Discrete Mathematics & Mathematical Reasoning
Chapter 10: Graphs

Kousha Etessami

U. of Edinburgh, UK

Overview

- Graphs and Graph Models
- Graph Terminology and Special Types of Graphs
- Representations of Graphs, and Graph Isomorphism
- Connectivity
- Euler and Hamiltonian Paths
- Brief look at other topics like graph coloring

What is a Graph?

Informally, a **graph** consists of a non-empty set of **vertices** (or **nodes**), and a set E of **edges** that connect (pairs of) nodes.

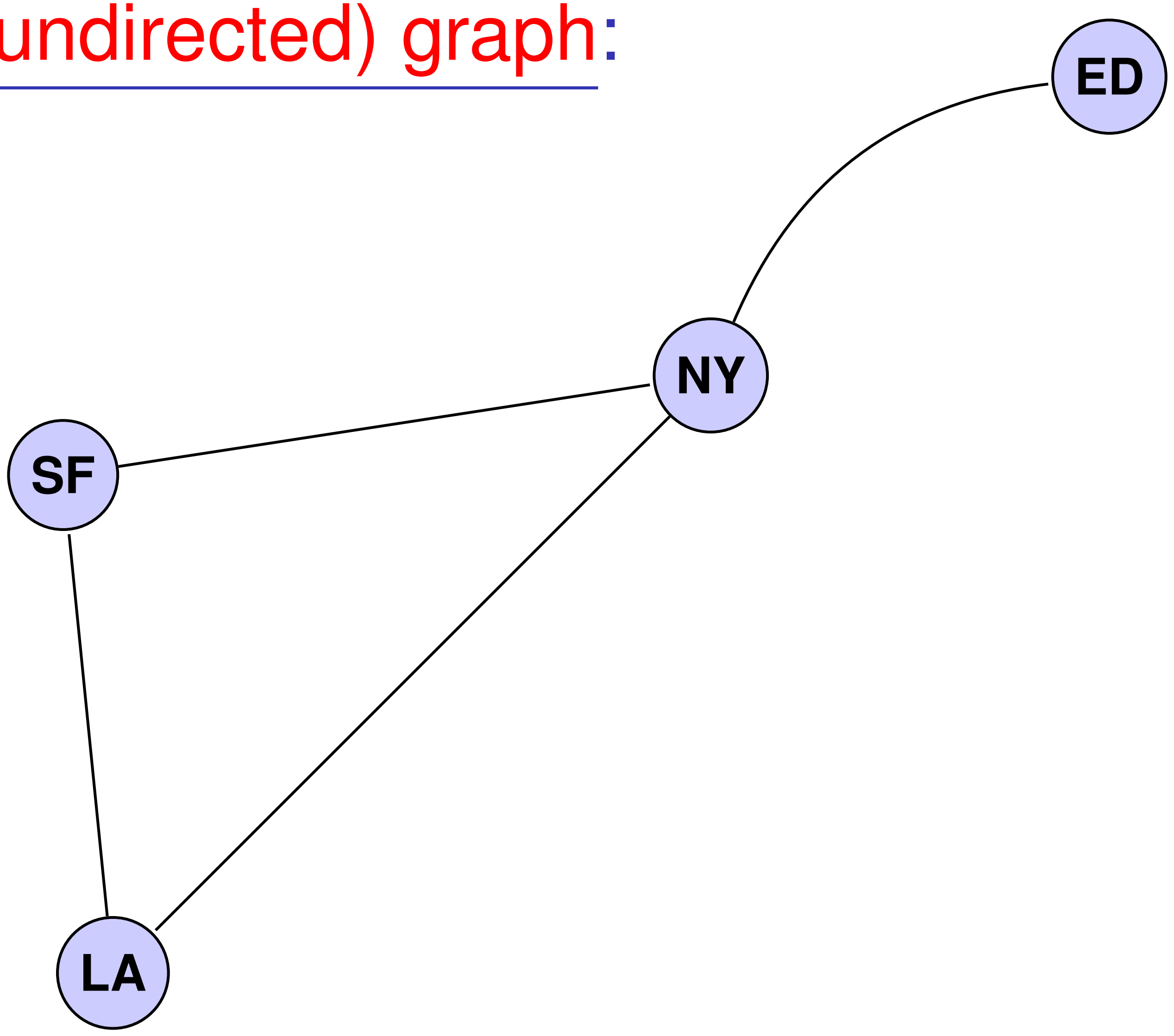
But different types of graphs (*undirected, directed, simple, multigraph, . . .*) have different formal definitions, depending on what kinds of edges are allowed.

This creates a lot of (often inconsistent) terminology.

Before formalizing, let's see some examples....

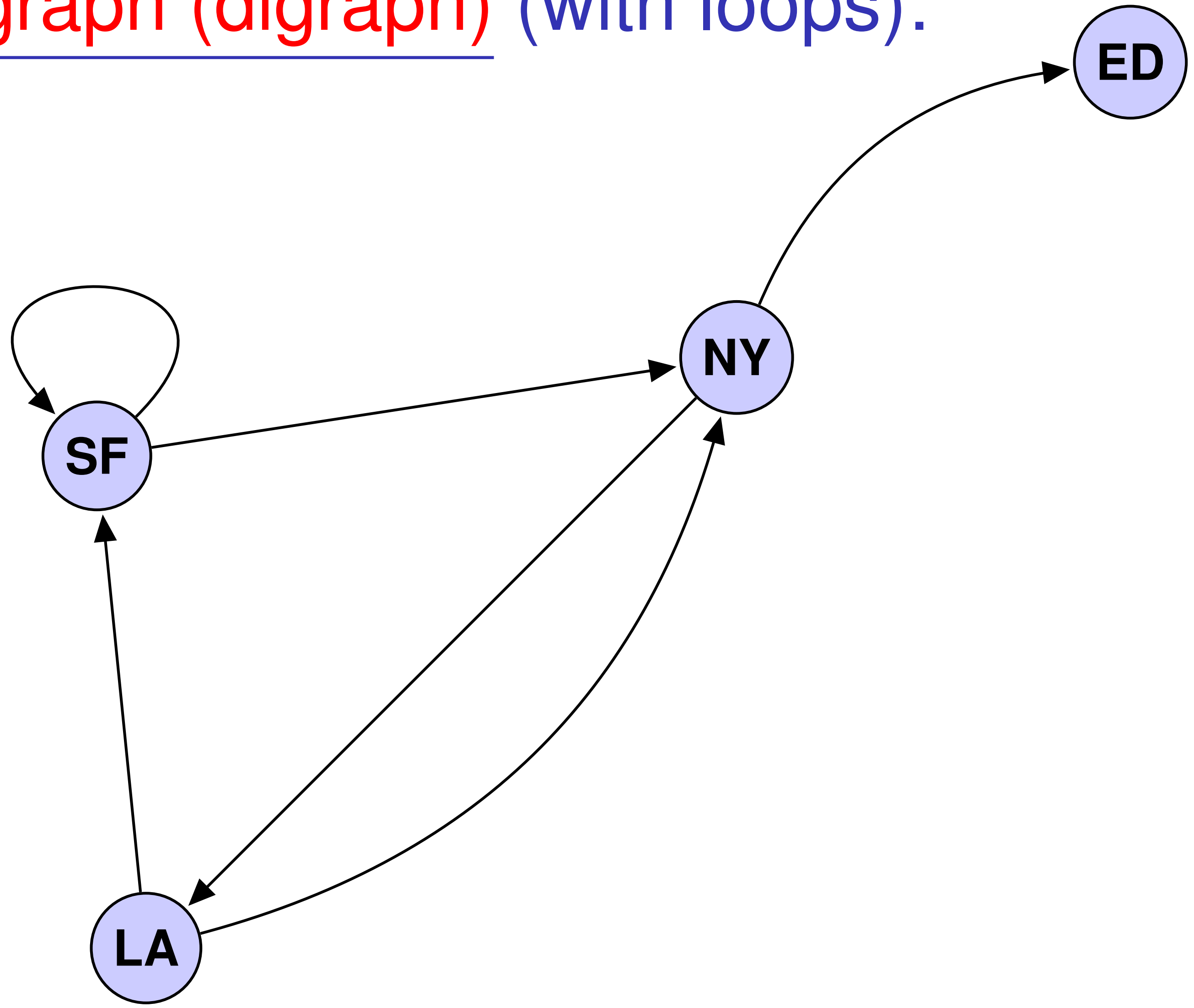
During this course, we focus almost exclusively on standard (undirected) graphs and directed graphs, which are our first two examples.

A (simple undirected) graph:



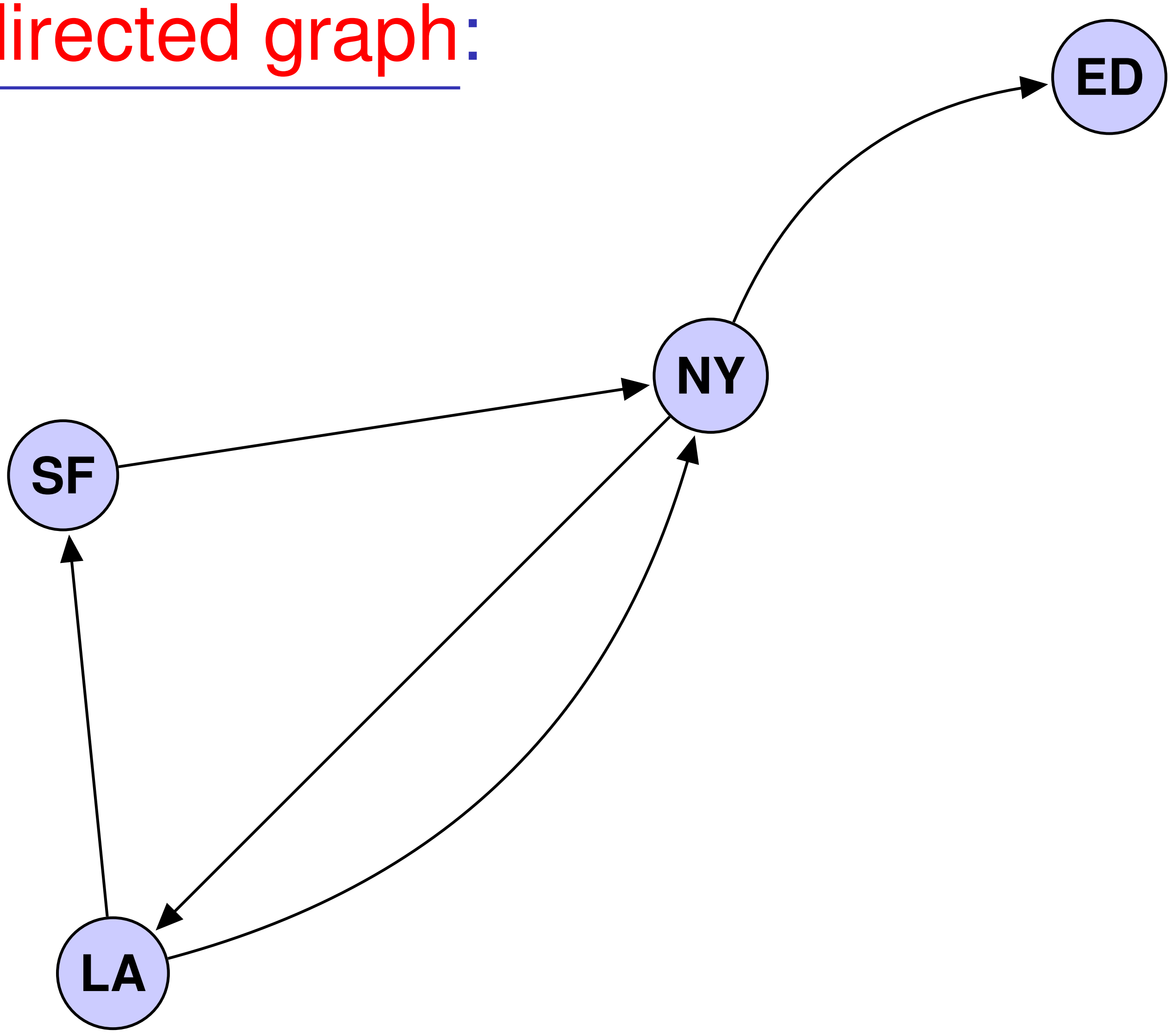
Only undirected edges; at most one edge between any pair of distinct nodes; and no loops (edges between a node and itself).

