
- **Induction step.** We consider two cases:
  - Every nonempty $A \subsetneq V_1$ satisfies $|A| < |N(A)|$.
  - There exists $A \subsetneq V_1$ such that $|A| = |N(A)|$.

Case 1

- Assume that every nonempty $A \subsetneq V_1$ satisfies $|A| < |N(A)|$.
  - Consider an edge $(a, b) \in E$. Remove a and b from G to obtain a graph $G' = (V_1' \cup V_2', E')$.
  - Every nonempty $A \subset V_1'$ satisfies $|A| \leq |N(A)|$.
  - By the induction hypothesis, G' contains a matching of size $|V_1'| = |V_1| - 1$.
  - Adding the edge (a, b) to the matching yields a matching of size $|V_1|$ in G.

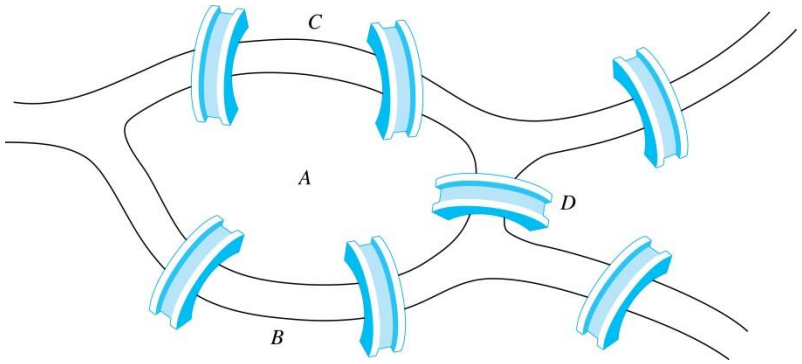
Case 2

- Assume that there exists a non-empty $A \subsetneq V_1$ such that $|A| = |N(A)|$.
  - By the hypothesis, the induced subgraph on $A \cup N(A)$ contains a matching of size $|A|$.
  - Consider the induced subgraph G' on $V_1 \setminus A$ and $V_2 \setminus N(A)$.
  - For any subset $A' \subset V_1 \setminus A$, since in G $|A \cup A'| \leq |N(A) \cup N(A')|$, then in G' $|A'| \leq |N(A')|$. By the hypothesis, G' contains a matching of size $|V_1 \setminus A|$.
- Thus, G contains a matching of size $|V_1|$.

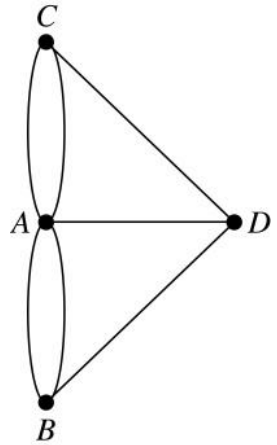


Euler Paths and Circuits: Introduction

- The town of Königsberg (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 it was possible to follow a path that crosses each bridge exactly once and returns to the starting point.
- Leonard Euler proved that no such path exists.



The 7 Bridges of Königsberg



Multigraph Model of the Bridges of Königsberg

Leonard Euler (1707-1783)

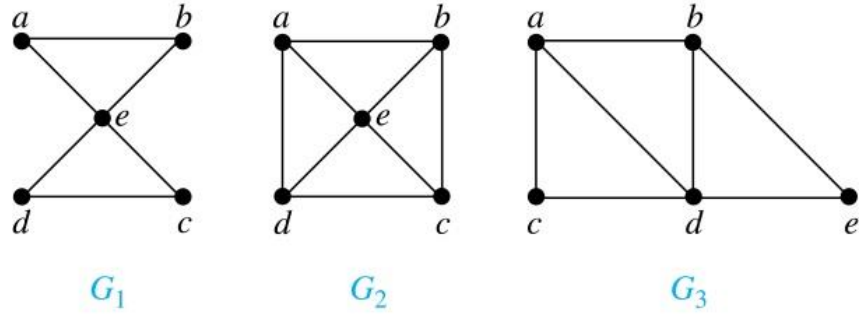
Path and Circuit

- A path in a graph is a sequence of edges
  - The length of a path is the number of edges in the sequence
- A path is a *circuit* if it begins and ends at the same vertex
- A path (or circuit) is *simple* if it does not contain the same edge more than once

