Relations

Chapter 9

Mr. SHOAIB
RAZA

Chapter Summary

- Relations and Their Properties
- Representing Relations
- Equivalence Relations
- Partial Orderings

Relations and Their Properties

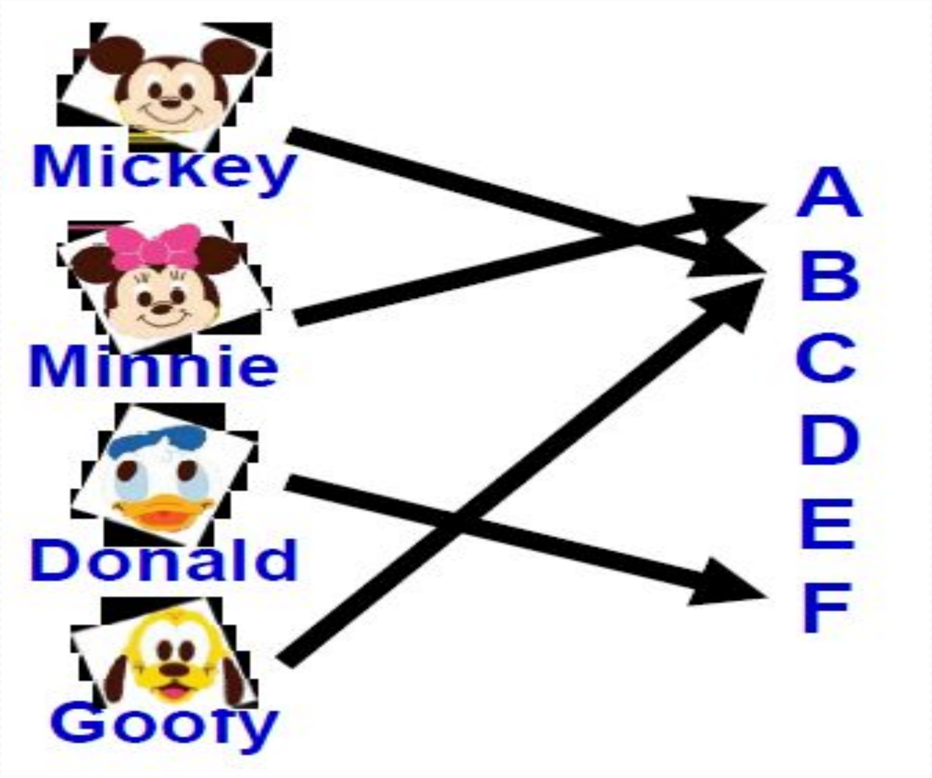
Section 9.1

Section Summary

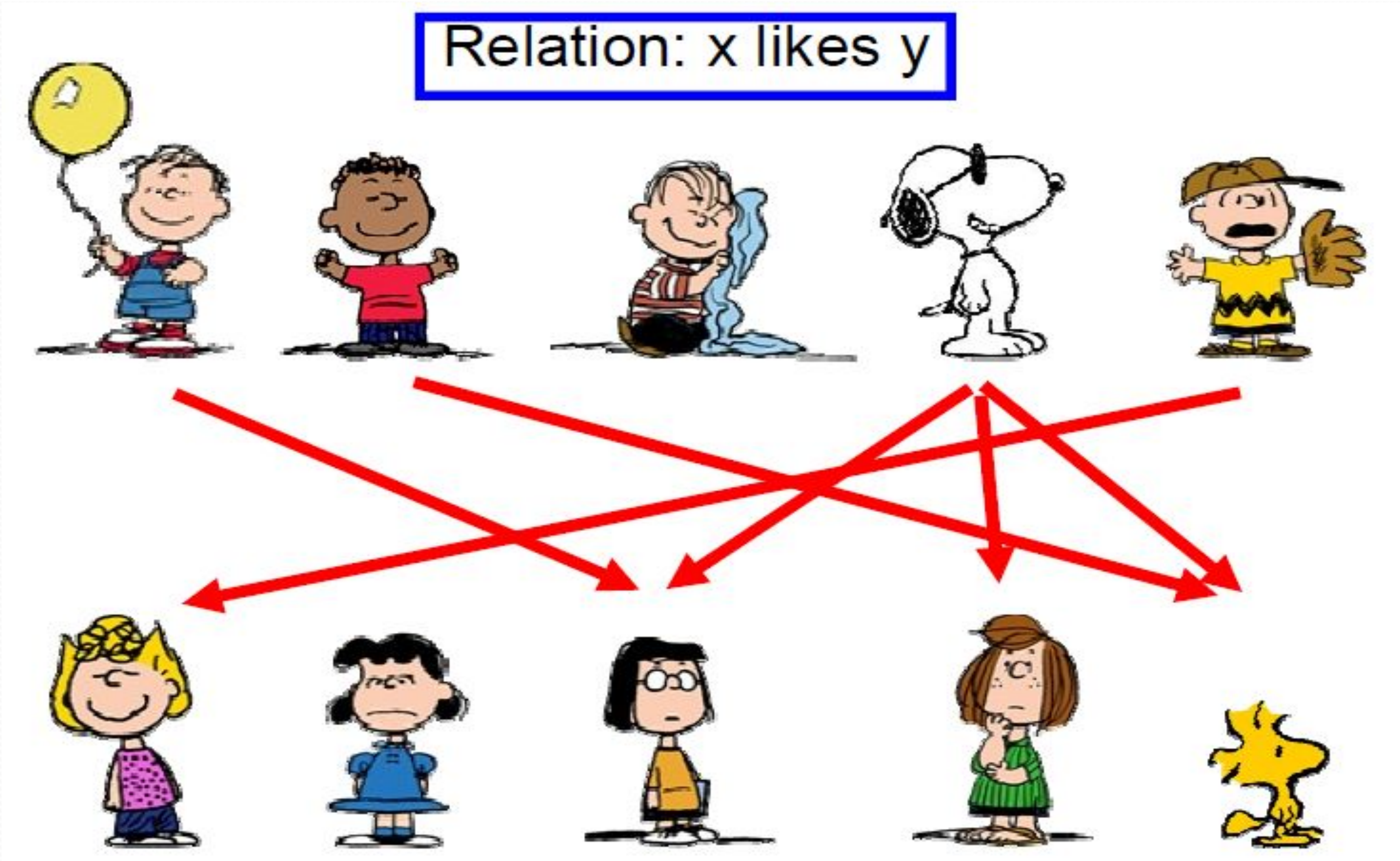
- Relations and Functions
- Properties of Relations
  - Reflexive Relations
  - Symmetric Relations
  - Antisymmetric Relations
  - Transitive Relations
  - Irreflexive Relations
  - Asymmetric Relations
- Combining Relations

Recall, Function is…

- Let **A** and **B** be nonempty sets Function ***f*** from **A** to **B** is an assignment of exactly one element of **B** to each element of **A.**
- By **defining** using a **relation**, a **function** from A to B contains **unique** ordered pair (a, b) for **every** element a ∈ A.



What is Relation?



Binary Relations

Definition: A *binary relation R* from a set *A* to a set *B* is a subset $R \subseteq A \times B$.

- Recall, for example:

  A = {a1, a2} and B = {b1, b2, b3}

  A x B = { (a1, b1), (a1, b2), (a1, b3),
  (a2, b1), (a2, b2), (a2, b3)}

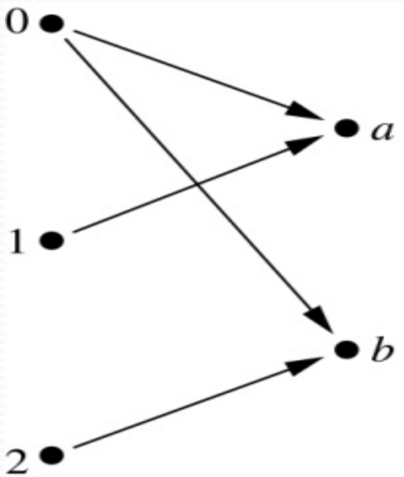
- Ordered pairs, which
  - First element comes from A
  - Second element comes from B
  - **aRb**: $(a, b) \in R$
  - $a \not R b$: $(a, b) \notin R$

Moreover, when (a, b) belongs to R, a is said to be related to b by R.

Binary Relations

Example:

- Let $A = \{0,1,2\}$ and $B = \{a,b\}$
- $\{(0, a), (0, b), (1,a) , (2, b)\}$ is a relation from A to B.
- We can represent relations from a set A to a set B graphically or using a table:



R	a	b
0	×	×
1	×	
2		×

Binary Relations

EXAMPLE:

- Let A = {eggs, milk, corn} and B = {cows, goats, hens}

Define a relation R from A to B by (a, b) $\in$R *iff* a is produced by b.

- Then R = {(eggs, hens), (milk, cows), (milk, goats)}
- Thus, with respect to this relation eggs R hens , milk R cows, etc.

Binary Relations

EXAMPLE #1:

- S = {Peter, Paul, Mary}
- C = {C++, DisMath}

- Given
  - Peter takes C++ — Peter R C++   Peter $\not R$ DisMath
  - Paul takes DisMath — Paul $\not R$ C++   Paul R DisMath
  - Mary takes none of them — Mary $\not R$ C++   Mary $\not R$ DisMath

- R = {(Peter, C++), (Paul, DisMath)}

Domain and Range of a Relation

DOMAIN OF A RELATION:

The domain of a relation R from A to B is the set of all first elements of the ordered pairs which belong to R denoted by Dom(R).

Symbolically, $\text{Dom}(R) = \{a \in A \mid (a, b) \in R\}$

RANGE OF A RELATION:

The range of a relation R from A to B is the set of all second elements of the ordered pairs which belong to R denoted Ran(R).

Symbolically, $\text{Ran}(R) = \{b \in B \mid (a, b) \in R\}$

Domain and Range of a Relation

EXERCISE:

Let $A = \{1, 2\}$, $B = \{1, 2, 3\}$,

Define a binary relation R from A to B as follows:

$R = \{(a, b) \in A \times B \mid a < b\}$ Then

- a. Find the ordered pairs in R.
- b. Find the Domain and Range of R.
- c. Is 1R3, 2R2?

SOLUTION:

Given $A = \{1, 2\}$, $B = \{1, 2, 3\}$,

$A \times B = \{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3)\}$

- a. $R = \{(a, b) \in A \times B \mid a < b\}$

  $R = \{(1,2), (1,3), (2,3)\}$

Domain and Range of a Relation

Given $A = \{1, 2\}$, $B = \{1, 2, 3\}$,

$$A \times B = \{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3)\}$$

- b. Find the Domain and Range of R.

Solution:

$\mathrm{Dom}(R) = \{1,2\}$ and $\mathrm{Ran}(R) = \{2, 3\}$

- c. Is 1R3, 2R2?

Solution:

c. Since $(1, 3) \in R$ so 1R3.

Since $(2, 2) \in R$ so 2R2.

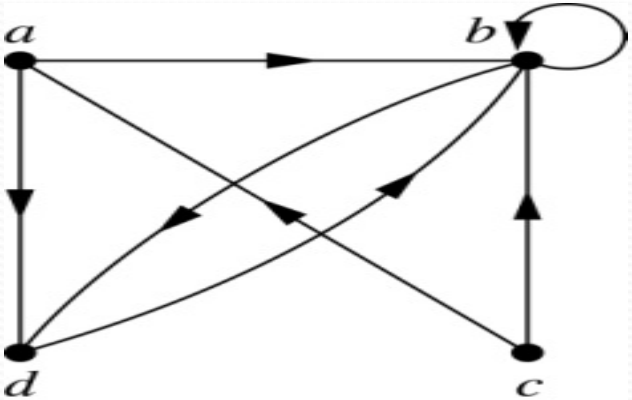
Representing Relations

Representing Relations Using Digraphs

Definition: A *directed graph*, or *digraph*, consists of a set V of *vertices* (or *nodes*) together with a set E of ordered pairs of elements of V called *edges* (or *arcs*). The vertex a is called the *initial vertex* of the edge (a,b), and the vertex b is called the *terminal vertex* of this edge.

- An edge of the form (a,a) is called a *loop*.

Example: A drawing of the directed graph with vertices a, b, c, and d, and edges (a, b), (a, d), (b, b), (b, d), (c, a), (c, b), and (d, b) is shown here.



Representing Relations Using Matrices

- A relation between finite sets can be represented using a zero-one matrix.
- Suppose R is a relation from $A = \{a_1, a_2, \ldots, a_m\}$ to $B = \{b_1, b_2, \ldots, b_n\}$.
  - The elements of the two sets can be listed in any particular arbitrary order. When $A = B$, we use the same ordering.
- The relation R is represented by the matrix $M_R = [m_{ij}]$, where

$$m_{ij} = \begin{cases} 1 \text{ if } (a_i, b_j) \in R, \\ 0 \text{ if } (a_i, b_j) \notin R. \end{cases}$$

- The matrix representing R has a 1 as its (i,j) entry when a_i is related to b_j and a 0 if a_i is not related to b_j.

Examples of Representing Relations Using Matrices

Example 1: Suppose that $A = \{1,2,3\}$ and $B = \{1,2\}$. Let R be the relation from A to B containing (a,b) if $a \in A$, $b \in B$, and $a > b$. What is the matrix representing R (assuming the ordering of elements is the same as the increasing numerical order)?

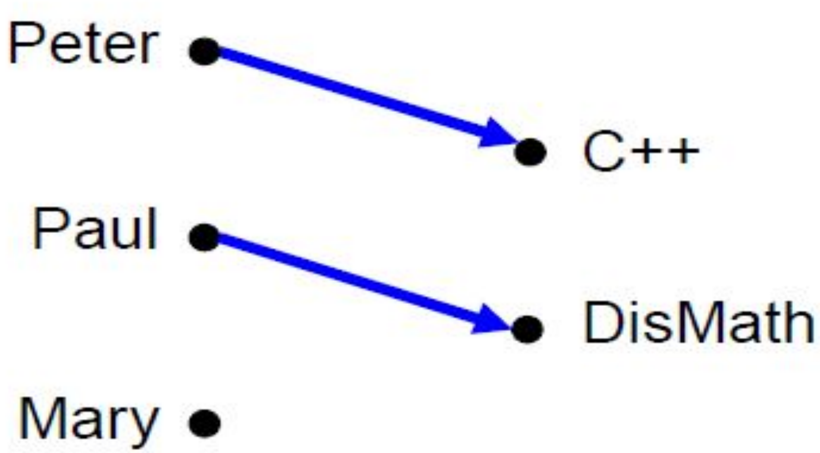
Solution: Because $R = \{(2,1), (3,1),(3,2)\}$, the matrix is

$$M_R = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

Binary Relations

EXAMPLE #1: (cont.)

- Peter R C++, Peter $\not R$ DisMath
  Paul $\not R$ C++, Paul R DisMath
  Mary $\not R$ C++, Mary $\not R$ DisMath



Directed Graph

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

Matrix

Binary Relation on a Set

Definition: A binary relation R *on a set* A is a subset of $A \times A$ or a relation from A to A.

Example:

- Suppose that $A = \{a,b,c\}$. Then $R = \{(a,a),(a,b), (a,c)\}$ is a relation on A.
- Let $A = \{1, 2, 3, 4\}$. The ordered pairs in the relation R $= \{(a,b) \mid a$ divides $b\}$ are
  $\{(1,1), (1, 2), (1,3), (1, 4), (2, 2), (2, 4), (3, 3)$, and $(4, 4)\}$.

REMARK:

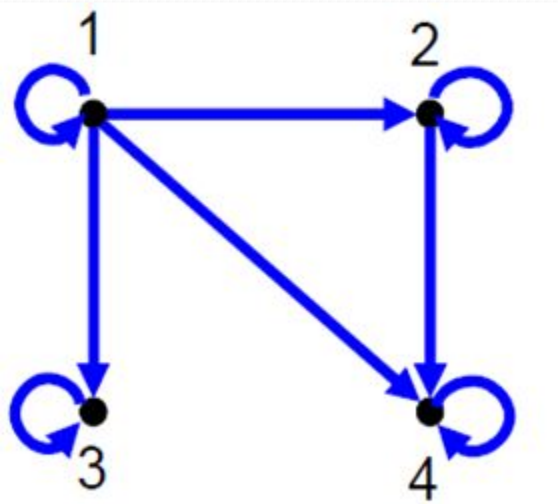
For any set A

1. A × A is known as the universal relation.
2. ∅ is known as the empty relation.

Binary Relation on a Set

- Let A be the set {1, 2, 3, 4}, which ordered pairs are in the relation R = {(a, b) | a divides b}?

- **R** = { (1,1), (1,2), (1,3), (1,4), (2,2), (2,4), (3,3), (4,4)}



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

Relations and Their Properties

Binary Relation on a Set (*cont.*)

- **Question**: How many different relations are there on a set A with n elements?

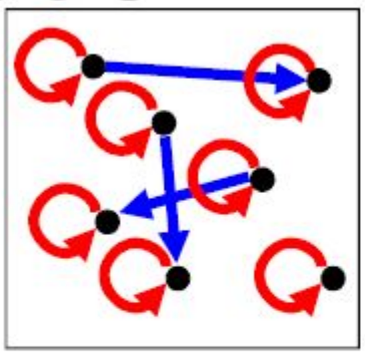
Solution:

- Suppose A has n elements
- Recall, a relation on a set A is a subset of A x A.
- A x A has n^2 elements.
- If a set has m element, its has 2^m subsets.
- Therefore, the answer is 2^{n^2}.

Reflexive Relations

Definition: R is *reflexive* iff $(a,a) \in R$ for every element $a \in A$. Written symbolically, R is reflexive if and only if

$$\forall a[a \in U \longrightarrow (a,a) \in R]$$



Reflexive

$\forall a\,((a, a) \in R)$

Every node has a self-loop

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

Reflexive

$\forall a\,((a, a) \in R)$

All 1's on diagonal

If $A = \varnothing$ then the empty relation is reflexive vacuously. That is the empty relation on an empty set is reflexive!

Reflexive Relations

EXAMPLE: Let A = {1, 2, 3, 4} and determine whether relations R1, R2, R3, and R4 are Reflexive?

R1 = {(1, 1), (3, 3), (2, 2), (4, 4)}

R2 = {(1, 1), (1, 4), (2, 2), (3, 3), (4, 3)}

R3 = {(1, 1), (1, 2), (2, 1), (2, 2), (3, 3), (4, 4)}

R4 = {(1, 3), (2, 2), (2, 4), (3, 1), (4, 4)}

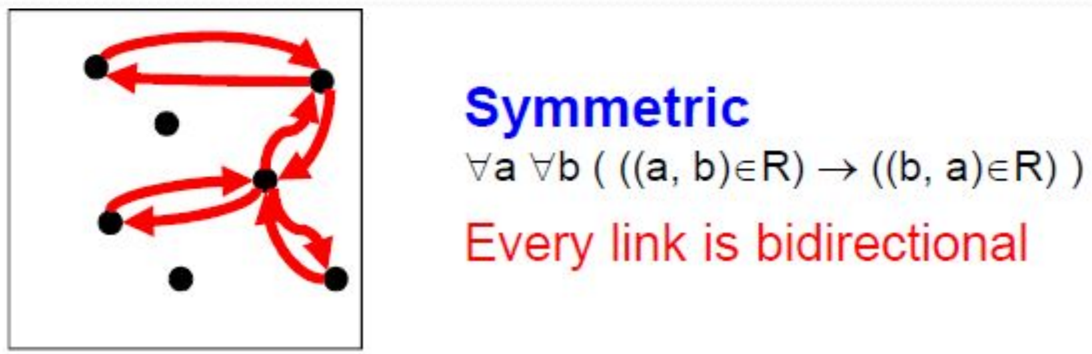
Solution:

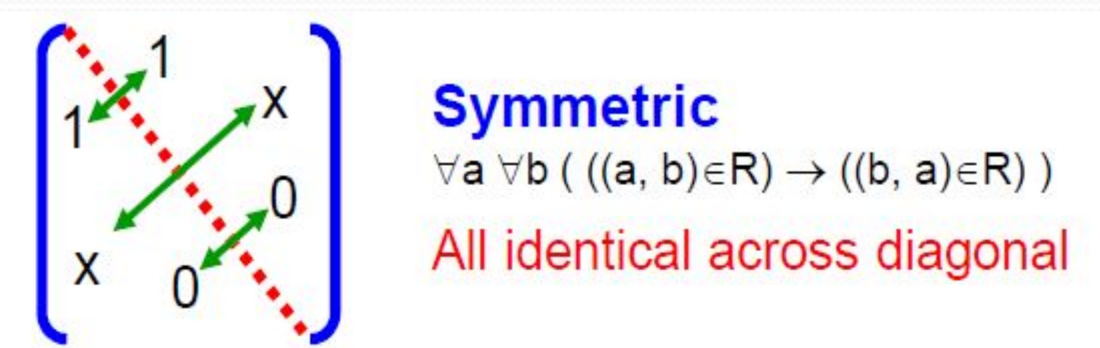
- R1 is reflexive, since $(a, a) \in R1$ for all $a \in A$.
- R2 is not reflexive, because $(4, 4) \notin R2$.
- R3 is reflexive, since $(a, a) \in R3$ for all $a \in A$.
- R4 is not reflexive, because $(1, 1) \notin R4$, $(3, 3) \notin R4$.

Symmetric Relations

Definition: R is *symmetric* iff $(b,a) \in R$ whenever $(a,b) \in R$ for all $a,b \in A$. Written symbolically, R is symmetric if and only if

$$\forall a \forall b\, [(a,b) \in R \longrightarrow (b,a) \in R]$$





Accordingly, R is symmetric if the elements in the ith row are the same as the elements in the ith column of the matrix M representing R. More precisely, M is a symmetric matrix i.e. M = M^t

Symmetric Relations

EXAMPLE: Let A = {1, 2, 3, 4} and determine whether relations R1, R2, R3, and R4 are Symmetric?

R1 = {(1, 1), (1, 3), (2, 4), (3, 1), (4,2)}

R2 = {(1, 1), (2, 2), (3, 3), (4, 4)}

R3 = {(2, 2), (2, 3), (3, 4)}

R4 = {(1, 1), (2, 2), (3, 3), (4, 3), (4, 4)}

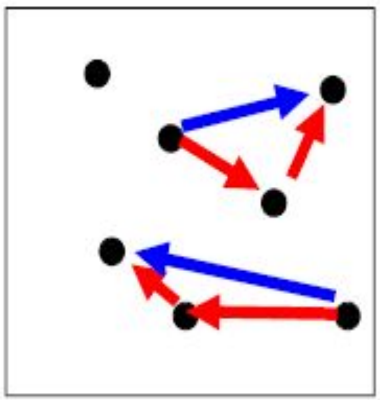
Solution:

- R1 is Symmetric, since (a, b) and (b, a) $\in$R1 for all (a, b) $\in$A.
- R2 is also symmetric. We say it is vacuously true.
- R3 is not symmetric, because (2,3) $\in$ R3 but (3,2) $\notin$ R3.
- R4 is not symmetric because (4,3) $\in$ R4 but (3,4) $\notin$ R4.

Transitive Relations

Definition: A relation R on a set A is called transitive if whenever $(a,b) \in R$ and $(b,c) \in R$, then $(a,c) \in R$, for all $a,b,c \in A$. Written symbolically, R is transitive if and only if

$$\forall a \forall b \forall c[(a,b) \in R \wedge (b,c) \in R \longrightarrow (a,c) \in R]$$



Transitive

$\forall a \forall b \forall c\ (((a,b) \in R \wedge (b,c) \in R) \rightarrow ((a,c) \in R))$

Every two adjacent forms a triangle
(Not easy to observe in Graph)

?

Transitive

$\forall a \forall b \forall c\ (((a,b) \in R \wedge (b,c) \in R) \rightarrow ((a,c) \in R))$

Not easy to observe in Matrix

For a transitive directed graph, whenever there is an arrow going from one point to the second, and from the second to the third, there is an arrow going directly from the first to the third.

Transitive Relations

EXAMPLE: Let A = {1, 2, 3, 4} and determine whether relations R1, R2 and R3 are Transitive?

R1 = {(1, 1), (1, 2), (1, 3), (2, 3)}

R2 = {(1, 2), (1, 4), (2, 3), (3, 4)}

R3 = {(2, 1), (2, 4), (2, 3), (3,4)}

Solution:

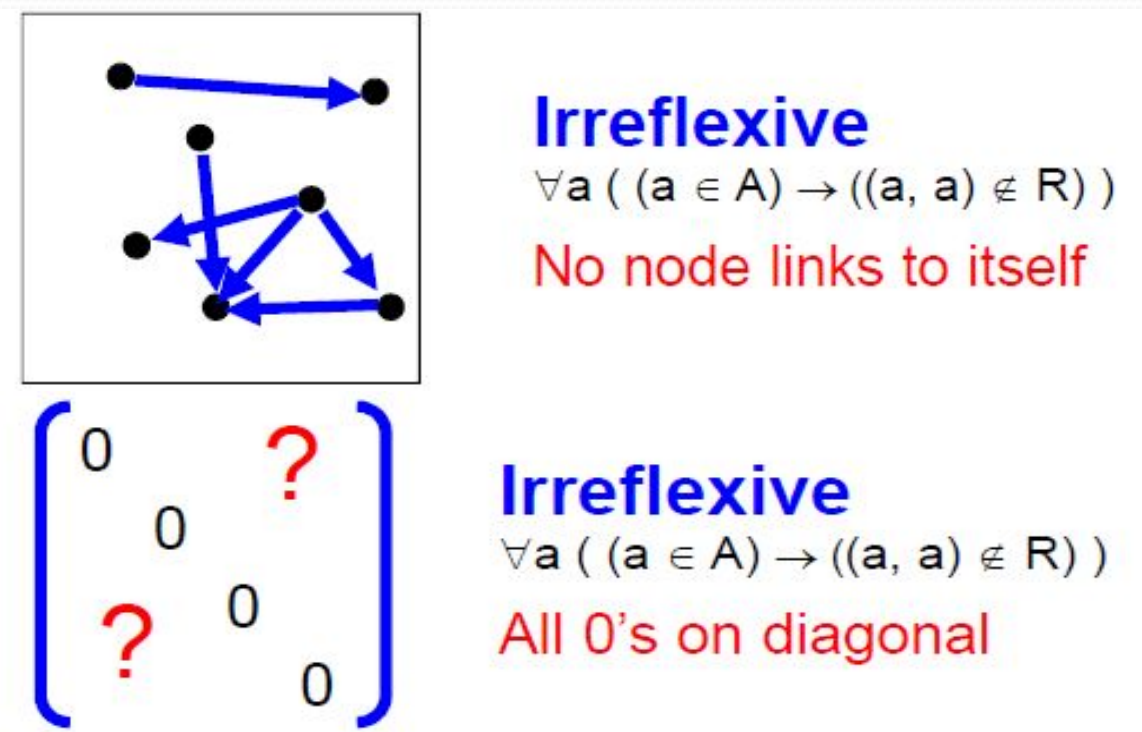
- R1 is transitive because (1, 1), (1, 2) are in R, then to be transitive relation(1,2) must be there and it belongs to R.
- R2 is not transitive since (1,2) and (2,3) $\in$ R2 but (1,3) $\notin$ R2.
- R3 is transitive.(check by definition)

Irreflexive Relations

Definition: R is irreflexive iff for all a $\in$ A,(a,a) $\notin$R. That is, R is irreflexive if no element in A is related to itself by R.

Written symbolically, R is irreflexive if and only if

$$\forall a\ [(a \in A) \longrightarrow (a\ ,\ a) \notin R]$$



R is not irreflexive iff there is an element a$\in$A such that (a,a) $\in$R.

Irreflexive Relations

EXAMPLE: Let A = {1, 2, 3, 4} and determine whether relations R1, R2 and R3 are Irreflexive?

R1 = {(1,3), (1,4), (2,3), (2,4), (3,1), (3,4)}

R2 = {(1,1), (1,2), (2,1), (2,2), (3,3), (4,4)}

R3= {(1,2), (2,3), (3,3), (3,4)}

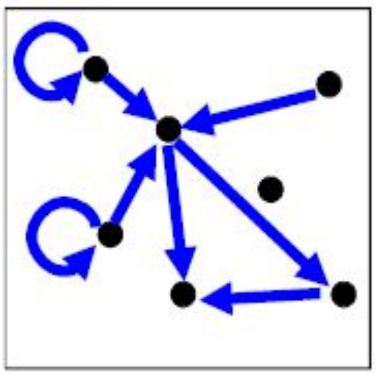
Solution:

- R1 is irreflexive since no element of A is related to itself in R1. i.e. $(1,1) \notin R1$, $(2,2) \notin R1$, $(3,3) \notin R1$, $(4,4) \notin R1$.
- R2 is not irreflexive, since all elements of A are related to themselves in R2.
- R3 is not irreflexive since $(3,3) \in R3$. Note that R3 is not reflexive.

Antisymmetric Relations

Definition: A relation R on a set A such that for all $a,b \in A$ if $(a,b) \in R$ and $(b,a) \in R$, then $a = b$ is called *antisymmetric*. Written symbolically, R is antisymmetric if and only if $\forall a \forall b$ $[(a,b) \in R \wedge (b,a) \in R \longrightarrow a = b]$

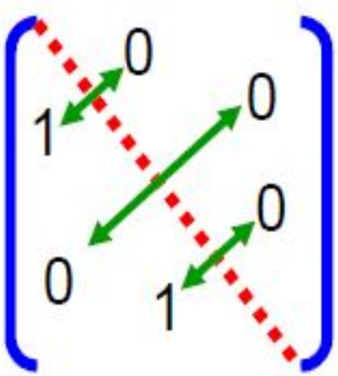
Note: (a,a) may be an element in R.



Antisymmetric

$\forall a \, \forall b \, (((a, b) \in R \wedge (b, a) \in R) \rightarrow (a = b))$

No link is bidirectional



Antisymmetric

$\forall a \, \forall b \, (((a, b) \in R \wedge (b, a) \in R) \rightarrow (a = b))$

All 1's are across from 0's

Let R be an anti-symmetric relation on a set A = {a1, a2, …, an}. Then if (ai, aj) $\in$ R for i $\neq$ j then (ai, aj) $\notin$R. Thus in the matrix representation of R there is a 1 in the ith row and jth column iff the jth row and ith column contains 0 vice versa.

Antisymmetric Relations

EXAMPLE: Let A = {1, 2, 3, 4} and determine whether relations R1, R2, R3, and R4 are Antisymmetric?

R1 = {(1,1),(2,2),(3,3)}

R2 = {(1,2),(2,2), (2,3), (3,4), (4,1)}

R3={(1,3),(2,2), (2,4), (3,1), (4,2)}

R4={(1,3),(2,4), (3,1), (4,3)}

Solution:

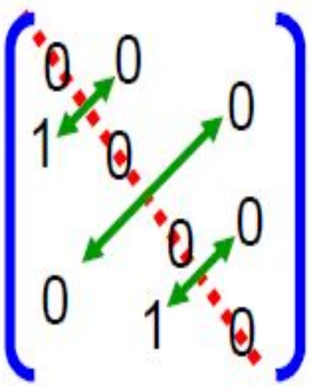
- R1 is anti-symmetric and symmetric.
- R2 is anti-symmetric but not symmetric because $(1,2) \in R2$ but $(2,1) \notin R2$.
- R3 is not anti-symmetric since $(1,3)$ & $(3,1) \in R3$ but $1 \neq 3$. Note that R3 is symmetric.
- R4 is neither anti-symmetric because $(1,3)$ & $(3,1) \in R4$ but $1 \neq 3$ nor symmetric because $(2,4) \in R4$ but $(4,2) \notin R4$.

Asymmetric Relations

Definition: R is Asymmetric iff for all $(a,b) \in R$ *than* $(b\ ,a) \notin R$.
Written symbolically, *R* is Asymmetric if and only if

$$\forall a\ \forall b[((a,b) \in R)) \longrightarrow ((b\ ,\ a) \notin R)]$$

Note:(a,a) cannot be an element in R.

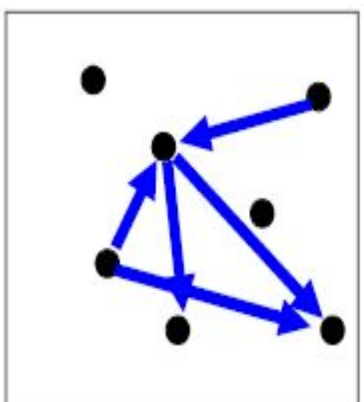


Asymmetric

$\forall a\ \forall b\ (\ ((a,\ b) \in R) \rightarrow ((b,\ a) \notin R)\ )$

All 1's are across from 0's (Antisymmetric)
All 0's on diagonal (Irreflexive)

Asymmetry = Antisymmetry + Irreflexivity



Asymmetric

$\forall a\ \forall b\ (\ ((a,\ b) \in R) \rightarrow ((b,\ a) \notin R)\ )$

No link is bidirectional (Antisymmetric)
No node links to itself (Irreflexive)

Asymmetric Relations

- **EXAMPLE**: Let A = {1, 2, 3, 4} and determine whether relations R1, R2 and R3 are Asymmetric?

  R1 = {(1,1), (1,2), (2,1), (2,2), (3,4), (4,1), (4,4)}

  R2 = {(1,2),(2,3),(3,4)}

  R3 = {(2,3),(3,3),(3,4)}

Solution:

- R1 is not Asymmetric since R1 is neither Antisymmetric nor Irreflexive.
- R2 is Asymmetric since R2 is both Antisymmetric and Irreflexive.
- R3 is not Asymmetric since it is Antisymmetric but not irreflexive.

Activity Time

Consider the following relations on {1, 2, 3, 4}:

$R1 = \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 4), (4, 1), (4, 4)\}$,
$R2 = \{(1, 1), (1, 2), (2, 1)\}$,
$R3 = \{(1, 1), (1, 2), (1, 4), (2, 1), (2, 2), (3, 3), (4, 1), (4, 4)\}$,
$R4 = \{(2, 1), (3, 1), (3, 2), (4, 1), (4, 2), (4, 3)\}$,
$R5 = \{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4)\}$,
$R6 = \{(3, 4)\}$.

Determine which of these relation are Reflexive, Symmetric, Transitive, Antisymmetric, Irreflexive and Asymmetric.

Combining Relations

As R is a subsets of A x B, the set operations can be applied

- Union (U)
- Intersection (∩)
- Difference (-)
- Symmetric Complement (⊕)

Given two relations R_1 and R_2, we can combine them using basic set operations to form new relations such as $R_1 \cup R_2$, $R_1 \cap R_2$, $R_1 - R_2$, $R_2 - R_1$ and $R_1 \oplus R_2$.