A directed graph (digraph) (with loops):



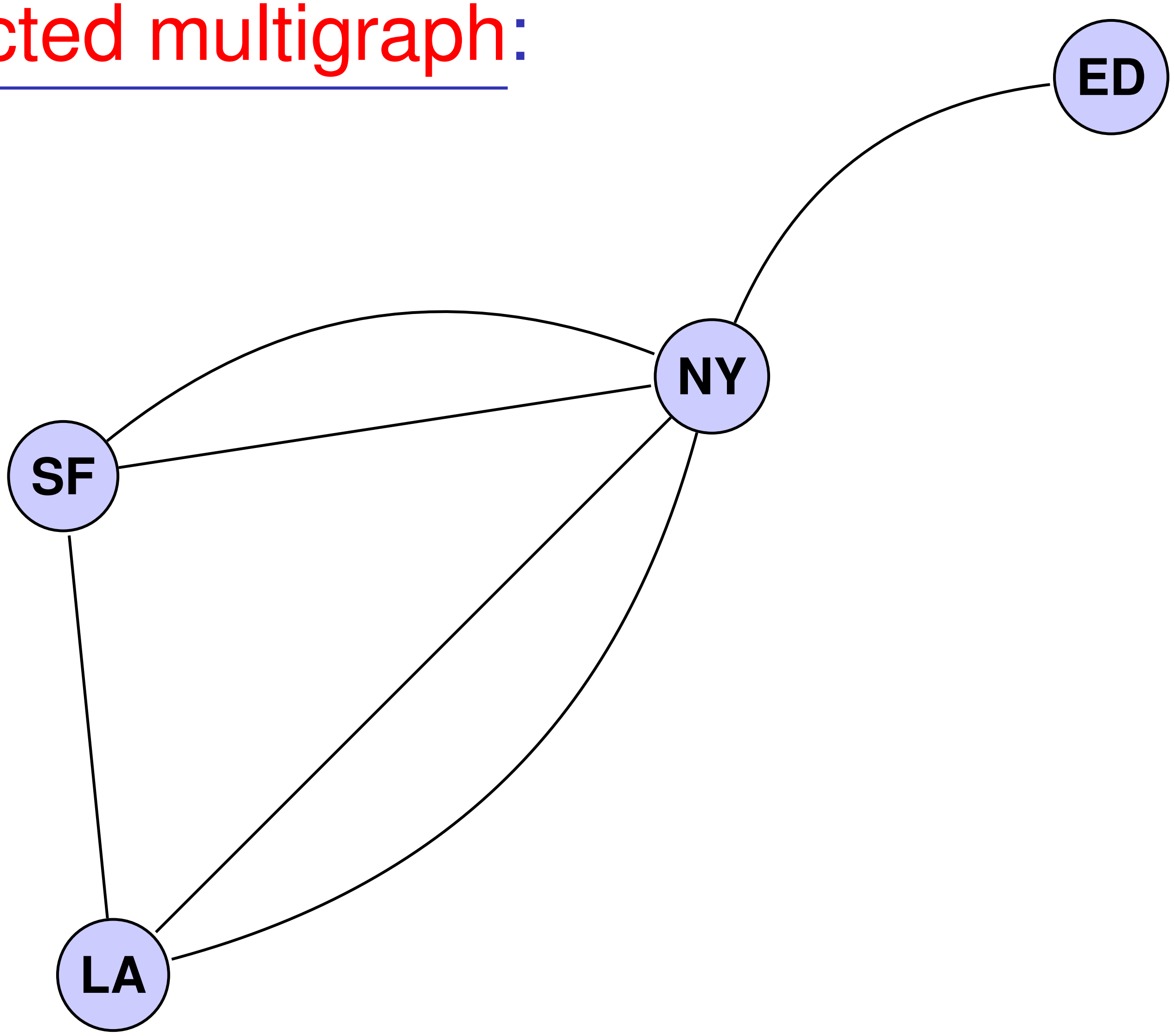
Only directed edges; at most one directed edge from any node to any node; and loops are allowed.

A simple directed graph:



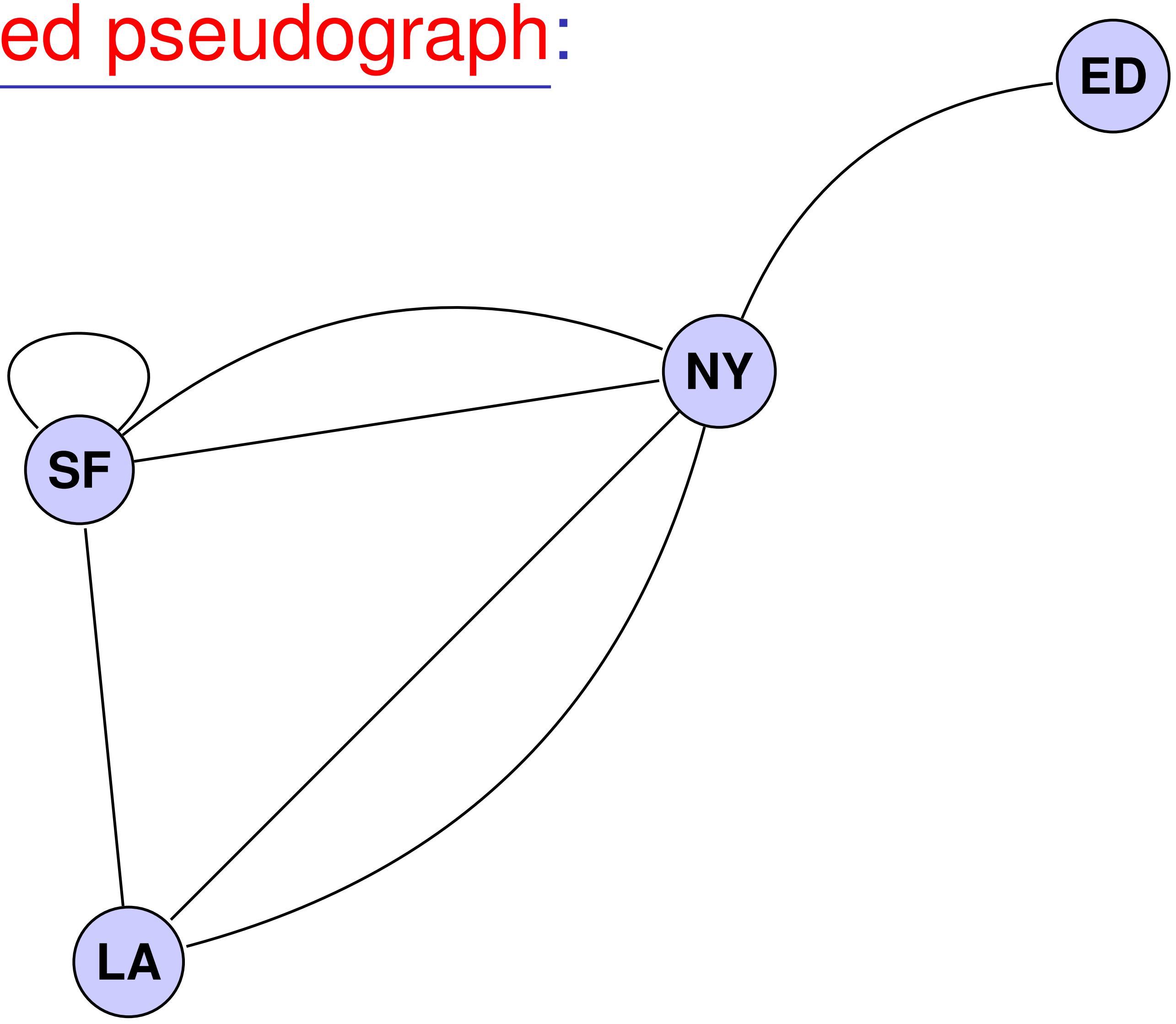
Only directed edges; at most one directed edge from any node to any other node; and no loops allowed.

An undirected multigraph:



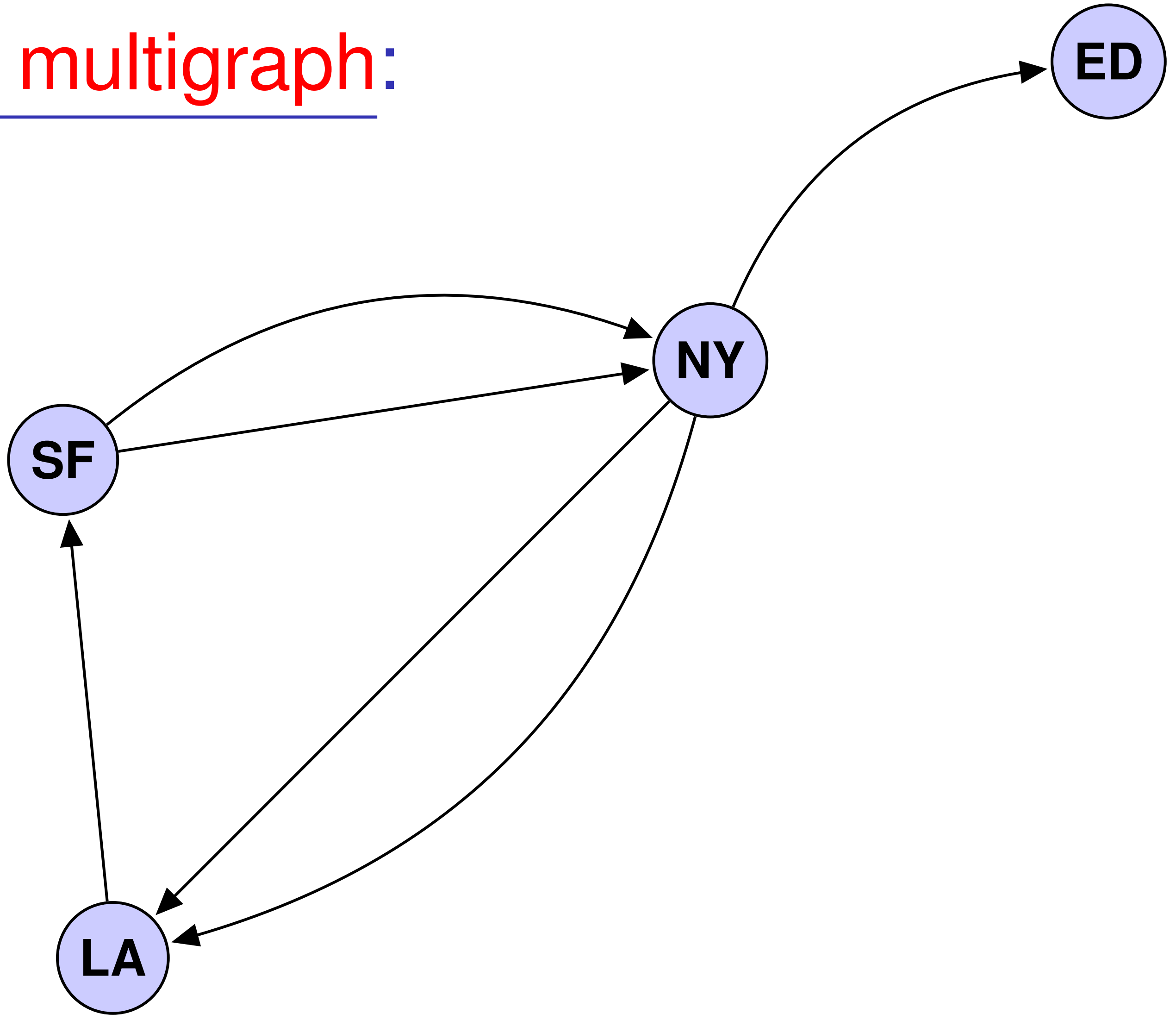
Only undirected edges; may contain multiple edges between a pair of nodes; but no loops.

An undirected pseudograph:



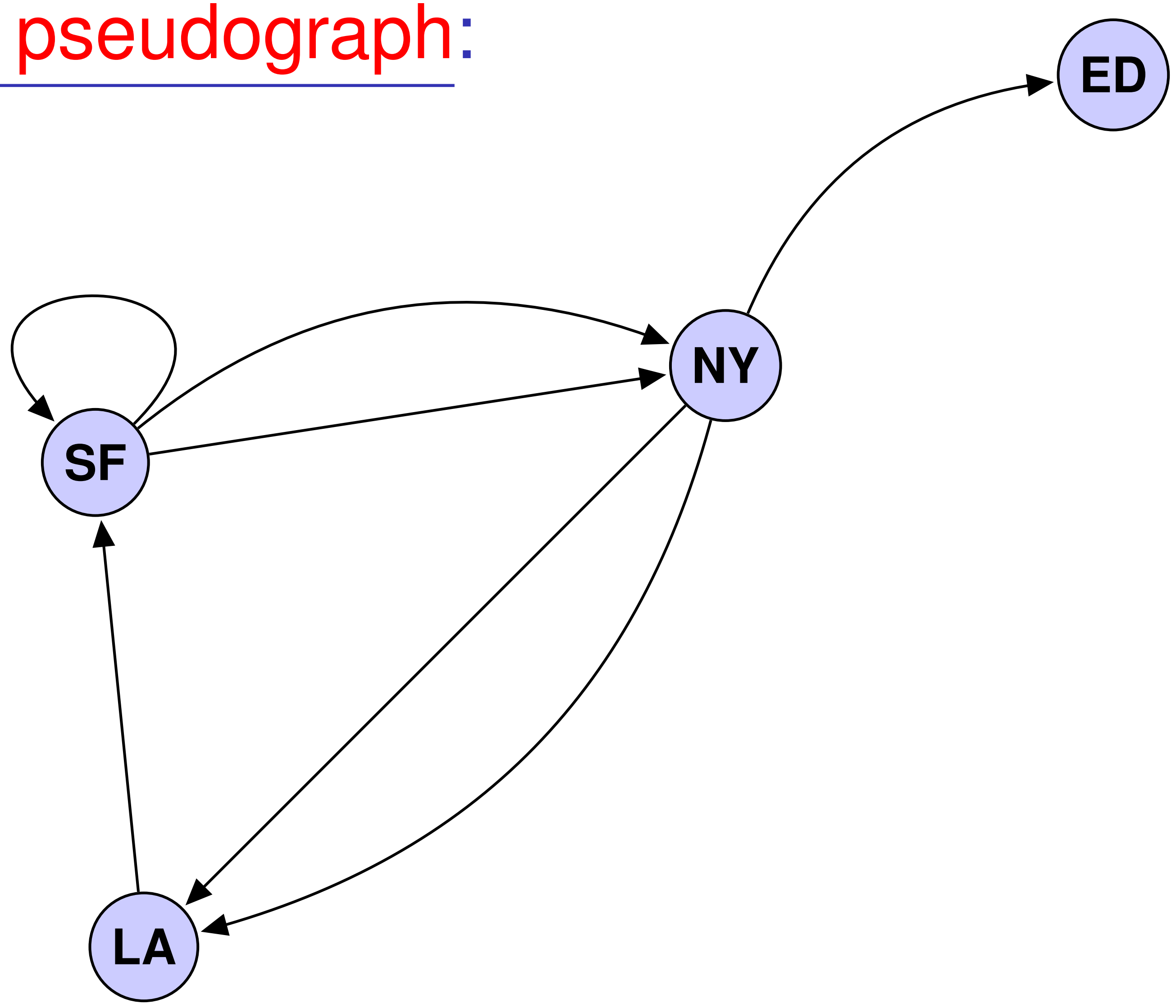
Only undirected edges; may contain multiple edges between a pair of nodes; and may contain loops (even multiple loops on the same node).

A directed multigraph:



Only directed edges; may contain multiple edges from one node to another; but no loops allowed.
Warning: this differs slightly from the Rosen book terminology. The book's notion of "directed multigraph" would allow loops.

An directed pseudograph:



Only directed edges; may contain multiple edges from one node to another; and may contain loops (even multiple loops on one node).