Euler Paths and Circuits

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's Circuit Theorem

Theorem

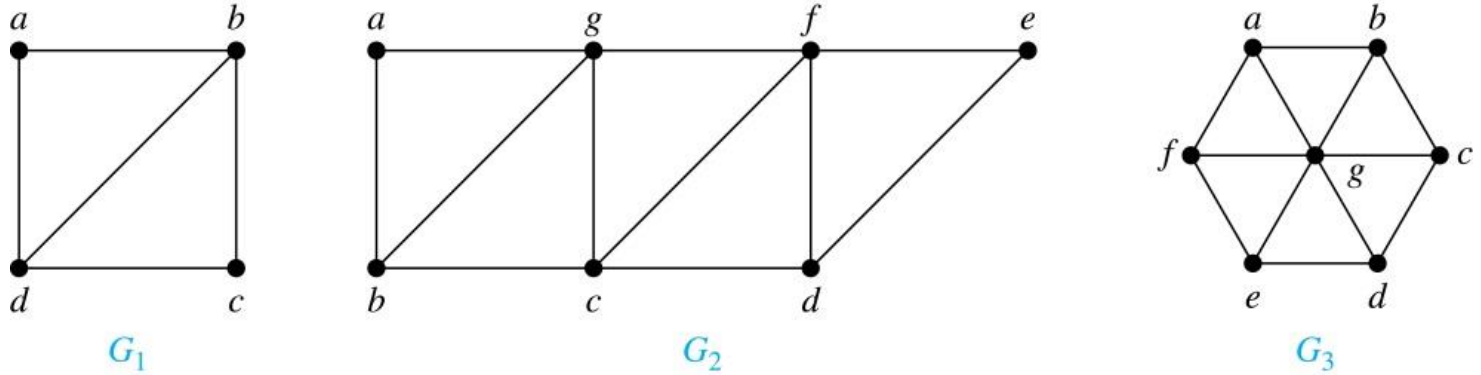
A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has an even degree.

Collorary

It has an Euler path if and only if it has exactly two vertices of odd degree.

Euler Circuits and Paths

Example:



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.

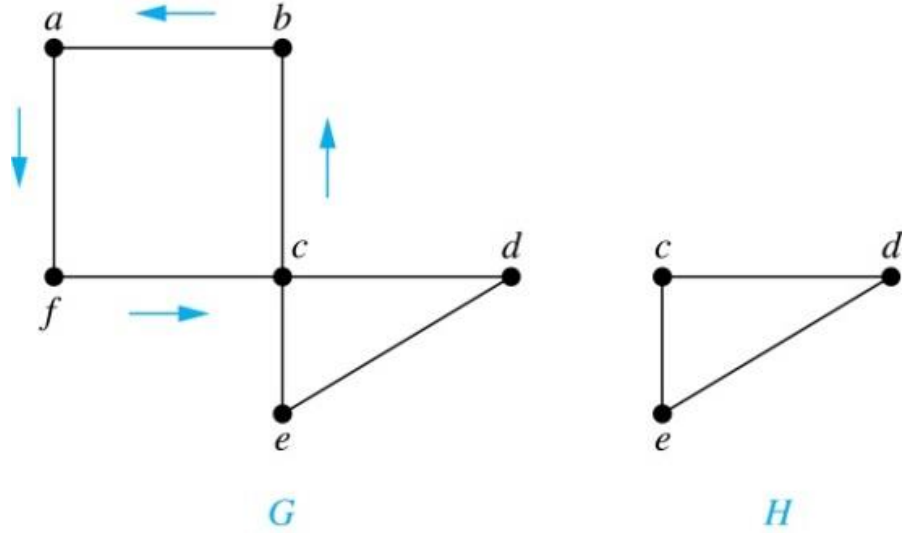
Necessary Conditions for Euler Circuits and Paths

- The degree of every vertex must also be even to have a Euler circuit.
  - 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 $\deg(a)$.
  - The circuit terminates where it started, contributing one to $\deg(a)$. Therefore $\deg(a)$ must be even.
  - Each time the circuit passes through a vertex it contributes two to the vertex's degree.