Combining Relations

Given, $A = \{1,2,3\}$, $B = \{1,2,3,4\}$

$R_1 = \{(1,1),(2,2),(3,3)\}$,

$R_2 = \{(1,1),(1,2),(1,3),(1,4)\}$

- $R_1 \cup R_2 = \{(1,1),(1,2),(1,3),(1,4),(2,2),(3,3)\}$
- $R_1 \cap R_2 = \{(1,1)\}$
- $R_1 - R_2 = \{(2,2),(3,3)\}$
- $R_2 - R_1 = \{(1,2),(1,3),(1,4)\}$
- $R_1 \oplus R_2 = \{(1,2),(1,3),(1,4),(2,2),(3,3)\}$

Composition of Relations

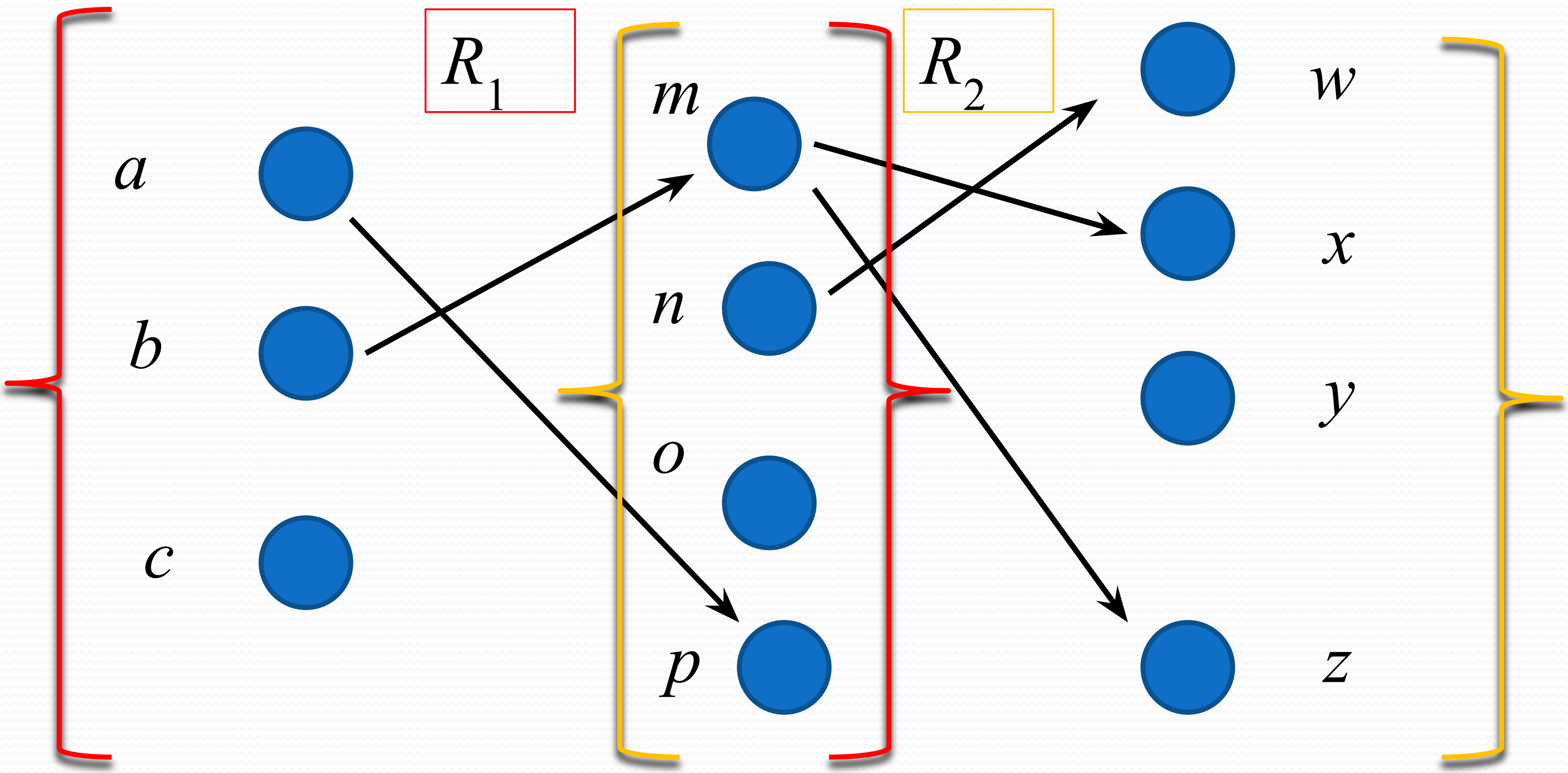
Definition: Suppose

- R_1 is a relation from a set A to a set B.
- R_2 is a relation from B to a set C.

Then the *composition* (or *composite*) of R_2 with R_1, is a relation from A to C where

- if (x,y) is a member of R_1 and (y,z) is a member of R_2, then (x,z) is a member of $R_2 \circ R_1$.

Representing the Composition of a Relation



$$R_1 \circ R_2 = \{(b,D),(b,B)\}$$

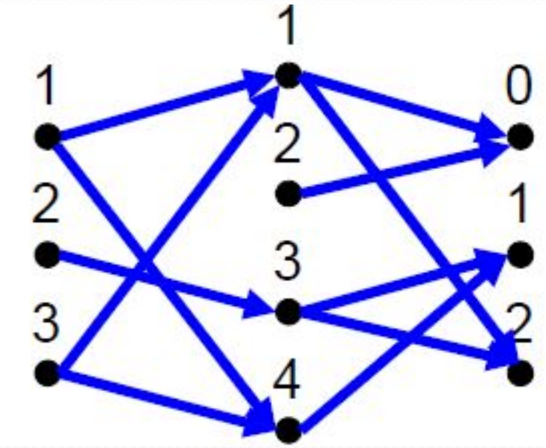
Composition of Relations

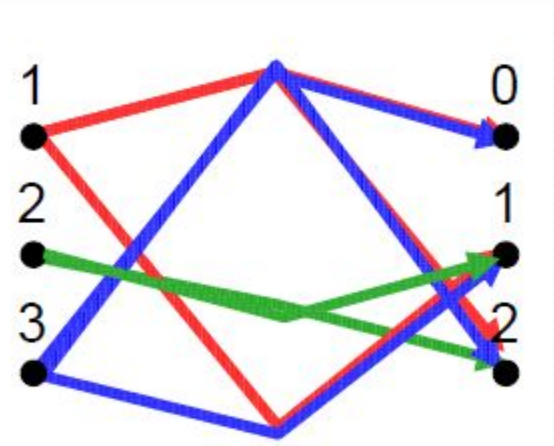
What is the composite of the relations R and S, where

- R is the relation from **{1,2,3}** to **{1,2,3,4}** with

  R = {(1,1),(1,4),(2,3),(3,1),(3,4)}
- S is the relation from **{1,2,3,4}** to **{0,1,2}** with

  S = {(1,0),(1,2),(2,0),(3,1),(3,2),(4,1)}?
- S o R = {(1,0),(1,2),(1,1), (2,2),(2,1), (3,0),(3,2),(3,1)}





INVERSE OF A RELATION

- Let R be a relation from A to B. The inverse relation R^{-1} from B to A is defined as:

$$R^{-1} = \{(b,a) \in B \times A \mid (a,b) \in R\}$$

More simply, the inverse relation R^{-1} of R is obtained by interchanging the elements of all the ordered pairs in R.

- **Example**

X = {a, b, c} and Y={1, 2}

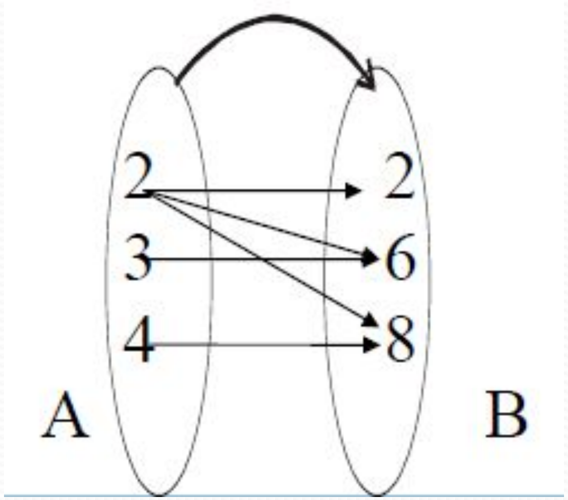
R = {(a, 1), (b, 2), (c, 1)}

- R^{-1} = {(1, a), (2, b), (1, c)}

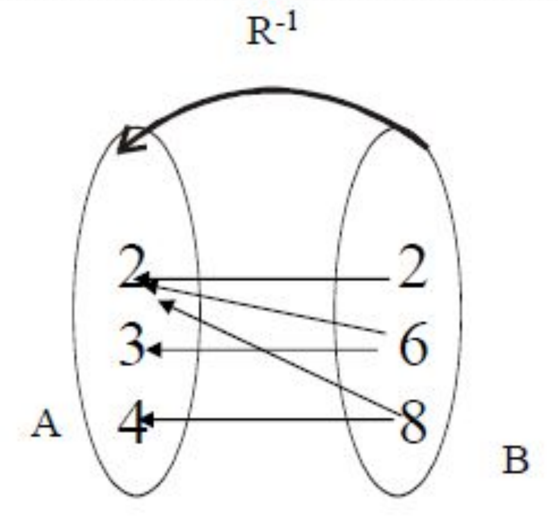
INVERSE OF A RELATION

The relation

R = {(2,2), (2,6), (2,8), (3,6), (4,8)} is represented by the arrow diagram.



Then inverse of the above relation can be obtained simply changing the directions of the arrows and hence the diagram is



Equivalence Relations

Equivalence Relations

Definition 1: A relation on a set *A* is called an *equivalence relation* if it is reflexive, symmetric, and transitive.

Definition 2: Two elements *a*, and *b* that are related by an equivalence relation are called *equivalent.* The notation $a \sim b$ is often used to denote that *a* and *b* are equivalent elements with respect to a particular equivalence relation.

Strings

Example:

Suppose that R is the relation on the set of strings of English letters such that aRb if and only if $l(a) = l(b)$, where $l(x)$ is the length of the string x. Is R an equivalence relation?

Solution: Show that all of the properties of an equivalence relation hold.

- *Reflexivity*: Because $l(a) = l(a)$, it follows that aRa for all strings a.
- *Symmetry*: Suppose that aRb. Since $l(a) = l(b)$, $l(b) = l(a)$ also holds and bRa.
- *Transitivity*: Suppose that aRb and bRc. Since $l(a) = l(b)$, and $l(b) = l(c)$, $l(a) = l(a)$ also holds and aRc.

Congruence Modulo *m*

Example: Let m be an integer with $m > 1$. Show that the relation

$$R = \{(a,b) \mid a \equiv b \pmod{m}\}$$

is an equivalence relation on the set of integers.

Solution: Recall that $a \equiv b \pmod{m}$ if and only if m divides $a - b$.

- *Reflexivity*: $a \equiv a \pmod{m}$ since $a - a = 0$ is divisible by m since $0 = 0 \cdot m$.
- *Symmetry*: Suppose that $a \equiv b \pmod{m}$. Then $a - b$ is divisible by m, and so $a - b = km$, where k is an integer. It follows that $b - a = (-k)\, m$, so $b \equiv a \pmod{m}$.
- *Transitivity*: Suppose that $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$. Then m divides both $a - b$ and $b - c$. Hence, there are integers k and l with $a - b = km$ and $b - c = lm$. We obtain by adding the equations:

$$a - c = (a - b) + (b - c) = km + lm = (k + l)\, m.$$

Therefore, $a \equiv c \pmod{m}$.

Divides

Example: Show that the "divides" relation on the set of positive integers is not an equivalence relation.

Solution: The properties of reflexivity, and transitivity do hold, but there relation is not transitive. Hence, "divides" is not an equivalence relation.

- *Reflexivity*: $a \mid a$ for all a.
- *Not Symmetric*: For example, $2 \mid 4$, but $4 \nmid 2$. Hence, the relation is not symmetric.
- *Transitivity*: Suppose that a divides b and b divides c. Then there are positive integers k and l such that $b = ak$ and $c = bl$. Hence, $c = a(kl)$, so a divides c. Therefore, the relation is transitive.

Partial Orderings

Partial Orderings

Definition 1: A relation *R* on a set S is called a *partial ordering,* or *partial order,* if it is reflexive, antisymmetric, and transitive.

A set together with a partial ordering *R* is called a *partially ordered set*, or *poset*, and is denoted by (*S*, *R*). Members of *S* are called *elements* of the poset.

Partial Orderings (*continued*)

Example 1: Show that the "greater than or equal" relation ($\geq$) is a partial ordering on the set of integers.

- *Reflexivity*: $a \geq a$ for every integer a.
- *Antisymmetry*: If $a \geq b$ and $b \geq a$, then $a = b$.
- *Transitivity*: If $a \geq b$ and $b \geq c$, then $a \geq c$.

These properties all follow from the order axioms for the integers. (*See Appendix* 1).

Partial Orderings (*continued*)

Example 2: Show that the divisibility relation ($|$) is a partial ordering on the set of integers.

- *Reflexivity*: $a \mid a$ for all integers a. (*see Example* 9 *in Section* 9.1)
- *Antisymmetry*: If a and b are positive integers with $a \mid b$ and $b \mid a$, then $a = b$. (*see Example* 12 *in Section* 9.1)
- *Transitivity*: Suppose that a divides b and b divides c. Then there are positive integers k and l such that $b = ak$ and $c = bl$. Hence, $c = a(kl)$, so a divides c. Therefore, the relation is transitive.

- $(\boldsymbol{Z}^+, |)$ is a poset.

Partial Orderings (*continued*)

Example 3: Show that the inclusion relation ($\subseteq$) is a partial ordering on the power set of a set S.

- *Reflexivity*: $A \subseteq A$ whenever A is a subset of S.
- *Antisymmetry*: If A and B are positive integers with $A \subseteq B$ and $B \subseteq A$, then $A = B$.
- *Transitivity*: If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$.

The properties all follow from the definition of set inclusion.

Sequences and Series(Sums)

Section Summary

- Sequences.
  - Examples: Geometric Progression, Arithmetic Progression
- Recurrence Relations
  - Example: Fibonacci Sequence
- Summations

Introduction

- Sequences are ordered lists of elements.
- EXAMPLES:
  - 1, 2, 3, 5, 8
  - 1, 3, 9, 27, 81, …….
  - 1, 2, 3, 4, 5, …
  - 4, 8, 12, 16, 20,…
  - 2, 4, 8, 16, 32, …
  - 1, 1/2, 1/3, 1/4, 1/5, …
  - 1, 4, 9, 16, 25, …
  - 1, -1, 1, -1, 1, -1, …
- Sequences arise throughout mathematics, computer science, and in many other disciplines, ranging from botany to music.
- We will introduce the terminology to represent sequences and sums of the terms in the sequences.

SEQUENCES IN COMPUTER PROGRAMMING:

- An important data type in computer programming consists of finite sequences known as one-dimensional arrays; a single variable in which a sequence of variables may be stored.

EXAMPLE:

- The names of k students in a class may be represented by an array of k elements “name” as:

  name [0], name[1], name[2], …, name[k-1]

Sequences

Definition: A *sequence* is a function from a subset of the integers (usually either the set {0, 1, 2, 3, 4, …..} or {1, 2, 3, 4, ….}) to a set *S*.

- The notation a_n is used to denote the image of the integer *n*. We can think of a_n as the equivalent of *f(n)* where *f* is a function from {0,1,2,…..} to *S*. We call a_n a *term* of the sequence.

OR

A sequence is just a list of elements usually written in a row.

Sequences

Example: Consider the sequence $\{a_n\}$ where

$$a_n = \frac{1}{n} \qquad \{a_n\} = \{a_1, a_2, a_3, \ldots\}$$

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4} \ldots$$

Sequences

- **EXAMPLE:**

Write the first four terms of the sequence defined by the formula: $b_j = 1 + 2^j$, for all integers $j \geq 0$

SOLUTION:

- $b_0 = 1 + 2^0 = 1 + 1 = 2$
- $b_1 = 1 + 2^1 = 1 + 2 = 3$
- $b_2 = 1 + 2^2 = 1 + 4 = 5$
- $b_3 = 1 + 2^3 = 1 + 8 = 9$

REMARK:

The formula $b_j = 1 + 2^j$, for all integers $j \geq 0$ defines an infinite sequence having infinite number of values.

Sequences

- **EXERCISE:**

Compute the first six terms of the sequence defined by the formula

- Cn = 1+ $(-1)^n$ for all integers n ≥ 0

SOLUTION :

- C0 = 1 + $(-1)^0$ = 1 + 1 = 2
- C1 = 1 + $(-1)^1$ = 1 + (-1) = 0
- C2 = 1 + $(-1)^2$ = 1 + 1 = 2
- C3 = 1 + $(-1)^3$ = 1 + (-1) = 0
- C4 = 1 + $(-1)^4$ = 1 + 1 = 2
- C5 = 1 + $(-1)^5$ = 1 + (-1) = 0

REMARK:

1) If n is even, then Cn = 2 and if n is odd, then Cn = 0. Hence, the sequence oscillates endlessly between 2 and 0.

2) An infinite sequence may have only a finite number of values.

Sequences

EXAMPLE:

Write the first four terms of the sequence defined by

$$C_n = \frac{(-1)^n n}{n+1} \quad \text{for all integers } n \geq 1$$

SOLUTION:

$$C_1 = \frac{(-1)^1(1)}{1+1} = \frac{-1}{2}, C_2 = \frac{(-1)^2(2)}{2+1} = \frac{2}{3}, C_3 = \frac{(-1)^3(3)}{3+1} = \frac{-3}{4}$$

$$And \quad fourth \quad term \ is C_4 = \frac{(-1)^4(4)}{4+1} = \frac{4}{5}$$

REMARK: A sequence whose terms alternate in sign is called an alternating sequence.

Sequences

Find explicit formulas for sequences with the initial terms given:

1) 0, 1, -2, 3, -4, 5, …

SOLUTION:

$a_n = (-1)^{n+1} n$ for all integers $n \geq 0$

2) $1 - \frac{1}{2}, \frac{1}{2} - \frac{1}{3}, \frac{1}{3} - \frac{1}{4}, \frac{1}{4} - \frac{1}{5}, \cdots$

SOLUTION:

$$b_k = \frac{1}{k} - \frac{1}{k+1} \quad \text{for all integers } n \geq 1$$

Sequences

3) 2, 6, 12, 20, 30, 42, 56, …

SOLUTION:

$C_n = n\,(n + 1)$ for all integers $n \geq 1$

4) 1/4, 2/9, 3/16, 4/25, 5/36, 6/49, …

SOLUTION:

$$d_i = \frac{i}{(i+1)^2} \quad \text{for all integers} \quad i \geq 1$$

OR

$$d_j = \frac{j+1}{(j+2)^2} \quad \text{for all integers} \quad j \geq 0$$

Arithmetic Progression OR Sequences

- A sequence in which every term after the first is obtained from the preceding term by adding a constant number is called an arithmetic sequence or arithmetic progression (A.P.)
- The constant number, being the difference of any two consecutive terms is called the common difference of A.P., commonly denoted by “d”.

EXAMPLES:

1. 5, 9, 13, 17, … (common difference = 4)
2. 0, -5, -10, -15, … (common difference = -5)
3. $x + a$, $x + 3a$, $x + 5a$, … (common difference = $2a$)

Arithmetic Sequences

GENERAL TERM OF AN ARITHMETIC SEQUENCE:

Let **a** be the first term and **d** be the common difference of an arithmetic sequence. Then the sequence is:

$$a, a+d, a+2d, a+3d, \ldots$$

If a_i, for $i \geq 1$, represents the terms of the sequence then

a_1 = first term = $a = a + (1-1)\,d$

a_2 = second term = $a + d = a + (2-1)\,d$

a_3 = third term = $a + 2d = a + (3-1)\,d$

By symmetry

a_n = nth term = $a + (n-1)d$ for all integers $n \geq 1$.

Arithmetic Sequences

Examples:

1. Let $a = -1$ and $d = 4$:

[redacted]

2. Let $a = 7$ and $d = -3$:

$$\{t_n\} = \{t_0, t_1, t_2, t_3, t_4, \ldots\} = \{7, 4, 1, -2, -5, \ldots\}$$

3. Let $a = 1$ and d = 2:

$$\{u_n\} = \{u_0, u_1, u_2, u_3, u_4, \ldots\} = \{1, 3, 5, 7, 9, \ldots\}$$

Arithmetic Sequences

EXAMPLE:

Find the 20th term of the arithmetic sequence

3, 9, 15, 21, …

SOLUTION:

- Here a = first term = 3
- d = common difference = 9 - 3 = 6
- n = term number = 20
- a_{20} = value of 20th term = ?
- Since $a_n = a + (n - 1)\,d;\ n \geq 1$

$\therefore a_{20} = 3 + (20 - 1)\,6$

$= 3 + 114$

$= 117$

Arithmetic Sequences

EXERCISE:

Find the 36th term of the arithmetic sequence whose 3rd term is 7 and 8th term is 17.

SOLUTION:

Let **a** be the first term and **d** be the common difference of the arithmetic sequence.

Then $a_n = a + (n - 1)d \quad n \geq 1$

$\Rightarrow a_3 = a + (3 - 1)d$ and $a_8 = a + (8 - 1)d$

Given that $a_3 = 7$ and $a_8 = 17$. Therefore

$7 = a + 2d$……………………….(1) and $17 = a + 7d$……………………...(2)

Subtracting (1) from (2), we get,

$10 = 5d \quad \Rightarrow d = 2$

Substituting $d = 2$ in (1) we have

$7 = a + 2(2)$ which gives $a = 3$

Arithmetic Sequences

Thus, $a_n = a + (n - 1) d$

$a_n = 3 + (n - 1) 2$ (using values of a and d)

Hence the value of 36th term is

$$a_{36} = 3 + (36 - 1) 2$$

$$= 3 + 70$$

$$a_{36} = 73$$

Geometric Progression OR Sequence

- A sequence in which every term after the first is obtained from the preceding term by multiplying it with a constant number is called a geometric sequence or geometric progression (G.P.)
- The constant number, being the ratio of any two consecutive terms is called the common ratio of the G.P. commonly denoted by "r".
- **EXAMPLE:**

1. 1, 2, 4, 8, 16, … (common ratio = 2)
2. 3, - 3/2, 3/4, - 3/8, … (common ratio = - 1/2)
3. 0.1, 0.01, 0.001, 0.0001, … (common ratio = 0.1 = 1/10)

GENERAL TERM OF A GEOMETRIC SEQUENCE:

Let **a** be the first tem and **r** be the common ratio of a geometric sequence. Then the sequence is a, ar, ar^2, ar^3, …

If a_i, for $i \geq 1$ represent the terms of the sequence, then

a_1 = first term = a = ar^{1-1}

a_2 = second term = ar = ar^{2-1}

a_3 = third term = ar^2 = ar^{3-1}

………………

………………

a_n = nth term = ar^{n-1}; for all integers $n \geq 1$

Geometric Progression

Examples:

1. Let $a = 1$ and $r = -1$. Then:

[redacted]

2. Let $a = 2$ and $r = 5$. Then:

[redacted]

3. Let $a = 6$ and $r = 1/3$. Then:

$$\{d_n\} = \{d_0, d_1, d_2, d_3, d_4, \ldots\} = \{6, 2, \frac{2}{3}, \frac{2}{9}, \frac{2}{27}, \ldots\}$$

Geometric Sequence

EXAMPLE:

Find the 8th term of the following geometric sequence

$$4, 12, 36, 108, \dots$$

SOLUTION:

Here a = first term = 4

r = common ratio = $\frac{12}{4} = 3$

n = term number = 8

a_8 = value of 8th term = ?

Since $a_n = ar^{n-1}; \quad n \geq 1$

$$\Rightarrow \quad a_8 = (4)(3)^{8-1}$$

$$= 4\,(2187)$$

$$= 8748$$

Geometric Sequence

EXERCISE:

Write the geometric sequence with positive terms whose second term is 9 and fourth term is 1.

SOLUTION:

Let **a** be the first term and **r** be the common ratio of the geometric sequence. Then

$$a_n = ar^{n-1} \qquad n \geq 1$$

Now $\quad a_2 = ar^{2-1}$

$\Rightarrow \quad 9 = ar$ ....(1)

Also $\quad a_4 = ar^{4-1}$

$1 = ar^3$ ....(2)

Dividing (2) by (1), we get,

$$\frac{1}{9} = \frac{ar^3}{ar}$$

$$\Rightarrow \quad \frac{1}{9} = r^2$$

$$\Rightarrow \quad r = \frac{1}{3} \qquad \left(\text{rejecting } r = -\frac{1}{3}\right)$$

Substituting r = 1/3 in (1), we get

$$9 = a\left(\frac{1}{3}\right)$$

$$\Rightarrow \quad a = 9 \times 3 = 27$$

Hence the geometric sequence is

27, 9, 3, 1, 1/3, 1/9, …

Useful Sequences

TABLE 1 **Some Useful Sequences.**

nth Term	*First 10 Terms*
n^2	1, 4, 9, 16, 25, 36, 49, 64, 81, 100, ...
n^3	1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, ...
n^4	1, 16, 81, 256, 625, 1296, 2401, 4096, 6561, 10000, ...
2^n	2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, ...
3^n	3, 9, 27, 81, 243, 729, 2187, 6561, 19683, 59049, ...
$n!$	1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, ...
f_n	1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

SERIES

The sum of the terms of a sequence forms a series. If $a_1, a_2, a_3, \ldots$ represent a sequence of numbers, then the corresponding series is

$$a_1 + a_2 + a_3 + \ldots = \sum_{k=1}^{\infty} a_k$$

SUMMATIONS

SUMMATION NOTATION:

The capital Greek letter sigma $\sum$ is used to write a sum in a short hand notation.

where k varies from 1 to n represents the sum given in expanded form by

$= a_1 + a_2 + a_3 + \ldots + a_n$

More generally if m and n are integers and $m \leq n$, then the summation from k equal m to n of a_k is

$$\sum_{k=m}^{n} a_k = a_m + a_{m+1} + a_{m+2} + \cdots + a_n$$

Here **k** is called the index of the summation; **m** the lower limit of the summation and **n** the upper limit of the summation.

SUMMATIONS

COMPUTING SUMMATIONS:

Let $a_0 = 2$, $a_1 = 3$, $a_2 = -2$, $a_3 = 1$ and $a_4 = 0$.Compute each of the summations:

(a) $\sum_{i=0}^{4} a_i$ (b) $\sum_{j=0}^{2} a_{2j}$ (c) $\sum_{k=1}^{1} a_k$

SOLUTION:

(a) $\sum_{i=0}^{4} a_i = a_0 + a_1 + a_2 + a_3 + a_4$
$= 2 + 3 + (-2) + 1 + 0 = 4$

(b) $\sum_{j=0}^{2} a_{2j} = a_0 + a_2 + a_4$
$= 2 + (-2) + 0 = 0$

(c) $\sum_{k=1}^{1} a_k = a_1$
$= 3$

ARITHMETIC SERIES:

The sum of the terms of an arithmetic sequence forms an arithmetic series (A.S). For example

$$1 + 3 + 5 + 7 + \ldots$$

is an arithmetic series of positive odd integers.

In general, if a is the first term and d the common difference of an arithmetic series, then the series is given as: $a + (a+d) + (a+2d) + \ldots$

SUM OF n TERMS OF AN ARITHMETIC SERIES:

Let a be the first term and d be the common difference of an arithmetic series. Then its nth term is:

$$a_n = a + (n - 1)d; \quad n \geq 1$$

If S_n denotes the sum of first n terms of the A.S, then

$$S_n = a + (a + d) + (a + 2d) + \ldots + [a + (n\text{-}1)\, d]$$
$$= a + (a+d) + (a + 2d) + \ldots + a_n$$
$$= a + (a+d) + (a + 2d) + \ldots + (a_n - d) + a_n \ldots\ldots\ldots(1)$$

where $a_n = a + (n - 1)\, d$

Rewriting the terms in the series in reverse order,

$$Sn = a_n + (a_n - d) + (a_n - 2d) + \ldots + (a + d) + a \ldots\ldots\ldots\ldots(2)$$

Adding (1) and (2) term by term, gives

$$2\, S_n = (a + a_n) + (a + a_n) + (a + a_n) + \ldots + (a + a_n) \qquad (n \text{ terms})$$
$$2\, S_n = n\,(a + a_n)$$
$$\Rightarrow \qquad S_n = n(a + a_n)/2$$
$$S_n = n(a + l)/2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

Where $\qquad l = an = a + (n - 1)d$

Therefore

$$S_n = n/2\, [a + a + (n - 1)\, d]$$
$$S_n \; n/2\, [2\, a + (n - 1)\, d] \ldots\ldots\ldots\ldots(4)$$

ARITHMETIC SERIES:

EXERCISE:

Find the sum of first n natural numbers.

SOLUTION:

Let $S_n = 1 + 2 + 3 + \ldots + n$

Clearly the right hand side forms an arithmetic series with

$a = 1, \quad d = 2 - 1 = 1 \qquad \text{and} \quad n = n$

$$\begin{aligned} \therefore \qquad S_n &= \frac{n}{2}[2a + (n-1)d] \\ &= \frac{n}{2}[2(1) + (n-1)(1)] \\ &= \frac{n}{2}[2 + n - 1] \\ &= \frac{n(n+1)}{2} \end{aligned}$$

GEOMETRIC SERIES:

The sum of the terms of a geometric sequence forms a geometric series (G.S.). For example

$$1 + 2 + 4 + 8 + 16 + \dots$$

is geometric series.

In general, if **a** is the first term and **r** the common ratio of a geometric series, then the series is given as: $a + ar + ar^2 + ar^3 + \dots$

Strings

Definition: A *string* is a finite sequence of characters from a finite set (an alphabet).

- Sequences of characters or bits are important in computer science.
- The *empty string* is represented by λ.
- The string *abcde* has *length* 5.

Recurrence Relations

Definition: A *recurrence relation* for the sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more of the previous terms of the sequence, namely, $a_0, a_1, \ldots, a_{n-1}$, for all integers n with $n \geq n_0$, where n_0 is a nonnegative integer.

- A sequence is called a *solution* of a recurrence relation if its terms satisfy the recurrence relation.
- The *initial conditions* for a sequence specify the terms that precede the first term where the recurrence relation takes effect.

Questions about Recurrence Relations

Example 1: Let $\{a_n\}$ be a sequence that satisfies the recurrence relation $a_n = a_{n-1} + 3$ for $n = 1,2,3,4,\ldots.$ and suppose that $a_0 = 2$. What are a_1, a_2 and a_3?

[Here $a_0 = 2$ is the initial condition.]

Solution: We see from the recurrence relation that

$$a_1 = a_0 + 3 = 2 + 3 = 5$$
$$a_2 = 5 + 3 = 8$$
$$a_3 = 8 + 3 = 11$$

Questions about Recurrence Relations

Example 2: Let $\{a_n\}$ be a sequence that satisfies the recurrence relation $a_n = a_{n-1} - a_{n-2}$ for $n = 2,3,4,\ldots.$ and suppose that $a_0 = 3$ and $a_1 = 5$. What are a_2 and a_3?
[Here the initial conditions are $a_0 = 3$ and $a_1 = 5$.]

Solution: We see from the recurrence relation that

$$a_2 = a_1 - a_0 = 5 - 3 = 2$$
$$a_3 = a_2 - a_1 = 2 - 5 = -3$$

Solving Recurrence Relations

- Finding a formula for the *n*th term of the sequence generated by a recurrence relation is called *solving the recurrence relation*.
- Such a formula is called a *closed formula*.
- Various methods for solving recurrence relations will be covered in Chapter 8 where recurrence relations will be studied in greater depth.
- Here we illustrate by example the method of iteration in which we need to guess the formula. The guess can be proved correct by the method of induction (Chapter 5).

Fibonacci Sequence

Definition: Define the *Fibonacci sequence*, $f_0, f_1, f_2, \ldots$, by:

- Initial Conditions: $f_0 = 0, f_1 = 1$
- Recurrence Relation: $f_n = f_{n-1} + f_{n-2}$

Example: Find f_2, f_3, f_4, f_5 and f_6 .

Answer:

$$f_2 = f_1 + f_0 = 1 + 0 = 1,$$
$$f_3 = f_2 + f_1 = 1 + 1 = 2,$$
$$f_4 = f_3 + f_2 = 2 + 1 = 3,$$
$$f_5 = f_4 + f_3 = 3 + 2 = 5,$$
$$f_6 = f_5 + f_4 = 5 + 3 = 8.$$

Iterative Solution Example

Method 1: Working upward, forward substitution

Let $\{a_n\}$ be a sequence that satisfies the recurrence relation $a_n = a_{n-1} + 3$ for $n = 2,3,4,\ldots.$ and suppose that $a_1 = 2$.

$$a_2 = 2 + 3$$
$$a_3 = (2 + 3) + 3 = 2 + 3 \cdot 2$$
$$a_4 = (2 + 2 \cdot 3) + 3 = 2 + 3 \cdot 3$$
$$\vdots$$
$$a_n = a_{n-1} + 3 = (2 + 3 \cdot (n - 2)) + 3 = 2 + 3(n - 1)$$

Iterative Solution Example

Method 2: Working downward, backward substitution

Let $\{a_n\}$ be a sequence that satisfies the recurrence relation $a_n = a_{n-1} + 3$ for $n = 2,3,4,\ldots.$ and suppose that $a_1 = 2$.

$$
\begin{aligned}
a_n &= a_{n-1} + 3 \\
&= (a_{n-2} + 3) + 3 = a_{n-2} + 3 \cdot 2 \\
&= (a_{n-3} + 3) + 3 \cdot 2 = a_{n-3} + 3 \cdot 3 \\
&\ \ \vdots \\
&= a_2 + 3(n-2) = (a_1 + 3) + 3(n-2) = 2 + 3(n-1)
\end{aligned}
$$

Some Useful Summation Formulae

TABLE 2 Some Useful Summation Formulae.

Sum	*Closed Form*
$\sum_{k=0}^{n} ar^k \ (r \neq 0)$	$\frac{ar^{n+1} - a}{r - 1}, r \neq 1$
$\sum_{k=1}^{n} k$	$\frac{n(n+1)}{2}$
$\sum_{k=1}^{n} k^2$	$\frac{n(n+1)(2n+1)}{6}$
$\sum_{k=1}^{n} k^3$	$\frac{n^2(n+1)^2}{4}$
$\sum_{k=0}^{\infty} x^k, \lvert x \rvert < 1$	$\frac{1}{1-x}$
$\sum_{k=1}^{\infty} kx^{k-1}, \lvert x \rvert < 1$	$\frac{1}{(1-x)^2}$

Geometric Series: We just proved this.

Later we will prove some of these by induction.

Proof in text (requires calculus)

Graphs

Chapter 10

Chapter Summary

- Graphs and Graph Models
- Graph Terminology and Special Types of Graphs
- Representing Graphs and Graph Isomorphism
- Connectivity
- Euler and Hamiltonian Graphs
- Shortest-Path Problems
- Planar Graphs
- Graph Coloring

Graphs and Graph Models

Section 10.1

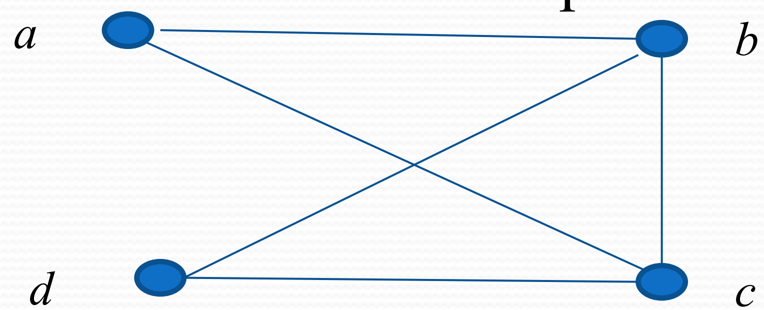
Section Summary

- Introduction to Graphs
- Graph Taxonomy
- Graph Models

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.

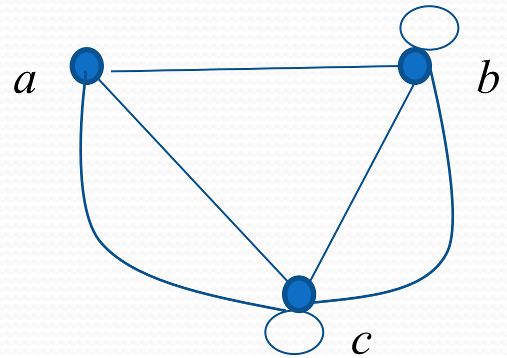
Remarks:

- 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

- 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

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*.

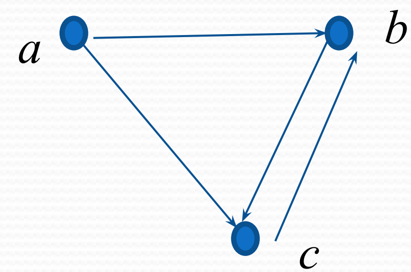
Remark:

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

Some Terminology (*continued*)

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

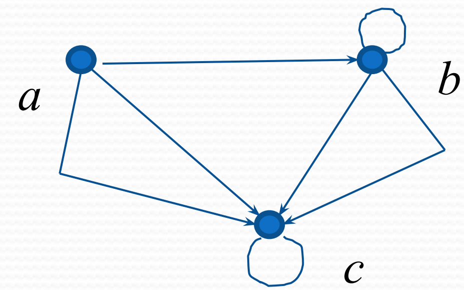
Example:
This is a directed graph with three vertices and four edges.



- 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 Terminology: Summary

- To understand the structure of a graph and to build a graph model, we ask these questions:
  - Are the edges of the graph undirected or directed (or both)?
  - If the edges are undirected, are multiple edges present that connect the same pair of vertices? If the edges are directed, are multiple directed edges present?
  - Are loops present?