Graph Terminology Zoo (ridiculous)

	Type	Edges	Multi-Edges?	Loops?
1.	(simple undirected) graph	Undirected	No	No
2.	(undirected) multigraph	Undirected	Yes	No
3.	(undirected) pseudograph	Undirected	Yes	Yes
4.	directed graph	Directed	No	Yes
5.	simple directed graph	Directed	No	No
6.	directed multigraph	Directed	Yes	No[1]
7.	directed pseudograph	Directed	Yes	Yes
8.	mixed graph	Both	Yes	Yes

We will focus on the two most standard types:

(1.) graphs (simple undirected), and
(4.) directed graphs (also known as digraphs).

[1] differs from book.

Formal Defintion of Directed Graphs

A **directed graph** (**digraph**), $G = (V, E)$, consists of a non-empty set, V, of **vertices** (or **nodes**), and a set $E \subseteq V \times V$ of **directed edges** (or **arcs**). Each directed edge $(u, v) \in E$ has a **start** (**tail**) vertex u, and a **end** (**head**) vertex v.
Note: a directed graph $G = (V, E)$ is simply a set V together with a binary relation E on V.

Definition of (Undirected) Graphs

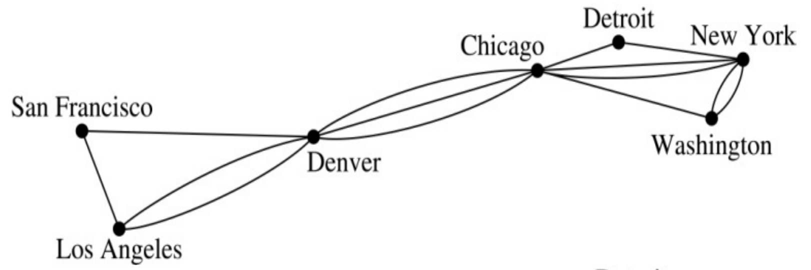
For a set V, let $[V]^k$ denote the set of k-element subsets of V.
(Equivalently, $[V]^k$ is the set of all k-combinations of V.)

A (simple,undirected) **graph**, $G = (V, E)$, consists of a non-empty set V of **vertices** (or **nodes**), and a set $E \subseteq [V]^2$ of (undirected) **edges**. Every edge $\{u, v\} \in E$ has two distinct vertices $u \neq v$ as **endpoints**, and such vertices u and v are then said to be **adjacent** in the graph G.

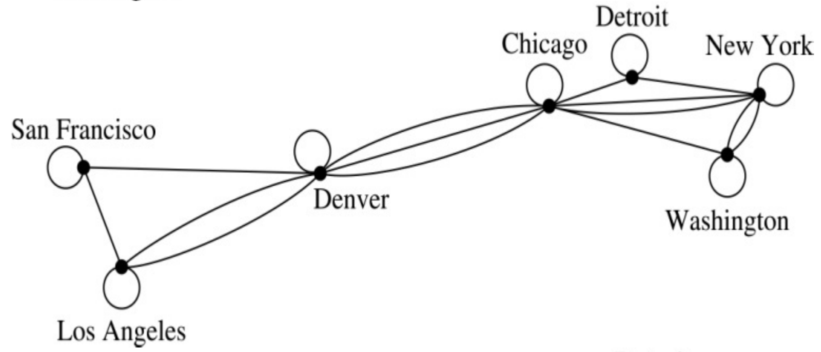
Note: the above definitions allow for infinite graphs, where $|V| = \infty$.
In this course we will focus on finite graphs.

Graph Models: Computer Networks

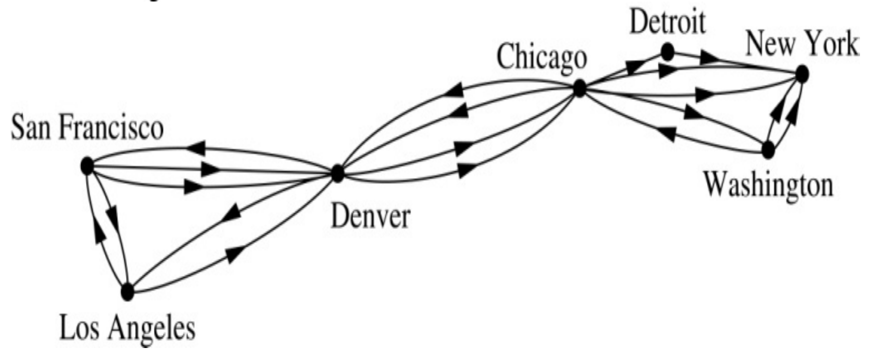
- network where we care about the number of links: we use a multigraph.



- diagnostic self-links at data centers: we use a pseudograph.



- network with multiple one-way links, we use a directed (multi)graph.



Applications of Graphs

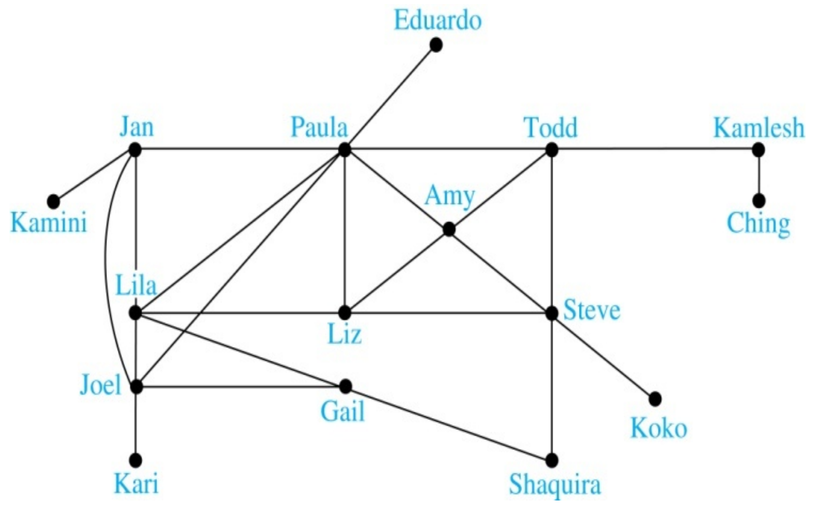
- "EVERYTHING IS A GRAPH"
  (labeled, directed, etc., ...)
- graph theory can be used in modelling of:
  - Social networks
  - Communications networks
  - Information networks
  - Software design
  - Transportation networks
  - Biological networks
  - ......

Graph Models: Social Networks

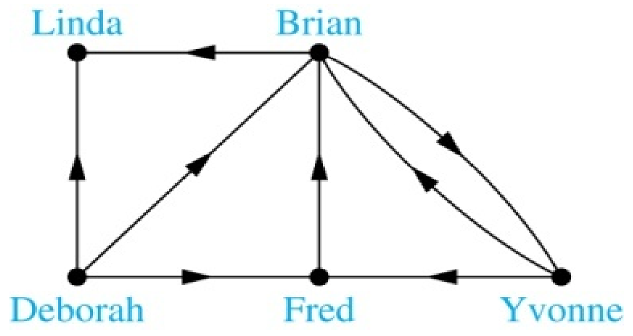
- model social structures: relationships between people or groups.
- vertices represent individuals or organizations, edges represent relationships between them.
- Useful graph models of social networks include:
  - *friendship graphs* - undirected graphs where two people are connected if they are friends (e.g., on Facebook)
  - *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

Graph Models: Social Networks

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



Example: An influence graph



Information Networks

- In a *web graph*, web pages are represented by vertices and links are represented by directed edges.
- In a *citation network*:
  - Research papers are represented by vertices.
  - When paper A cites paper B, there is an edge from the vertex representing paper A to the vertex representing paper B.

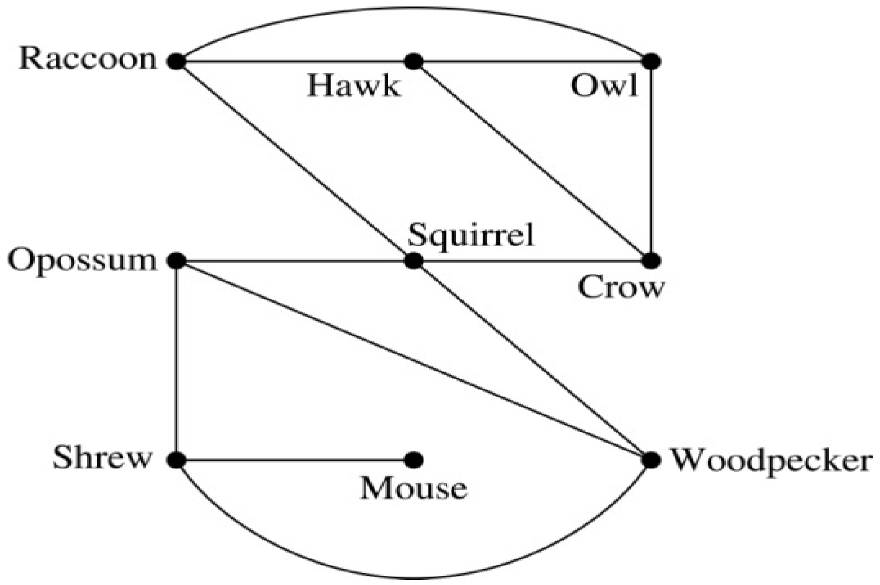
Transportation Graphs

- Graph models are extensively used to study 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 modeled using graphs

Biological Applications

- Graph models are used extensively in many areas of the biological science.
- *Niche overlap graphs* model competition between species in an ecosystem:

Example: niche overlap graph for a forest ecosystem.



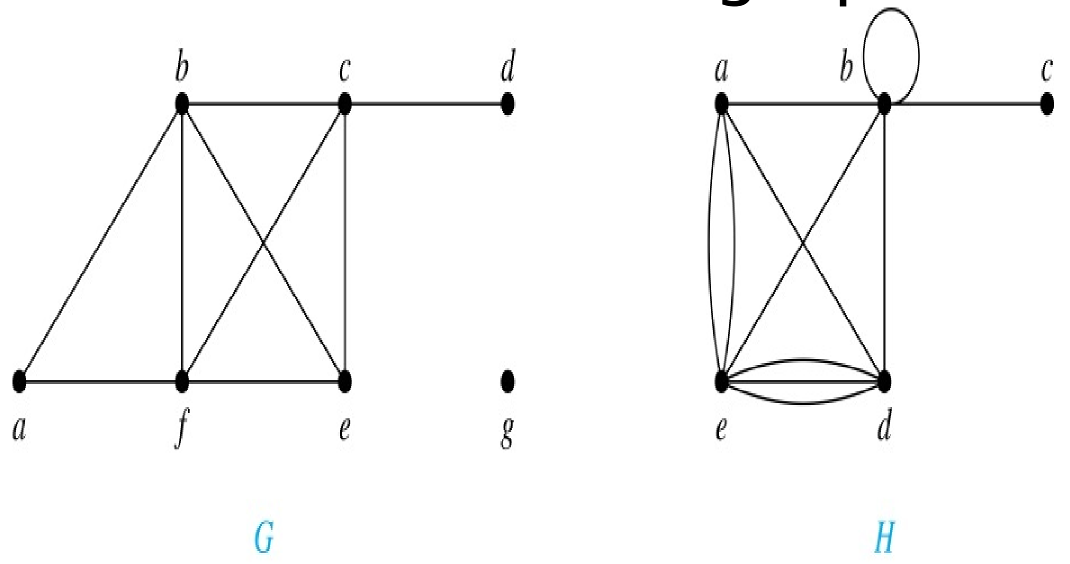
Degree and neighborhood of a vertex

Definition 3. The *degree of a vertex v in a undirected graph* is the number of edges incident with it. The degree of the vertex *v* is denoted by deg(*v*).

Definition 3. The neighborhood (neighbor set) *of a vertex v in a undirected graph, denoted N(v)* is the set of vertices adjacent to v.

Degrees and Neighborhoods of Vertices

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



Solution: deg(a) = 2, deg(b) = 4, deg(d) = 1, $N(a) = \{b, f\}$, $N(b) = \{a, c, e, f\}$, $N(d) = \{c\}$.

Handshaking Theorem

THEOREM 1 (Handshaking Lemma): If G=(V,E) is a 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. QED

Degree of Vertices (*continued*)

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 graph *G* = (*V*, *E*) with *m* edges. Then

$$\underset{\text{even}}{\underbrace{2m}} = \sum_{v\in V}\deg(v) = \sum_{v\in V_1}\deg(v) + \sum_{v\in V_2}\deg(v).$$

even

must be even since deg(*v*) is even for each *v* ∈ *V*1

must be even because 2*m* is even and the sum of degrees of vertices of even degree is even. Thus, since this is the sum of degrees of all vertices of odd degree, there must be an even number of them.

Handshaking Theorem:Examples

Example: How many edges are there in a graph with 10 vertices, each having degree six?

Solution: the sum of the degrees of the vertices is $6 \cdot 10 = 60$. The handshaking theorem says $2m = 60$.

So the number of edges is $m = 30$.

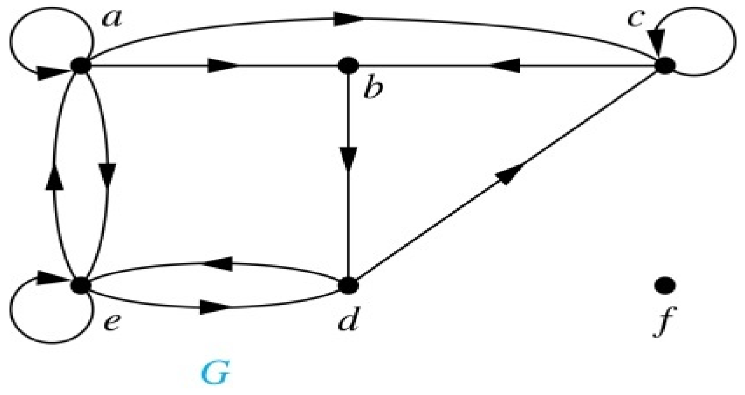
Example: If a graph has 5 vertices, can each vertex have degree 3?

Solution: This is not possible by the handshaking thorem, because the sum of the degrees of the vertices $3 \cdot 5 = 15$ is odd.