- 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.

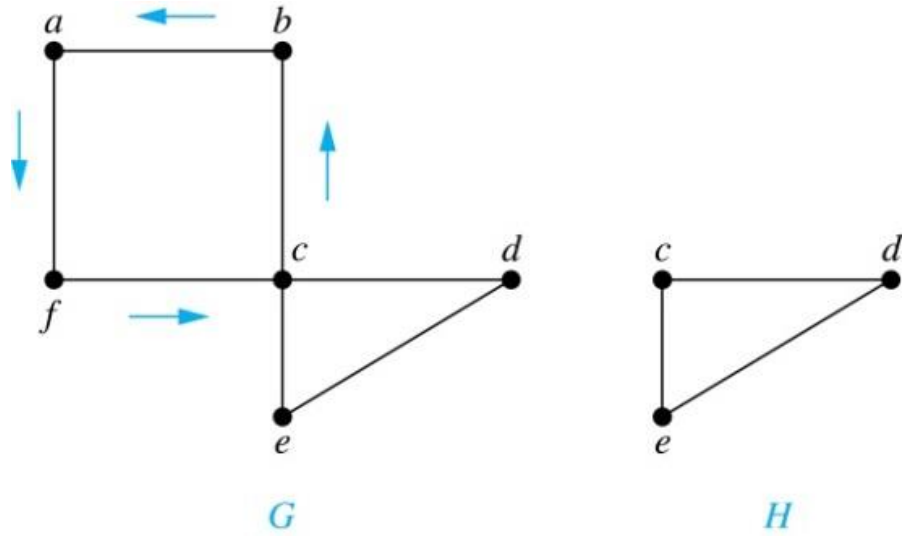
Sufficient Conditions for Euler Circuits and Paths (1/2)

- For a connected multigraph G having ≥ 2 vertices with even degree.
- Let $x_0 = a$ be a beginning vertex, and choose an edge $\{x_0, x_1\}$ and proceed to build a simple path $\{x_0, x_1\}, \{x_1, x_2\}, \ldots, \{x_{n-1}, x_n\}, \{x_n, x_0\}$ by adding edges one by one until another edge can not be added.
  - Since each vertex has an even degree, every time we enter a vertex other than a, we can leave it. Therefore, the path can only end at a.
- Get H from G by deleting the edges already used. H must have at least one vertex in common with the circuit that has been deleted.



Sufficient Conditions for Euler Circuits and Paths (2/2)

- Every vertex in H must have even degree. Beginning with the shared vertex construct a path ending in the same vertex. Then splice this new circuit into the original circuit.
- 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.



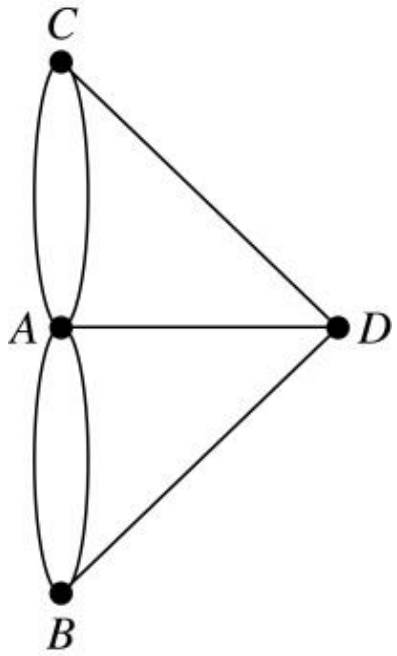
Algorithm for Constructing an Euler Circuits

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 s*ubcircuit* inserted at the appropriate vertex.
return *circuit*{*circuit* is an Euler circuit}

Euler's Circuit Theorem

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.