TABLE 1 **Graph Terminology.**

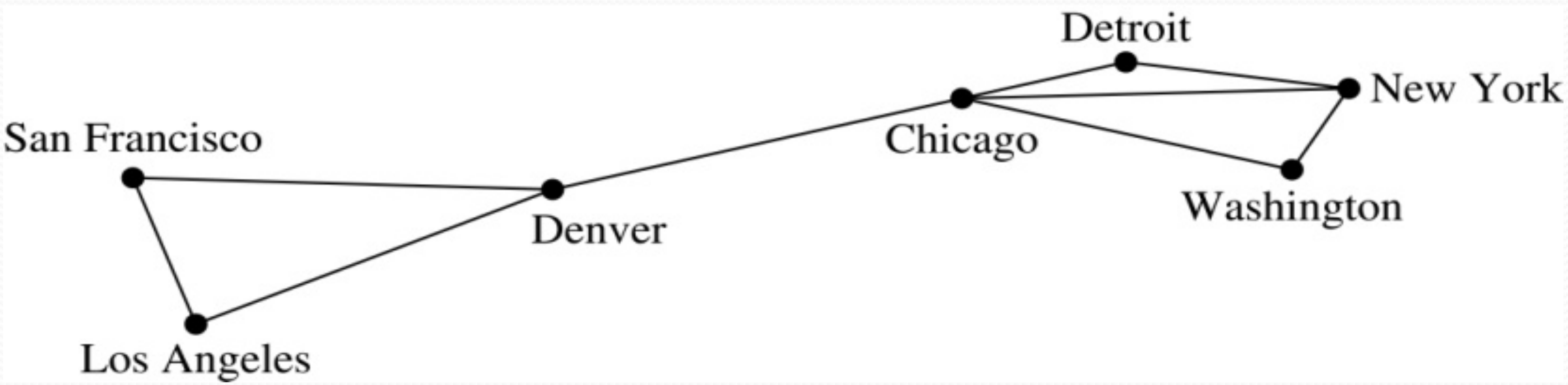
Type	*Edges*	*Multiple Edges Allowed?*	*Loops Allowed?*
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed graph	Directed and undirected	Yes	Yes

Other Applications of Graphs

- We will illustrate how graph theory can be used in models of:
  - Social networks
  - Communications networks
  - Information networks
  - Software design
  - Transportation networks
  - Biological networks
- It's a challenge to find a subject to which graph theory has not yet been applied. Can you find an area without applications of graph theory?

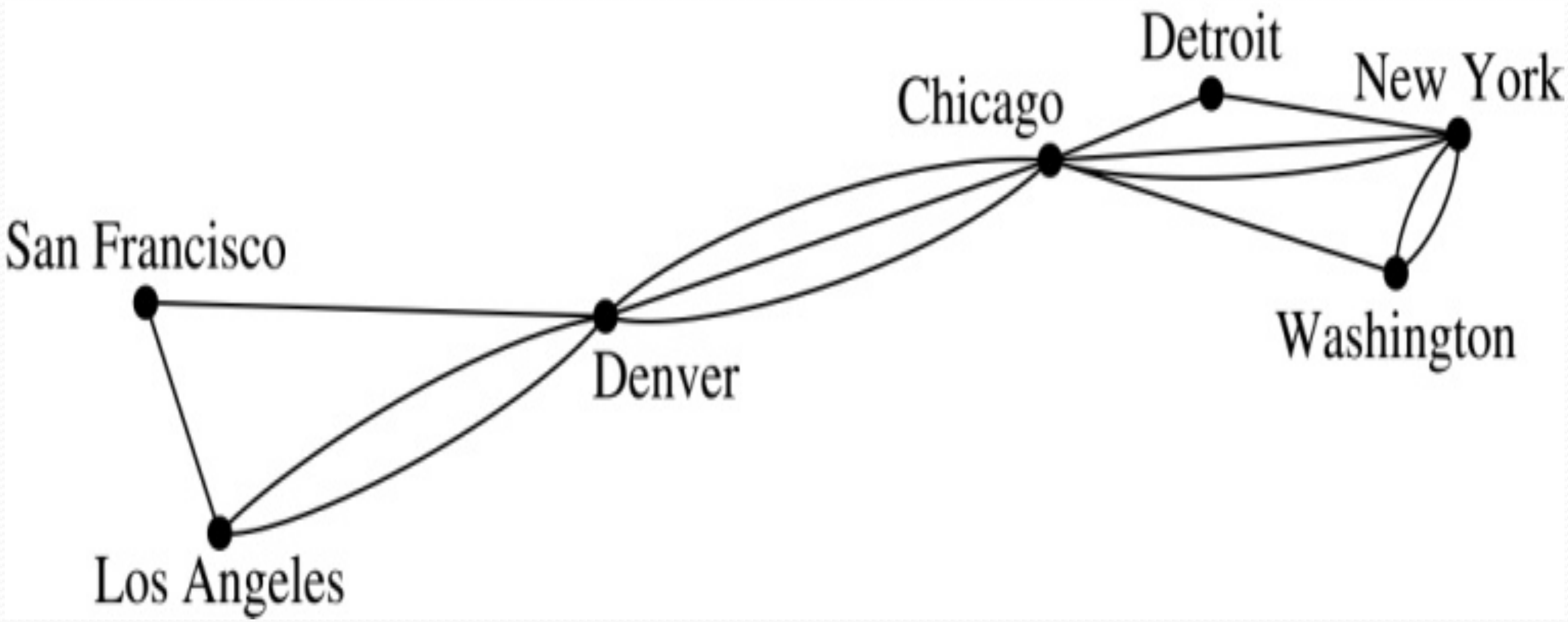
Graph Models: Computer Networks

- When we build a graph model, we use the appropriate type of graph to capture the important features of the application.
- We illustrate this process using graph models of different types of computer networks. In all these graph models, the vertices represent data centers and the edges represent communication links.
- To model a computer network where we are only concerned whether two data centers are connected by a communications link, we use a simple graph. This is the appropriate type of graph when we only care whether two data centers are directly linked (and not how many links there may be) and all communications links work in both directions.



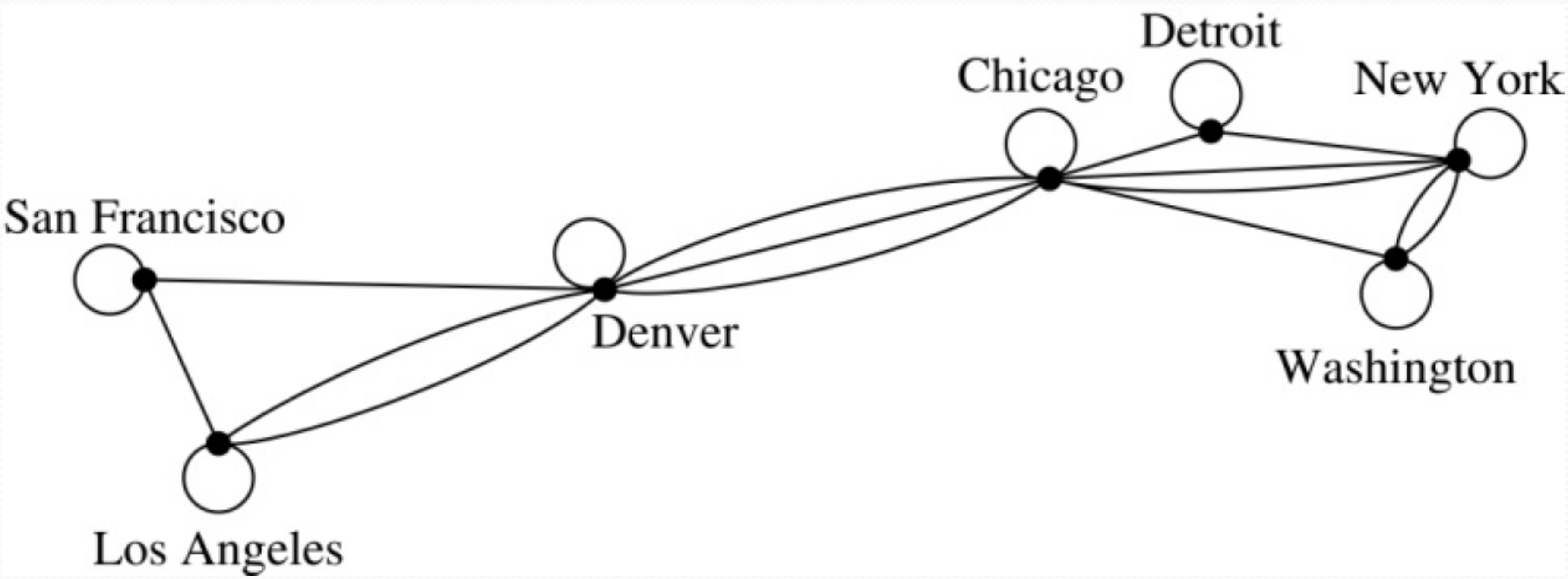
Graph Models: Computer Networks (*continued*)

- To model a computer network where we care about the number of links between data centers, we use a multigraph.



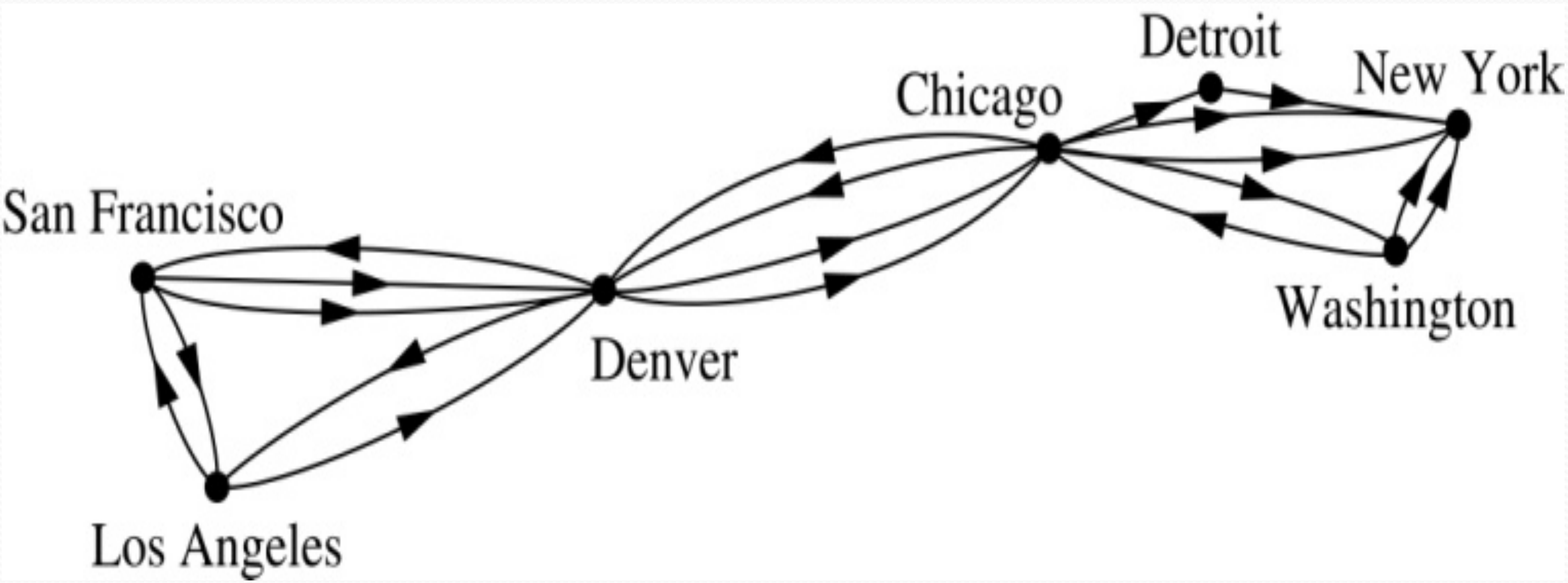
Graph Models: Computer Networks

- To model a computer network with diagnostic links at data centers, we use a pseudograph, as loops are needed.



Graph Models: Computer Networks

- To model a network with multiple one-way links, we use a directed multigraph. Note that we could use a directed graph without multiple edges if we only care whether there is at least one link from a data center to another data center.

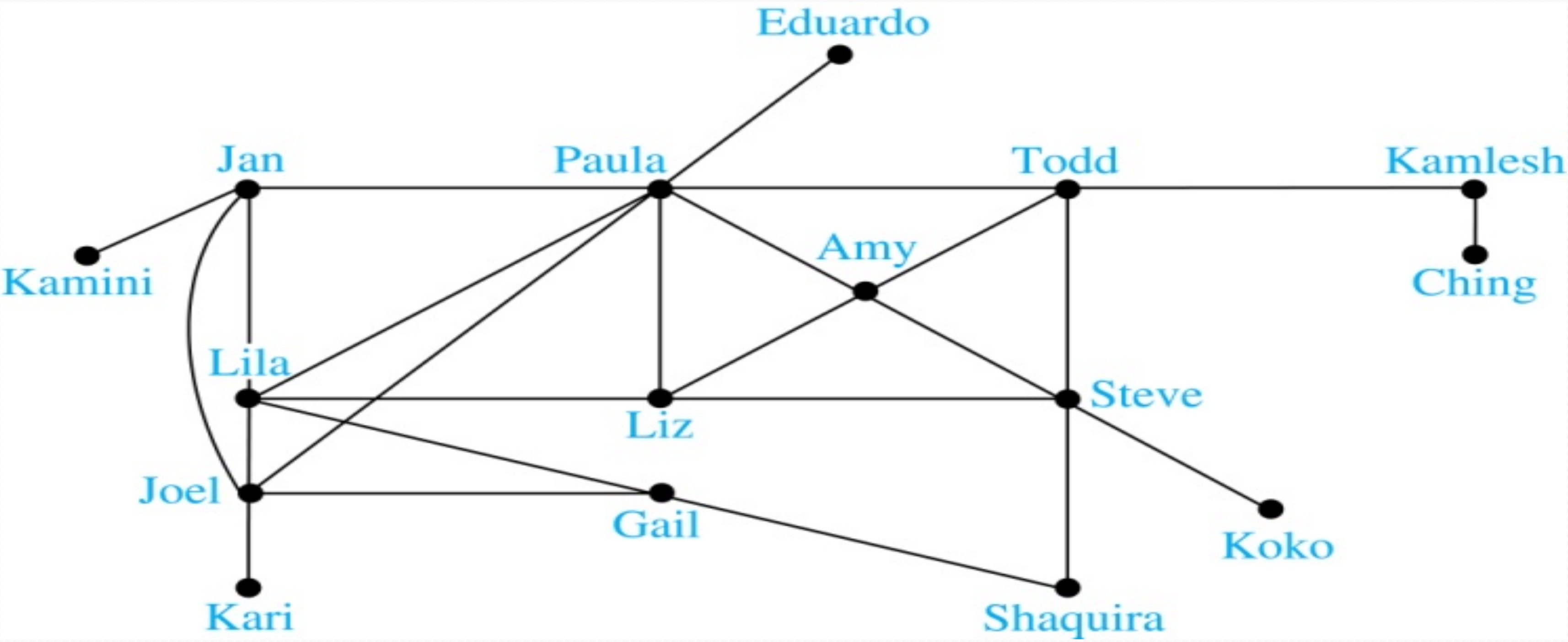


Graph Models: Social Networks

- 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

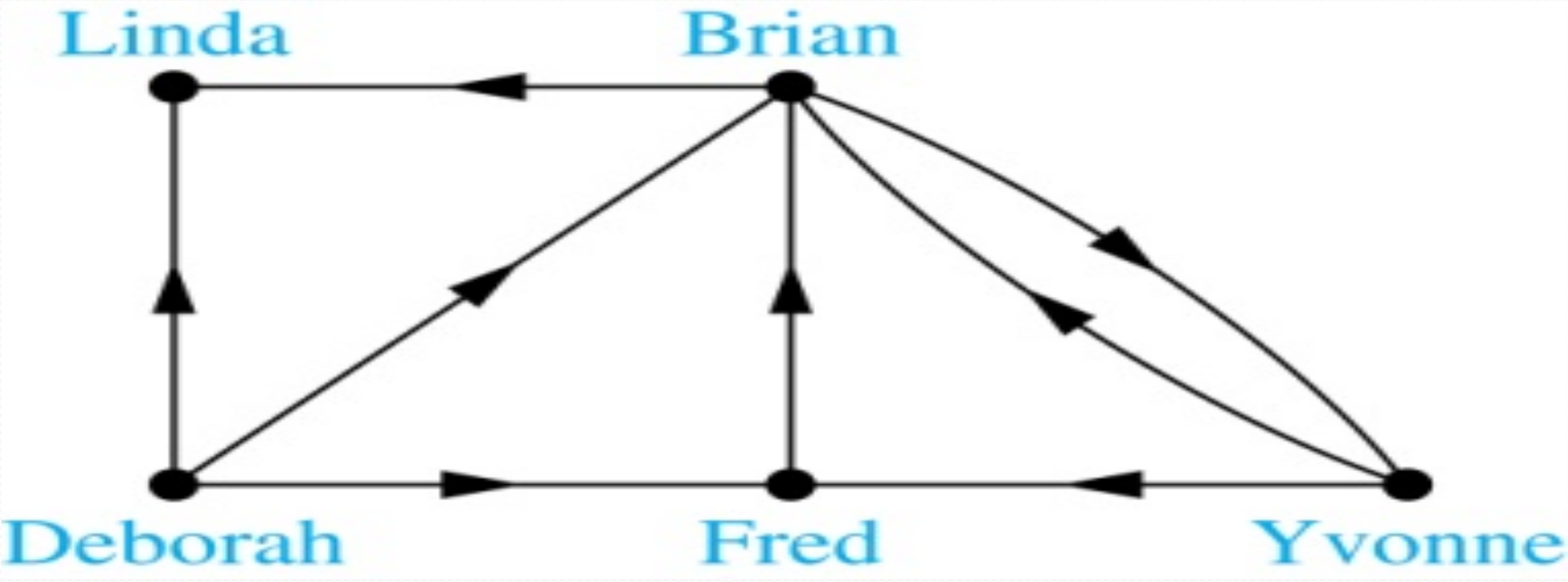
Graph Models: Social Networks

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



Graph Models: Social Networks

Example: An influence graph



Applications to Information Networks

- Graphs can be used to model different types of networks that link different types of information.
- In a *web graph*, web pages are represented by vertices and links are represented by directed edges.
  - A web graph models the web at a particular time.
  - We will explain how the web graph is used by search engines in Section 11.4.
- In a *citation network*:
  - Research papers in a particular discipline are represented by vertices.
  - When a paper cites a second paper as a reference, there is an edge from the vertex representing this paper to the vertex representing the second paper.

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.

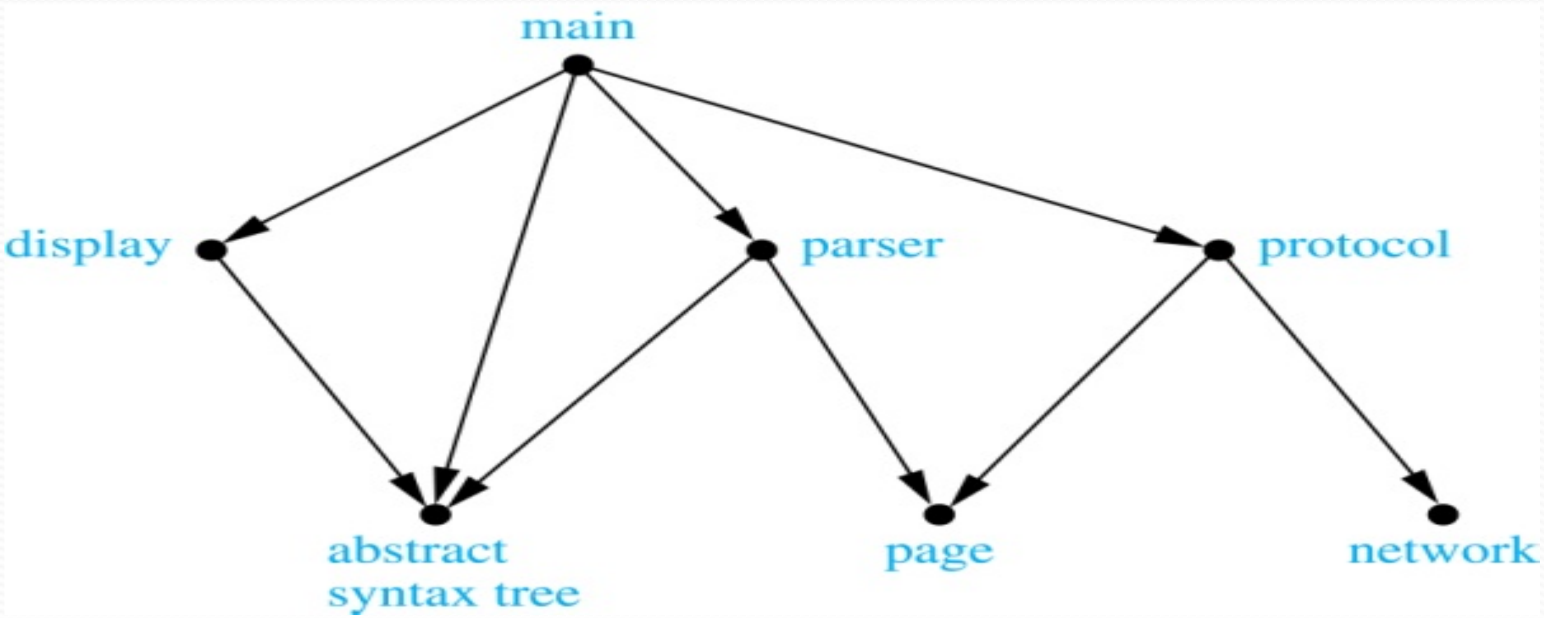
Software Design Applications

- Graph models are extensively used in software design. We will introduce two such models here; one representing the dependency between the modules of a software application and the other representing restrictions in the execution of statements in computer programs.
- When a top-down approach is used to design software, the system is divided into modules, each performing a specific task.
- We use a *module dependency graph* to represent the dependency between these modules. These dependencies need to be understood before coding can be done.

Software Design Applications

- In a module dependency graph vertices represent software modules and there is an edge from one module to another if the second module depends on the first.

Example: The dependencies between the seven modules in the design of a web browser are represented by this module dependency graph.

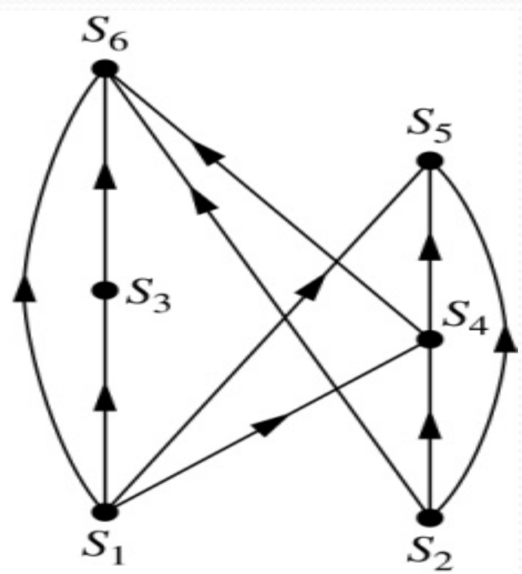


Software Design Applications

- 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.

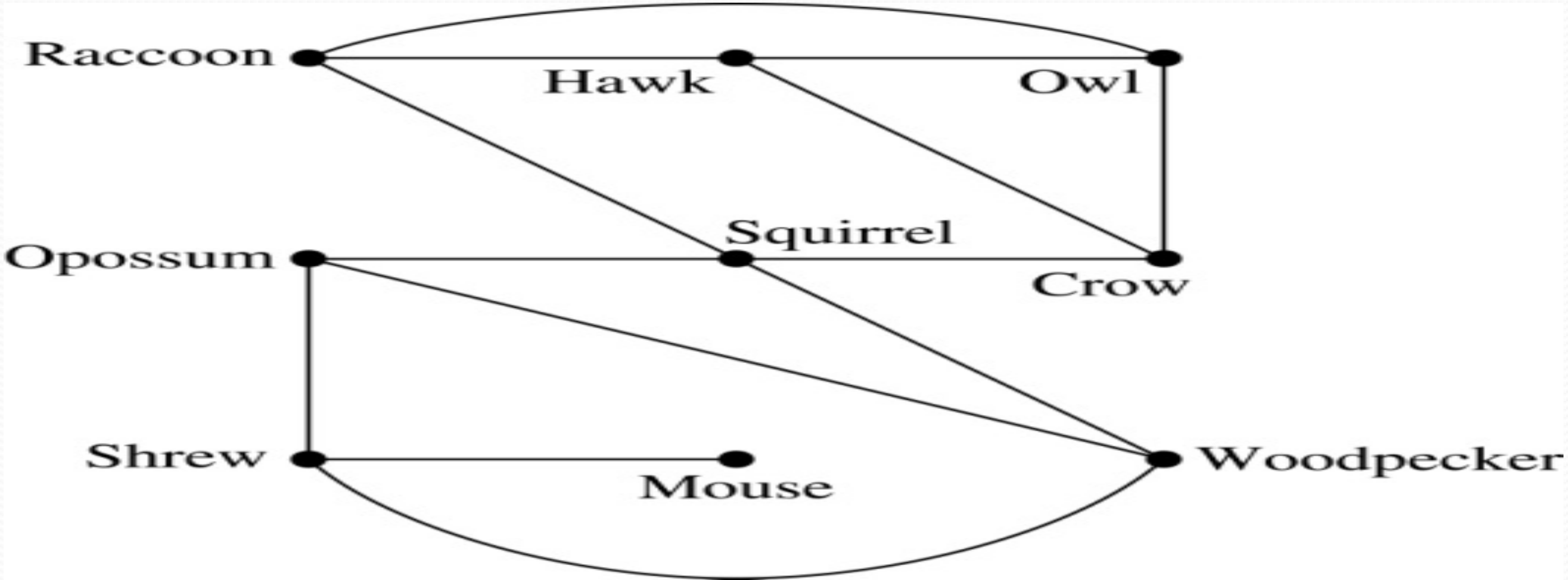
S_1 $a := 0$
S_2 $b := 1$
S_3 $c := a + 1$
S_4 $d := b + a$
S_5 $e := d + 1$
S_6 $e := c + d$



Biological Applications

- 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.

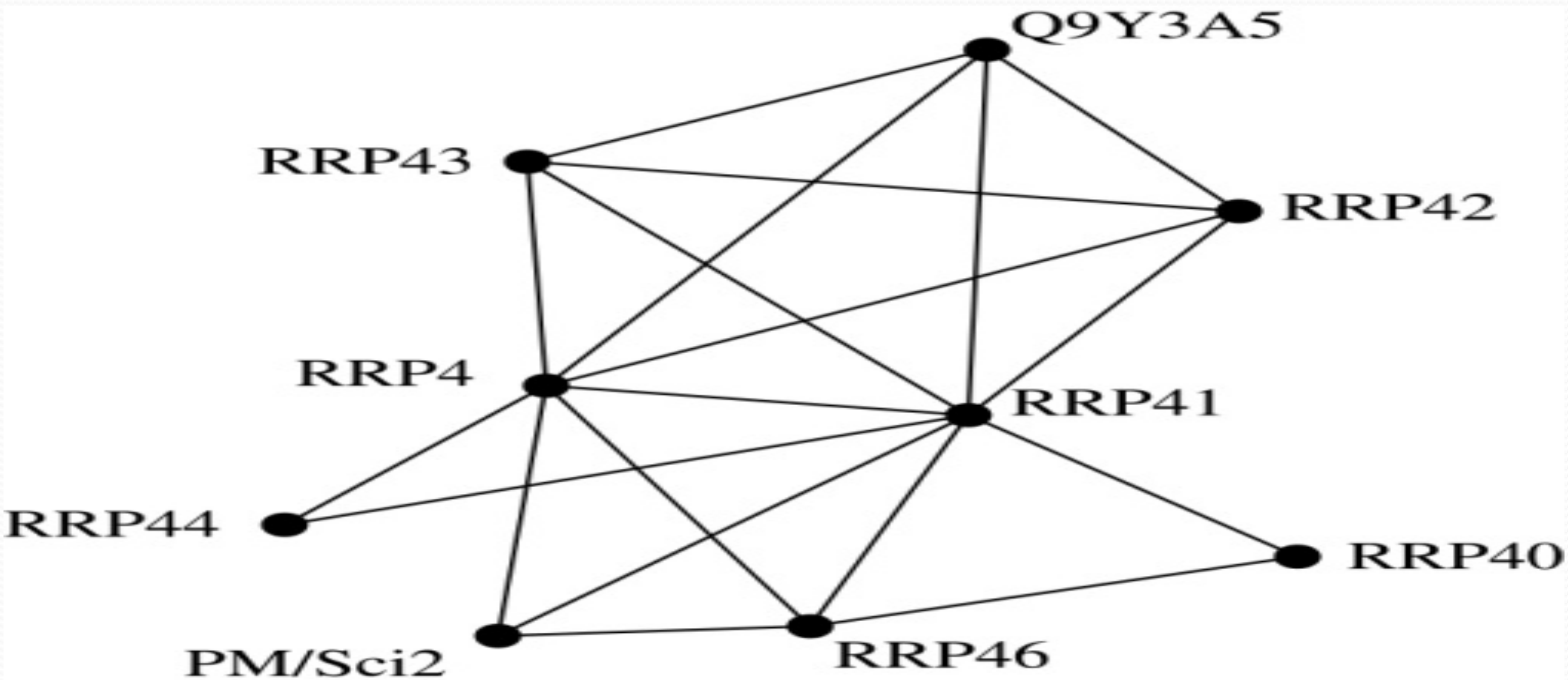


Biological Applications

- 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.

Biological Applications

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



Graph Terminology and Special Types of Graphs

Section 10.2

Section Summary

- Basic Terminology
- Some Special Types of Graphs
- Bipartite Graphs
- Bipartite Graphs and Matchings
- Some Applications of Special Types of Graphs
- New Graphs from Old

Basic Terminology

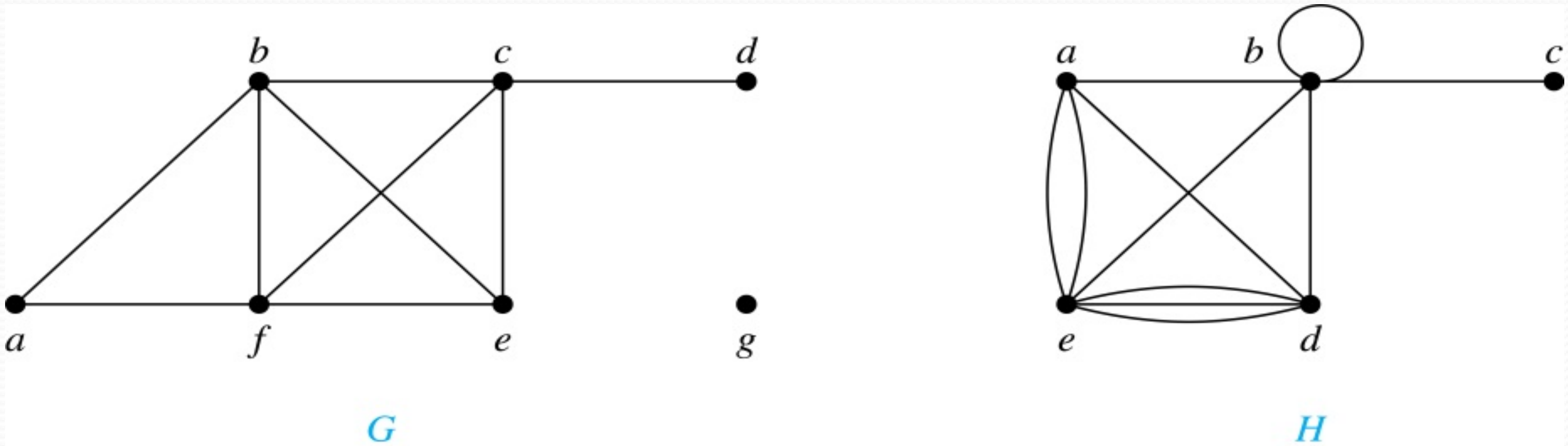
Definition 1. Two vertices u, v in an undirected graph G are called *adjacent* (or *neighbors*) in G if there is an edge e between u and v. Such an edge e is called *incident with* the vertices u and v and e is said to *connect* u and v.

Definition 2. 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. So, $N(A) = \bigcup_{v \in A} N(v)$.

Definition 3. The *degree of a vertex in a undirected graph* is the number of edges incident with it, except that a loop at a vertex contributes two to the degree of that 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*?



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

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

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

Handshaking Theorem

Theorem: If G is any graph, then the sum of the degrees of all the vertices of G equals twice the number of edges of G.

Specifically, if the vertices of G are v**1**, v**2**, …, v**n**, where n is a positive integer, then

the total degree of G = deg(v1) + deg(v2) + … + deg(vn)
= 2 . (the number of edges of G)

PROOF:

- Each edge "e" of G connects its end points vi and vj. This edge, therefore contributes 1 to the degree of vi and 1 to the degree of vj.
- If "e" is a loop, then it is counted twice in computing the degree of the vertex on which it is incident.
- Accordingly, each edge of G contributes 2 to the total degree of G.

Thus, the total degree of G = 2. (the number of edges of G)

COROLLARY: The total degree of G is an even number

Degree of Vertices

Theorem: 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

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

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.

Handshaking Theorem

We now give two examples illustrating the usefulness of the handshaking theorem.

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

Solution: Because the sum of the degrees of the vertices is $6 \cdot 10 = 60$, the handshaking theorem tells us that $2m = 60$. So the number of edges $m = 30$.

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

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

Directed Graphs

Recall the definition of a directed graph.

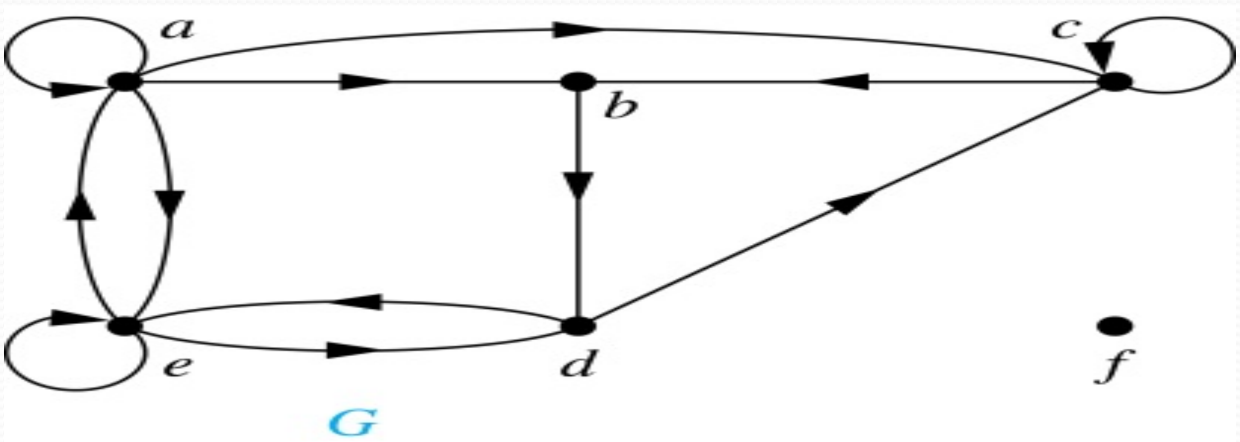
Definition: An *directed graph* $G = (V, E)$ consists of V, a nonempty set of *vertices* (or *nodes*), and E, a set of *directed edges* or *arcs*. Each edge is an ordered pair of vertices. The directed edge (u,v) is said to start at u and end at v.

Definition: Let (u,v) be an edge in G. Then u is the *initial vertex* of this edge and is *adjacent to* v and v is the *terminal* (or *end*) *vertex* of this edge and is *adjacent from* u. The initial and terminal vertices of a loop are the same.