Directed Graphs

Definition: The *in-degree of a vertex v*, denoted *deg−(v)*, is the number of edges directed into *v*. The *out-degree of v*, denoted *deg+(v)*, is the number of edges directed out of *v*. Note that a loop at a vertex contributes 1 to both in-degree and out-degree.

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.

Directed Graphs (*continued*)

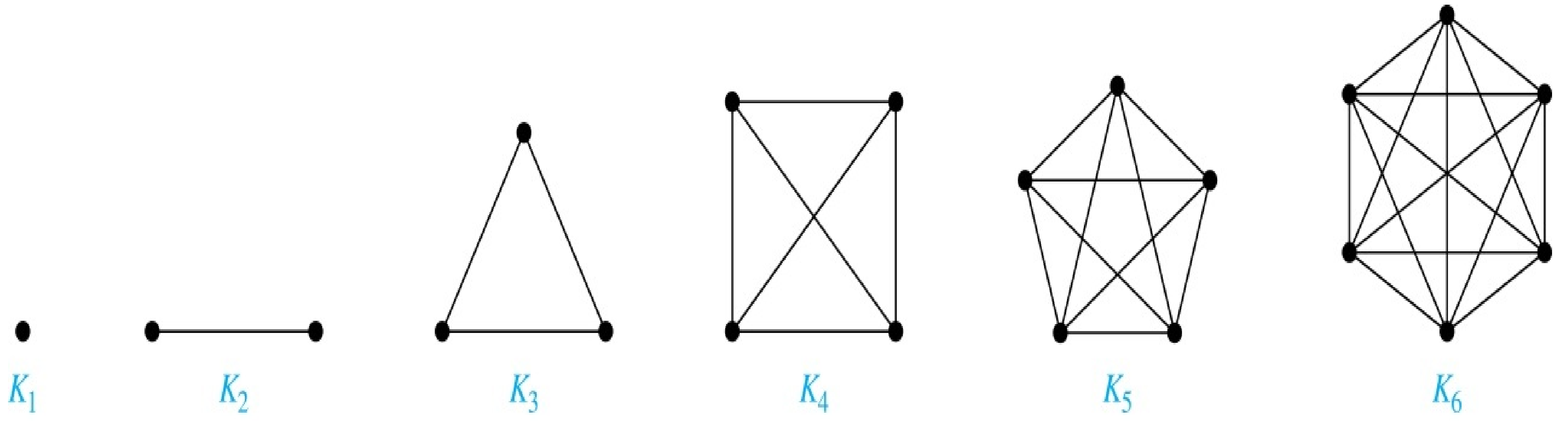
Theorem 3: Let $G = (V, E)$ be a directed graph. Then:

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

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. Both sums must be |E|.

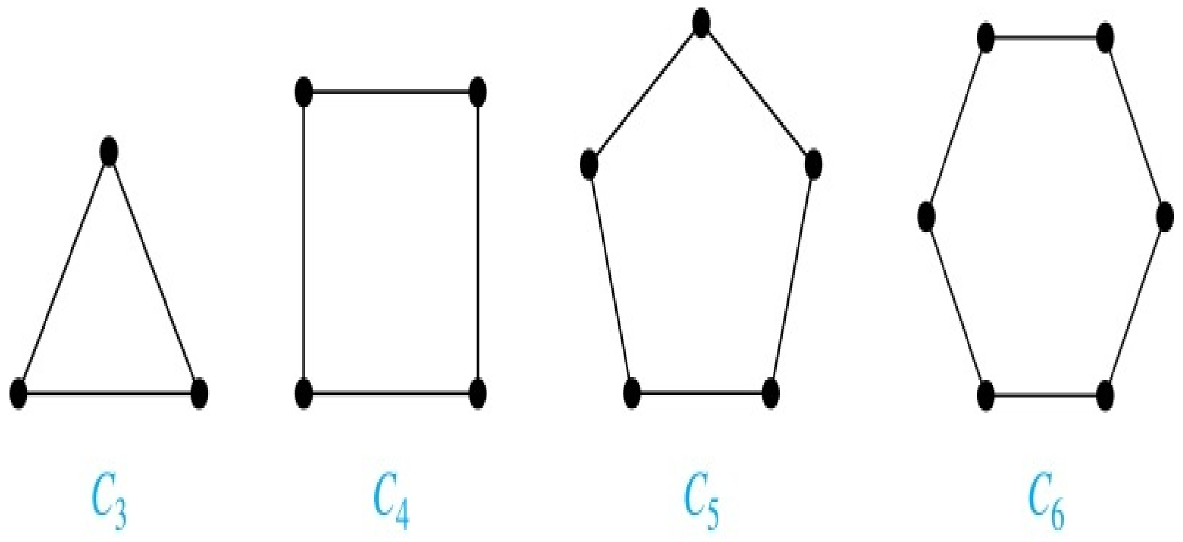
Special Types of Graphs: Complete Graphs

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.



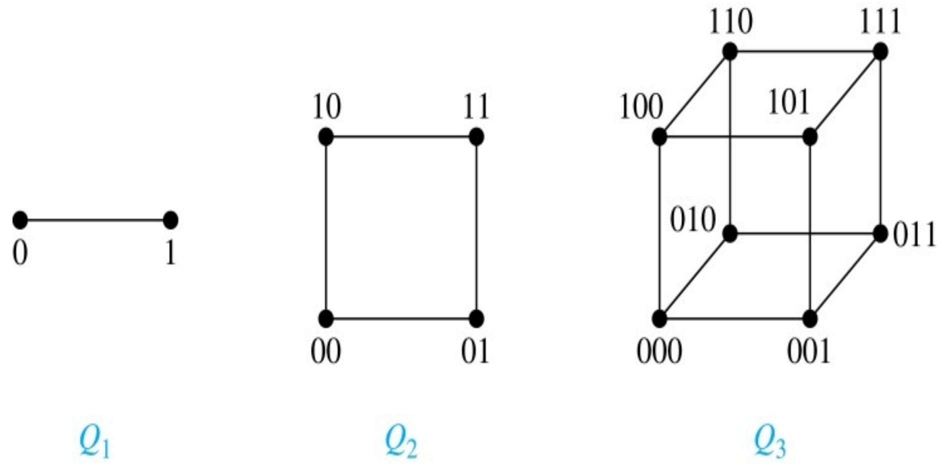
Special Types of Graphs: Cycles

A *cycle* C_n for *n* ≥ 3 consists of *n* vertices *v*1, *v*2 ,⋯ , *v*n, and edges {*v*1, *v*2}, {*v*2, *v*3} ,⋯ , {*vn*-1, *vn*}, {*vn*, *v*1}.



Special Types of Simple Graphs: *n*-Cubes

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

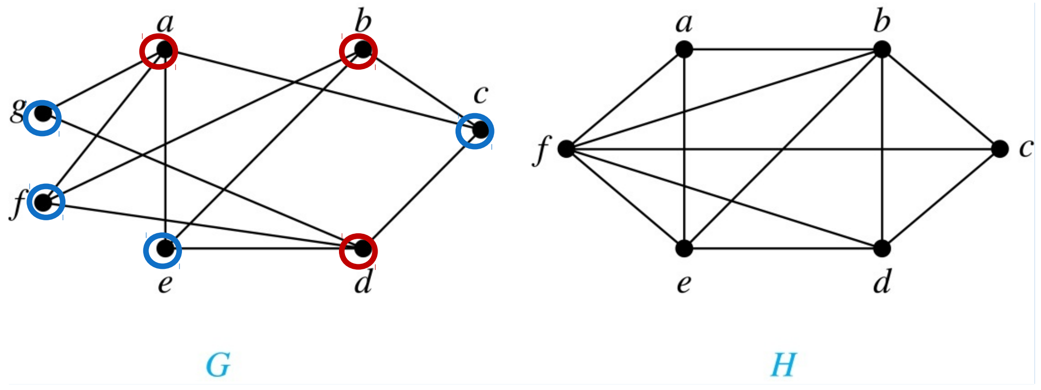


Bipartite Graphs

Definition:

An equivalent definition of a bipartite graph is one where it is possible to color the vertices either red or blue so that no two adjacent vertices are the same color.

G is bipartite

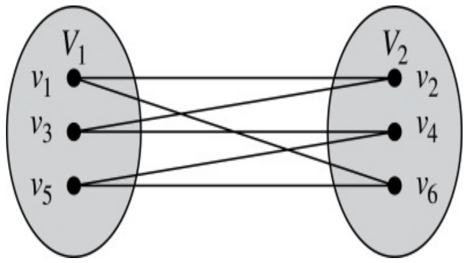


H is not bipartite: if we color *a* red, then its neighbors *f* and *b* must be blue. But f and b are adjacent.

Bipartite Graphs (*continued*)

Example: Show that C_6 is bipartite.

Solution: Partition the vertex set into *V*1 = {*v*1, *v*3, *v*5} and *V*2 = {*v*2, *v*4, *v*6}:



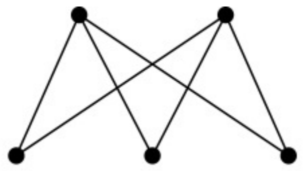
Example: Show that *C_3* is not bipartite.

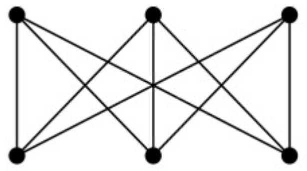
Solution: If we partition vertices of *C*_3 into two nonempty sets, one set must contain two vertices. But every vertex is connected to every other. So, the two vertices in the same partition are connected. Hence, *C*_3 is not bipartite.

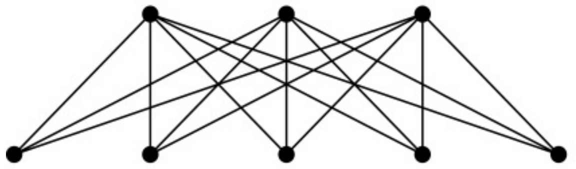
Complete Bipartite Graphs

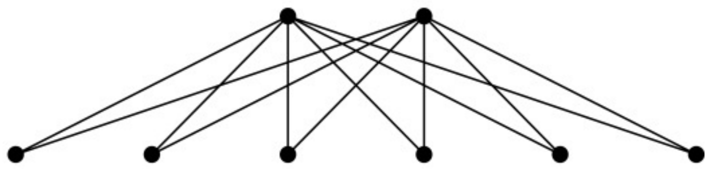
Definition: A *complete bipartite graph* $K_{m,n}$ is a graph that has its vertex set partitioned into two subsets *V_1* of size *m* and *V_2* of size *n* such that there is an edge from every vertex in *V_1* to every vertex in *V_2.*

Examples:


$K_{2,3}$


$K_{3,3}$

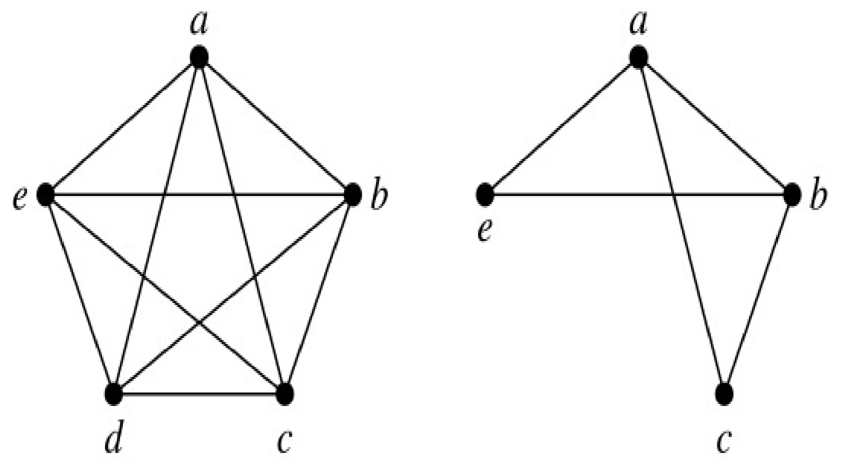

$K_{3,5}$


$K_{2,6}$

Subgraphs

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

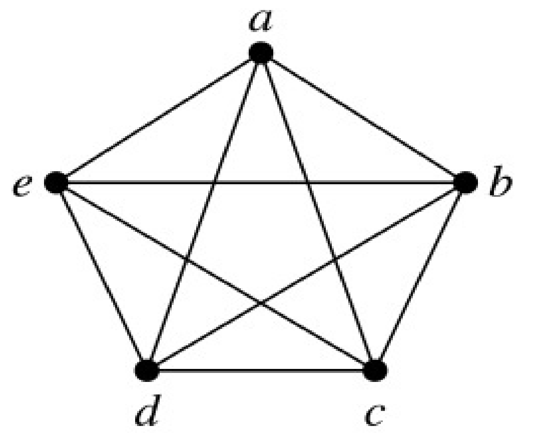
Example: here is K_5 and one of its (proper) subgraphs:

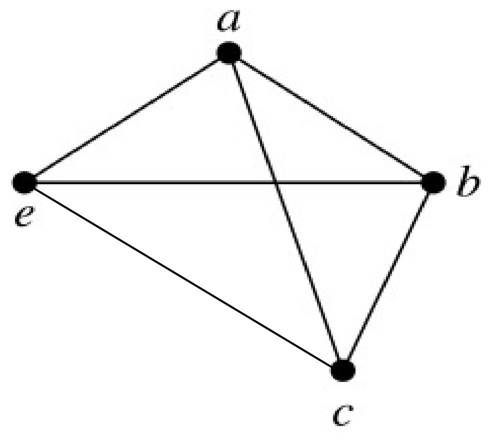


Induced Subgraphs

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

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

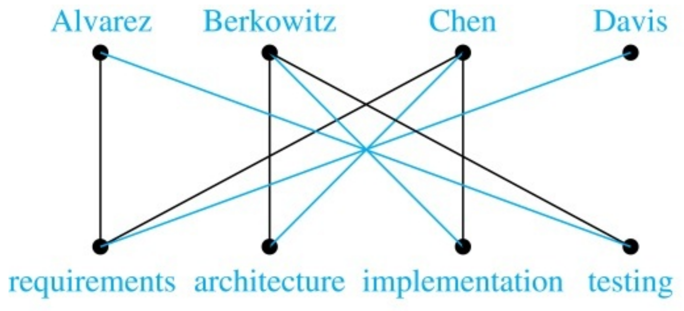




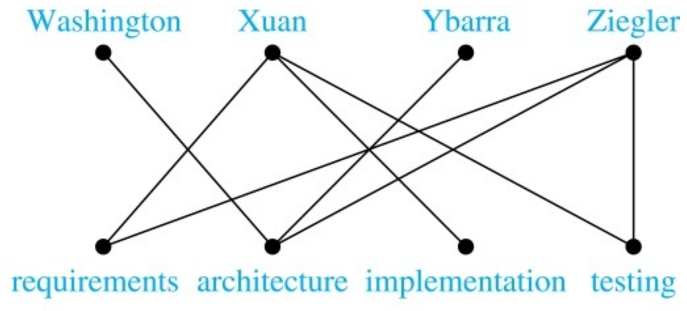
Bipartite Graphs and Matchings

Bipartite graphs used extensively in app's involving matching elements of two sets:

Job assignments - vertices represent the jobs and the employees, edges link employees with jobs they are qualified for. Maximize # of employees matched to jobs.