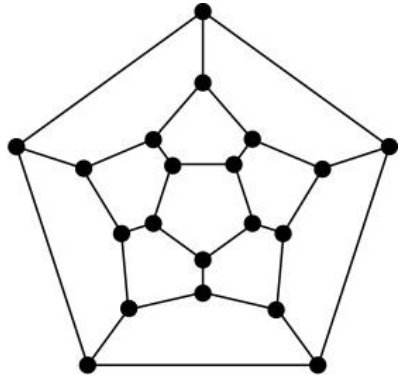
Applications of Euler Paths and Circuits

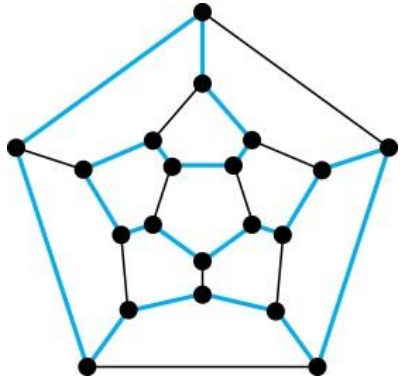
- Euler paths and circuits can be used to solve many practical problems such as finding a path or circuit that traverses each
  - street in a neighborhood,
  - road in a transportation network,
  - connection in a utility grid,
  - link in a communications network.
- Other applications are found in the
  - 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*.

William Hamilton invented the *Icosian puzzle* in 1857. It consisted of a wooden dodecahedron (with 12 regular pentagons as faces), with a peg at each vertex, labeled with the names of different cities. String was used to used to plot a circuit visiting 20 cities exactly once

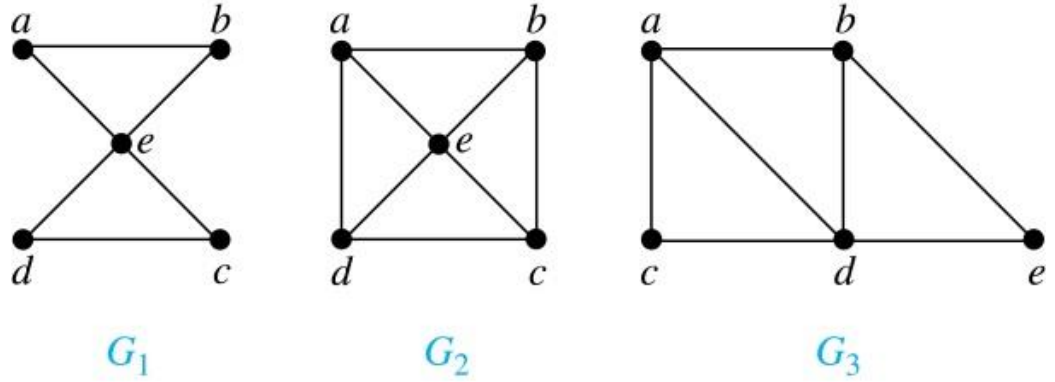




William Rowan Hamilton (1805- 1865)

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?

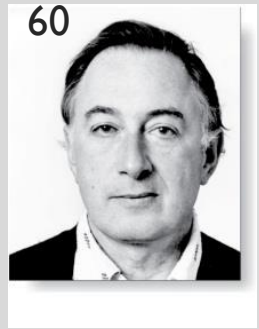
Necessary Conditions for Hamilton Circuits

- Unlike for an Euler circuit, no simple necessary and sufficient conditions are known for the existence of a Hamiton circuit.

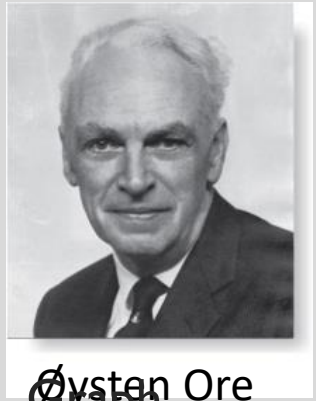
- However, there are some useful necessary conditions. We describe two of these now.