Directed Graphs (*continued*)

Definition: 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 (*continued*)

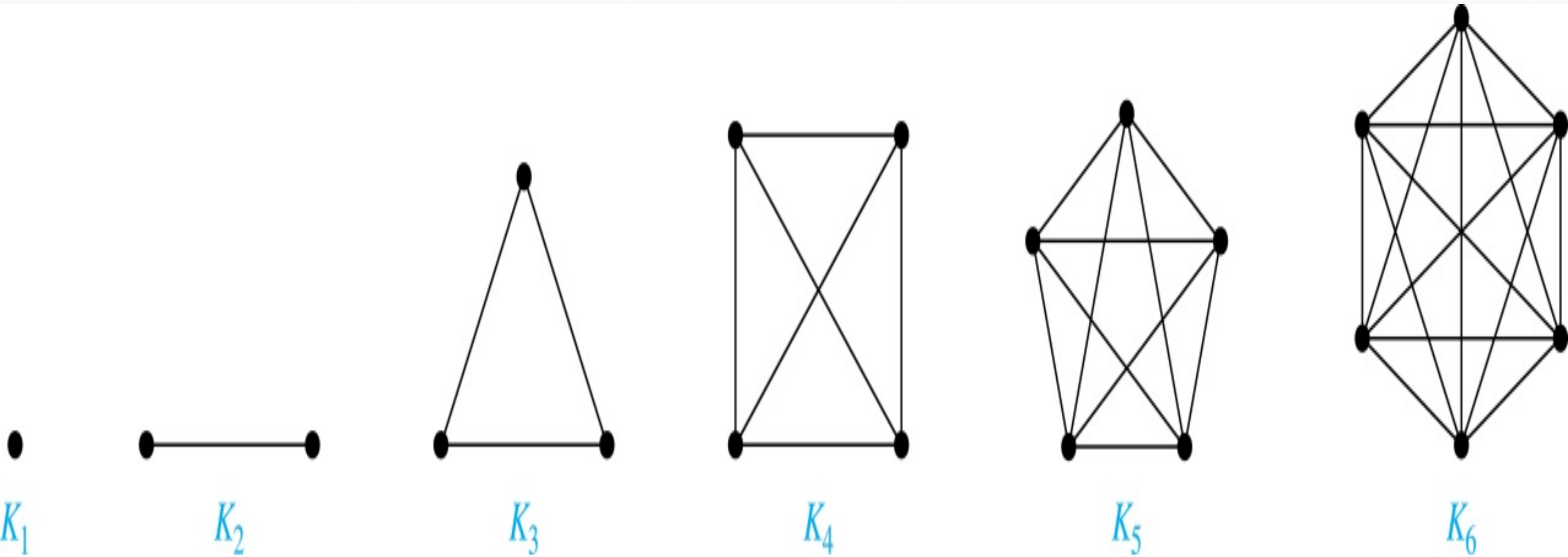
Theorem 3: Let $G = (V, E)$ be a graph with directed edges. Then:

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

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.

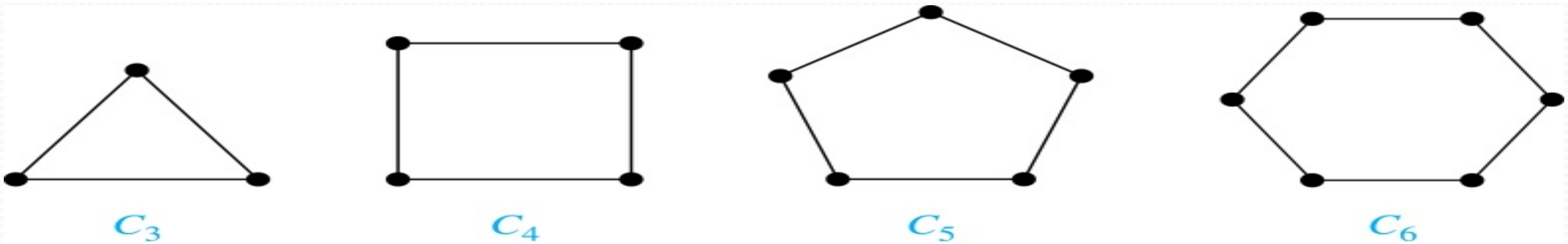
Special Types of Simple Graphs: Complete Graphs

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.

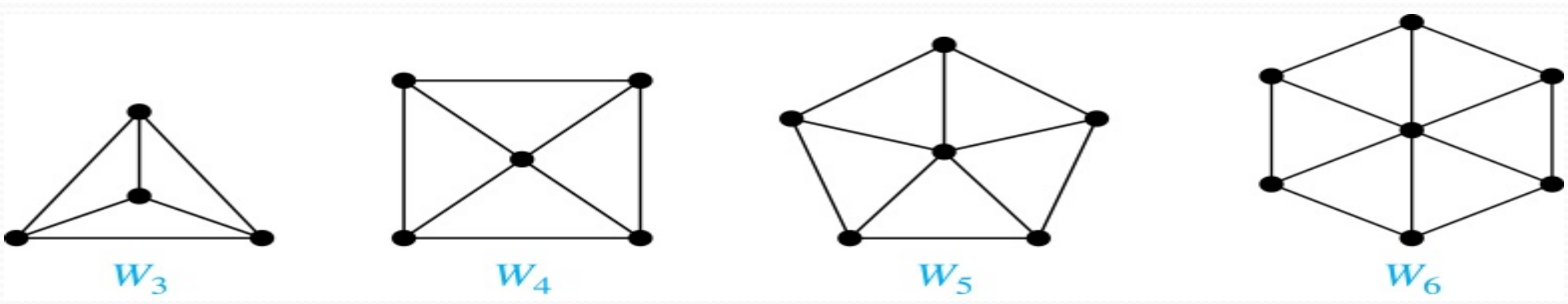


Special Types of Simple Graphs: Cycles and Wheels

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

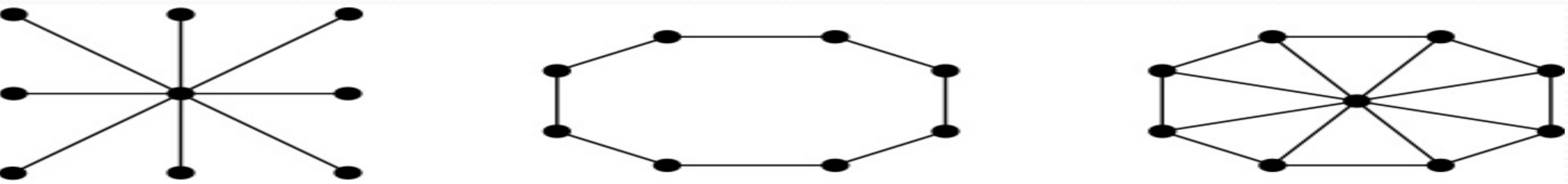


A *wheel* W_n is obtained by adding an additional vertex to a cycle C_n for $n \geq 3$ and connecting this new vertex to each of the n vertices in C_n by new edges.



Special Types of Graphs and Computer Network Architecture

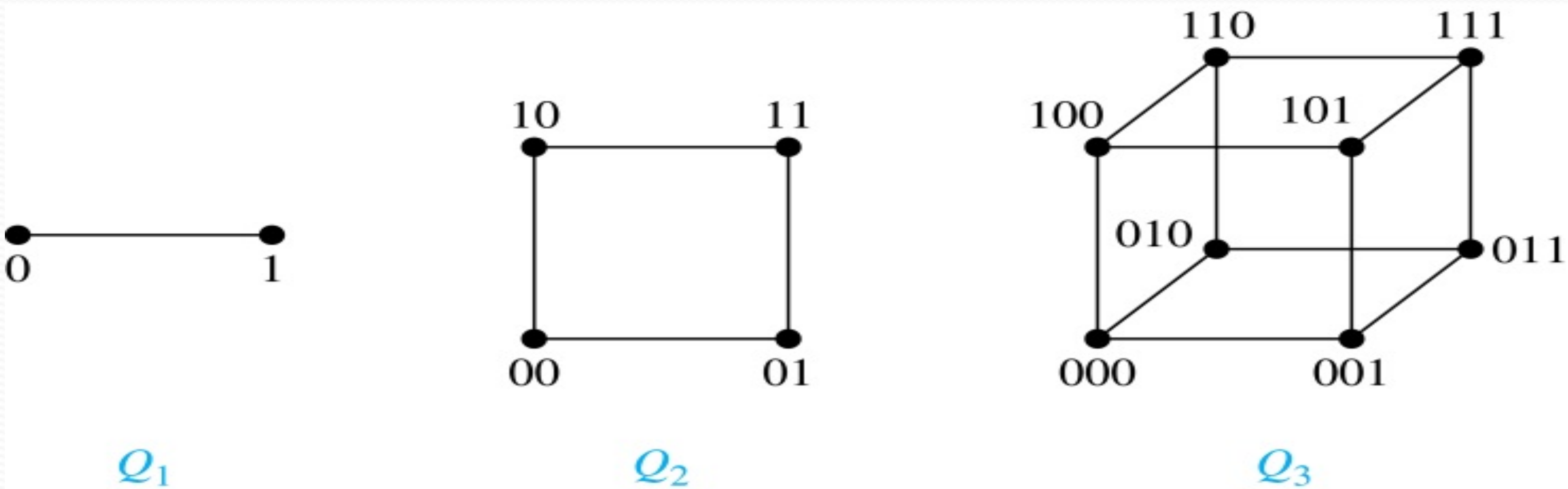
Various special graphs play an important role in the design of computer networks.


(a) (b) (c)

- Some local area networks use a *star topology*, which is a complete bipartite graph $K_{1,n}$, as shown in (a). All devices are connected to a central control device.
- Other local networks are based on a *ring topology*, where each device is connected to exactly two others using C_n, as illustrated in (b). Messages may be sent around the ring.
- Others, as illustrated in (c), use a W_n – based topology, combining the features of a star topology and a ring topology.

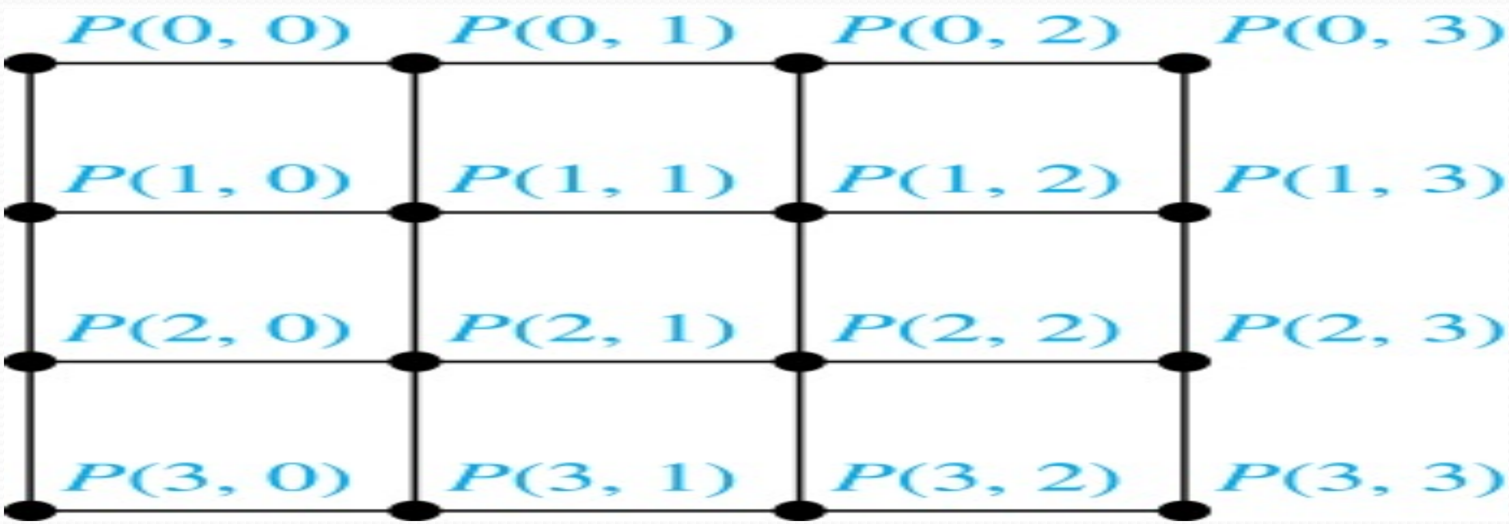
Special Types of Simple Graphs: *n*-Cubes

An *n-dimensional hypercube*, or *n-cube,* $\boldsymbol{Q_n}$, 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.



Special Types of Graphs and Computer Network Architecture

- Various special graphs also play a role in parallel processing where processors need to be interconnected as one processor may need the output generated by another.
- The n-dimensional hypercube, or n-cube, Qn, is a common way to connect processors in parallel, e.g., Intel Hypercube.
- Another common method is the mesh network, illustrated here for 16 processors.



Bipartite Graphs

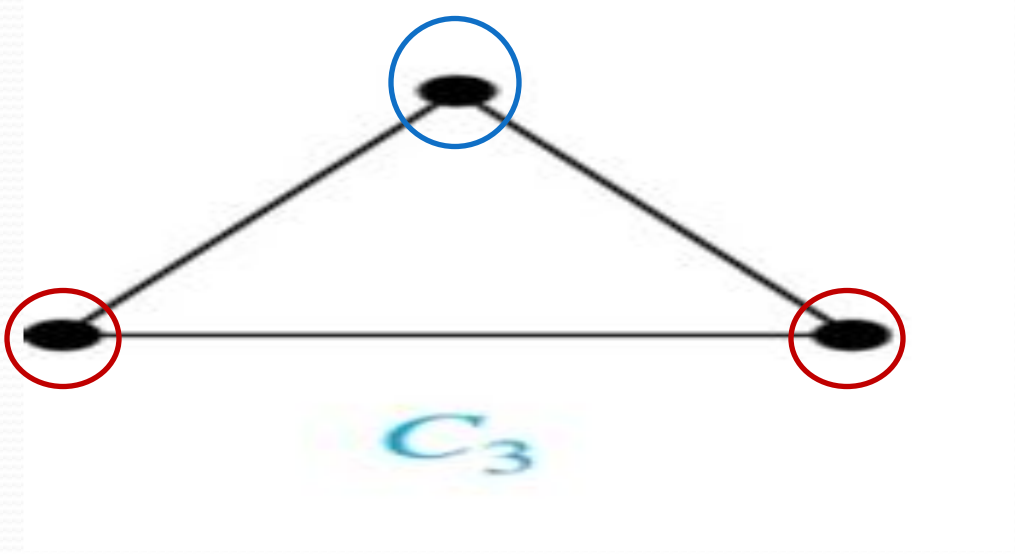
Definition: A simple graph G is bipartite if V can be partitioned into two disjoint subsets V_1 and V_2 such that every edge connects a vertex in V_1 and a vertex in V_2. In other words, there are no edges which connect two vertices in V_1 or in V_2.

- It is not hard to show that an equivalent definition of 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.

Bipartite Graphs (*continued*)

Example: Show that C_3 is not bipartite.

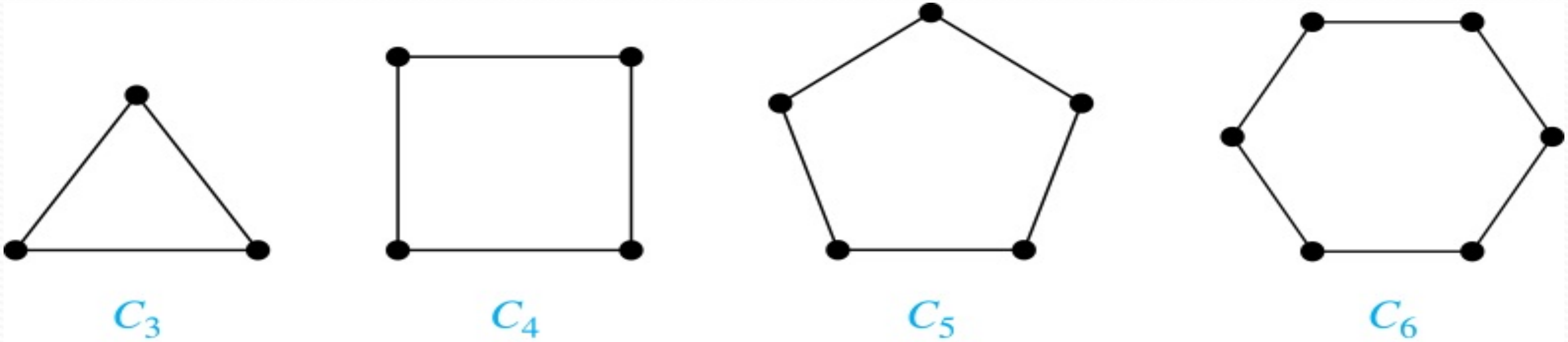
Solution: If we divide the vertex set of C_3 into two nonempty sets, one of the two must contain two vertices. But in C_3 every vertex is connected to every other vertex. Therefore, the two vertices in the same partition are connected. Hence, C_3 is not bipartite.

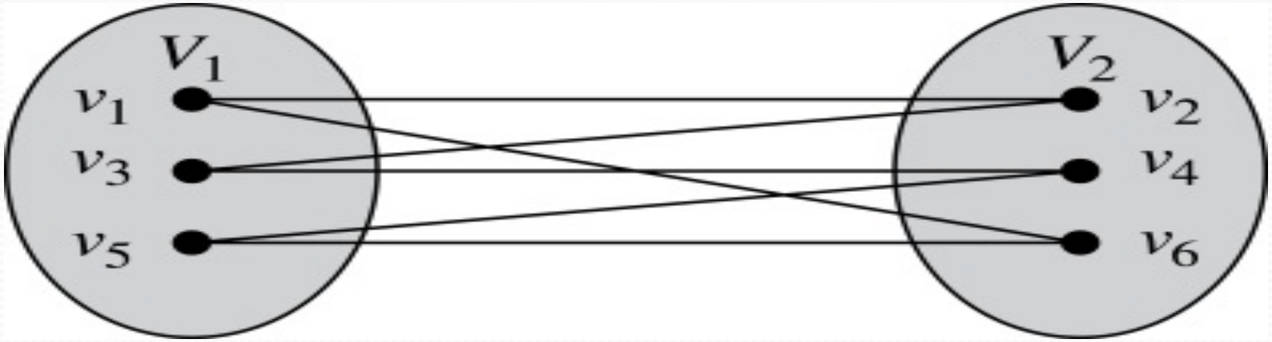


Bipartite Graphs (*continued*)

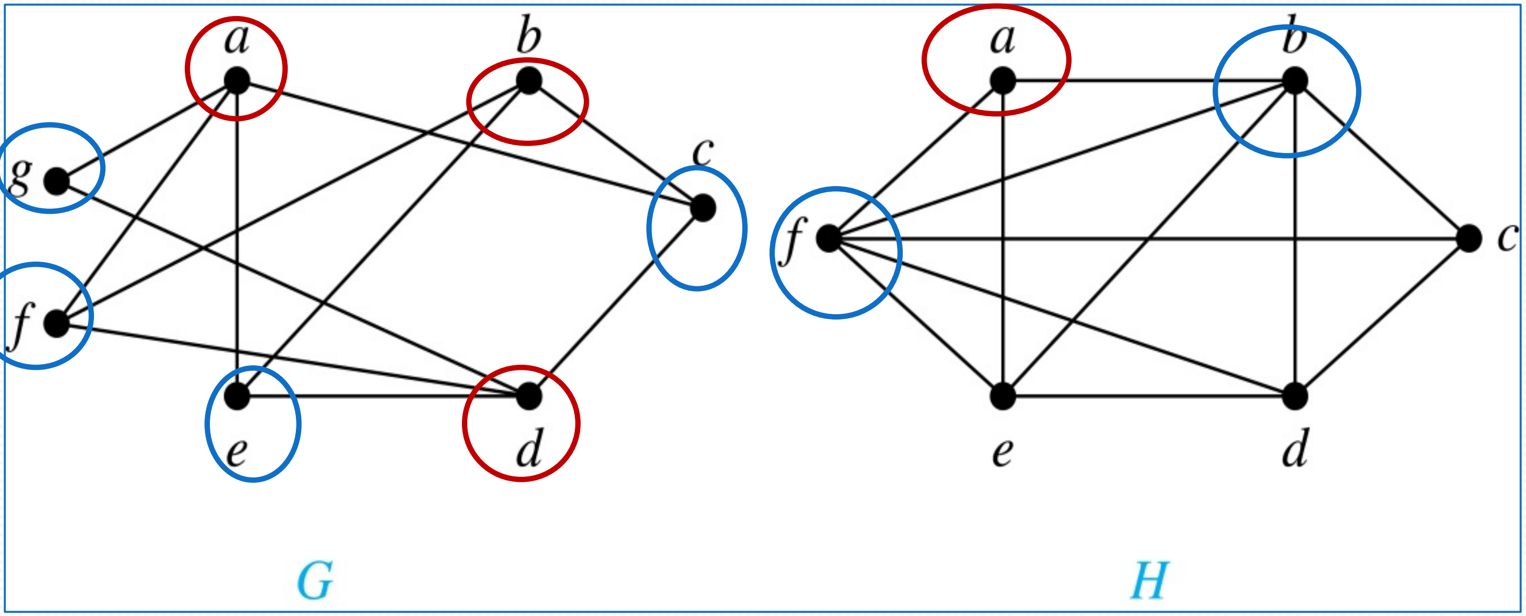
Example: Show that C_6 is bipartite.

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 .





Bipartite Graphs

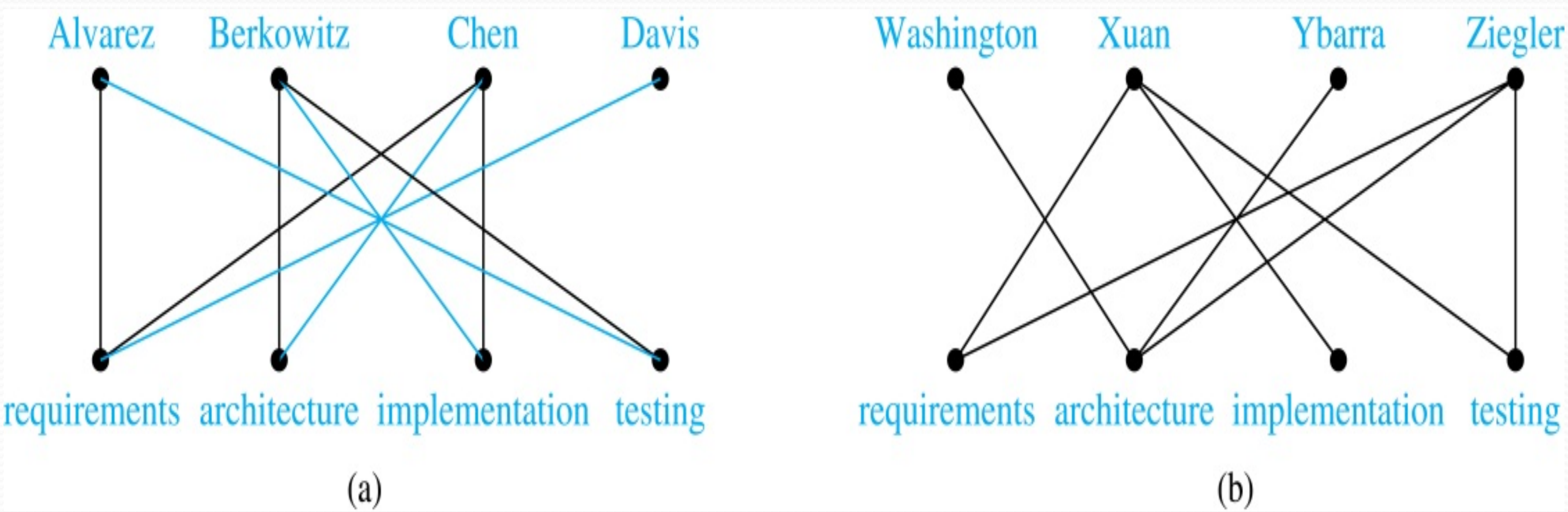


G is bipartite

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

Bipartite Graphs and Matchings

- Bipartite graphs are used to model applications that involve matching the elements of one set to elements in another, for example:
- *Job assignments* - vertices represent the jobs and the employees, edges link employees with those jobs they have been trained to do. A common goal is to match jobs to employees so that the most jobs are done.



Bipartite Graphs and Matchings

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

Example:

Match.com an online dating website receive applications from 5 boys (**Andy, Bobby, Colin, Dan and Elliott**) and 5 girls (**Rita, Sarah, Tracey, Uma and Victoria**). After filling in their details the match.com algorithms work out the following:

- Andy appears compatible with all 5 girls.

- Bobby appears compatible with Uma.

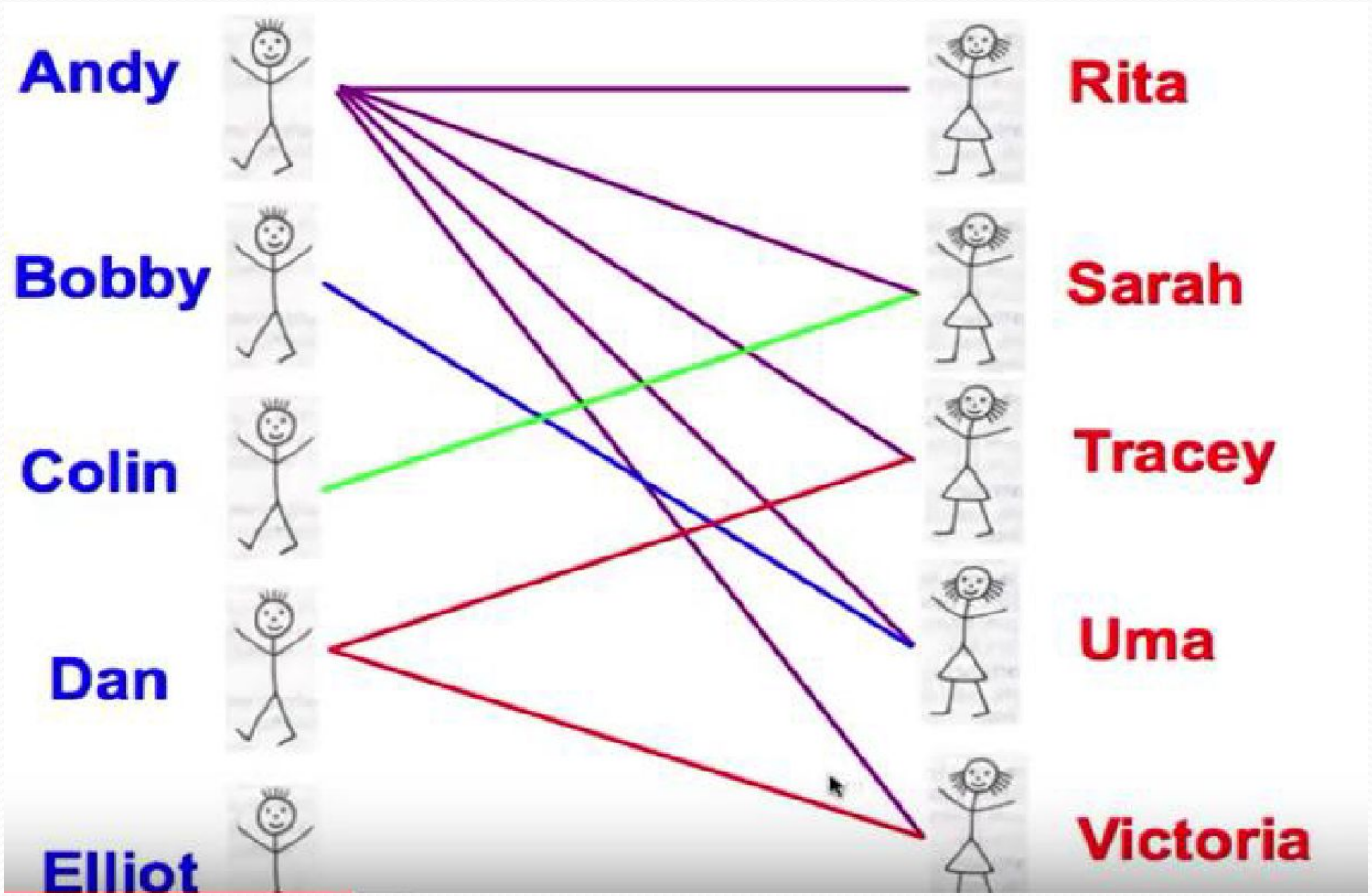
- Colin appears compatible with Sarah.

- Dan appears compatible with Tracey and Victoria.

- Elliott does not appear compatible with any of the girls.

Draw a bipartite graph to model this situation.

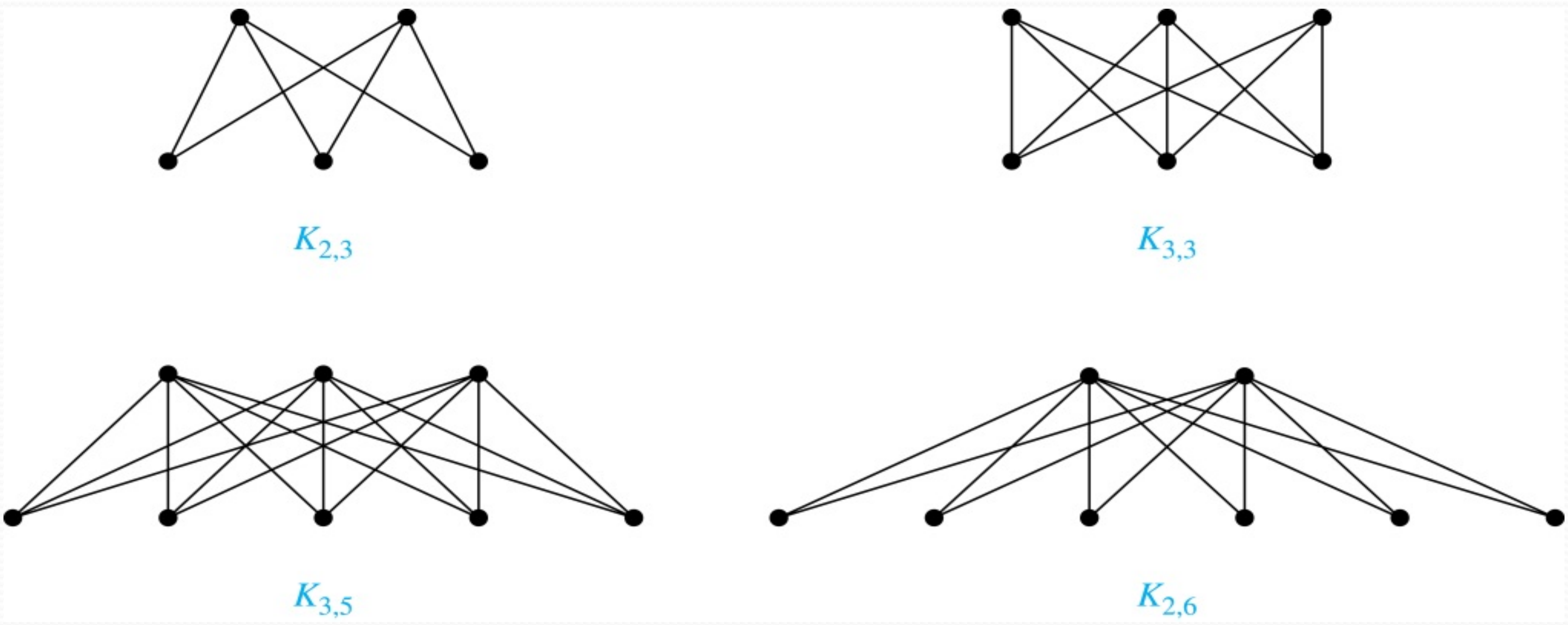
Bipartite Graphs and Matchings



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.

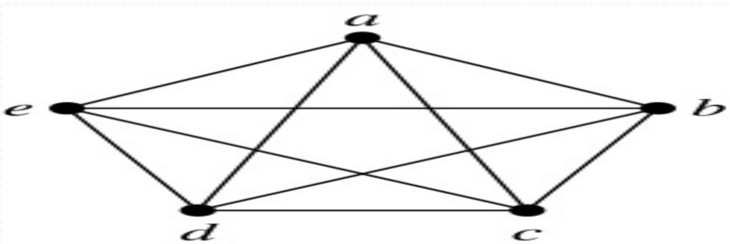
Example: We display four complete bipartite graphs here.

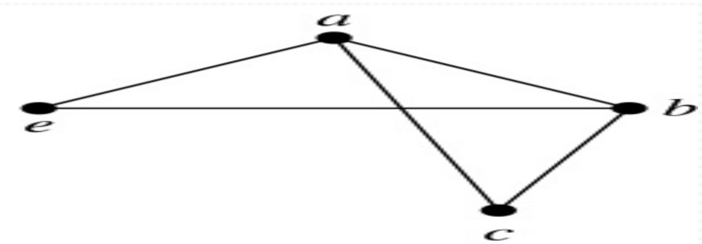


New Graphs from Old

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 $H \neq G$.

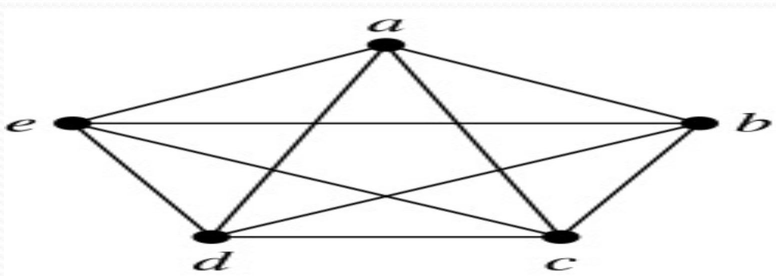
Example: Here we show K_5 and one of its subgraphs.

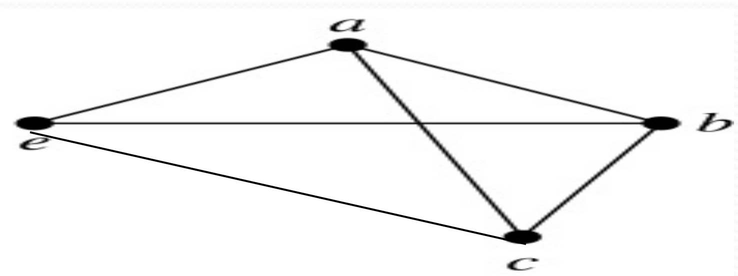




Definition: Let $G = (V, E)$ be a simple graph. The *subgraph induced* by a subset W of the vertex set V is the graph (W,F), where the edge set F contains an edge in E if and only if both endpoints are in W.

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

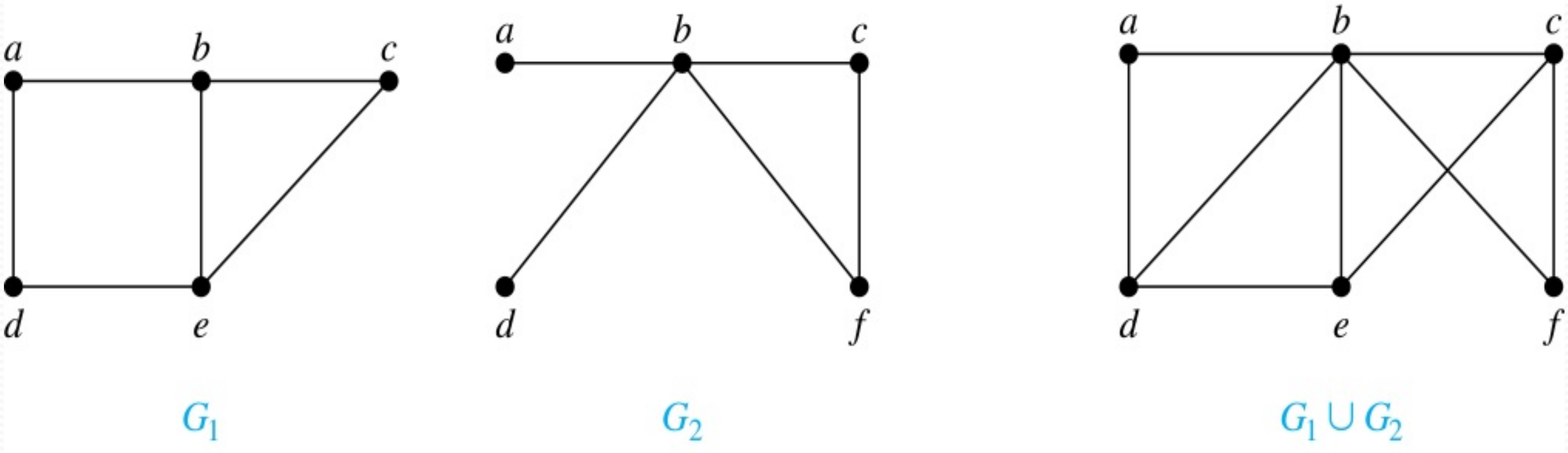




New Graphs from Old (*continued*)

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:



(a) (b)

Representations of Graphs

Section 10.3

Section Summary

- Adjacency Lists
- Adjacency Matrices
- Incidence Matrices

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:

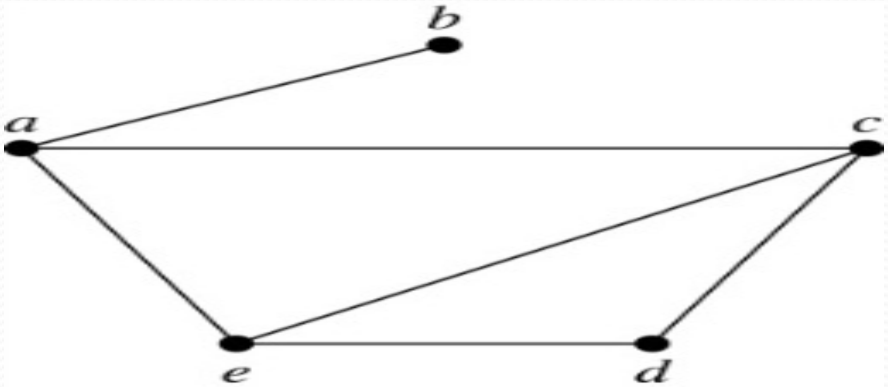


TABLE 1 An Adjacency List for a Simple Graph.