(a)



(b)

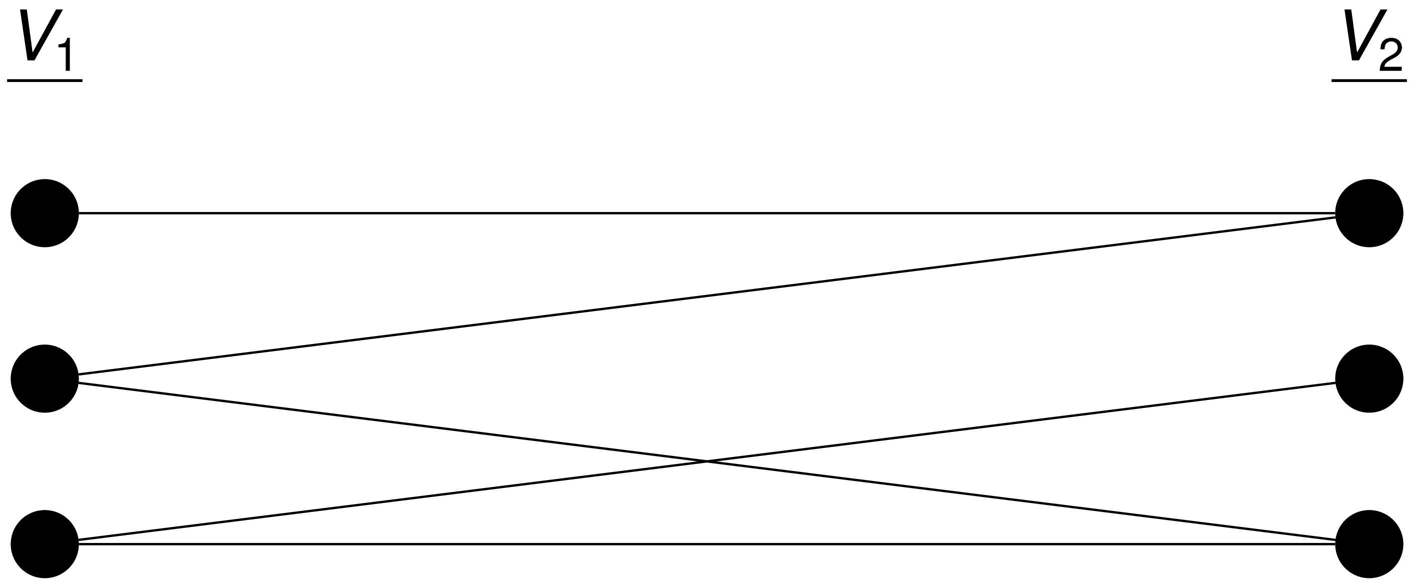
Marriage/dating - vertices represent men & women and edges link a man & woman if they are acceptable to each other as partners.

Bipartite graphs

A bipartite graph is a (undirected) graph $G = (V, E)$ whose vertices can be partitioned into two disjoint sets (V_1, V_2), with $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$, such that for every edge $e \in E$, $e = \{u, v\}$ such that $u \in V_1$ and $v \in V_2$. In other words, every edge connects a vertex in V_1 with a vertex in V_2.

Equivalently, a graph is bipartite if and only if it is possible to color each vertex red or blue such that no two adjacent vertices are the same color.

Example of a Bipartite Graph



Matchings in Bipartite Graphs

A **matching**, M, in a graph, $G = (V, E)$, is a subset of edges, $M \subseteq E$, such that there does not exist two distinct edges in M that are incident on the same vertex. In other words, if $\{u, v\}, \{w, z\} \in M$, then either $\{u, v\} = \{w, z\}$ or $\{u, v\} \cap \{w, z\} = \emptyset$.

A **maximum matching** in graph G is a matching in G with the maximum possible number of edges.

Perfect/complete matchings

For a graph $G = (V, E)$, we say that a subset of edges, $W \subseteq E$, **covers** a subset of vertices, $A \subseteq V$, if for all vertices $u \in A$, there exists an edge $e \in W$, such that e is incident on u, i.e., such that $e = \{u, v\}$, for some vertex v.

In a bipartite graph $G = (V, E)$ with bipartition (V_1, V_2), a **complete matching** with respect to V_1, is a matching $M' \subseteq E$ that covers V_1, and a **perfect matching** is a matching, $M^* \subseteq E$, that covers V.

Question: When does a bipartite graph have a perfect matching?

Hall's Marriage Theorem

For a bipartite graph $G = (V, E)$, with bipartition (V_1, V_2), there exists a matching $M \subseteq E$ that covers V_1 if and only if for all $S \subseteq V_1$, $|S| \leq |N(S)|$.

Proof: For $G = (V, E)$, with $A \subseteq V$, let $N_G(A)$ denote the neighbors of A in G.
First, "only if" direction: Suppose there is a matching M in G that covers V_1. We show that $\forall S \subseteq V_1$, $|S| \leq |N_G(S)|$. Suppose, for contradiction, that there is a subset $S \subseteq V_1$ such that $|S| > |N_G(S)|$. Then no matching M could possibly covers S, because there aren't enough neighbors $N_G(S)$. Done.

The "if" direction of the proof is harder...

proof of Hall’s Theorem, continued...

“If” direction: Suppose $\forall S \subseteq V_1$, $|S| \leq |N_G(S)|$. Then we prove a matching M exists which covers V_1, by **induction** on the size $|V_1|$.

Base case: $|V_1| = 1$. Since $|V_1| \leq |N_G(V_1)|$, there must be an edge covering the vertex u in $V_1 = \{u\}$.

Inductive step: Suppose (by **inductive hypothesis**) that the claim holds for bipartite graphs G' with $|V_1'| = j \leq k$. Suppose $|V_1| = k + 1$.

proof of Hall's Theorem (continued)

Case 1: Suppose that for every nonempty strict subset $S \subset V_1$, we have $|S| \leq |N_G(S)| - 1$. Take any $\{u, v\} \in E$, with $u \in V_1$. Remove u and v (and the edges incident on them) from G. Call the resulting bipartite graph G', with bipartition $(V_1 - \{u\}, V_2 - \{v\})$.
By the induction hypothesis, there must exist a matching M' in G' that covers $V_1 - \{u\}$, because for every subset $S \subseteq V_1 - \{u\}$, $N_G(S) \subseteq N_{G'}(S) \cup \{v\}$, and thus $|N_{G'}(S)| \geq |N_G(S)| - 1 \geq |S|$.
But then $M = M' \cup \{\{u, v\}\}$ is a matching in G which covers V_1.

Case 2: Suppose, on the contrary, that there exists a nonempty strict subset $S \subset V_1$ with $|S| = |N_G(S)|$.

Any matching that covers V_1 must match S to $N_G(S)$.

By the induction hypothesis, there is a matching M' covering S on the bipartite subgraph G' of G induced by $S \cup N_G(S)$. And furthermore, the bipartite subgraph G'' of G induced by $(V_1 - S) \cup (V_2 - N_G(S))$ also satisfies the condition, and contains a matching M'' that covers $(V_1 - S)$. This is because if $A \subseteq V_1 - S$ has $|A| > |N_{G''}(A)|$, this implies $|A \cup S| > |N_G(A \cup S)|$, which violates the assumption about G.

Letting $M = M' \cup M''$, M defines a matching in G that covers V_1. $\square$

More on Matchings

Corollary A bipartite graph $G = (V, E)$ with bipartition (V_1, V_2) has a perfect matching if and only if $|V_1| = |V_2|$ and $\forall S \subseteq V_1$, $|S| \leq |N_G(S)|$.

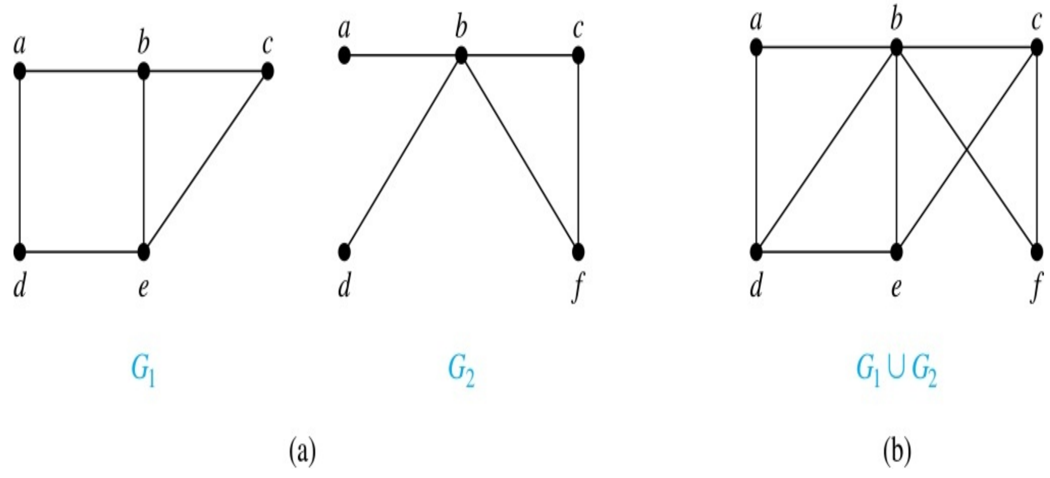
Unfortunately, the proof we have given is not constructive enough: it doesn't yield an (efficient) algorithm to compute a maximum matching in a bipartite graph.

An alternative proof of Hall's theorem (which we do not give) based on alternating paths and augmenting paths, is constructive & yields an efficient (polynomial time) algorithm for computing a maximum matching.

New Graphs from Old

Definition: The *union* of two simple graphs $G1 = (V1, E1)$ and $G2 = (V2, E2)$ is the simple graph with vertex set $V1 \cup V2$ and edge set $E1 \cup E2$. The union of $G1$ and $G2$ is denoted by $G1 \cup G2$.

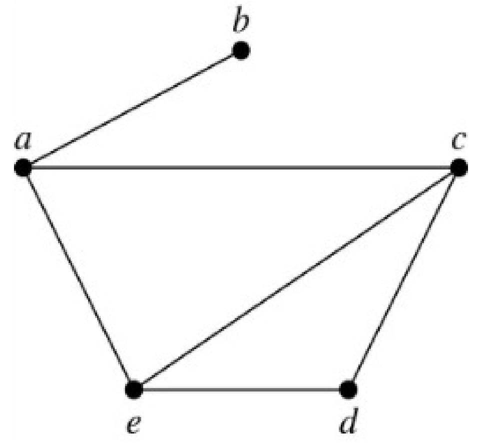
Example:



Representing Graphs: Adjacency Lists

Definition: An *adjacency list* represents a graph (with no multiple edges) by specifying the vertices that are adjacent to each vertex.

Example:



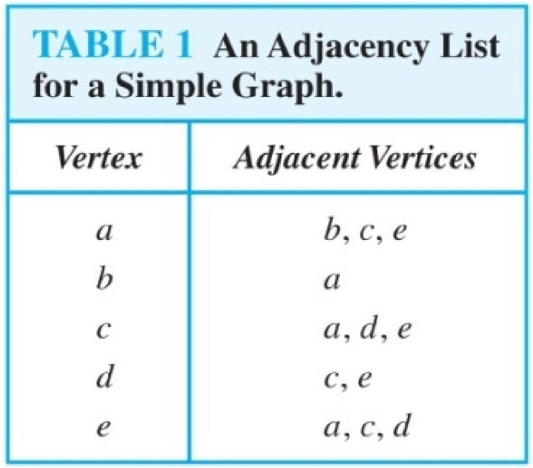
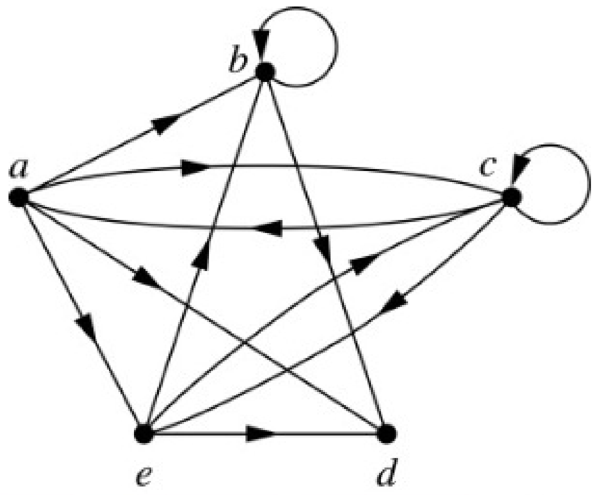


TABLE 1 An Adjacency List for a Simple Graph.