  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.



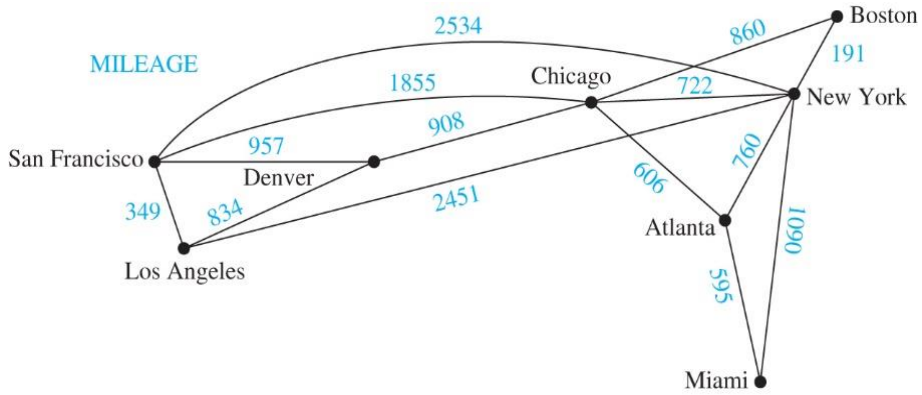
Gabriel Andrew Dirac (1925-1984)

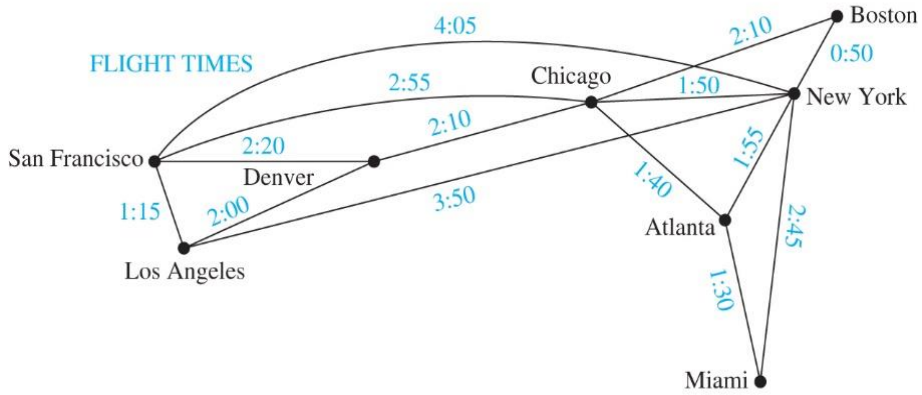


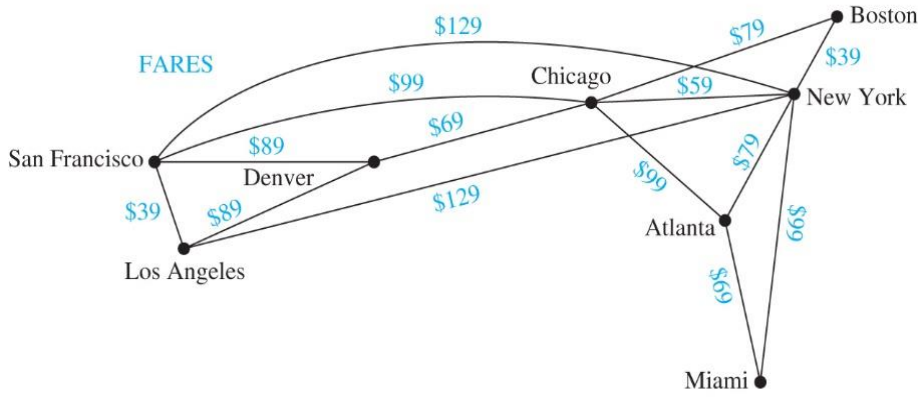
Øystein Ore (1899-1968)

Shortest Path Problem

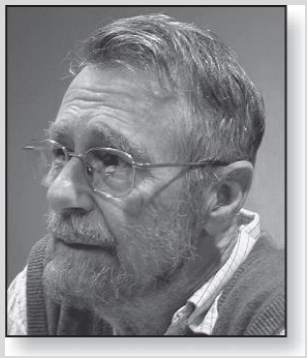
- **Weight graphs** that have a number assigned to each edge
- **The length of a path** in a weighted graph is the sum of weights of the edges of the path.
- What is a shortest path between two verticies?
- What is a circuit of shortest total length that visits every vertex once?