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

Representing Graphs: Adjacency Lists

Example:

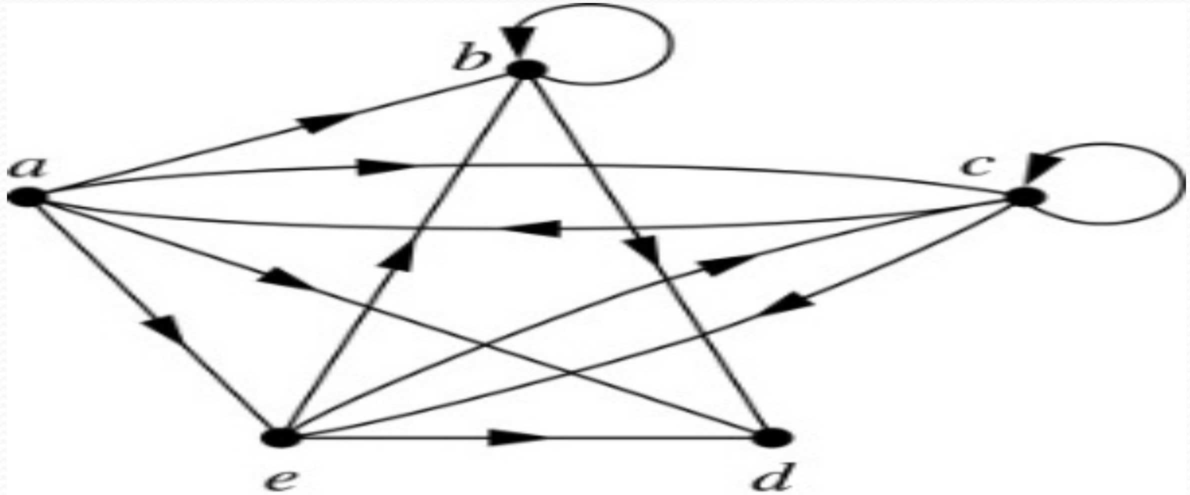


TABLE 2 **An Adjacency List for a Directed Graph.**

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

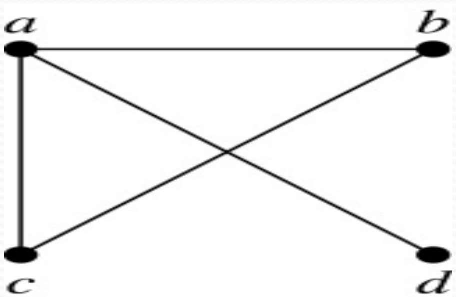
Definition: Suppose that $G = (V, E)$ is a simple graph where $|V| = n$. Arbitrarily list the vertices of G as $v_1, v_2, \ldots, v_n$. The *adjacency matrix* $\mathbf{A}_G$ of G, with respect to the listing of vertices, is the $n \times n$ zero-one matrix with 1 as its (i, j)th entry when v_i and v_j 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

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

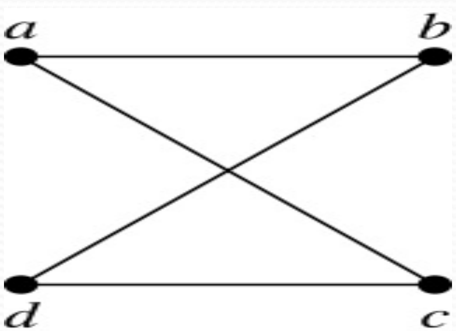
Adjacency Matrices (*continued*)

Example:



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

The ordering of vertices is a, b, c, d.



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

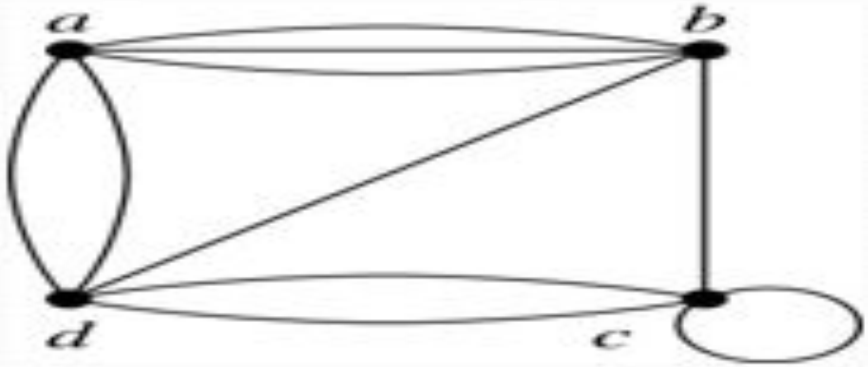
The ordering of vertices is a, b, c, d.

When a graph is sparse, that is, it has few edges relatively to the total number of possible edges, it is much more efficient to represent the graph using an adjacency list than an adjacency matrix. But for a dense graph, which includes a high percentage of possible edges, an adjacency matrix is preferable.

Note: The adjacency matrix of a simple graph is symmetric, i.e., $a_{ij} = a_{ji}$
Also, since there are no loops, each diagonal entry a_{ij} for $i = 1, 2, 3, \ldots, n$, is 0.

Adjacency Matrices (*continued*)

- Adjacency matrices can also be used to represent graphs with loops and multiple edges.
- A loop at the vertex vi is represented by a 1 at the (i, j)th position of the matrix.
- When multiple edges connect the same pair of vertices vi and vj, (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.



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

Adjacency Matrices (*continued*)

- Adjacency matrices can also be used to represent directed graphs. The matrix for a directed graph $G = (V, E)$ has a 1 in its (i, j)th position if there is an edge from v_i to v_j, where $v_1, v_2, \ldots v_n$ is a list of the vertices.
  - In other words, if the graphs adjacency matrix is $\mathbf{A}_G = [a_{ij}]$, then

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

  - The adjacency matrix for a directed graph does not have to be symmetric, because there may not be an edge from v_i to v_j, when there is an edge from v_j to v_i.
  - To represent directed multigraphs, the value of a_{ij} is the number of edges connecting v_i to v_j.

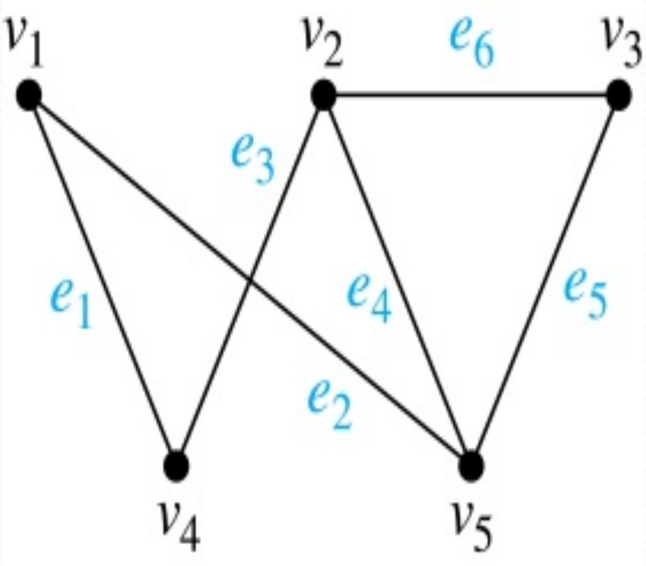
Representation of Graphs: Incidence Matrices

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

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

Incidence Matrices (*continued*)

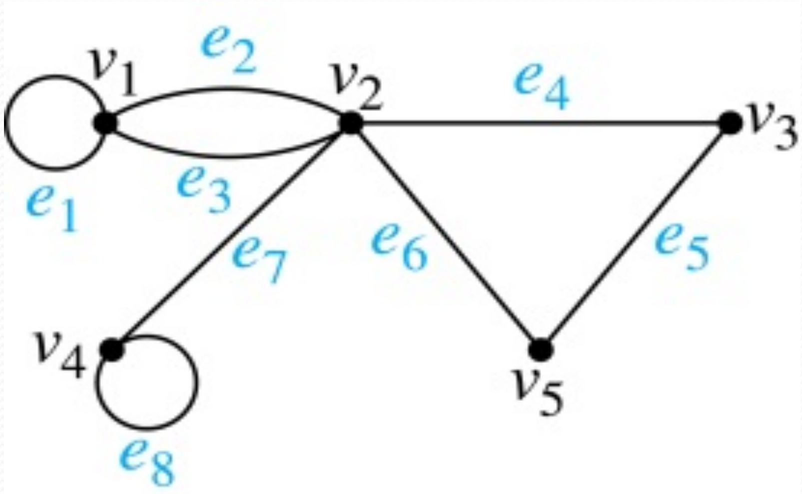
Example: Simple Graph and Incidence Matrix



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

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.

Connectivity

Section 10.4

Section Summary

- Paths
- Connectedness in Undirected Graphs
- Connectedness in Directed Graphs
- Counting Paths between Vertices

Paths

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

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

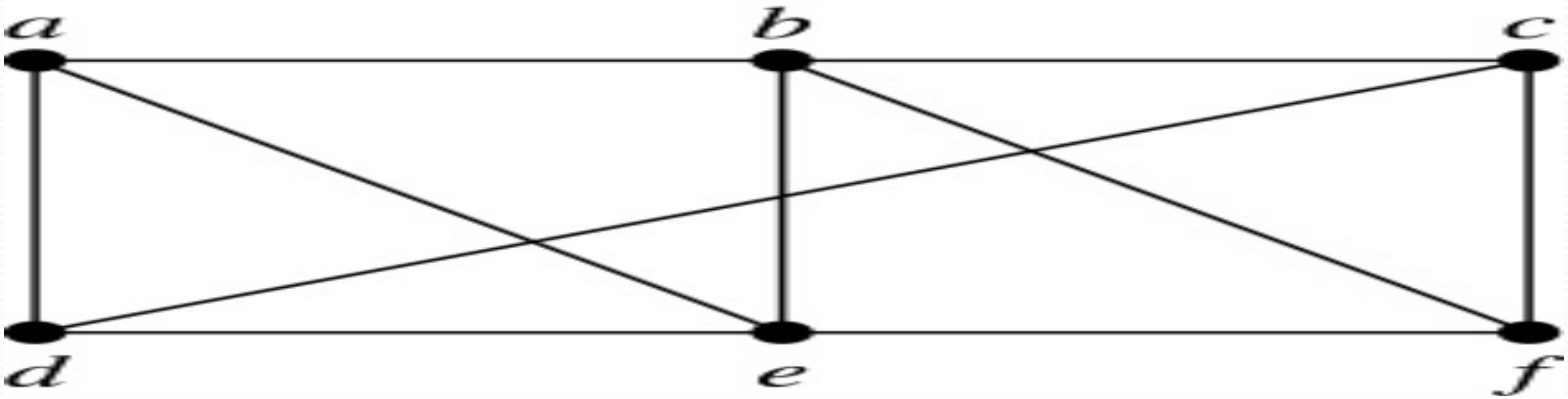
- determining whether a message can be sent between two computers.
- efficiently planning routes for mail delivery.

Paths

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

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

Paths (*continued*)



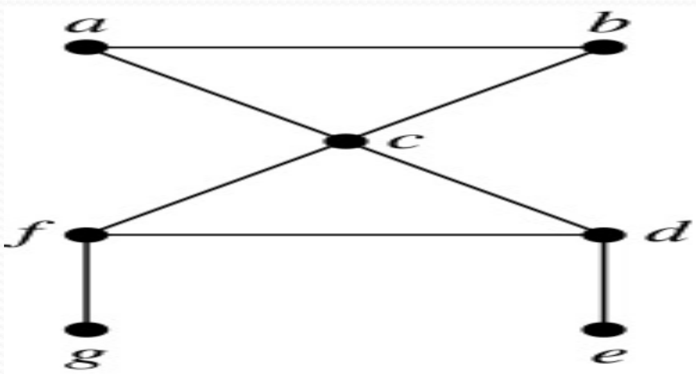
Example: In the simple graph here:

- a, d, c, f, e is a simple path of length 4.
- d, e, c, a is not a path because e is not connected to c.
- b, c, f, e, b is a circuit of length 4.
- a, b, e, d, a, b is a path of length 5, but it is not a simple path.

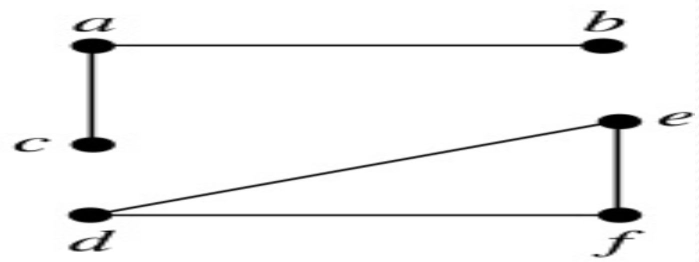
Connectedness in Undirected Graphs

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

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



G_1



G_2

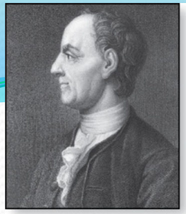
Euler and Hamiltonian Graphs

Section 10.5

Section Summary

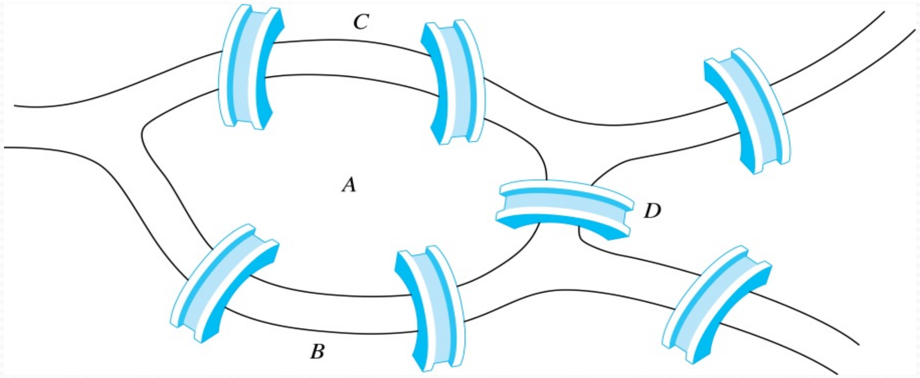
- Euler Paths and Circuits
- Hamilton Paths and Circuits
- Applications of Hamilton Circuits

Euler Paths and Circuits

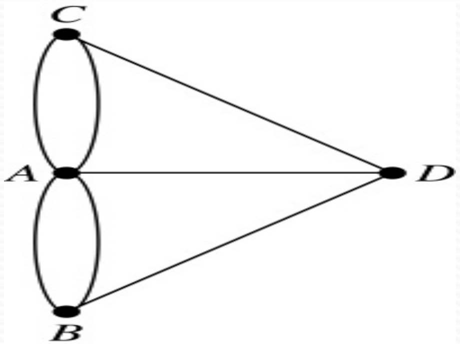


Leonard Euler
(1707-1783)

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



The 7 Bridges of Königsberg

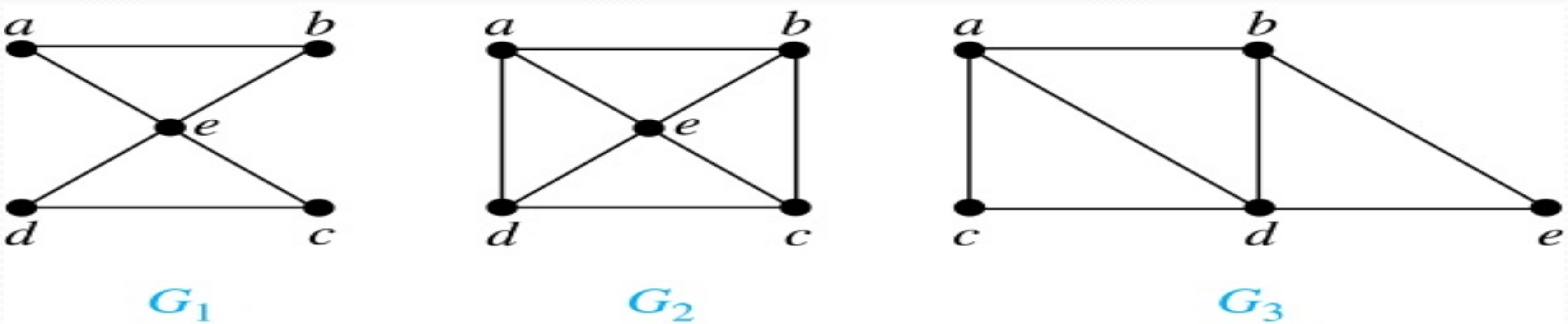


Multigraph Model of the Bridges of Königsberg

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.

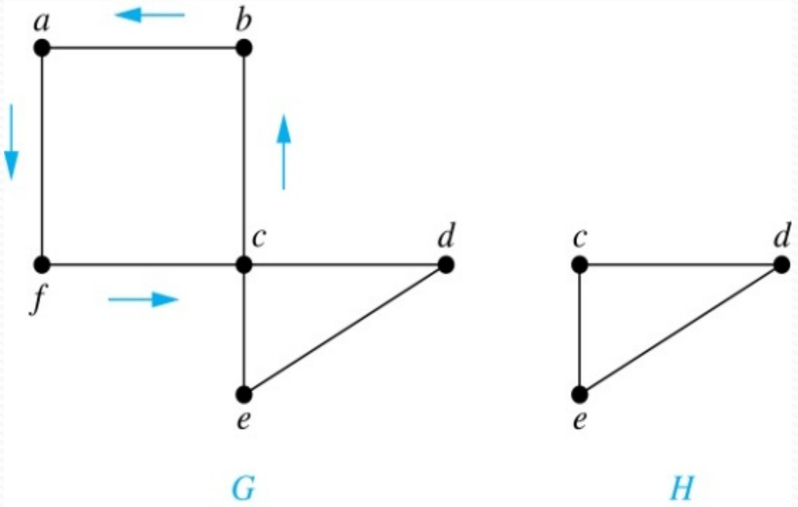
Necessary Conditions for Euler Circuits and Paths

- An Euler circuit begins with a vertex a and continues with an edge incident with a, say $\{a, b\}$. The edge $\{a, b\}$ contributes one to $\deg(a)$.
- Each time the circuit passes through a vertex it contributes two to the vertex's degree.
- Finally, the circuit terminates where it started, contributing one to $\deg(a)$. Therefore $\deg(a)$ must be even.
- We conclude that the degree of every other vertex must also be even.
- By the same reasoning, we see that the initial vertex and the final vertex of an Euler path have odd degree, while every other vertex has even degree. So, a graph with an Euler path has exactly two vertices of odd degree.
- In the next slide we will show that these necessary conditions are also sufficient conditions.

Sufficient Conditions for Euler Circuits and Paths

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

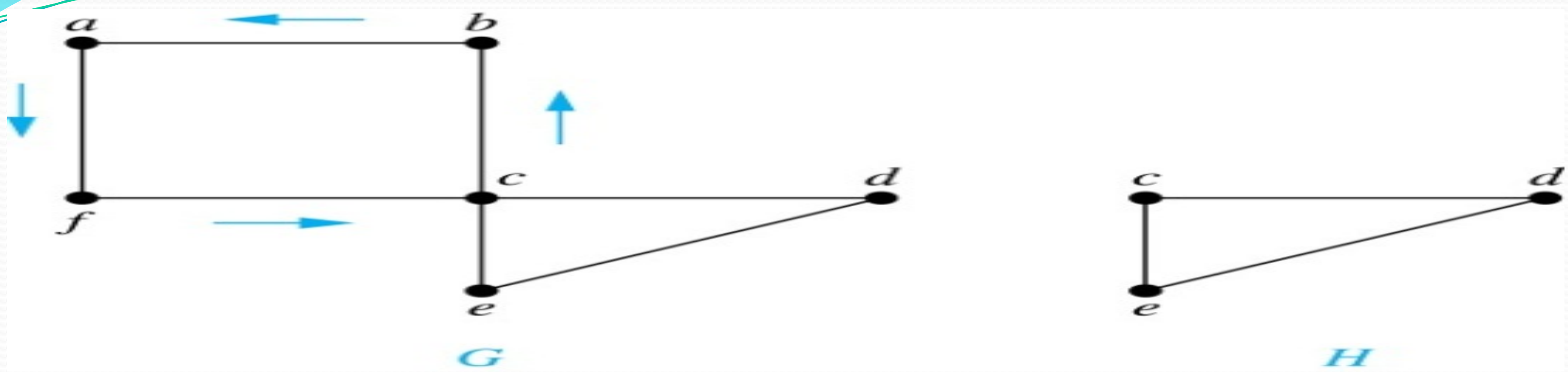
We illustrate this idea in the graph G here. We begin at a and choose the edges $\{a, f\}$, $\{f, c\}$, $\{c, b\}$, and $\{b, a\}$ in succession.



- The path begins at a with an edge of the form $\{a, x\}$; we show that it must terminate at a with an edge of the form $\{y, a\}$. Since each vertex has an even degree, there must be an even number of edges incident with this vertex. Hence, every time we enter a vertex other than a, we can leave it. Therefore, the path can only end at a.
- If all of the edges have been used, an Euler circuit has been constructed. Otherwise, consider the subgraph H obtained from G by deleting the edges already used.

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

Sufficient Conditions for Euler Circuits and Paths



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

In the example, the vertex is *c*.

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

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

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

Algorithm for Constructing an Euler Circuits

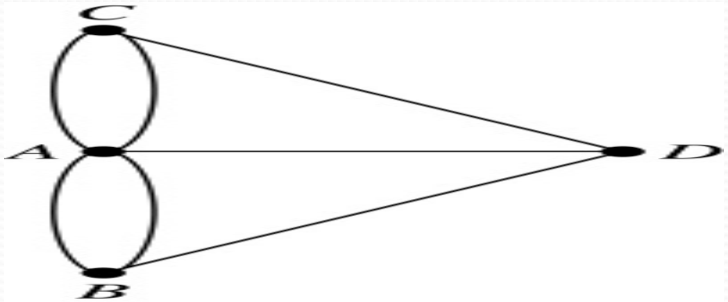
In our proof we developed this algorithms for constructing a Euler circuit in a graph with no vertices of odd degree.

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

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

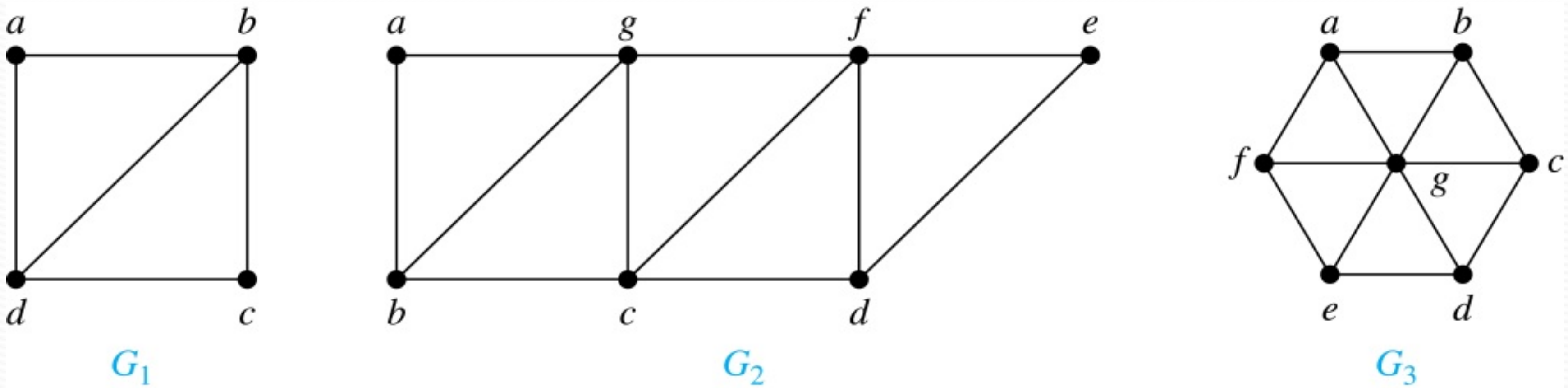
Theorem: A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has an even degree and it has an Euler path if and only if it has exactly two vertices of odd degree.

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



Euler Circuits and Paths

Example:



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

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

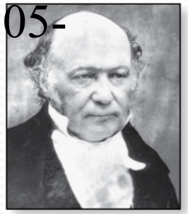
G_3 has six vertices of odd degree. Hence, it does not have an Euler path.

Applications of Euler Paths and Circuits

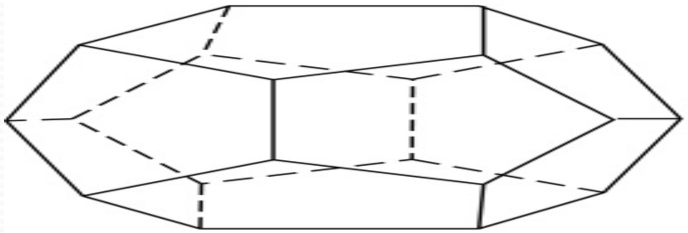
- Euler paths and circuits can be used to solve many practical problems such as finding a path or circuit that traverses each
  - street in a neighborhood,
  - road in a transportation network,
  - connection in a utility grid,
  - link in a communications network.
- Other applications are found in the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

Hamilton Paths and Circuits

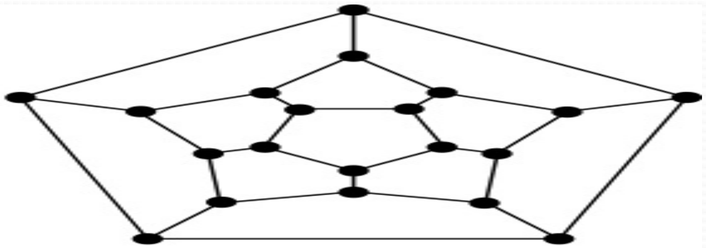
William Rowan Hamilton (1805-1865)



- Euler paths and circuits contained every edge only once.
  Now we look at paths and circuits that contain every vertex exactly once.
- William Hamilton invented the *Icosian puzzle* in 1857. It consisted of a wooden dodecahedron (with 12 regular pentagons as faces), illustrated in (a), 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
- The graph form of the puzzle is given in (b).

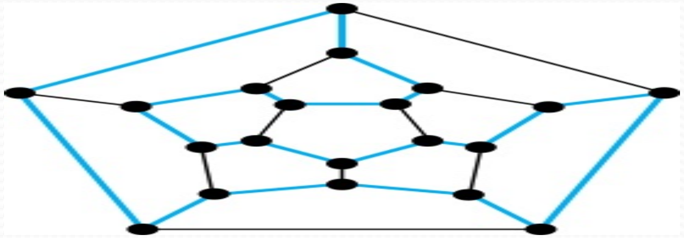


(a)



(b)

- The solution (a Hamilton circuit) is given here.



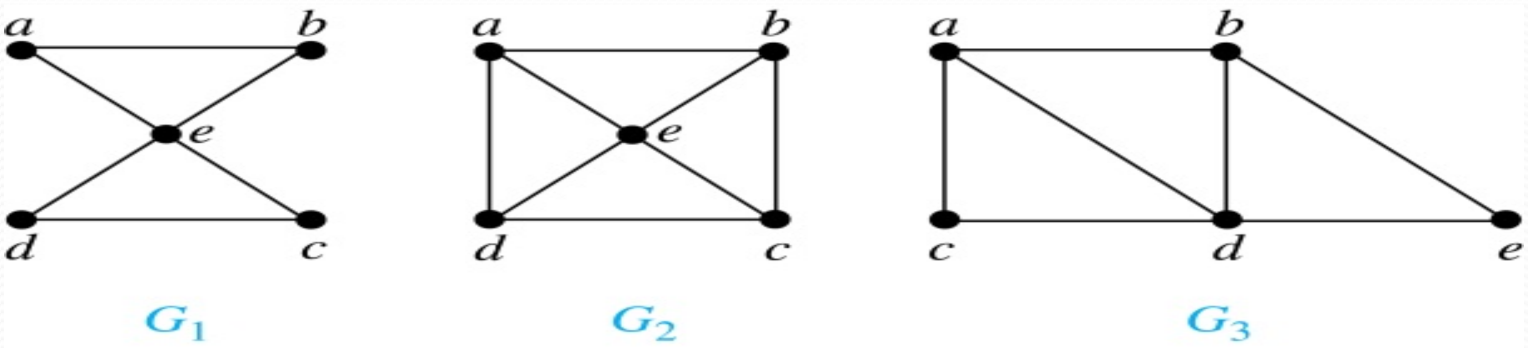
Hamilton Paths and Circuits

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

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

Hamilton Paths and Circuits

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



Solution:

G_1 does not have a Hamilton circuit (Why?), but does have a Hamilton path : *a*, *b* ,e , *c*, *d*.

G_2 has a Hamilton circuit: *a*, *b*, *c*, *d*, *e*, *a*.

G_3 has a Hamilton circuit: a,b,e,d,c,a

Necessary Conditions for Hamilton Circuits

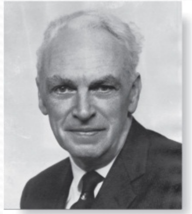
Gabriel Andrew Dirac
(1925-1984)

- 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.

Øysten Ore
(1899-1968)



Applications of Hamilton Paths and Circuits

- Applications that ask for a path or a circuit that visits each intersection of a city, each place pipelines intersect in a utility grid, or each node in a communications network exactly once, can be solved by finding a Hamilton path in the appropriate graph.
- The famous *traveling salesperson problem* (*TSP*) asks for the shortest route a traveling salesperson should take to visit a set of cities. This problem reduces to finding a Hamilton circuit such that the total sum of the weights of its edges is as small as possible.
- A family of binary codes, known as *Gray codes*, which minimize the effect of transmission errors, correspond to Hamilton circuits in the n-cube Q_n.

ISOMORPHISM OF GRAPHS

Section 10.3

Isomorphism of Graphs

Definition: The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there is a one-to-one and onto function f from V_1 to V_2 with the property that a and b are adjacent in G_1 if and only if $f(a)$ and $f(b)$ are adjacent in G_2, for all a and b in V_1. Such a function f is called an *isomorphism*. Two simple graphs that are not isomorphic are called *nonisomorphic*.

Isomorphism of Graphs

- It is difficult to determine whether two simple graphs are isomorphic using brute force because there are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with n vertices.
- The best algorithms for determining weather two graphs are isomorphic have exponential worst case complexity in terms of the number of vertices of the graphs.
- Sometimes it is not hard to show that two graphs are not isomorphic. We can do so by finding a property, preserved by isomorphism, that only one of the two graphs has. Such a property is called *graph invariant*.
- There are many different useful graph invariants that can be used to distinguish nonisomorphic graphs, such as the number of vertices, number of edges, and degree sequence (list of the degrees of the vertices in nonincreasing order). We will encounter others in later sections of this chapter.

ISOMORPHIC INVARIANT

- A property P is called an isomorphic invariant if, and only if, given any graphs G and G',
- If G has property P and G' is isomorphic to G, then G' has property P.

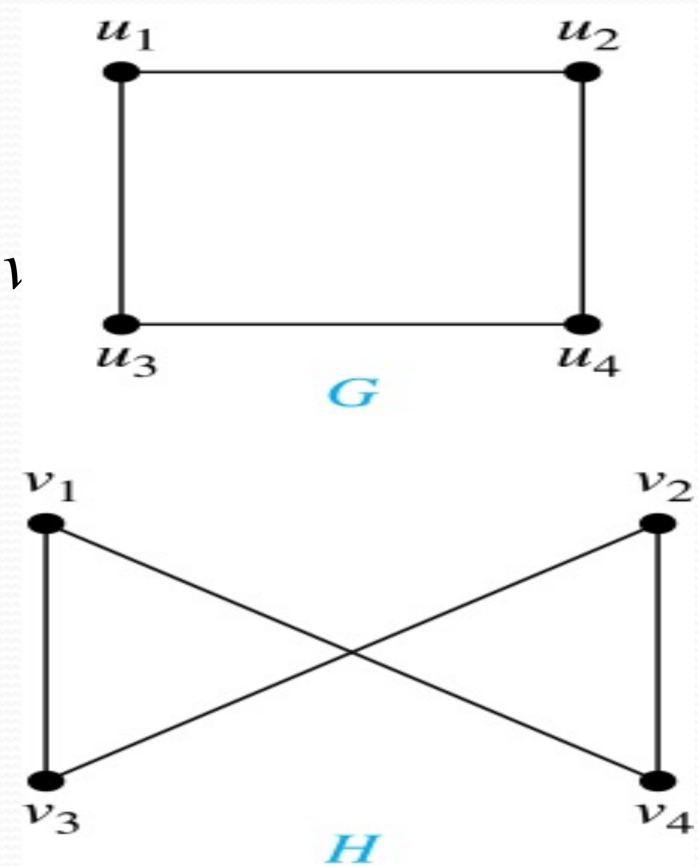
<u>THEOREM OF ISOMORPHIC INVARIANT</u>

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

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

Isomorphism of Graphs (*cont.*)

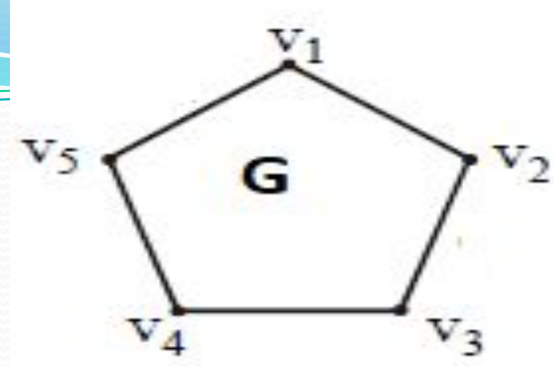
Example: Show that the graphs $G = (V, E)$ and $H = (W, F)$ are isomorphic.

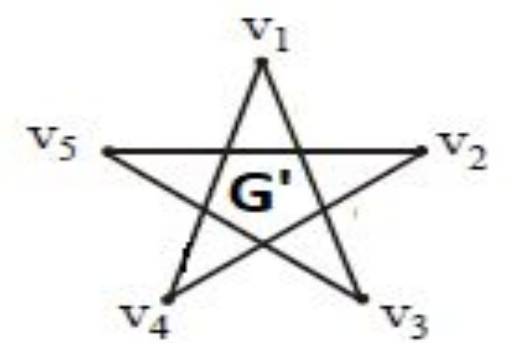


Solution: The function f with
$f(u_1) = v_1, f(u_2) = v_4, f(u_3) = v_3$, and $f(u_4) = v$
correspondence between V and W.
Note that adjacent vertices in G are
u_1 and u_2, u_1 and u_3, u_2 and u_4, and u_3 and u_4. Each of the pairs
$f(u_1) = v_1$ and $f(u_2) = v_4$, $f(u_1) = v_1$ and $f(u_3) = v_3$, $f(u_2) = v_4$ 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.