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

Example:



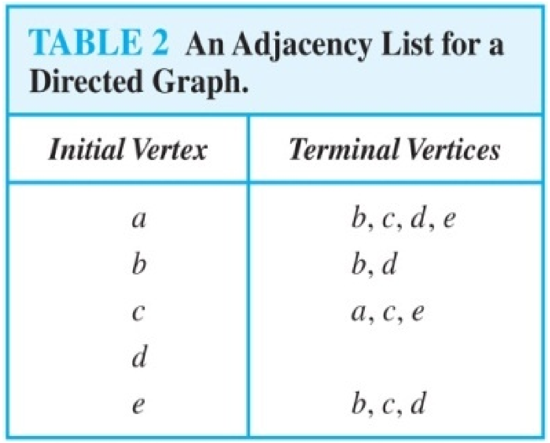


TABLE 2 An Adjacency List for a Directed Graph.

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

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, … , v_n.

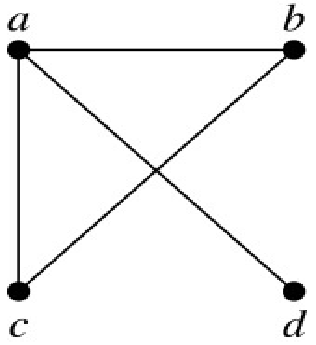
The *adjacency matrix*, **A**, of G, with respect to this listing of vertices, is the $n \times n$ 0-1 matrix with its (i, j)th entry = 1 when v_i and v_j are adjacent, and =0 when they are not adjacent.

- In other words: $A = [a_{ij}]$ and:

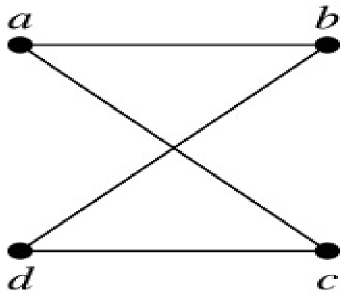
$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

Adjacency Matrices (*continued*)

Example: *The vertex ordering is is a, b, c, d.*



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



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

A sparse graph has few edges relative to the number of possible edges. Sparse graphs are more efficient to represent using an adjacency list than an adjacency matrix. But for a dense graph, an adjacency matrix is often preferable.

Note: The adjacency matrix of an undirected graph is symmetric:

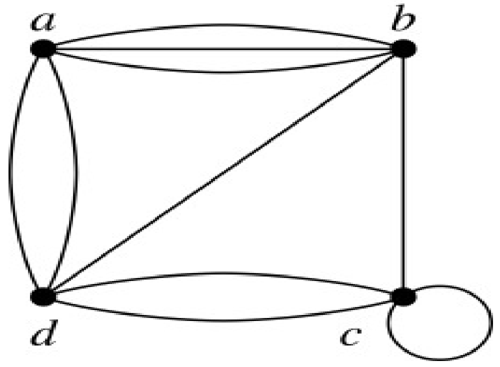
$$a_{ij} = a_{ji}, \forall i, j$$

Also, since there are no loops, each diagonal entry *is zero:* $a_{ii} = 0, \forall i$

Adjacency Matrices (*continued*)

- Adjacency matrices can also be used to represent graphs with loops and multi-edges.
- When multiple edges connect vertices *vi* and *vj*, (or if multiple loops present at the same vertex), the (*i*, *j*)th entry equals the number of edges connecting the pair of vertices.

Example: Adjacency matrix of a pseudograph, using vertex ordering *a*, *b*, *c*, *d:*



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

Adjacency Matrices (*continued*)

- Adjacency matrices can represent directed graphs in exactly the same way. The matrix A for a directed graph *G* = (*V*, *E*) has a 1 in its (*i*, *j*)th position if there is an edge from *vi* to *vj*, where *v*1, *v*2, … *vn* is a list of the vertices.
  - In other words,

$$a_{ij}=1 \ \ if \ \ (i,j)\in E$$
$$a_{ij}=0 \ \ if \ \ (i,j)\notin E$$

- Note: the adjacency matrix for a directed graph need not be symmetric.

Isomorphism of Graphs

Definition: Two (undirected) graphs *G1* = (*V1*, *E*1) and *G2* = (*V*2, *E*2) are *isomorphic* if there is a bijection, $f : V_1 \to V_2$, with the property that for all vertices $a, b \in V_1$

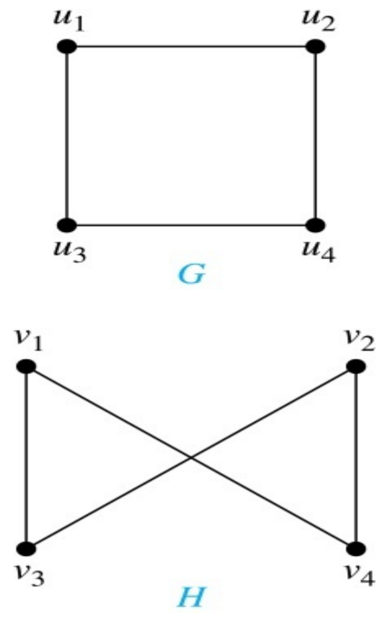
$$\{a, b\} \in E_1 \quad \text{if and only if} \quad \{f(a), f(b)\} \in E_2$$

Such a function *f* is called an *isomorphism*.

Intuitively, isomorphic graphs are "THE SAME", except for "renamed" vertices.

Isomorphism of Graphs (*cont.*)

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



Solution: The function f with $f(u1) = v1$, $f(u2) = v4$, $f(u3) = v3$, and $f(u4) = v2$ is a one-to-one correspondence between V and W.

Isomorphism of Graphs (*cont.*)

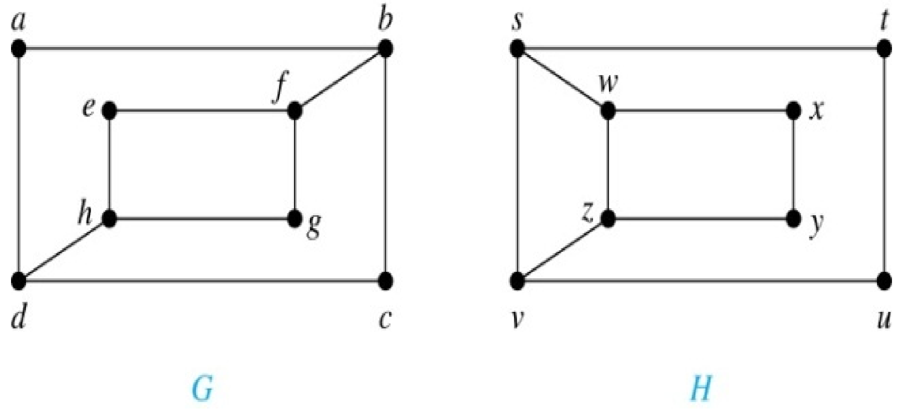
It is difficult to determine whether two graphs are isomorphic by brute force: there are n! bijections between vertices of two n-vertex graphs.

Often, we can show two graphs are not isomorphic by finding a property that only one of the two graphs has. Such a property is called *graph invariant*:

- e.g., number of vertices of given degree, the degree sequence (list of the degrees), ...

Isomorphism of Graphs (*cont.*)

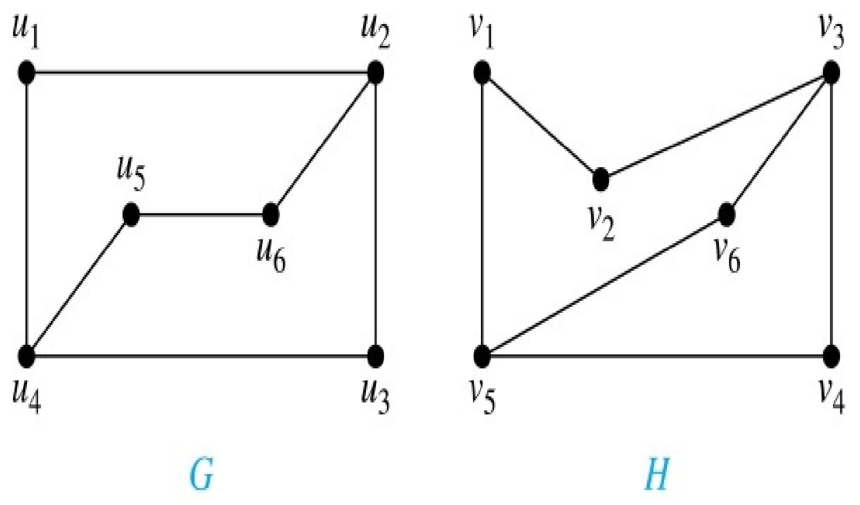
Example: Are these graphs are isomorphic?



Solution: No! Since *deg*(*a*) = 2 in *G*, *a* must correspond to *t*, *u*, *x*, or *y*, since these are the vertices of degree 2 in H. But each of these vertices is adjacent to another vertex of degree 2 in *H*, which is not true for *a* in *G*. So, G and H can not be isomorphic.

Isomorphism of Graphs (*cont.*)

Example: Determine whether these two graphs are isomorphic.



Solution: The function f is defined by: $f(u1) = v6$, $f(u2) = v3$, $f(u3) = v4$, $f(u4) = v5$, $f(u5) = v1$, and $f(u6) = v2$ is a bijection.

Algorithms for Graph Isomorphism

- There is no known polynomial time algorithm for graph isomorphism.

  In fact, it was a decades-old open problem to obtain an algorithm for deciding whether two graphs with n vertices are isomorphic with worst-case running time better than $2^{O(\sqrt{n})}$ (exponential in $\sqrt{n}$).

- BREAKTHROUGH: [Babai, 2016]: “Graph Isomorphism in Quasipolynomial time”. (Quasipolynomial time means time $2^{O((\log(n))^k)}$ for some fixed constant $k > 1$. This is much better than $2^{O(\sqrt{n})}$ time, but worse than polynomial time.)

- Babai's proof is over 80 pages long (and uses sophisticated finite group theory).
- In practice, there are actually very good algorithms for checking isomorphism, which run very efficiently for nearly all graph instances that arise in practice.

  See, e.g., the publicly available software called NAUTY ([B. McKay, 1984. . .2016]) for graph isomorphism.

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 new compound is isomorphic to the graph of an already known one.

Section Summary

- Paths
- Connectedness in Undirected Graphs
- (strong) Connectedness in Directed Graphs

Paths (in undirected graphs)

Informally, a path is a sequence of edges connecting vertices.
Formally:

Definition: For an undirected graph $G = (V, E)$, an integer $n \geq 0$, and vertices $u, v \in V$, a path (or walk) of length n from u to v in G is a sequence:

$$x_0, e_1, x_1, e_2, \ldots, x_{n-1}, e_n, x_n$$

of interleaved vertices $x_j \in V$ and edges $e_i \in E$,
such that $x_0 = u$ and $x_n = v$, and such that $e_i = \{x_{i-1}, x_i\} \in E$ for all $i \in \{1, \ldots, n\}$.

Such a path starts at u and ends at v. A path of length $n \geq 1$ is called a circuit (or cycle) if $n \geq 1$ and the path starts and ends at the same vertex, i.e., $u = v$.

A path or circuit is called simple if it does not contain the same edge more than once.

More on paths

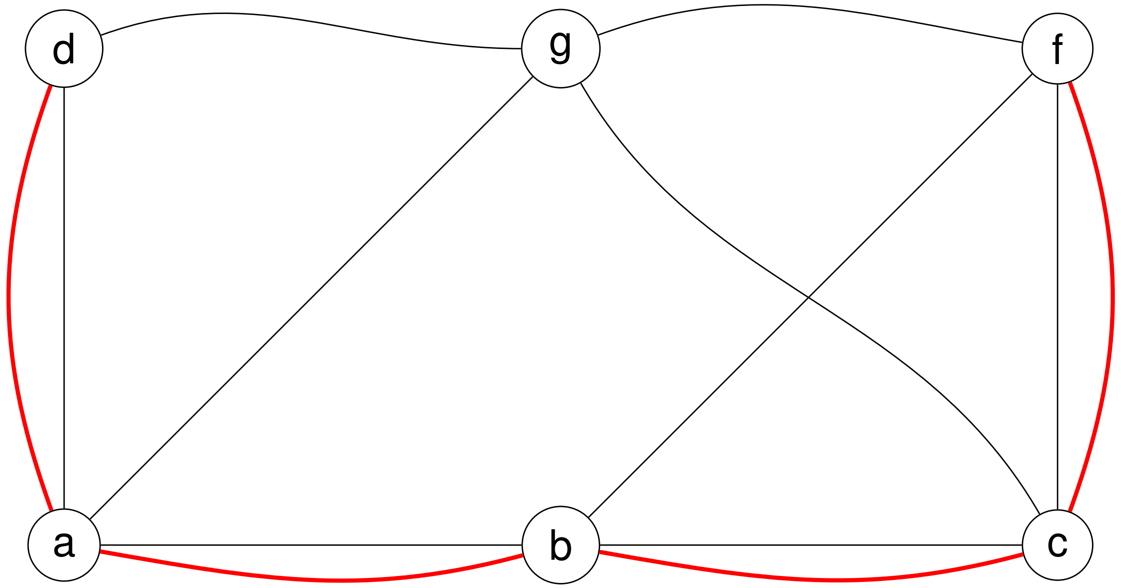
When $G = (V, E)$ is a simple undirected graph a path $x_0, e_1, \ldots, e_n, x_n$ is determined uniquely by the sequence of vertices $x_0, x_1, \ldots, x_n$. So, for simple undirected graphs we can denote a path by its sequence of vertices $x_0, x_1, \ldots, x_n$.

Note 1: The word “simple” is overloaded. Don’t confuse a simple undirected graph with a simple path. There can be a simple path in a non-simple graph, and a non-simple path in a simple graph.

Note 2: The terms “path” and “simple path” used in Rosen’s book are not entirely standard. Other books use the terms “walk” and “trail” to denote “path” and “simple path”, respectively. Furthermore, others use “path” itself to mean a walk that doesn’t re-visit any vertex, except possibly the first and last in case it is a circuit. To stick to Rosen’s terminology, we shall use the non-standard term tidy path to refer to such a walk.