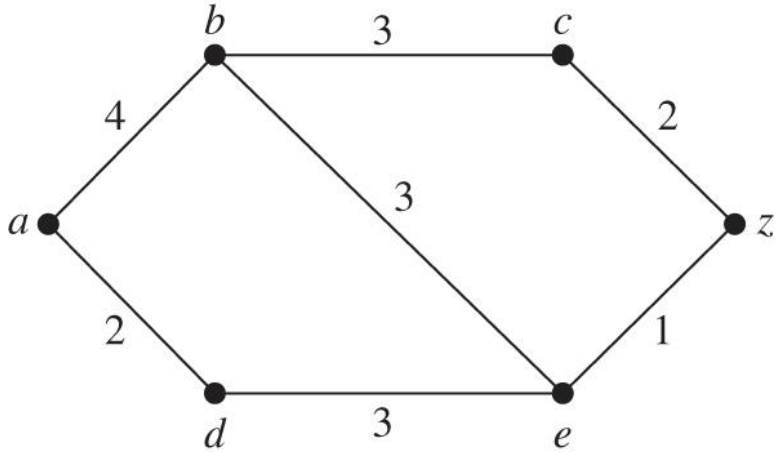




Dijkstra's Algorithm - Motivation



- A greedy algorithm discovered by Edgar W. Dijkstra working for graphs with positive weights
- Idea
  - find and check the closest vertex among the neighbors of the starting vertex
  - find and check the next closest vertex among the neighbors of the checked one until the destination vertex is checked
- Example:
  - find the shortest path from *a* to *z*



Dijkstra's Algorithm

Input

G=(V, E, w), a weighted undirected graph

$a \in V$, the starting vertext

$z \in V$, the destination vertex

Procedure

foreach $v \in V$:

    $L(v) \leftarrow \infty$

$L(a) \leftarrow 0$

$S \leftarrow \emptyset$

while $z \notin S$:

    $u \leftarrow$ a vertex $\in V \setminus S$ with a minimal $L(u)$

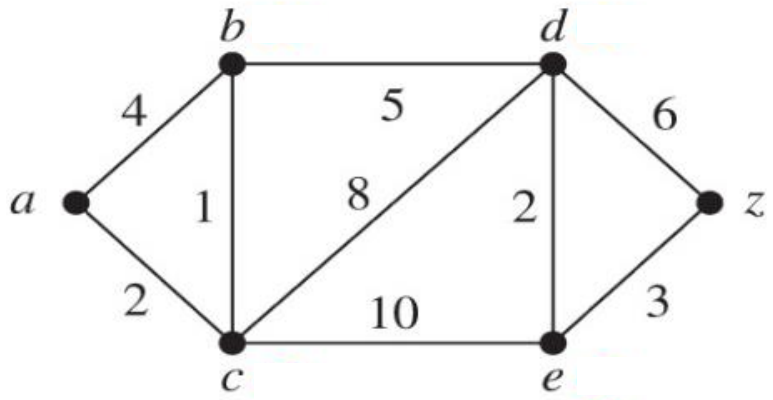
    $S \leftarrow S \cup \{u\}$

    foreach $v \in V \setminus S$

        if $L(u) + w(u, v) < L(v)$:

            $L(v) \leftarrow L(u) + w(u, v)$

return $L(z)$



Proof

Input

G=(V, E, w), a weighted undirected graph

$a \in V$, the starting vertext

$z \in V$, the destination vertex

Procedure

foreach $v \in V$:

$L(v) \leftarrow \infty$

$L(a) \leftarrow 0$

$S \leftarrow \emptyset$

while $z \notin S$:

$u \leftarrow$ a vertex $\in V \setminus S$ with a minimal $L(u)$

$S \leftarrow S \cup \{u\}$

foreach $v \in V \setminus S$

if $L(u) + w(u, v) < L(v)$:

$L(v) \leftarrow L(u) + w(u, v)$

return $L(z)$

Inductive hypotheses:

- $L(v)$ for $v \in S$ is the length of a shortest path from a to v
- $L(u)$ for $u \in V \setminus S$ is the length of a shortest path from a to u, that contains only the verticies in S

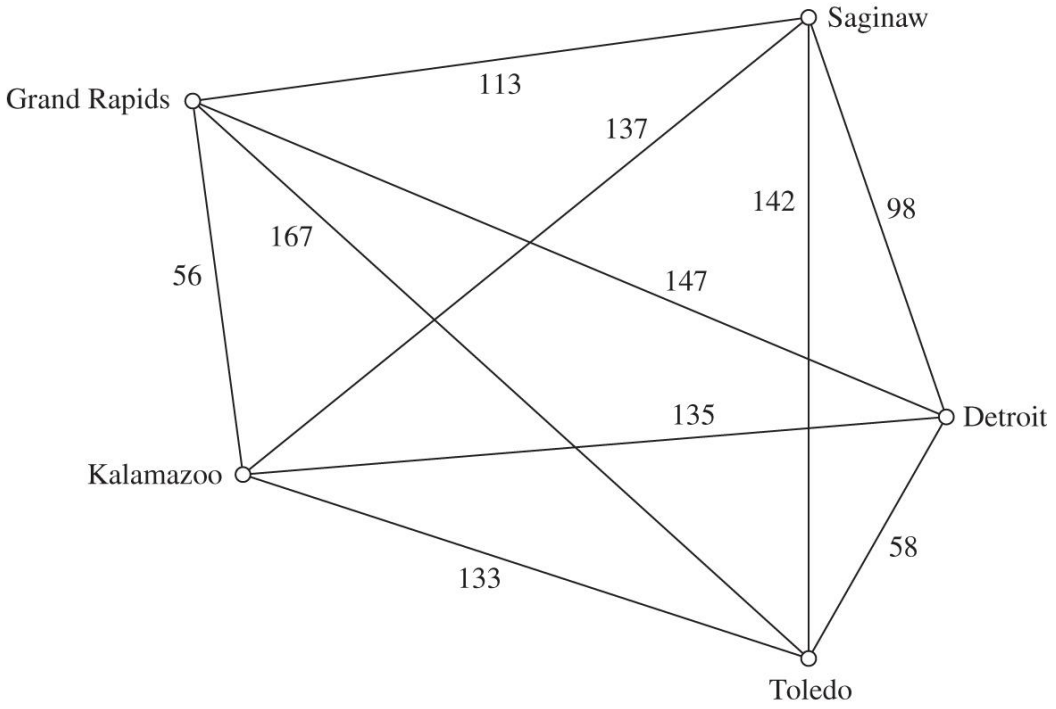
Induction basis: obvious

Induction step:

- Let u is the vertex added to S at (k+1)-th iteration. Then, $L(u)$ must be the length of a shortest path from a to u.
  - Assume that $L(u)$ is not the length of shortest path from a to u.
  - Then, there will be another path from a to u, which contains $v' \notin S$
  - Such v' must be already in S since $L(v') < L(u)$, thus, it's a contradiction.

Traveling Salesperson Problem

- Find a circuit of the minimum total weight that visits each vertex exactly once
- No algorithm exists (so far) to find the exact answer without checking all possible Hamiltonian circuits in the graph.
  - Yet, many approximation algorithms have been developed



In this example, the answer is Detroit-Toledo-Grand Rapids-Saginaw-Kalamazoo-Detroit which takes 610