Isomorphism of Graphs

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





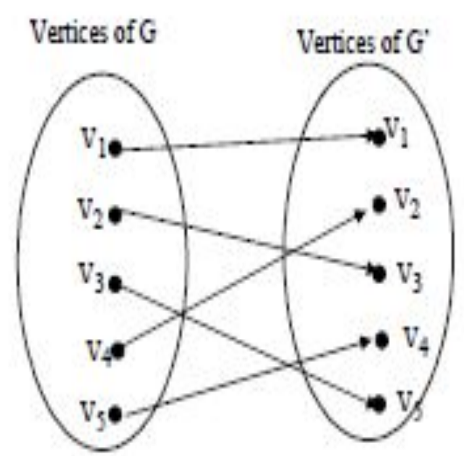
Solution: The function f with

f(v1) = v1,f(v2) = v3, f(v3) = v5, f(v4) = v2 and f(v5) = v4 is a one-to-one correspondence between V and W.

Note that adjacent vertices in G are v1 and v2, v2 and v3, v3 and v4, v4 and v5 and v5 and v1.

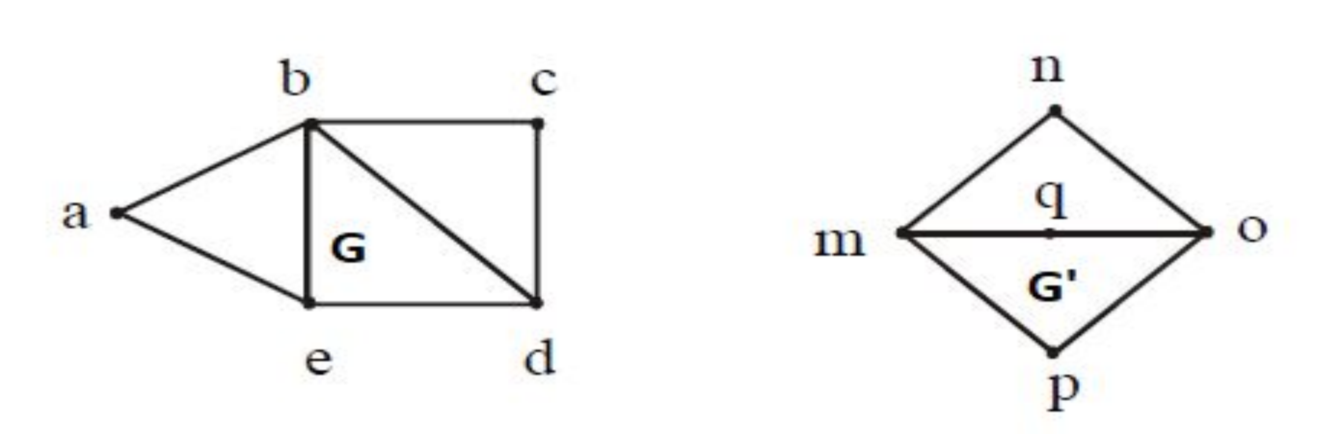
Each of the pairs f(v1) = v1 and f(v2) = v3, f(v2) = v3 and f(v3) = v5 , f(v3) = v5 and

f(v4) = v2 , f(v4) = v2 and f(v5) = v4 and f(v5) = v4 and f(v1) = v1 consists of two adjacent vertices in H.



Isomorphism of Graphs

EXAMPLE: Determine whether the graph G and G’ given below are isomorphic.



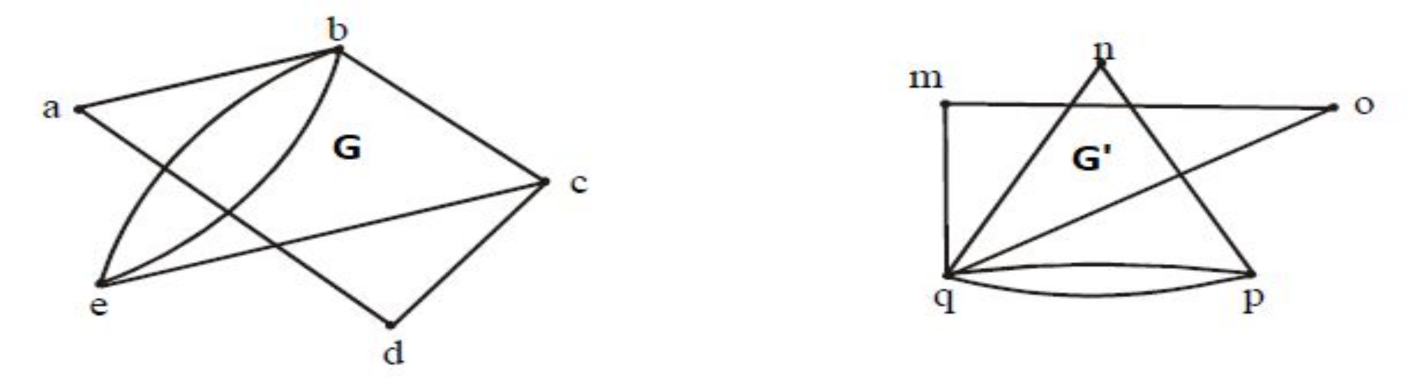
SOLUTION:

As both the graphs have the same number of vertices. But the graph G has 7 edges and the graph G’ has only 6 edges. Therefore the two graphs are not isomorphic.

Note: As the edges of both the graphs G and G’ are not same then how the one-one correspondence is possible ,that the reason the graphs G and G’ are not isomorphic.

Isomorphism of Graphs

EXAMPLE: Determine whether the graph G and G’ given below are isomorphic.

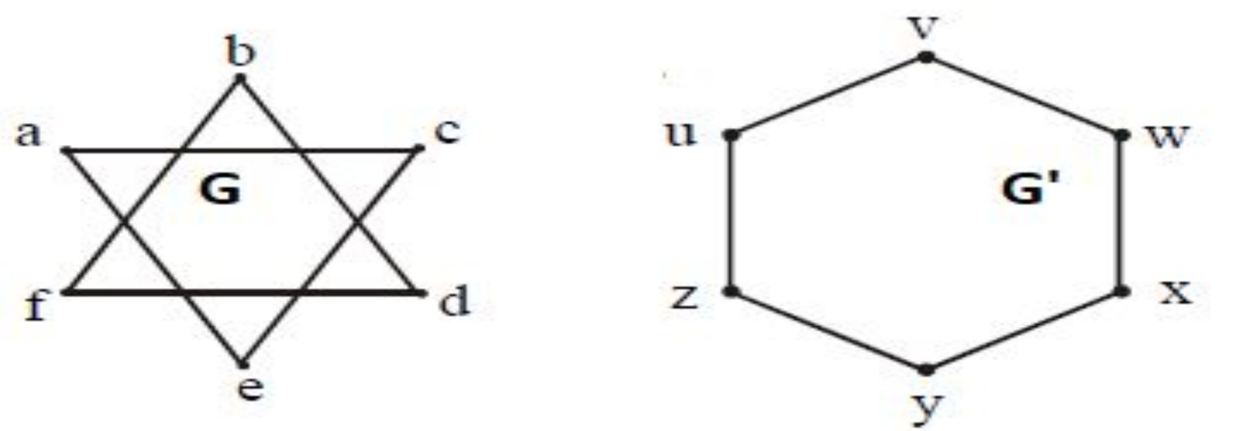


SOLUTION:

Both the graphs have 5 vertices and 7 edges. The vertex q of G’ has degree 5. However G does not have any vertex of degree 5 (so one-one correspondence is not possible). Hence, the two graphs are not isomorphic.

Isomorphism of Graphs

EXAMPLE: Determine whether the graph G and G’ given below are isomorphic.

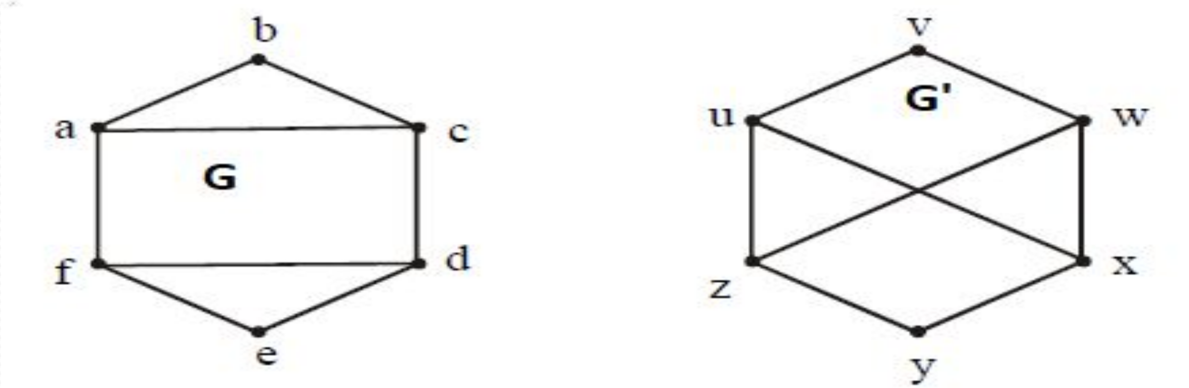


SOLUTION:

Clearly the vertices of both the graphs G and G’ have the same degree (i.e “2”) and having the same number of vertices and edges but isomorphism is not possible. As the graph G’ is a connected graph but the graph G is not connected due to have two components (eca and bdf). Therefore the two graphs are non isomorphic.

Isomorphism of Graphs

EXAMPLE: Determine whether the graph G and G' given below are isomorphic.



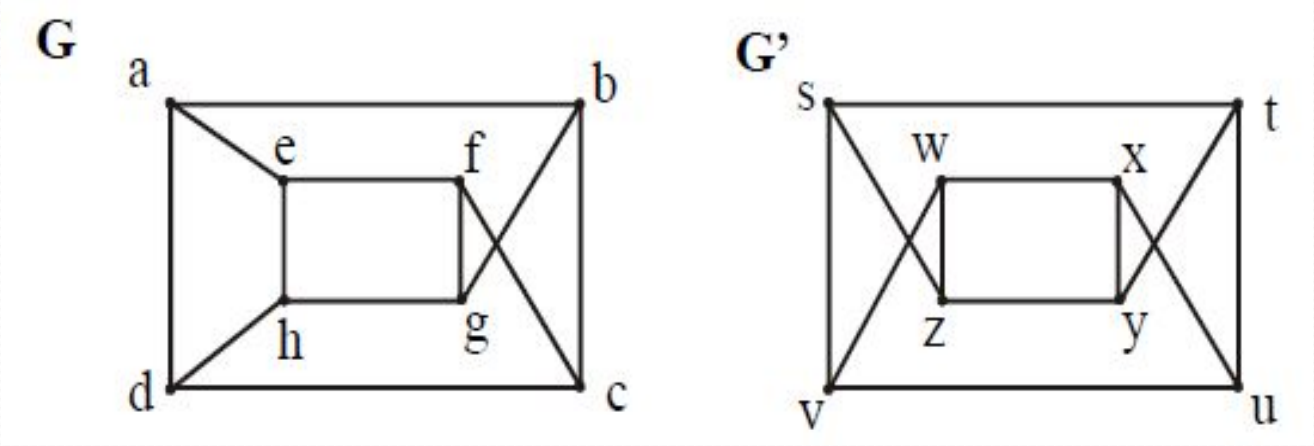
SOLUTION:

Clearly G has six vertices, G' also has six vertices. And the graph G has two simple circuits of length 3; one is abca and the other is defd. But G' does not have any simple circuit of length 3(as one simple circuit in G' is uxwv of length 4).Therefore the two graphs are non-isomorphic.

Note: A simple circuit is a circuit that does not have any other repeated vertex except the first and last.

Isomorphism of Graphs

EXAMPLE: Determine whether the graph G and G' given below are isomorphic.



SOLUTION:

Both the graph G and G' have 8 vertices and 12 edges and both are also called regular graph(as each vertex has degree 3).The graph G has two simple circuits of length 5; abcfea(i.e starts and ends at a) and cdhgfc(i.e starts and ends at c). But G' does not have any simple circuit of length 5 (it has simple circuit tyxut,vwxuv of length 4 etc). Therefore the two graphs are non-isomorphic.

Isomorphism of Graphs

EXAMPLE: Determine whether the given graph G and H are isomorphic.

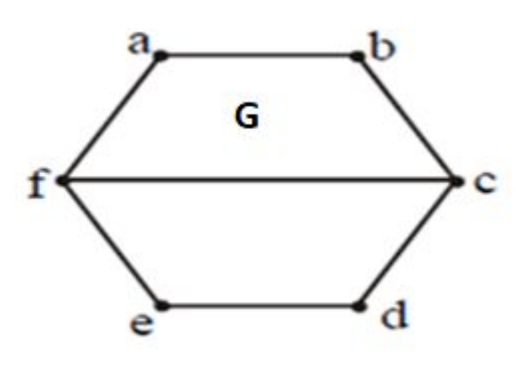
Solution:

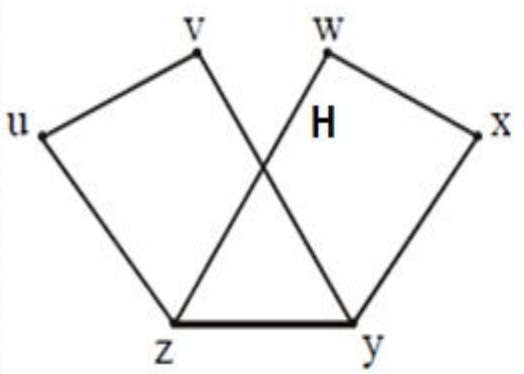
Solution: The function f with

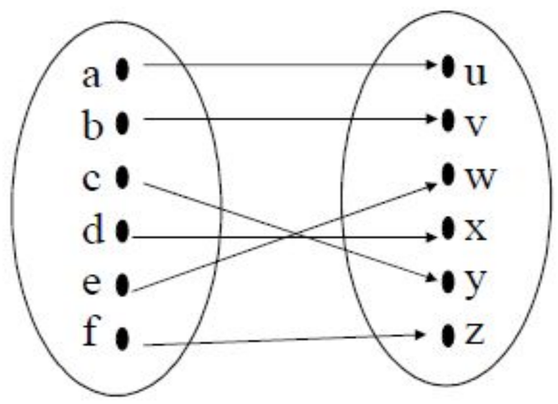
f(a) = u, f(b) = v, f(c) = y, f(d) = x, f(e) = w and f(f) = z is a one-to-one correspondence between G and H.

Note that adjacent vertices in G are a and b, b and c, c and d, c and f, d and e, e and f and f and a.

Each of the pairs f(a) = u and f(b) = v, f(b) = v and f(c) = y , f(c) = y and f(d) = x , f(c) = y and f(f) = z , f(d) = x and f(e) = w, f(e) = w and f(f) = z and f(f) = z and f(a) = u consists of two adjacent vertices in H.

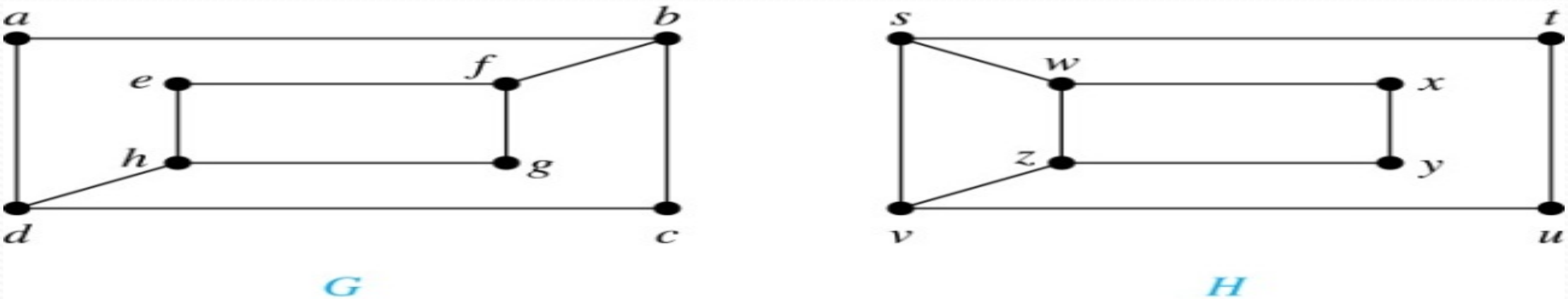






Isomorphism of Graphs (*cont.*)

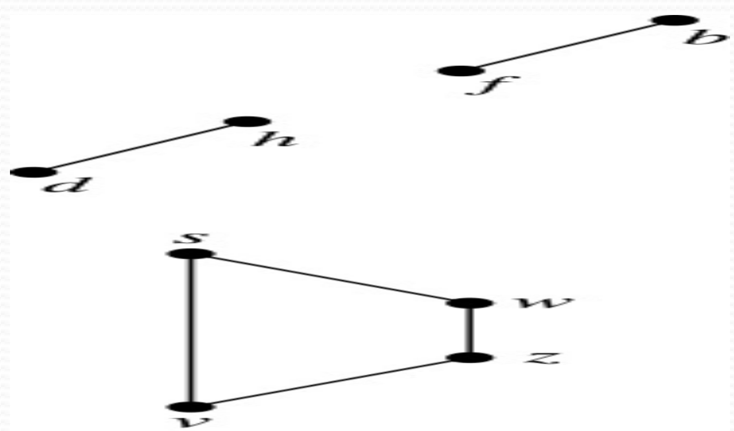
Example: Determine whether these two graphs are isomorphic.



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

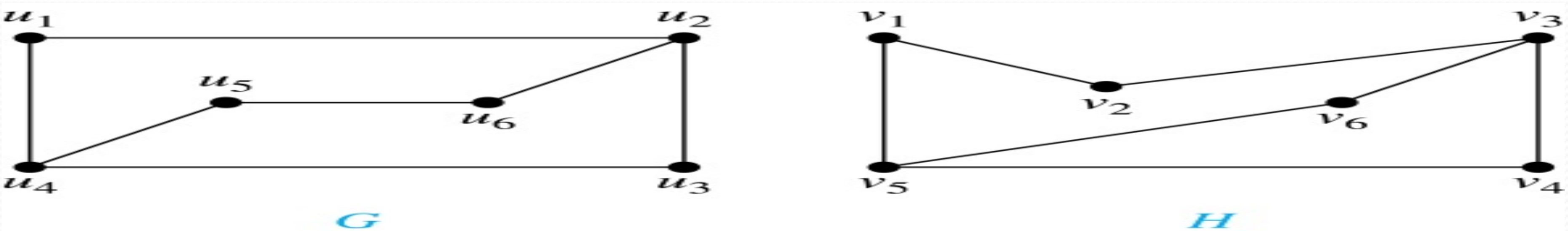
However, G and H are not isomorphic. Note that since $deg(a) = 2$ in G, a must correspond to t, u, x, or y in H, because these are the vertices of degree 2. But each of these vertices is adjacent to another vertex of degree two in H, which is not true for a in G.

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



Isomorphism of Graphs

Example: Determine whether these two graphs 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.

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 isomoprhic.
- 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).

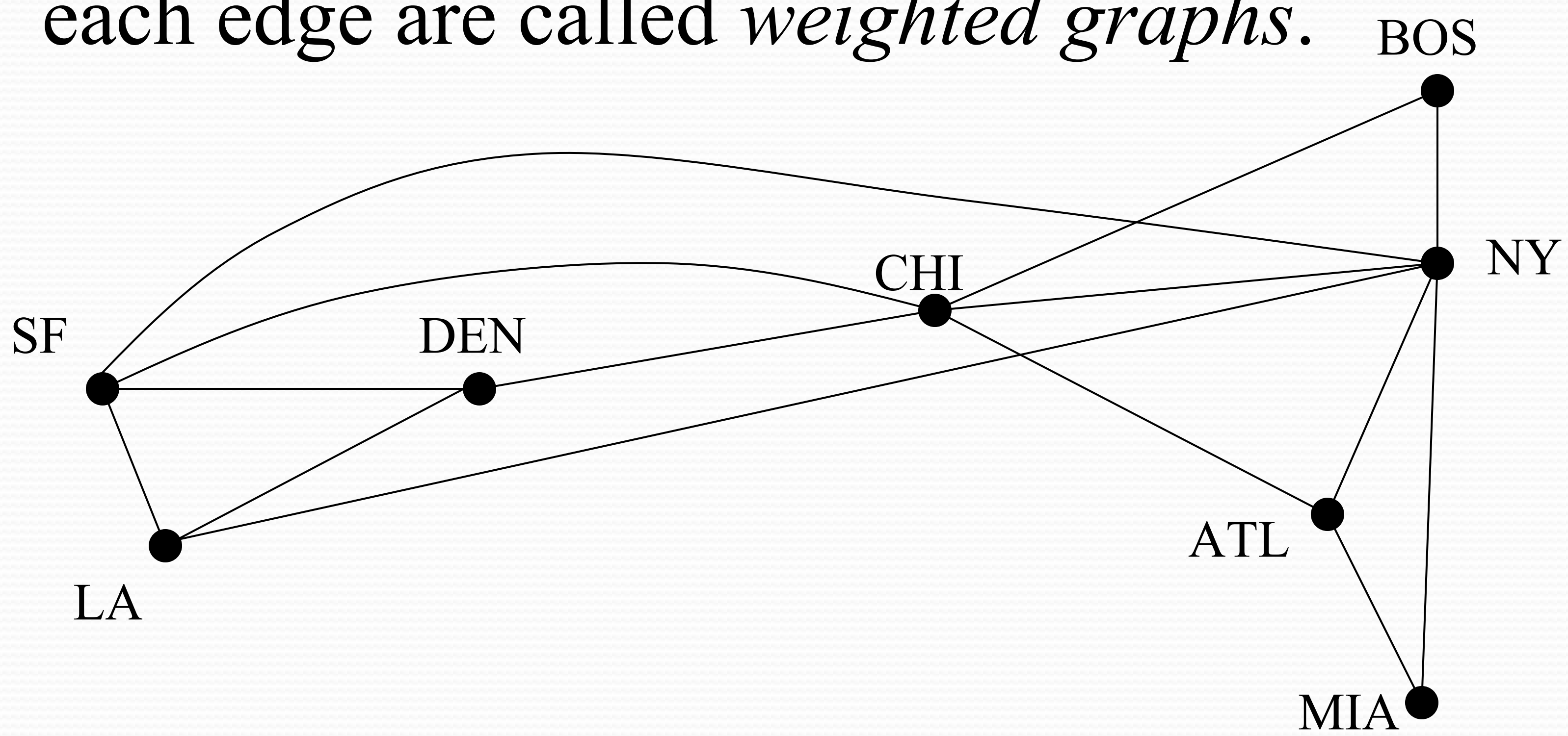
Applications of Graph Isomorphism

- 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.

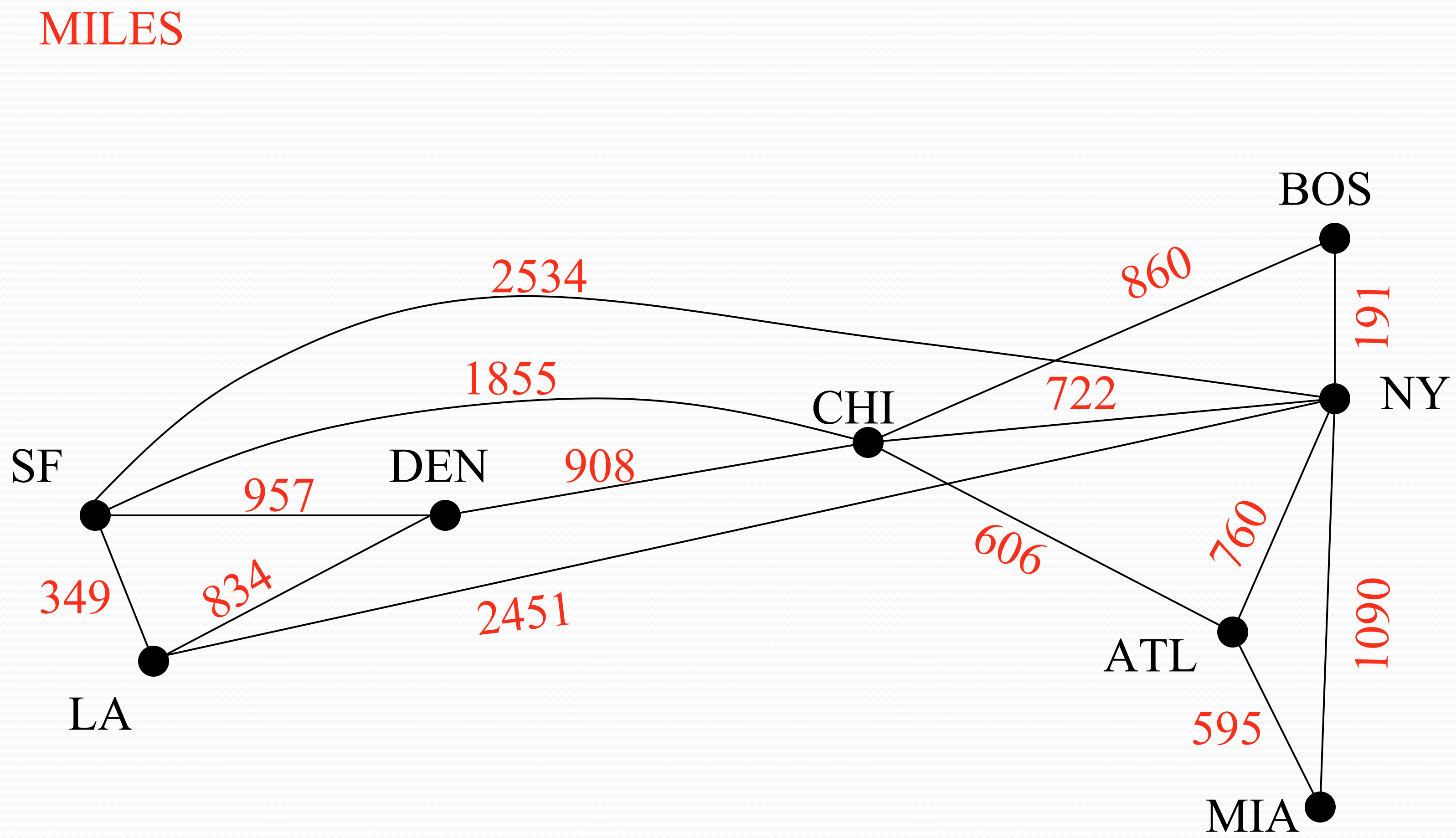
Shortest Paths

Weighted Graphs

Graphs that have a number assigned to each edge are called *weighted graphs*.

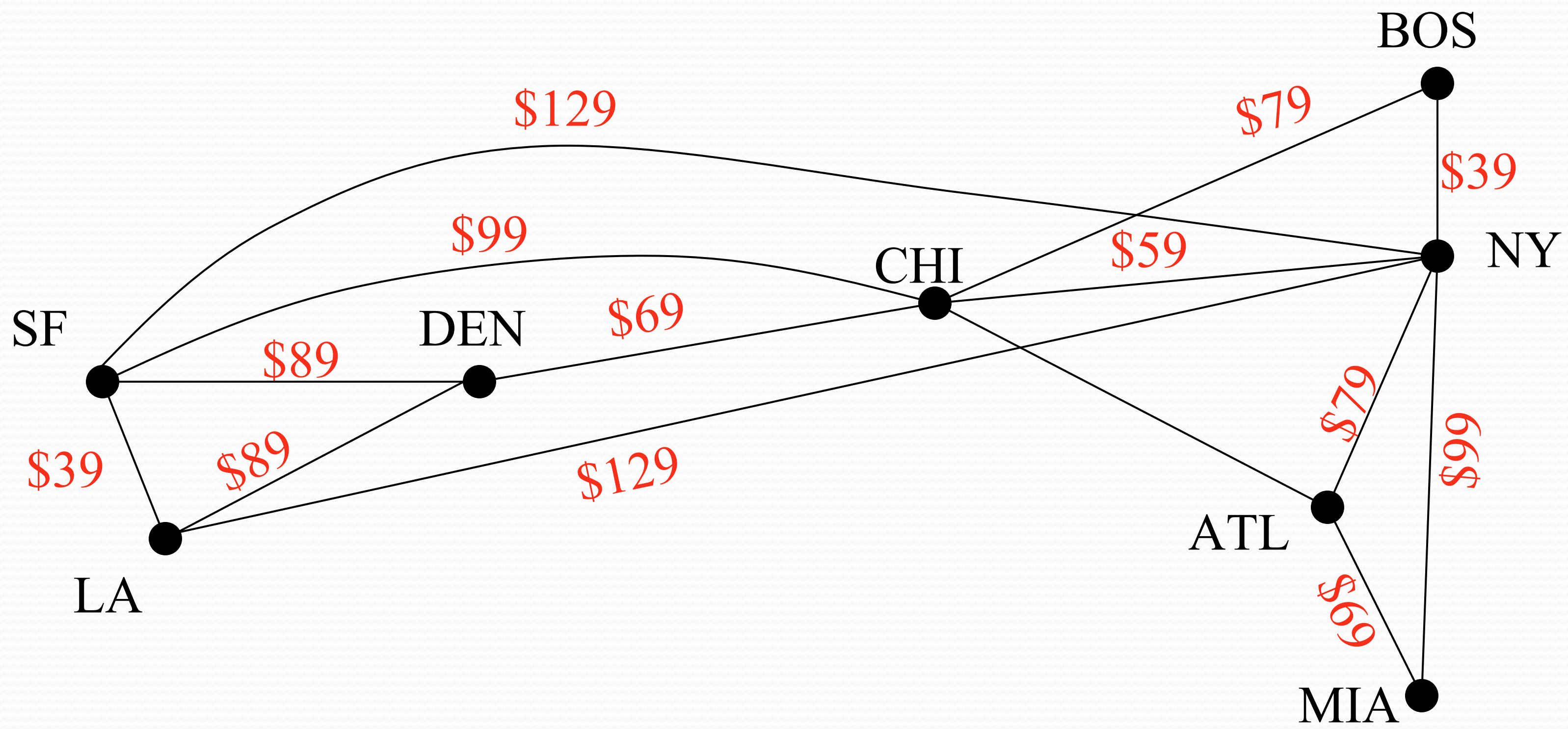


Weighted Graphs

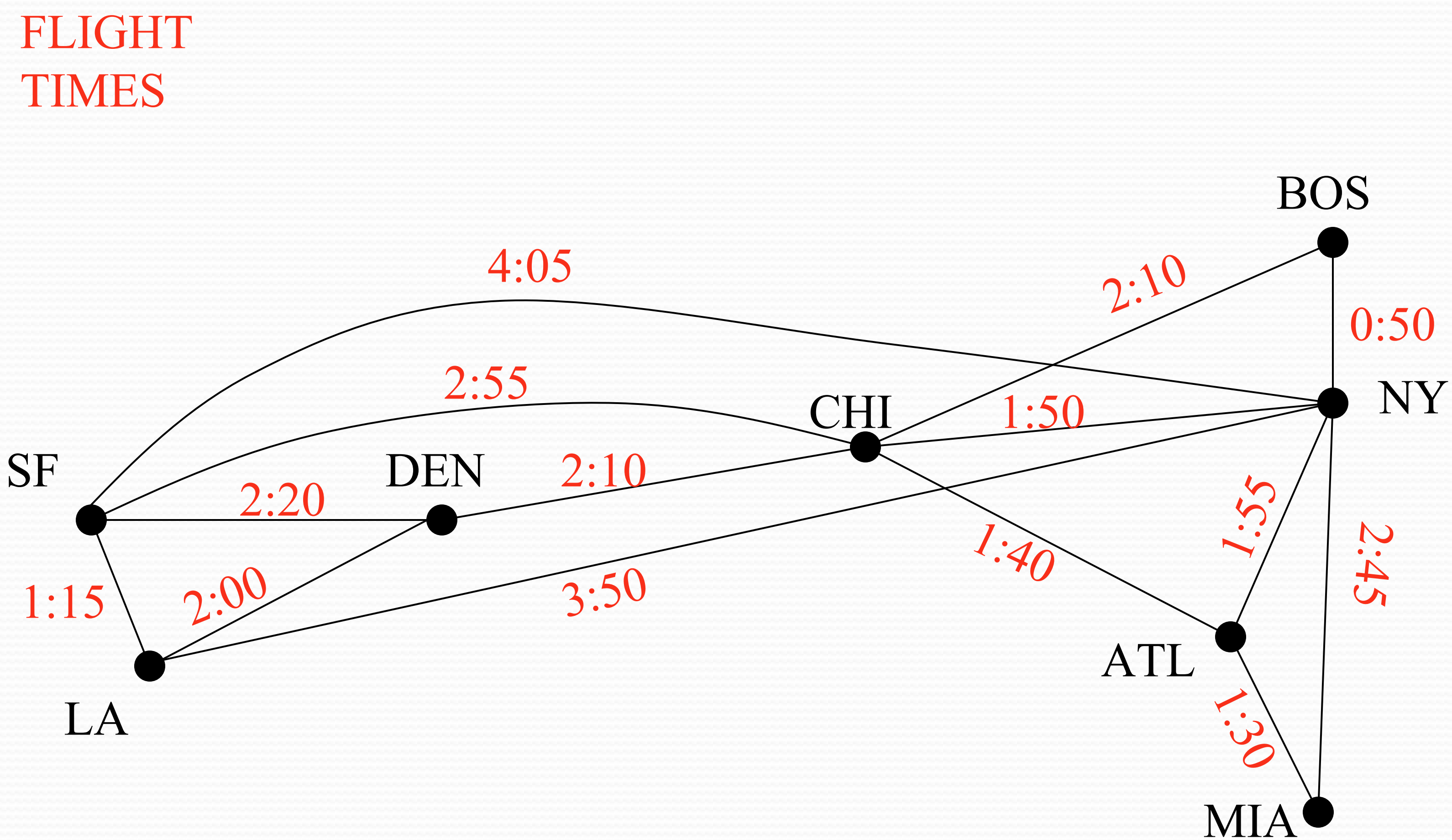


Weighted Graphs

FARES



Weighted Graphs



Weighted Graphs

- A weighted graph is a graph in which each edge (u, v) has a weight w(u, v). Each weight is a real number.
- Weights can represent distance, cost, time, capacity, etc.
- The length of a path in a weighted graph is the sum of the weights on the edges.
- Dijkstra's Algorithm finds the shortest path between two vertices.

Dijkstra's Algorithm

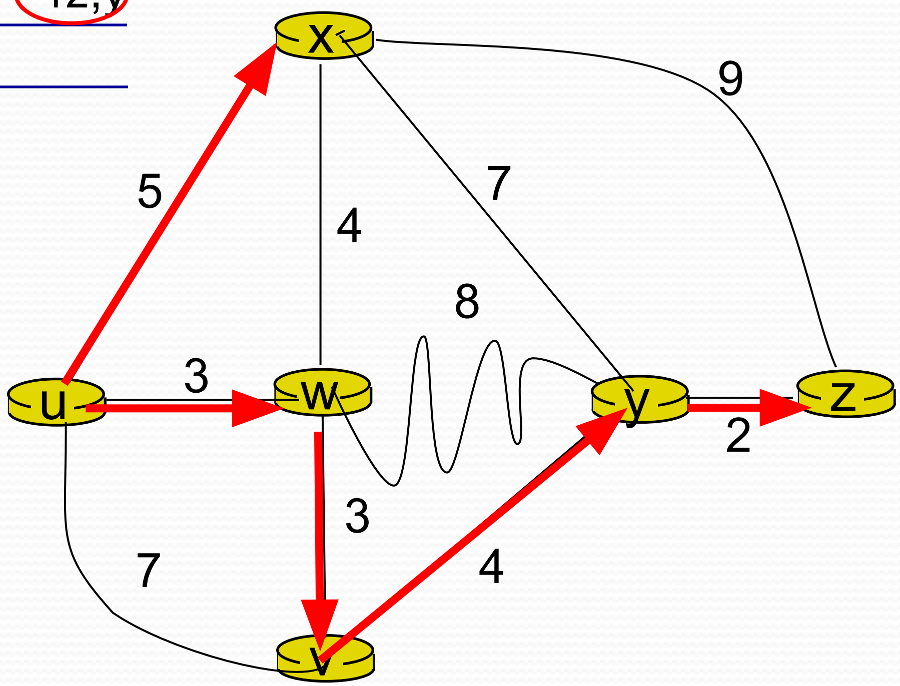
- Dijkstra's algorithm is used in problems relating to finding the shortest path.
- Each node is given a temporary label denoting the length of the shortest path *from* the start node *so far*.
- This label is replaced if another shorter route is found.
- Once it is certain that no other shorter paths can be found, the temporary label becomes a permanent label.
- Eventually all the nodes have permanent labels.
- At this point the shortest path is found by retracing the path backwards.