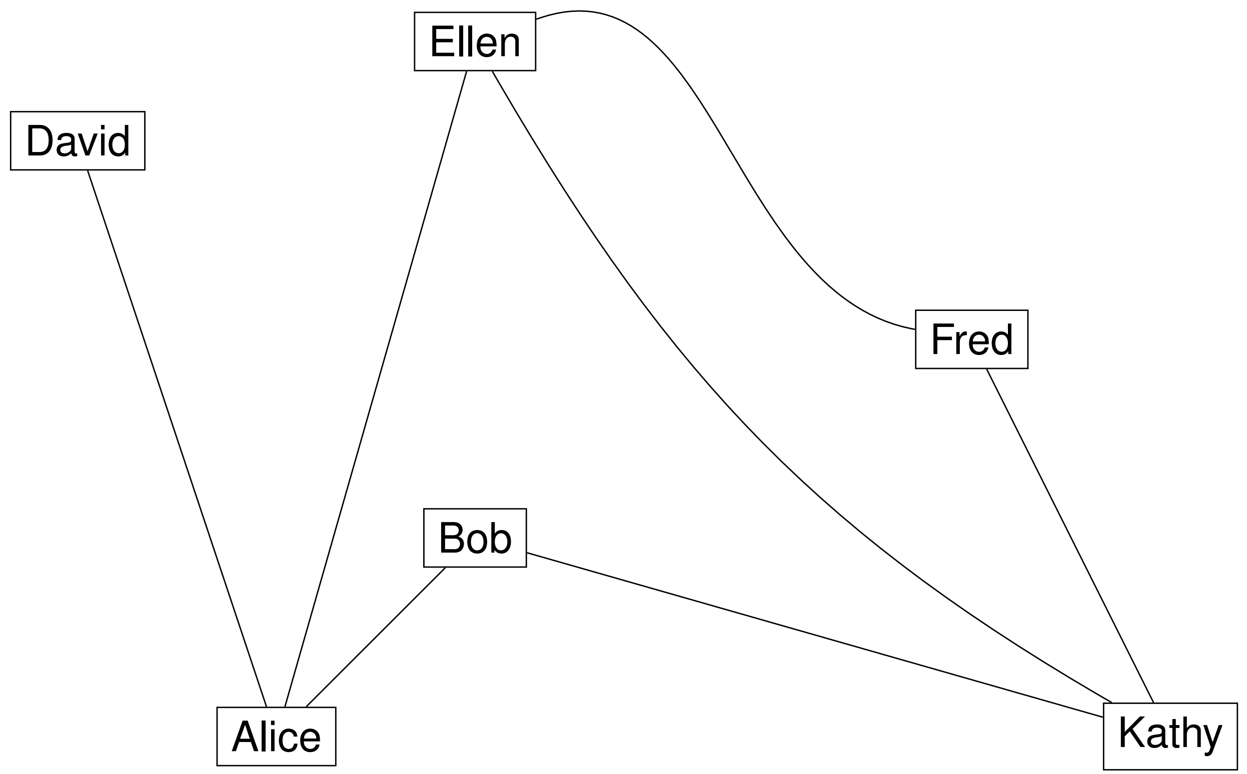
Example

Here is a simple undirected graph:



- d, a, b, c, f is a simple (and tidy) path of length 4.
- d, g, c, b, a, d is a simple (and tidy) circuit of length 5.
- a, b, g, f **is not** a path, because $\{b, g\}$ is not an edge.
- d, a, b, c, f, b, a, g **is** a path, but it **is not** a simple path, because the edge $\{a, b\}$ occurs twice in it.
- c, g, a, d, g, f is a simple path, but it **is not** a tidy path, because vertex g occurs twice in it.

Example: an acquantance graph



Paths in directed graphs (same definitions)

Definition: For an directed graph $G = (V, E)$, an integer $n \geq 0$, and vertices $u, v \in V$, a path (or walk) of length n from u to v in G is a sequence of vertices and edges $x_0, e_1, x_1, e_2, \ldots, x_n, e_n$, such that $x_0 = u$ and $x_n = v$, and such that $e_i = (x_{i-1}, x_i) \in E$ for all $i \in \{1, \ldots, n\}$.

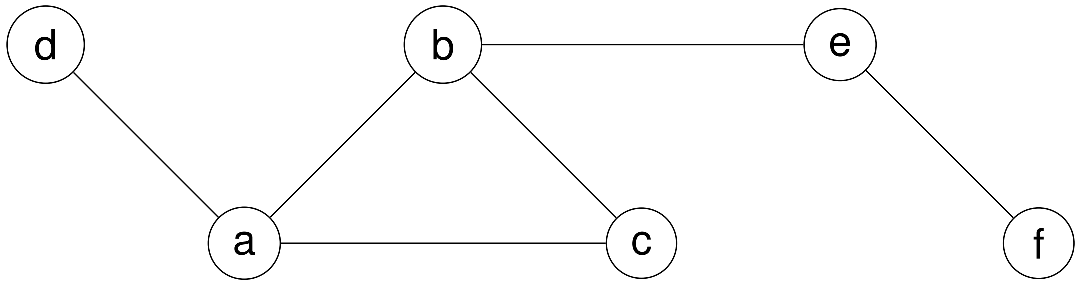
When there are no multi-edges in the directed graph G, the path can be denoted (uniquely) by its vertex sequence $x_0, x_1, \ldots, x_n$.

A path of length $n \geq 1$ is called a circuit (or cycle) if the path starts and ends at the same vertex, i.e., $u = v$.

A path or circuit is called simple if it does not contain the same edge more than once. (And we call it tidy if it does not contain the same vertex more than once, except possibly the first and last in case $u = v$ and the path is a circuit (cycle).)

Connectness in undirected graphs

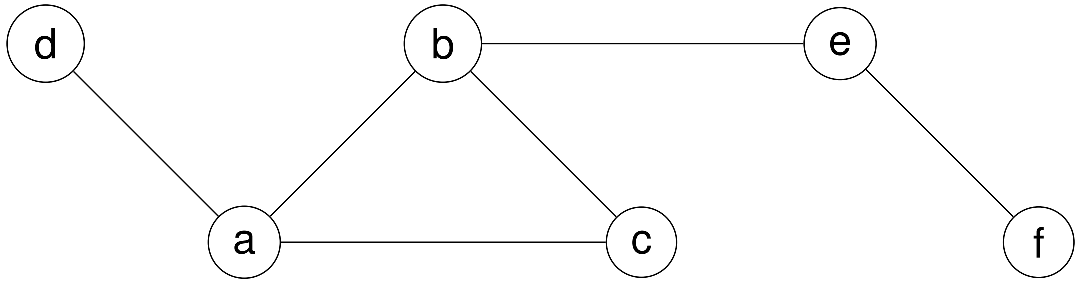
Definition: An undirected graph $G = (V, E)$ is called connected, if there is a path between every pair of distinct vertices.
It is called disconnnected otherwise.



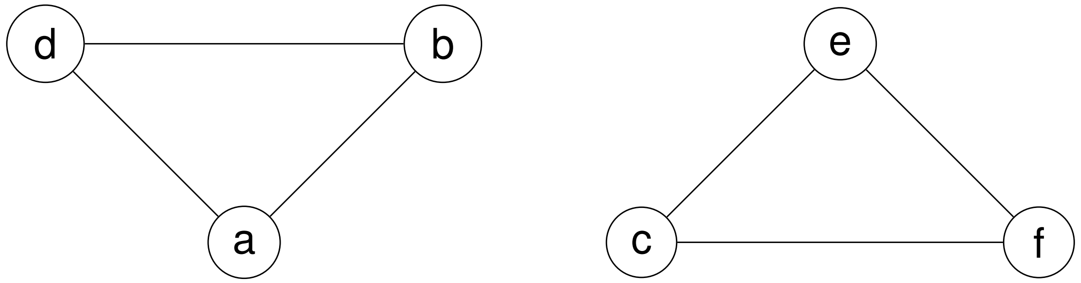
This graph is connected

Connectness in undirected graphs

Definition: An undirected graph $G = (V, E)$ is called connected, if there is a path between every pair of distinct vertices.
It is called disconnnected otherwise.



This graph is connected



This graph is **not** connected

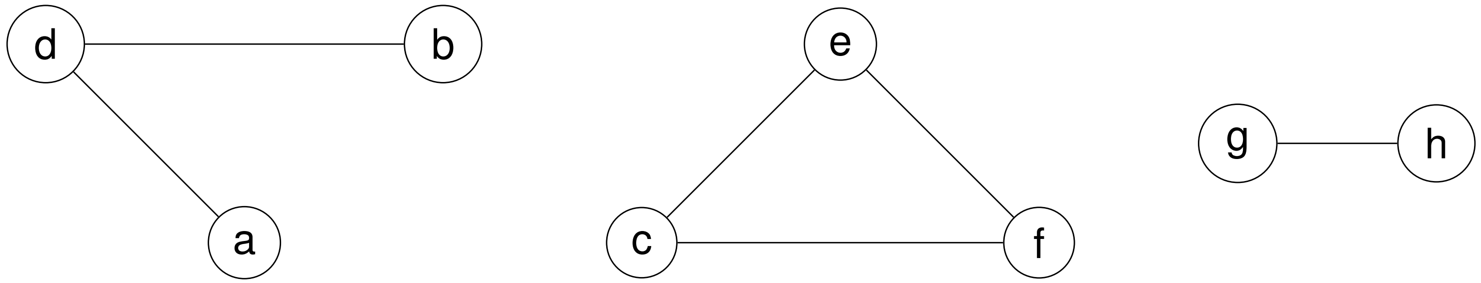
Proposition

There is always a simple, and tidy, path between any pair of vertices u, v of a connected undirected graph G.

Proof: By definition of connectedness, for every pair of vertices u, v, there must exist a **shortest** path $x_0, e_1, x_1, \ldots, e_n, x_n$ in G such that $x_0 = u$ and $x_n = v$.
Suppose this path is not tidy, and $n \geq 1$. (If $n = 0$, the Proposition is trivial.) Then $x_j = x_k$ for some $0 \leq j < k \leq n$. But then $x_0, e_1, x_1, \ldots, x_j, e_{k+1}, x_{k+1}, \ldots, e_n, x_n$ is a shorter path from u to v, contradicting the assumption that the original path was shortest. $\square$

connected components of undirected graphs

Definition: A **connected component** $H = (V', E')$ of a graph $G = (V, E)$ is a *maximal* connected subgraph of G, meaning H is connected and $V' \subseteq V$ and $E' \subseteq E$, but H is not a proper subgraph of a larger connected subgraph R of G.



This graph, $G = (V, E)$, has 3 connected components.
(It is thus a disconnected graph.)
One connected component of G is $H_1 = (V_1', E_1')$,
where $V_1' = \{d, a, b\}$ and $E_1' = \{\{d, a\}, \{d, b\}\}$.

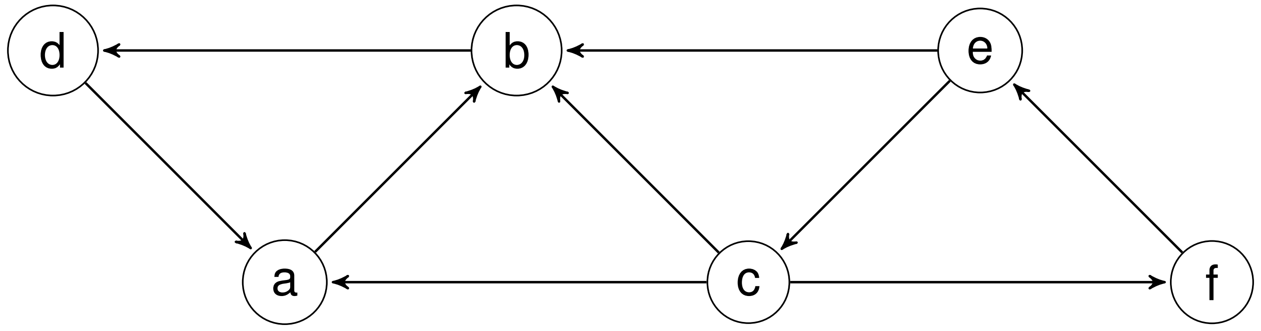
Connectedness in directed graphs

Definition: A directed graph $G = (V, E)$ is called strongly connected, if for every pair of vertices u and v in V, there is a (directed) path from u to v, *and* a directed path from v to u.

$(G = (V, E)$ is weakly connected if there is a path between every pair of vertices in V in the underlying undirected graph (meaning when we ignore the direction of edges in E.)

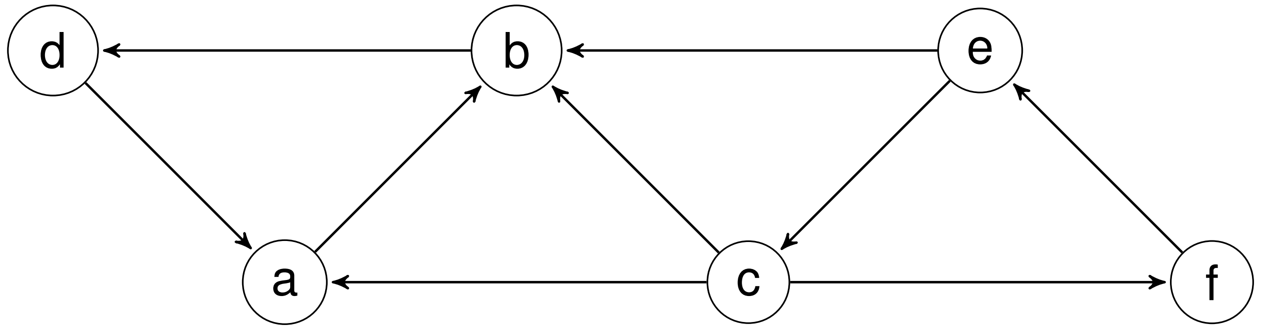
A strongly connected component (SCC) of a directed graph G, is a maximal strongly connected subgraph H of G which is not contained in a larger strongly connected subgraph of G.

Example



This digraph, G, is **not** strongly connected, because, for example, there is no directed path from b to c.
Question: what are the strongly connected components (SCCs) of G?

Example



This digraph, G, is **not** strongly connected, because, for example, there is no directed path from b to c.
Question: what are the strongly connected components (SCCs) of G?

One strongly connected component (SCC) of G is $H_1 = (V'_1, E'_1)$, where $V'_1 = \{d, a, b\}$ and $E'_1 = \{(d, a), (a, b), (b, d)\}$.