Dijkstra's algorithm: example

Step	N'	D(v)	D(w)	D(x)	D(y)	D(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

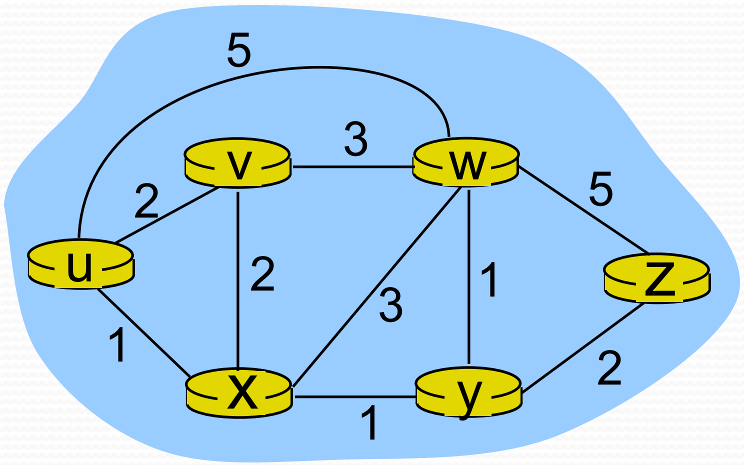
notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



Dijkstra's algorithm: another example

Step	N'	D(v)	D(w)	D(x)	D(y)	D(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



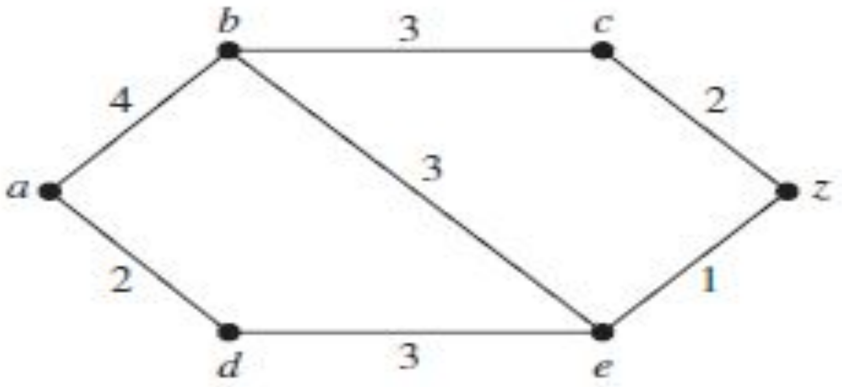
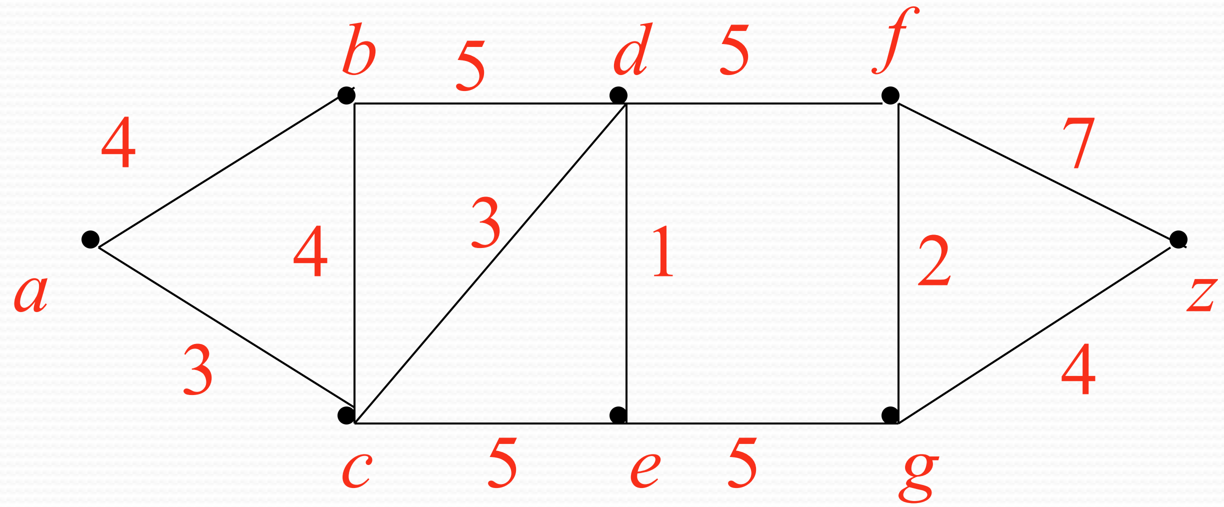


FIGURE 3 **A Weighted Simple Graph.**

What is the length of a shortest path between a and z in the weighted graph shown in Figure 3?

Problem: shortest path from a to z



The Traveling Salesman Problem

- The **traveling salesman problem** is one of the classical problems in computer science.
- A traveling salesman wants to visit a number of cities and then return to his starting point. Of course he wants to save time and energy, so he wants to determine the **shortest cycle** for his trip.
- We can represent the cities and the distances between them by a weighted, complete, undirected graph.
- The problem then is to find the **shortest cycle (of minimum total weight that visits each vertex exactly one)**.
- Finding the shortest cycle is different than Dijkstra's shortest path.
  It is much harder too, no polynomial time algorithm exists!

The Traveling Salesman Problem

- Importance:
  - Variety of scheduling application can be solved as a traveling salesmen problem.
  - Examples:
    - Ordering drill position on a drill press.
    - School bus routing.
  - The problem has theoretical importance because it represents a class of difficult problems known as NP-hard problems.

Travelling Salesman problem

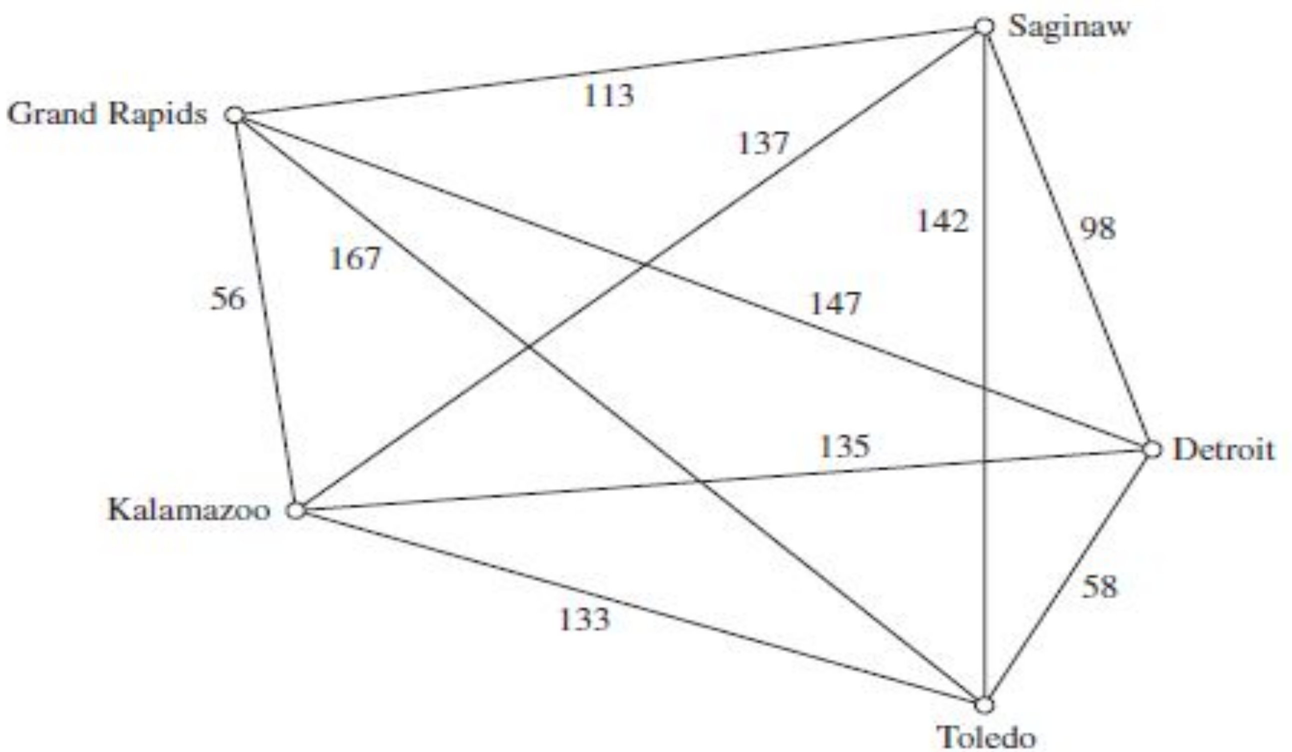


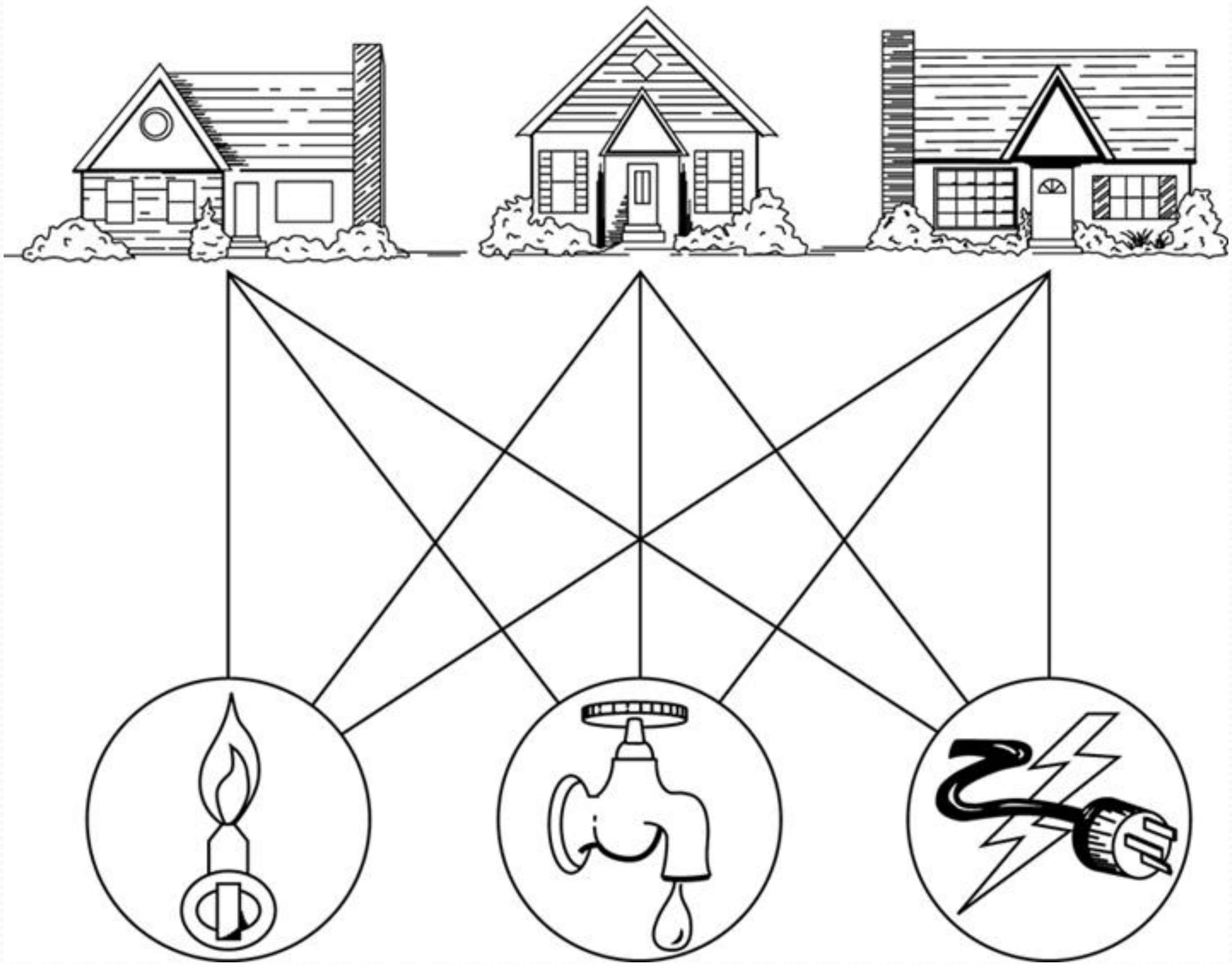
FIGURE 5 **The Graph Showing the Distances between Five Cities.**

Travelling Salesman problem

Route	*Total Distance (miles)*
Detroit–Toledo–Grand Rapids–Saginaw–Kalamazoo–Detroit	610
Detroit–Toledo–Grand Rapids–Kalamazoo–Saginaw–Detroit	516
Detroit–Toledo–Kalamazoo–Saginaw–Grand Rapids–Detroit	588
Detroit–Toledo–Kalamazoo–Grand Rapids–Saginaw–Detroit	458
Detroit–Toledo–Saginaw–Kalamazoo–Grand Rapids–Detroit	540
Detroit–Toledo–Saginaw–Grand Rapids–Kalamazoo–Detroit	504
Detroit–Saginaw–Toledo–Grand Rapids–Kalamazoo–Detroit	598
Detroit–Saginaw–Toledo–Kalamazoo–Grand Rapids–Detroit	576
Detroit–Saginaw–Kalamazoo–Toledo–Grand Rapids–Detroit	682
Detroit–Saginaw–Grand Rapids–Toledo–Kalamazoo–Detroit	646
Detroit–Grand Rapids–Saginaw–Toledo–Kalamazoo–Detroit	670
Detroit–Grand Rapids–Toledo–Saginaw–Kalamazoo–Detroit	728

Planar Graphs

The House-and-Utilities Problem

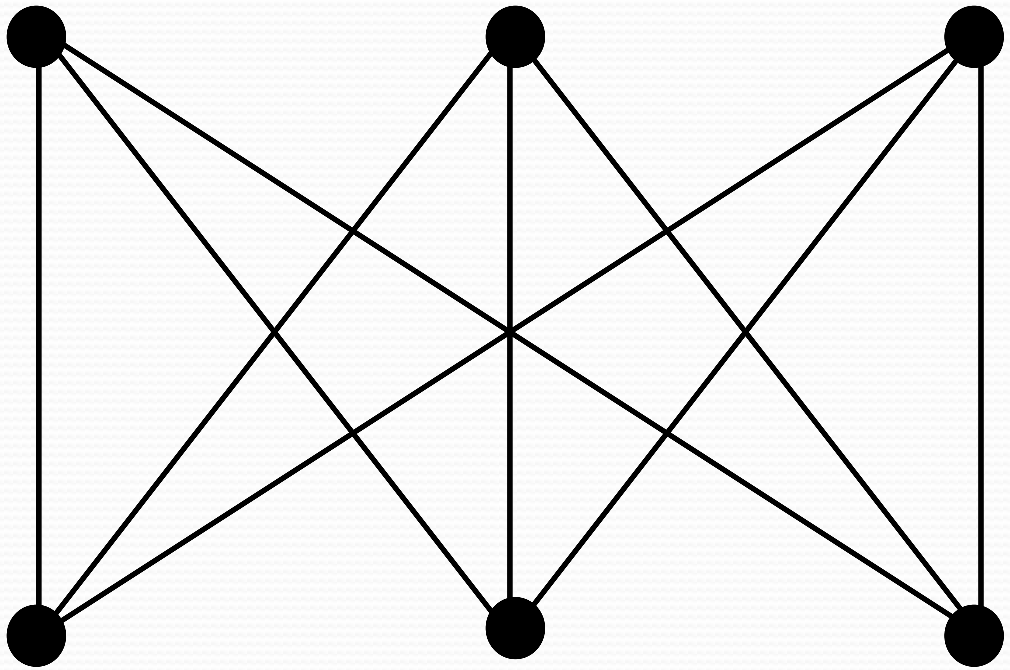


Planar Graphs

- Consider the previous slide. Is it possible to join the three houses to the three utilities in such a way that none of the connections cross?

Planar Graphs

- Phrased another way, this question is equivalent to: Given the complete bipartite graph $K_{3,3}$, can $K_{3,3}$ be drawn in the plane so that no two of its edges cross?



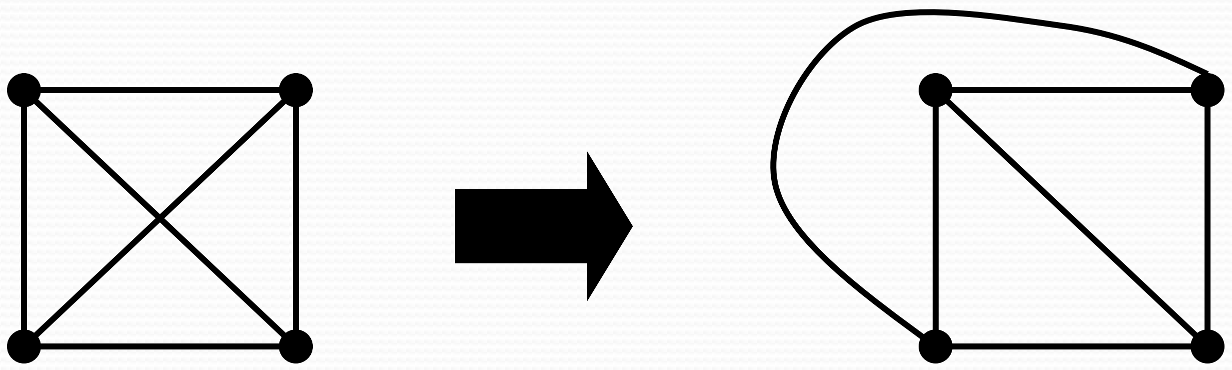
$K_{3,3}$

Planar Graphs

- A graph is called *planar* if it can be drawn in the plane without any edges crossing.
- A crossing of edges is the intersection of the lines or arcs representing them at a point other than their common endpoint.
- Such a <u>drawing</u> is called a *planar representation* of the graph.

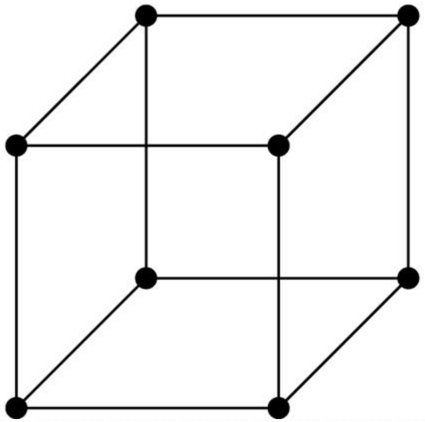
Example

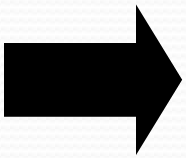
A graph may be planar even if it is usually drawn with crossings, since it may be possible to draw it in another way without crossings.

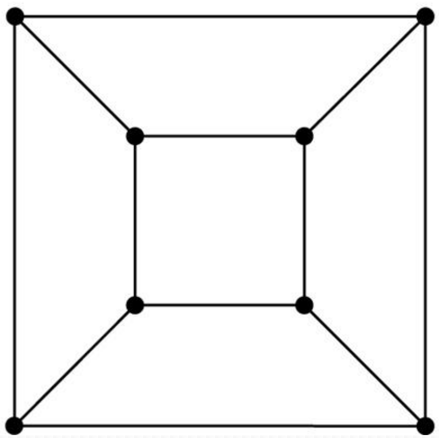


Example

A graph may be planar even if it represents a 3-dimensional object.





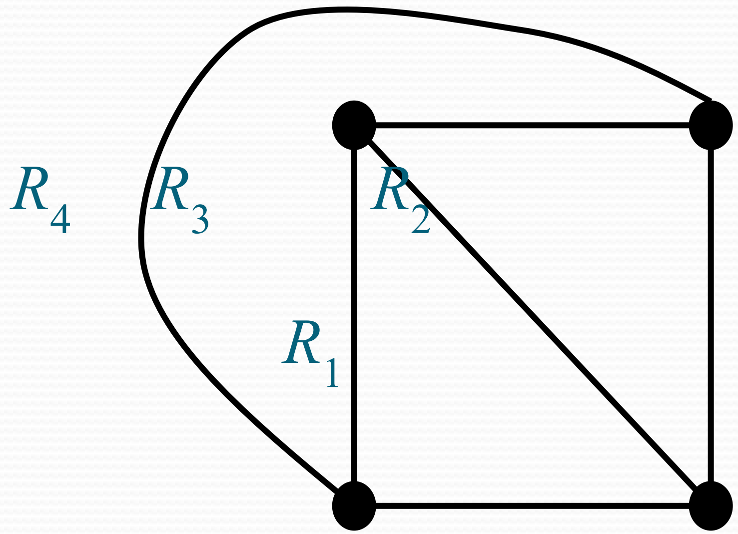


Planar Graphs

- We can prove that a particular graph is planar by showing how it can be drawn without any crossings.
- However, not all graphs are planar.
- It may be difficult to show that a graph is nonplanar. We would have to show that there is *no way* to draw the graph without any edges crossing.

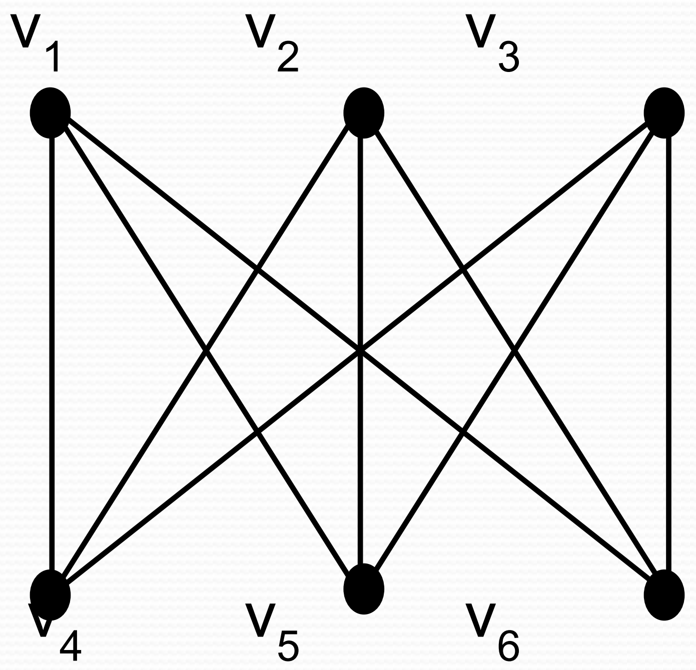
Regions

- Euler showed that all planar representations of a graph split the plane into the same number of *regions*, including an unbounded region.



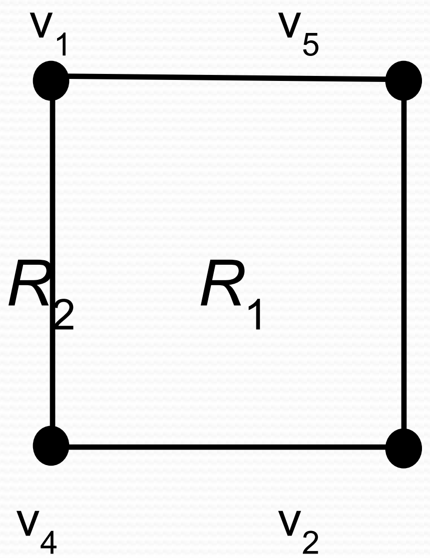
Regions

- In any planar representation of $K_{3,3}$, vertex v_1 must be connected to both v_4 and v_5, and v_2 also must be connected to both v_4 and v_5.



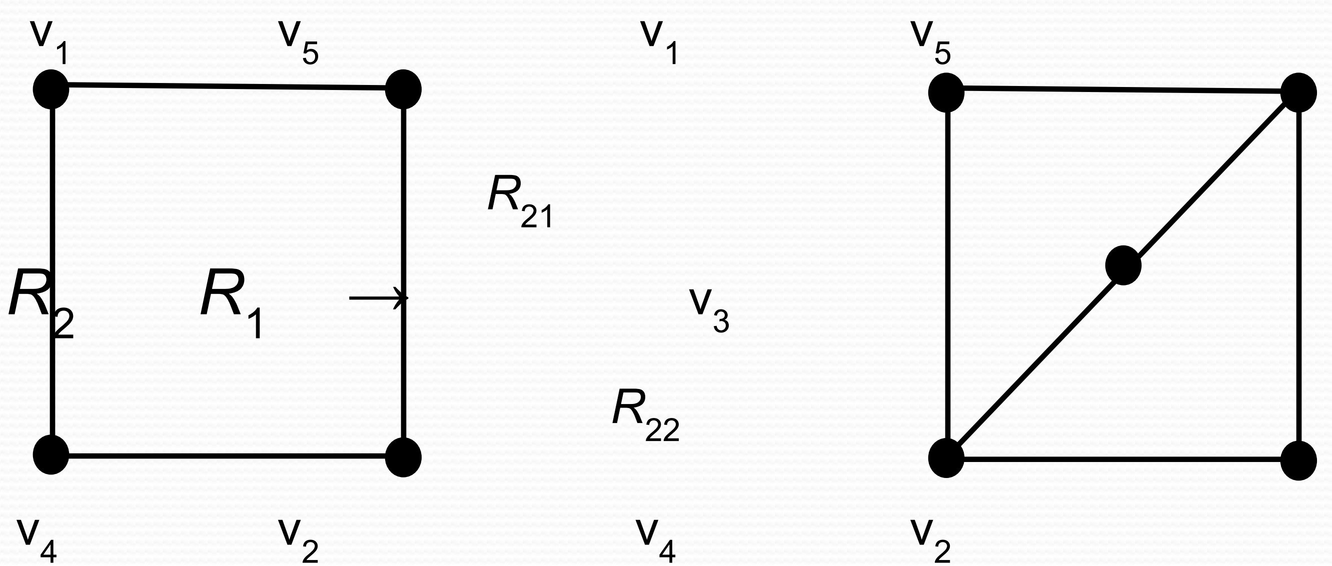
Regions

- The four edges $\{v_1, v_4\}$, $\{v_4, v_2\}$, $\{v_2, v_5\}$, $\{v_5, v_1\}$ form a closed curve that splits the plane into two regions, R_1 and R_2.



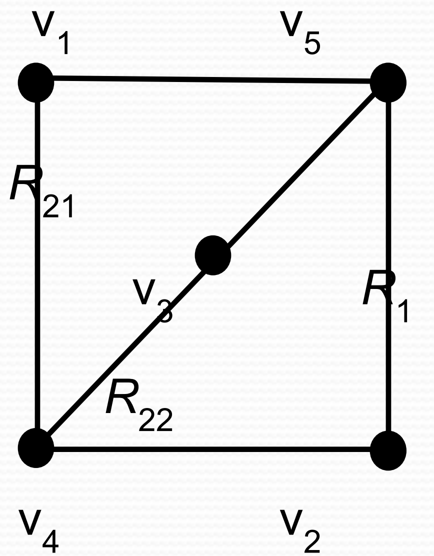
Regions

- Next, we note that v_3 must be in either R_1 or R_2.
- Assume v_3 is in R_2. Then the edges $\{v_3, v_4\}$ and $\{v_4, v_5\}$ separate R_2 into two subregions, R_{21} and R_{22}.



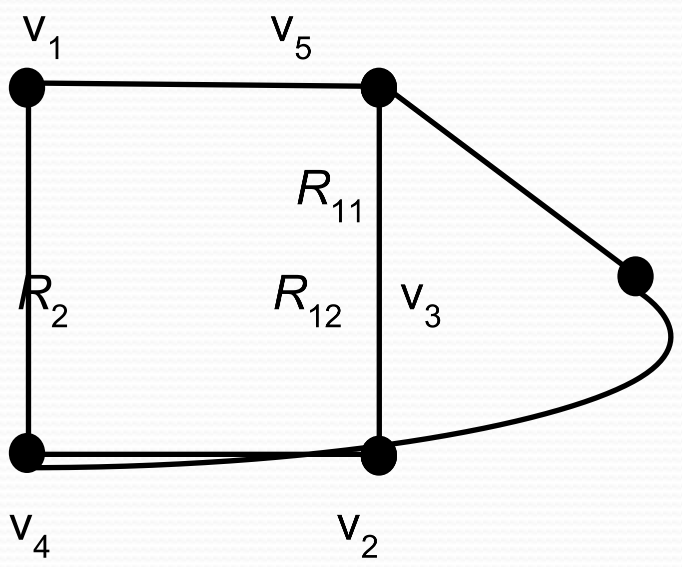
Regions

- Now there is no way to place vertex v_6 without forcing a crossing:
  - If v_6 is in R_1 then $\{v_6, v_3\}$ must cross an edge
  - If v_6 is in R_{21} then $\{v_6, v_2\}$ must cross an edge
  - If v_6 is in R_{22} then $\{v_6, v_1\}$ must cross an edge



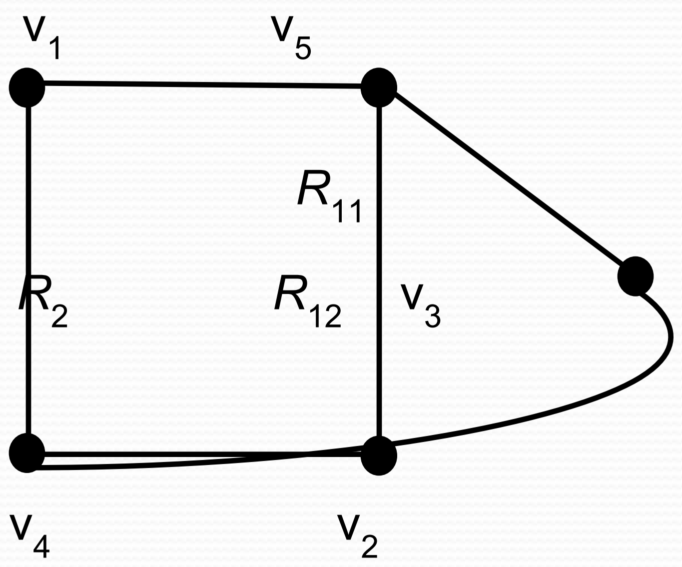
Regions

- Alternatively, assume v_3 is in R_1. Then the edges $\{v_3, v_4\}$ and $\{v_4, v_5\}$ separate R_1 into two subregions, R_{11} and R_{12}.



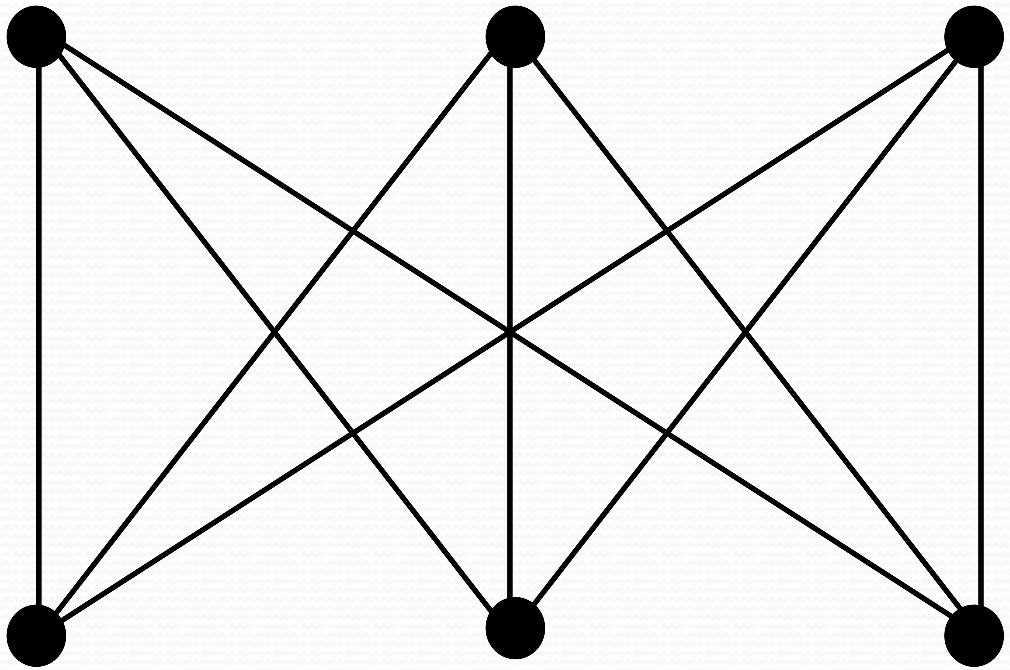
Regions

- Now there is no way to place vertex v_6 without forcing a crossing:
  - If v_6 is in R_2 then $\{v_6, v_3\}$ must cross an edge
  - If v_6 is in R_{11} then $\{v_6, v_2\}$ must cross an edge
  - If v_6 is in R_{12} then $\{v_6, v_1\}$ must cross an edge



Planar Graphs

- Consequently, the graph $K_{3,3}$ must be nonplanar.



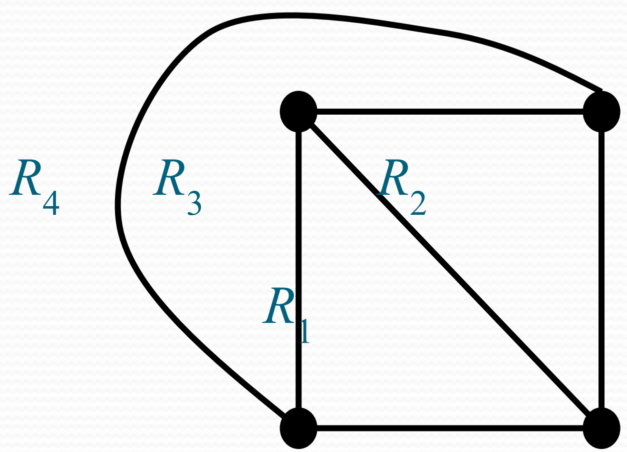
$K_{3,3}$

Regions

- Euler devised a formula for expressing the relationship between the number of vertices, edges, and regions of a planar graph.
- These *may* help us determine if a graph can be planar or not.

Euler's Formula

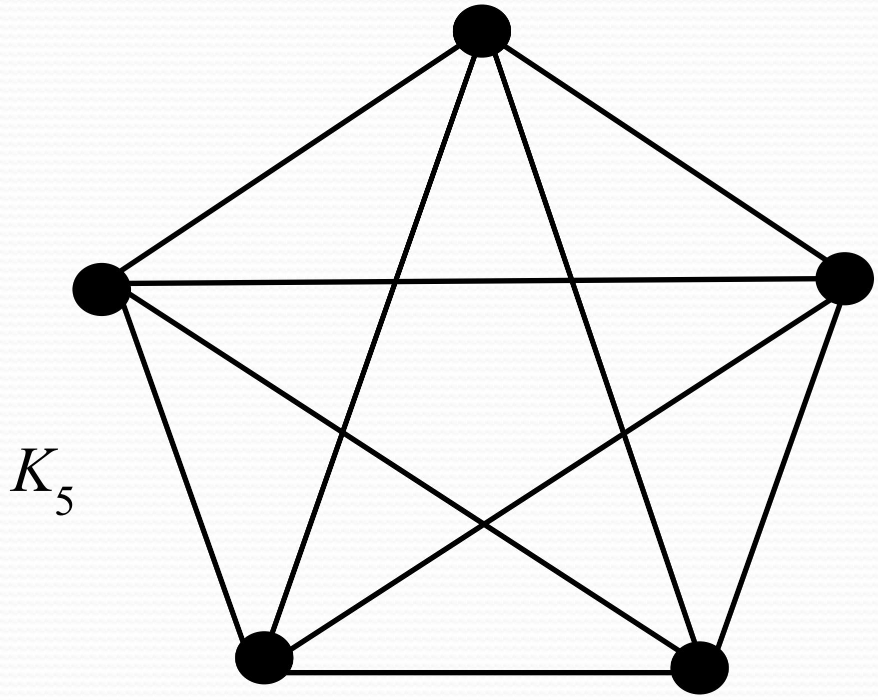
- Let G be a connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then $r = e - v + 2$.