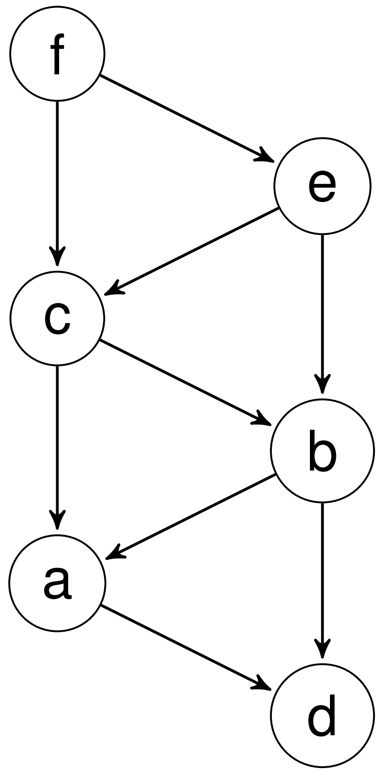
Another SCC of G is $H_2 = (V'_2, E'_2)$, where $V'_2 = \{e, c, f\}$ and $E'_2 = \{(e, c), (c, f), (f, e)\}$.

There are no other SCCs in G.

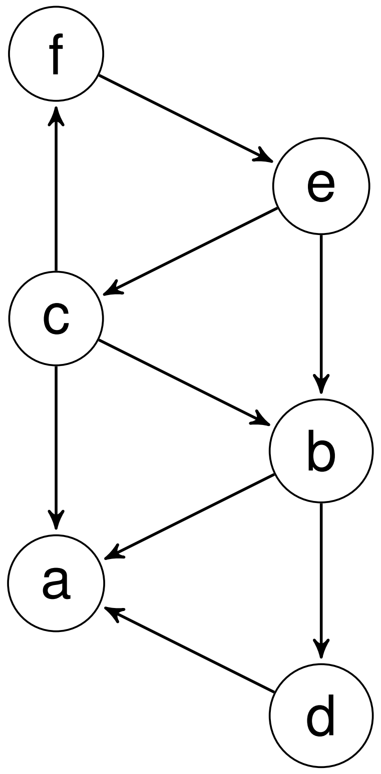
Directed Acyclic Graphs

A **Directed Acyclic Graph (DAG)**, is a directed graph that contains no circuits or loops.

Example:



This is a DAG

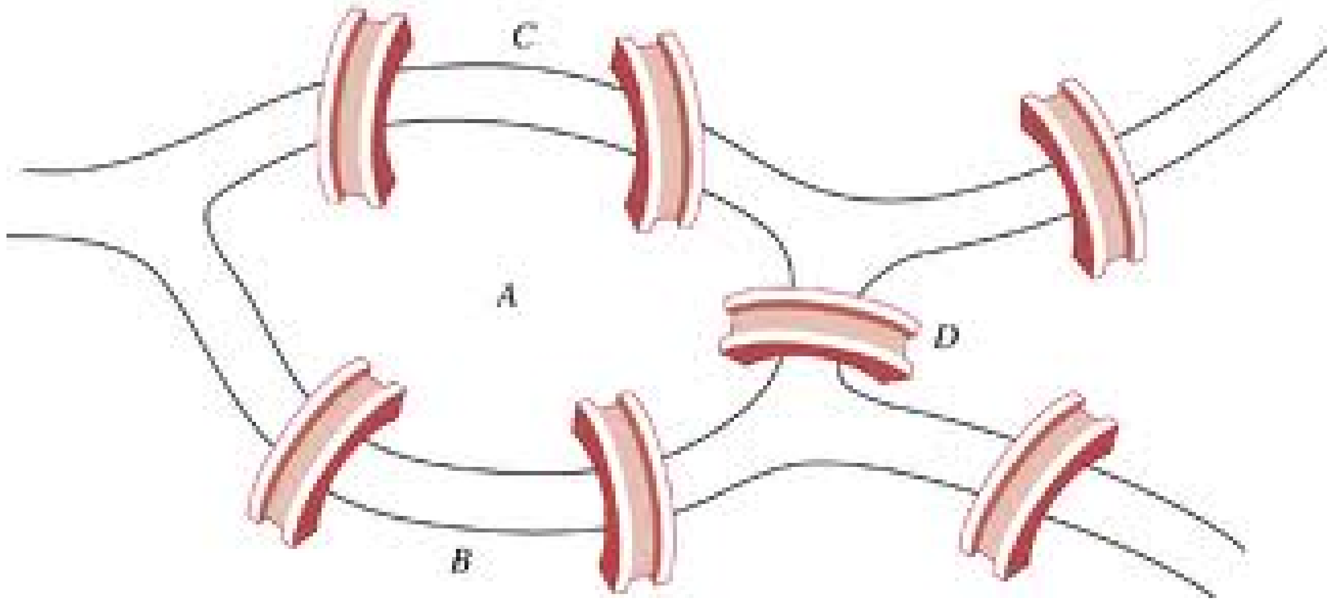


This is NOT a DAG

- Euler Paths and Euler Circuits
- Hamiltonian Paths and Hamiltonian Circuits

The Königsberg Bridge Problem

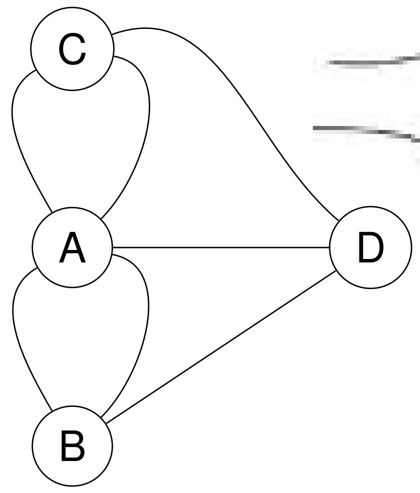
Leonard Euler (1707-1783) was asked to solve the following:

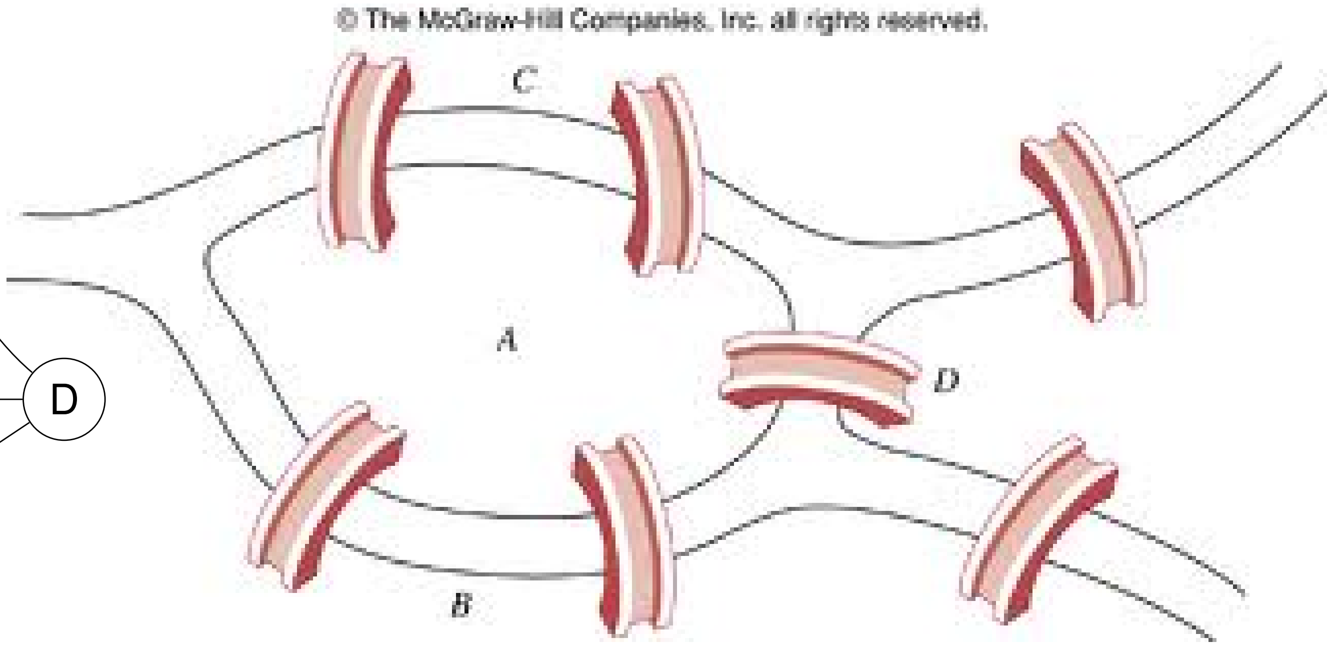


Question: Can you start a walk somewhere in Königsberg, walk across each of the 7 bridges exactly once, and end up back where you started from?

The Königsberg Bridge Problem

Leonard Euler (1707-1783) was asked to solve the following:





Question: Can you start a walk somewhere in Königsberg, walk across each of the 7 bridges exactly once, and end up back where you started from?

Euler (in 1736) used “graph theory” to answer this question.

Euler paths and Euler Circuits

Recall that an (undirected) multigraph does not have any loops, but can have multiple edges between the same pair of vertices.

Definition: An Euler path in a multigraph G is a simple path that contains every edge of G.
(So, every edge occurs exactly once in the path.)

An Euler circuit in an multigraph G is a simple circuit that contains every edge of G.
(So, every edge occurs exactly once in the circuit.)

Question: Is there a simple criterion for determining whether a multigraph G has an Euler path (an Euler circuit)?

Euler paths and Euler Circuits

Recall that an (undirected) multigraph does not have any loops, but can have multiple edges between the same pair of vertices.

Definition: An Euler path in a multigraph G is a simple path that contains every edge of G.
(So, every edge occurs exactly once in the path.)

An Euler circuit in an multigraph G is a simple circuit that contains every edge of G.
(So, every edge occurs exactly once in the circuit.)

Question: Is there a simple criterion for determining whether a multigraph G has an Euler path (an Euler circuit)?
Answer: Yes.

Euler's Theorem

Euler's Theorem (1736)

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

Proof: "Only if" direction: Suppose a multigraph $G = (V, E)$ has an Euler circuit, $x_0 e_1 x_1 e_2 \ldots e_m x_m$, where $x_0 = x_m = u$.
For every vertex $v \in V$, $v \neq u$, each time we enter v via an edge e_i, we must leave v via a different edge e_{i+1}. So, in total, since we see all edges incident to v exactly once, all such vertices v must have even degree.
Likewise, the initial (and final) vertex $u = x_0 = x_m$, must also have even degree, because the edges e_1 and e_m pair up in the same way.

Proof of Euler's Theorem (continued)

The (harder) "if" direction: Suppose $G = (V, E)$ is connected and every vertex in V has even degree.
We give a constructive proof that, given such a multigraph G, shows how to construct an Euler circuit (efficiently).
Start a "walk" at any vertex v, never re-using an edge, walking for as long as possible until you can not do so any more.
Claim: Such a "walk" (simple path), w_1, must end at the vertex v where it started (i.e., it must be a circuit).
Reason: For any vertex z other than v, whenever the walk enters z via an edge, there must be an odd number of edges incident to z "remaining". Note: zero is not an odd number! After leaving z, there must be an even number of edges of z "remaining".
If the simple circuit w_1 covers every edge of G, we are done.
If not, ...

Proof of Euler's theorem (final part)

Note that every vertex has even degree "remaining" after the edges of the simple circuit w_1 are removed.

If the simple circuit w_1 does not cover every edge of G, since G is connected, there must be some vertex x' on the circuit w_1 which is incident to an edge not in w_1. So, $w_1 = w_1' x w_1''$

We start a new walk at the vertex x', on the "remaining" graph without the edges of w_1. This yields a new circuit w_2 that must start and end at x'.

We can then then "splice" w_2 inside w_1 (at the point where x' occurs) in order to get a new longer Euler circuit: $w_1' w_2 w_1''$.

We can do this same process repeatedly until there are no edges remaining. $\square$

Note: this also yields a reasonably efficient algorithm for computing an Euler circuit in a connected multigraph where every vertex has even degree.

Euler's theorem for paths

Euler's Theorem for paths

A connected undirected multigraph G has an Euler path which is not an Euler circuit if and only if G has exactly two vertices of odd degree.

The proof is very similar to the case of Euler circuits: just start the initial walk at one of the vertices of odd degree.

The proof is thus similarly constructive, and yields an efficient algorithm to construct an Euler path, if one exists.

Hamiltonian Paths

Definition: A Hamiltonian path in a (undirected) graph G is a simple path that visits every vertex exactly once. (In other words, it is a tidy path that visits every vertex.)

A Hamiltonian circuit in a (undirected) graph G is a simple circuit that passes through every vertex exactly once (except for the common start and end vertex, which is seen exactly twice).

Question: Is there a simple criterion for determining whether a (simple undirected) graph has a Hamiltonian path, or Hamiltonian circuit?

Hamiltonian Paths

Definition: A Hamiltonian path in a (undirected) graph G is a simple path that visits every vertex exactly once. (In other words, it is a tidy path that visits every vertex.)

A Hamiltonian circuit in a (undirected) graph G is a simple circuit that passes through every vertex exactly once (except for the common start and end vertex, which is seen exactly twice).

Question: Is there a simple criterion for determining whether a (simple undirected) graph has a Hamiltonian path, or Hamiltonian circuit?

Answer: No. Nobody knows any efficient algorithm for determining whether a given (arbitrary) graph G has a Hamiltonian path/circuit. The problem is "**NP-complete**".

More on Hamiltonian paths/circuits

There are sufficient criteria that guarantee existence of a Hamiltonian circuit. For example:

Ore's Theorem

Every simple undirected graph, $G = (V, E)$, with $n \geq 3$ vertices, in which $deg(u) + deg(v) \geq n$ for every two non-adjacent vertices u and v in V, has a Hamiltonian circuit.

Corollary (Dirac's Theorem)

Every simple undirected graph, $G = (V, E)$, with $n \geq 3$ vertices, in which $deg(u) \geq n/2$ for all vertices $u \in V$, has a Hamiltonian circuit.

We will NOT prove these theorems, and we will NOT assume that you know these theorems.