# of edges, $e = 6$
# of vertices, $v = 4$
# of regions, $r = e - v + 2 = 4$

Euler's Formula (Cont.)

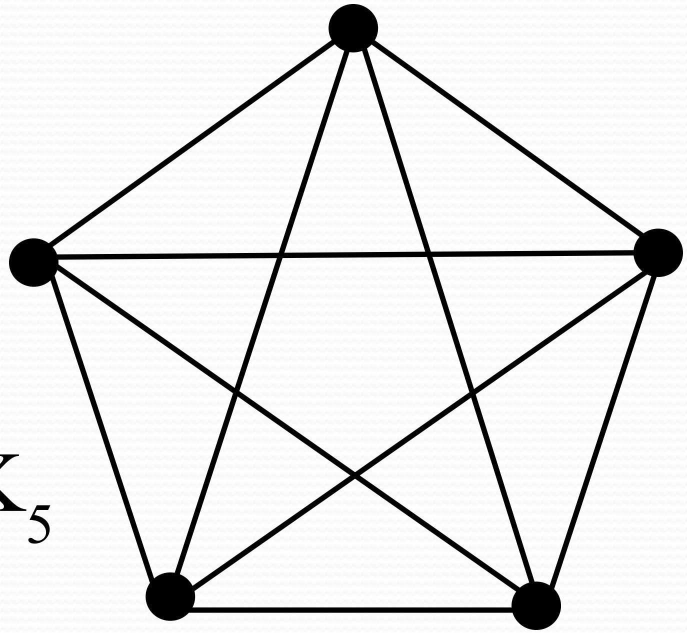
- Corollary 1: If G is a connected planar simple graph with e edges and v vertices where $v \geq 3$, then $e \leq 3v - 6$.
- Is K_5 planar?



Euler's Formula (Cont.)

- K_5 has 5 vertices and 10 edges.
- We see that $v \geq 3$.
- So, if K_5 is planar, it must be true that $e \leq 3v - 6$.
- $3v - 6 = 3*5 - 6 = 15 - 6 = 9$.
- So e must be ≤ 9.
- But $e = 10$.
- So, K_5 is nonplanar.

K_5



Euler's Formula (Cont.)

- Corollary 2: If G is a connected planar simple graph, then G must have a vertex of degree not exceeding 5.

If G has one or two vertices, it is true;
thus, we assume that G has at least three vertices.

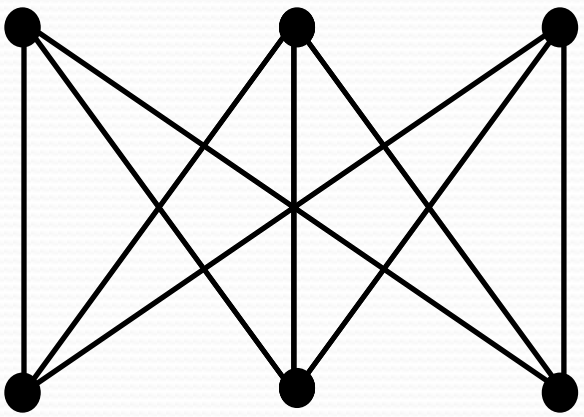
If the degree of each vertex were at least 6, then by Handshaking Theorem,
2e ≥ 6v, i.e., e ≥ 3v,

but this contradicts the inequality from
Corollary 1: 1 e ≤ 3v – 6.

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

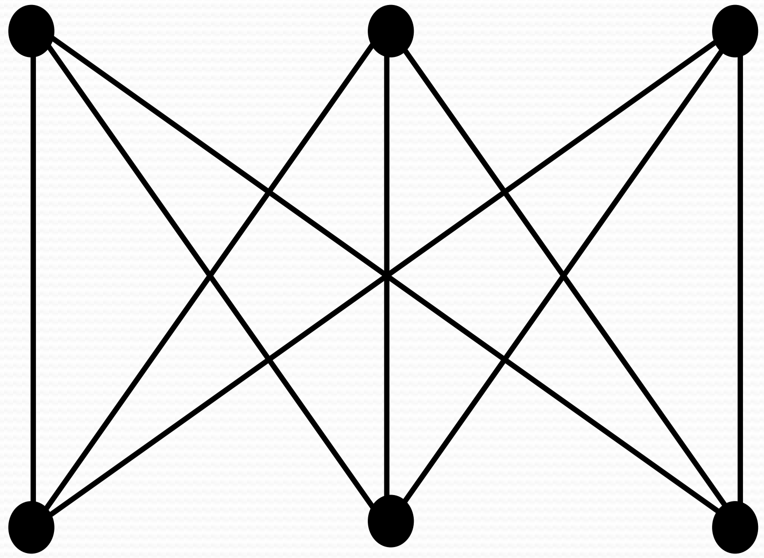
Euler's Formula (Cont.)

- Corollary 3: If a connected planar simple graph has e edges and v vertices with $v \geq 3$ and no circuits of length 3, then $e \leq 2v - 4$.
- Is $K_{3,3}$ planar?



Euler's Formula (Cont.)

- $K_{3,3}$ has 6 vertices and 9 edges.
- Obviously, v ≥ 3 and there are no circuits of length 3.
- If $K_{3,3}$ were planar, then $e \leq 2v - 4$ would have to be true.
- $2v - 4 = 2*6 - 4 = 8$
- So e must be ≤ 8.
- But $e = 9$.
- So $K_{3,3}$ is nonplanar.



$K_{3,3}$

Trees

Chapter 11

Chapter Summary

- Introduction to Trees
- Applications of Trees
- Tree Traversal
- Spanning Trees
- Minimum Spanning

Introduction to Trees

Section 11.1

Section Summary

- Introduction to Trees
- Rooted Trees
- Trees as Models
- Properties of Trees

Trees

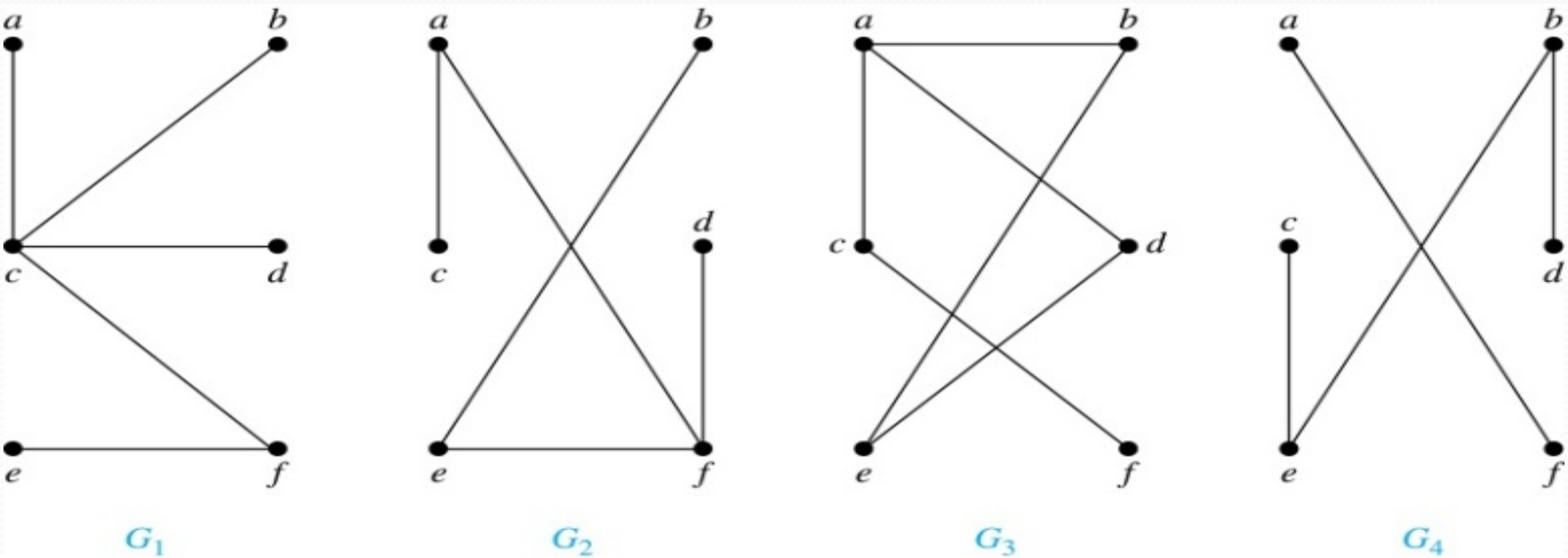
Definition: A *tree* is a connected undirected graph with no simple circuits.

Definition: An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices. A tree cannot contain multiple edges or loops.

Definition: An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

Trees

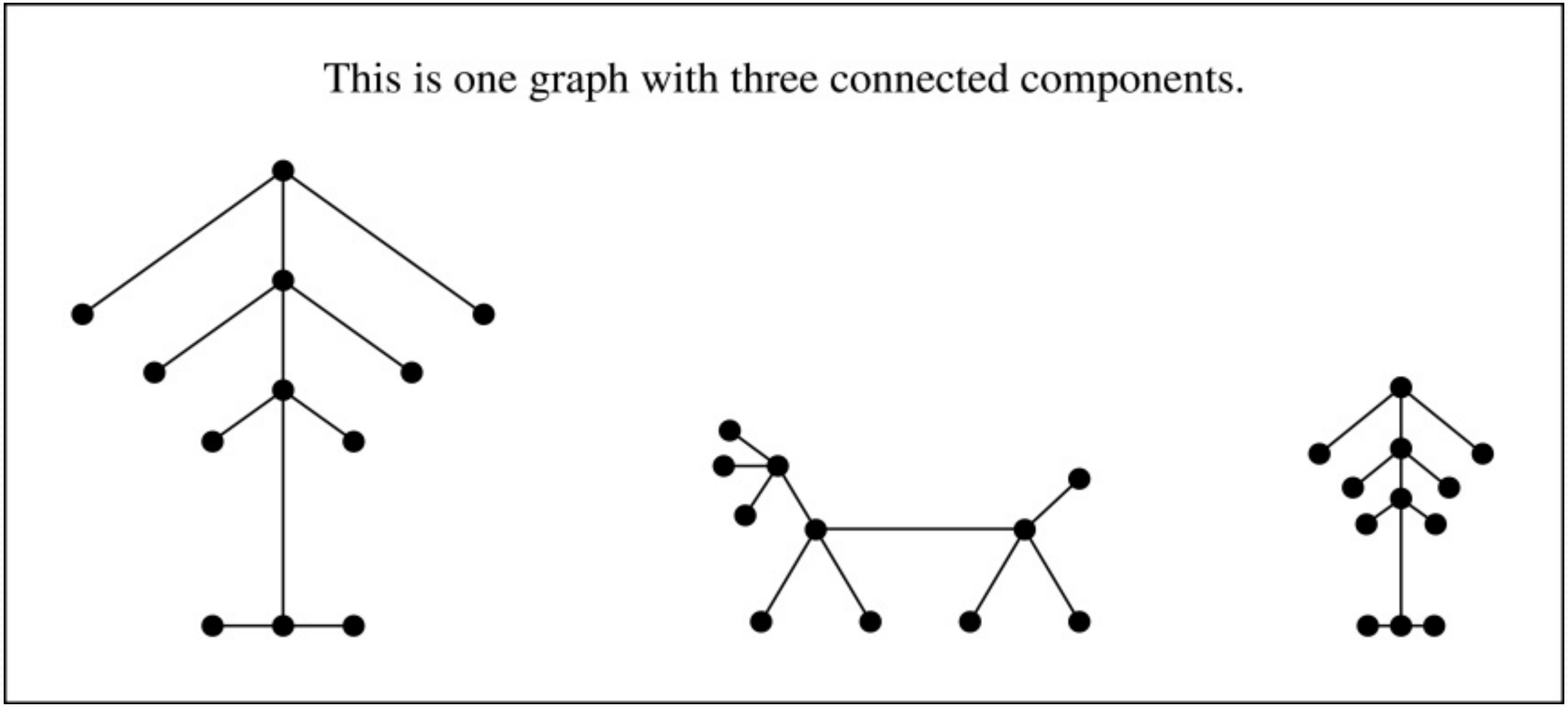
Example: Which of these graphs are trees?



Solution: G_1 and G_2 are trees - both are connected and have no simple circuits. G_3 is not a tree because e, b, a, d, e is a simple circuit,. G_4 is not a tree because it is not connected.

FOREST

Definition: A *forest* is a graph that has no simple circuit, but is not connected. Each of the connected components in a forest is a tree.

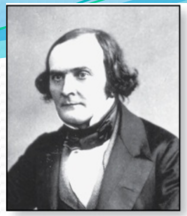


Applications of Trees

Section 11.2

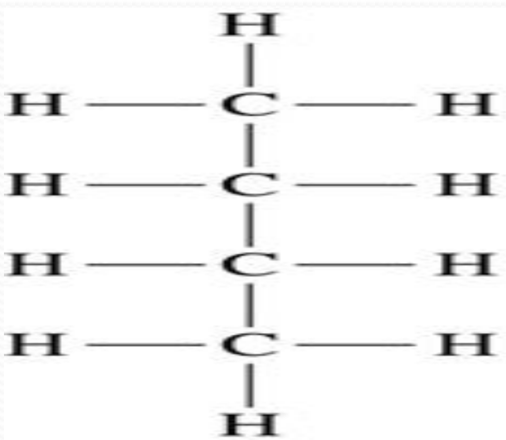
Trees as Models

Arthur Cayley
(1821-1895)

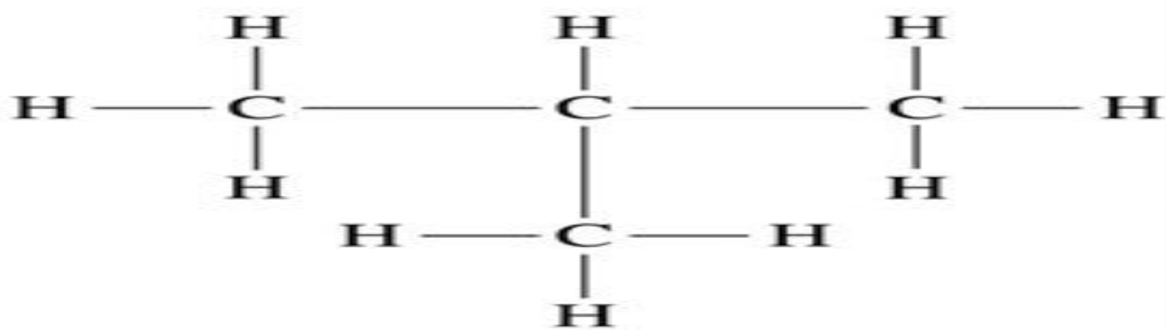


- Trees are used as models in computer science, chemistry, geology, botany, psychology, and many other areas.

- Trees were introduced by the mathematician Cayley in 1857 in his work counting the number of isomers of saturated hydrocarbons. The two isomers of butane are:



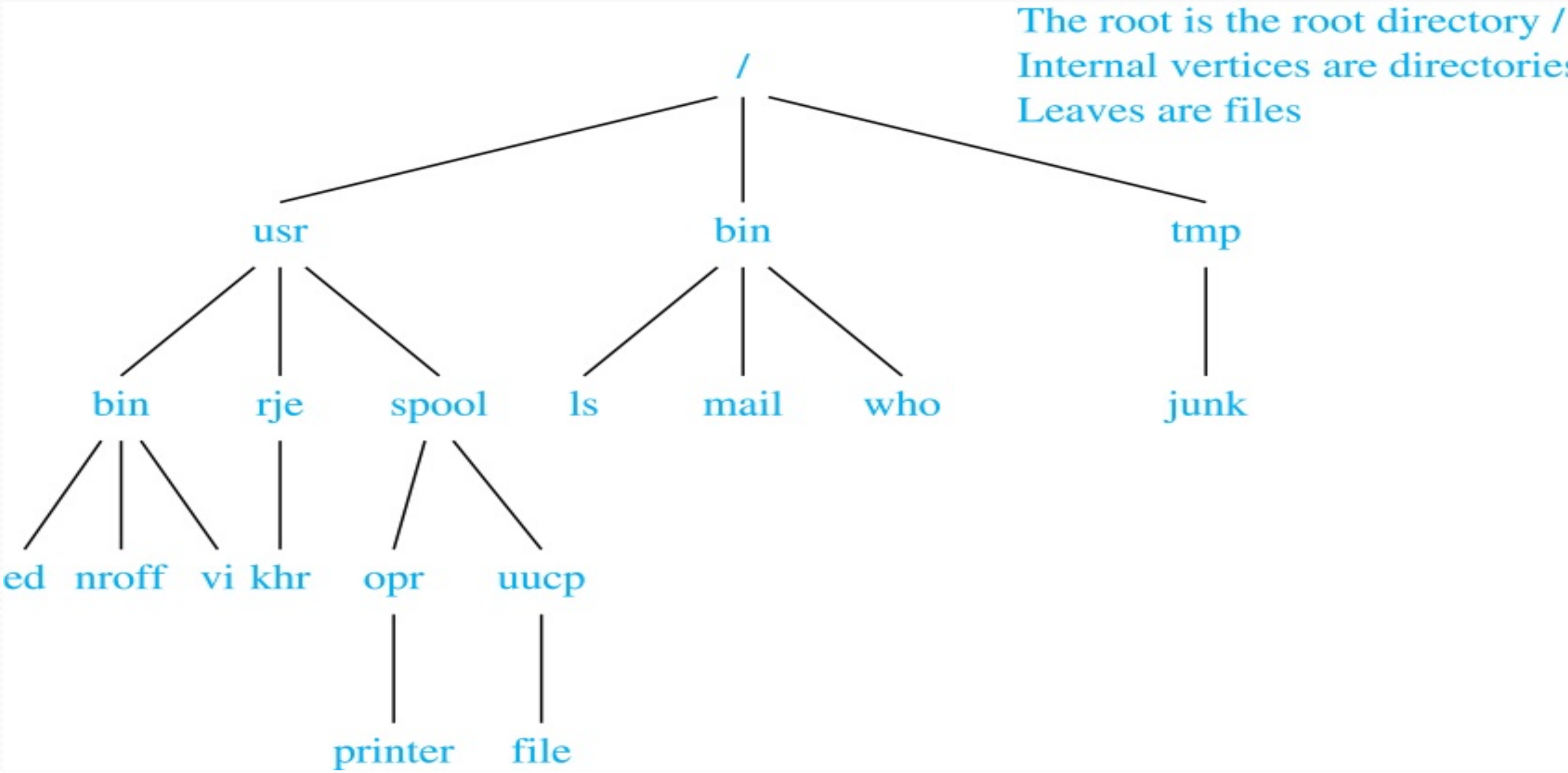
Butane



Isobutane

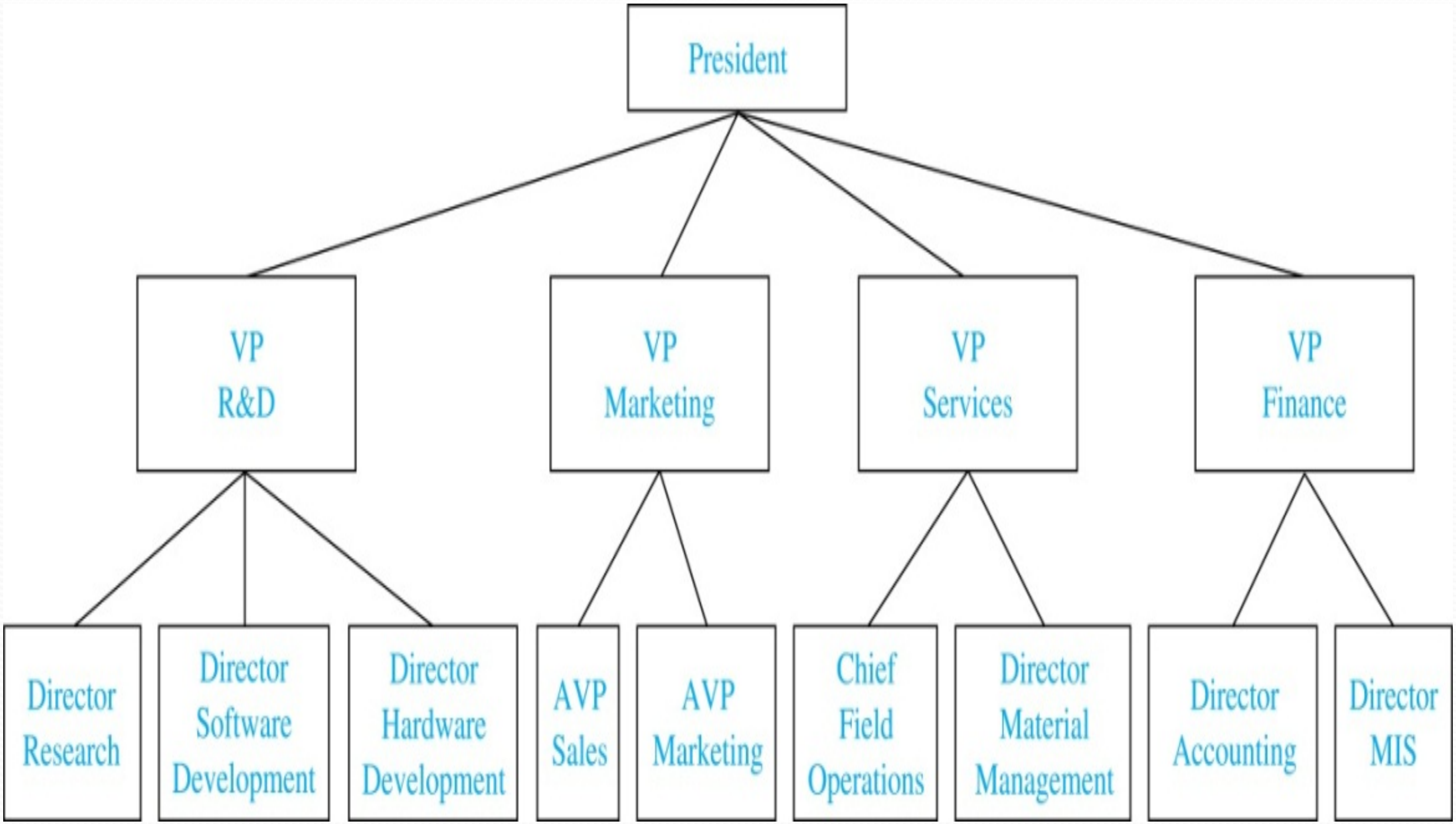
Trees as Models

- The organization of a computer file system into directories, subdirectories, and files is naturally represented as a tree.



Trees as Models

- Trees are used to represent the structure of organizations.

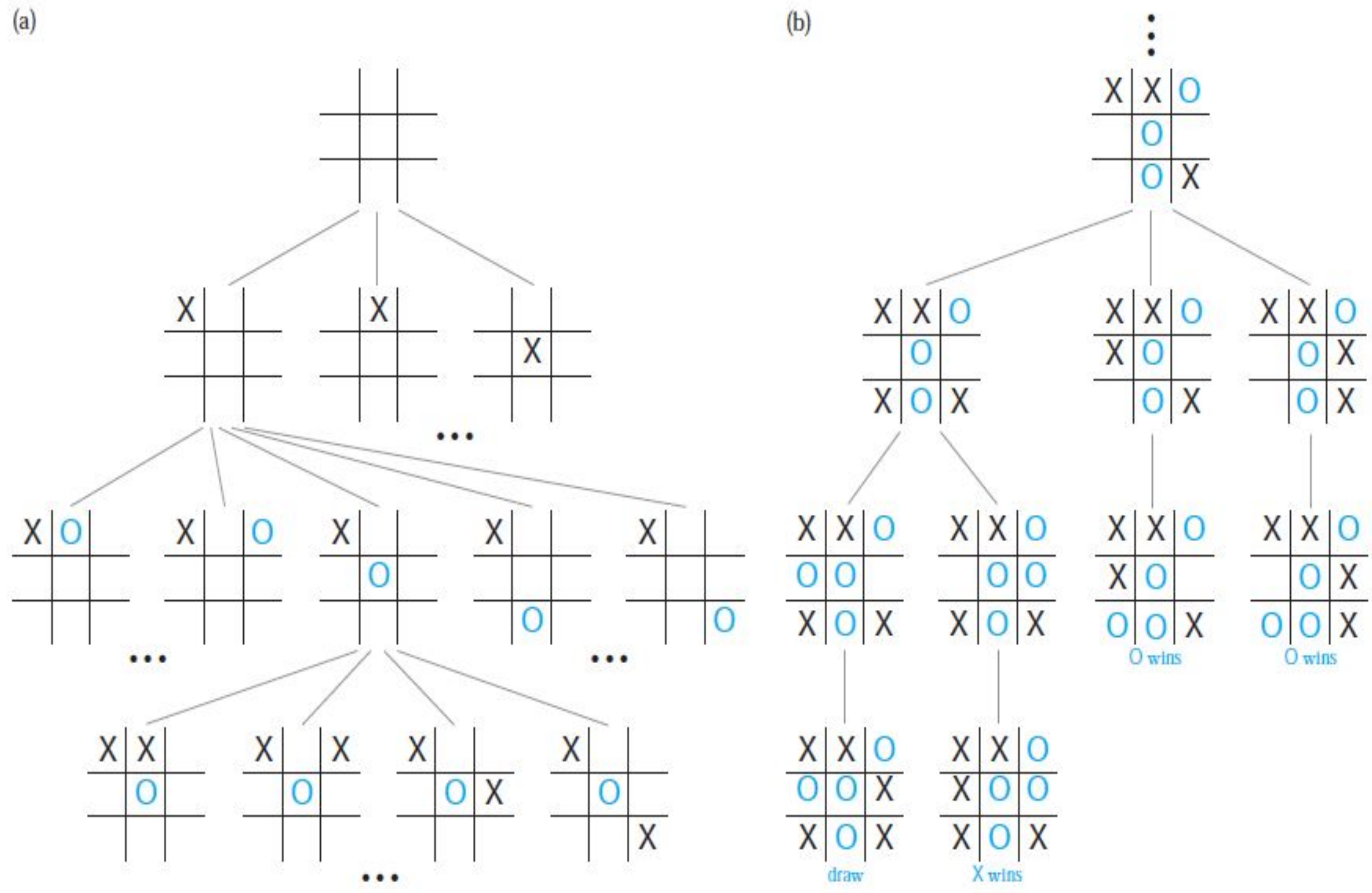


Applications of Trees

- **Game Trees**

Trees can be used to analyze certain types of games such as tic-tac-toe, nim, checkers, and chess.

Game Tree for Tic-Tac-Toe



Universal Address Systems

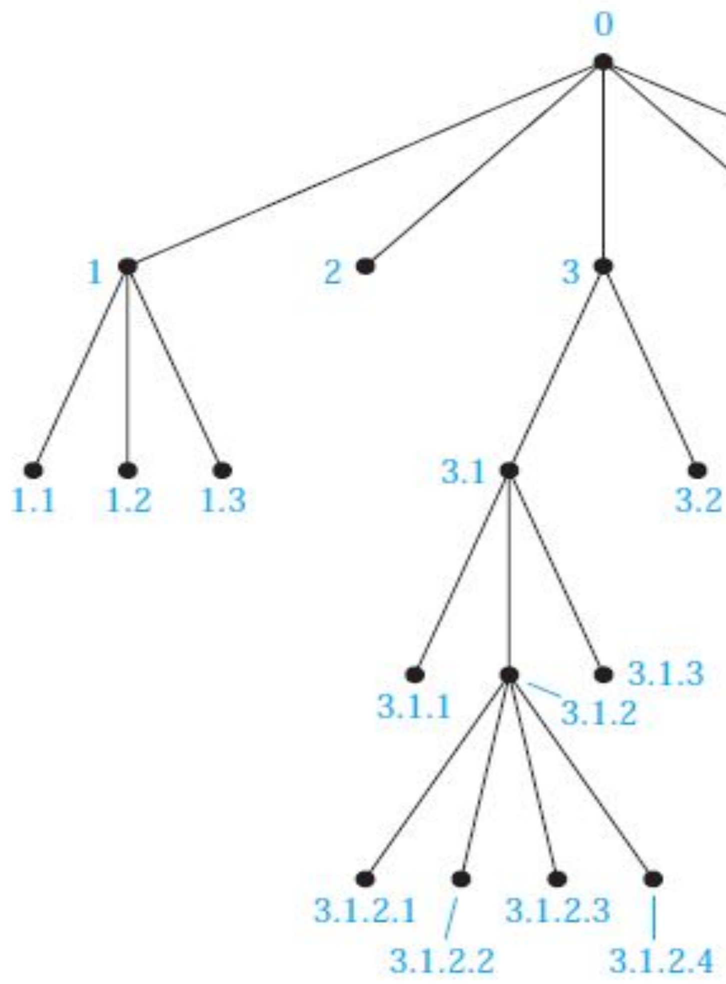


FIGURE 1 The Universal Address System of an Ordered Rooted Tree.

Prefix code

Definition: A code that has the property that the code of a character is never a prefix of the code of another character.

- A prefix code can be represented using a binary tree, where the characters are the labels of the leaves in the tree.
- The edges of the tree are labeled so that an edge leading to a left child is assigned a 0 and an edge leading to a right child is assigned a 1.
- The bit string used to encode a character is the sequence of labels of the edges in the unique path from the root to the leaf that has this character as its label.
- For instance, the tree in Figure 5 represents the encoding of *e* by 0, *a* by 10, *t* by 110, *n* by 1110, and *s* by 1111.

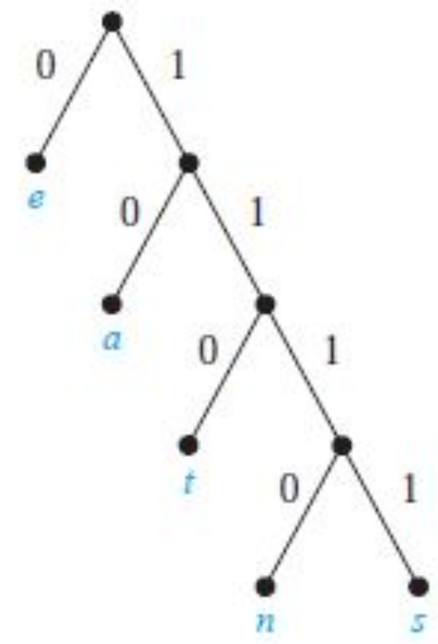


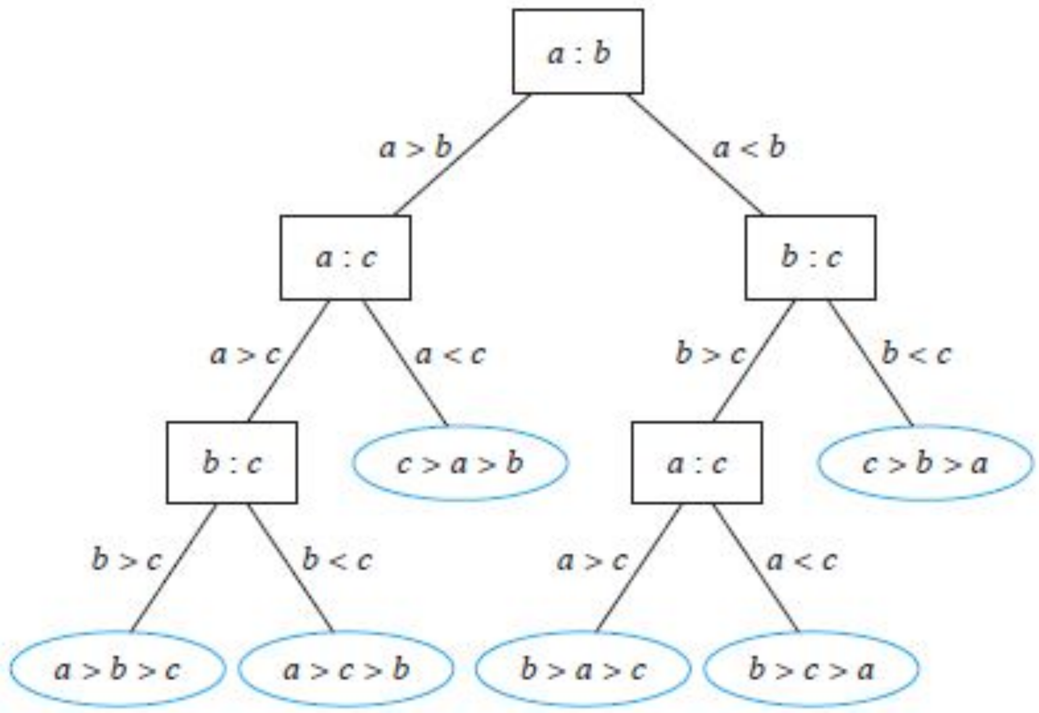
FIGURE 5 A Binary Tree with a Prefix Code.

Decision Trees

Definition: A rooted tree where each vertex represents a possible outcome of a decision and the leaves represent the possible solutions of a problem.

- Rooted trees can be used to model problems in which a series of decisions leads to a solution.
- The possible solutions of the problem correspond to the paths to the leaves of this rooted tree.

Example : A decision tree that orders the elements of the list *a*, *b*, *c*.

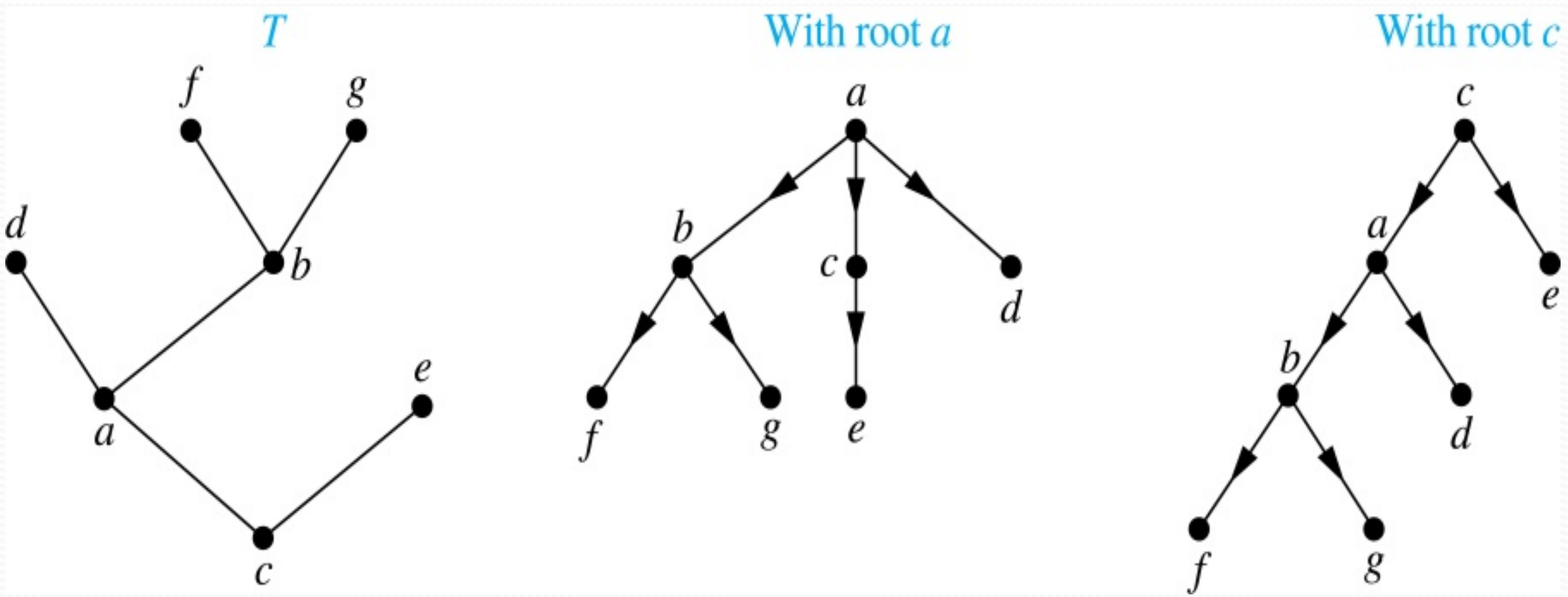


A Decision Tree for Sorting Three Distinct Elements.

Rooted Trees

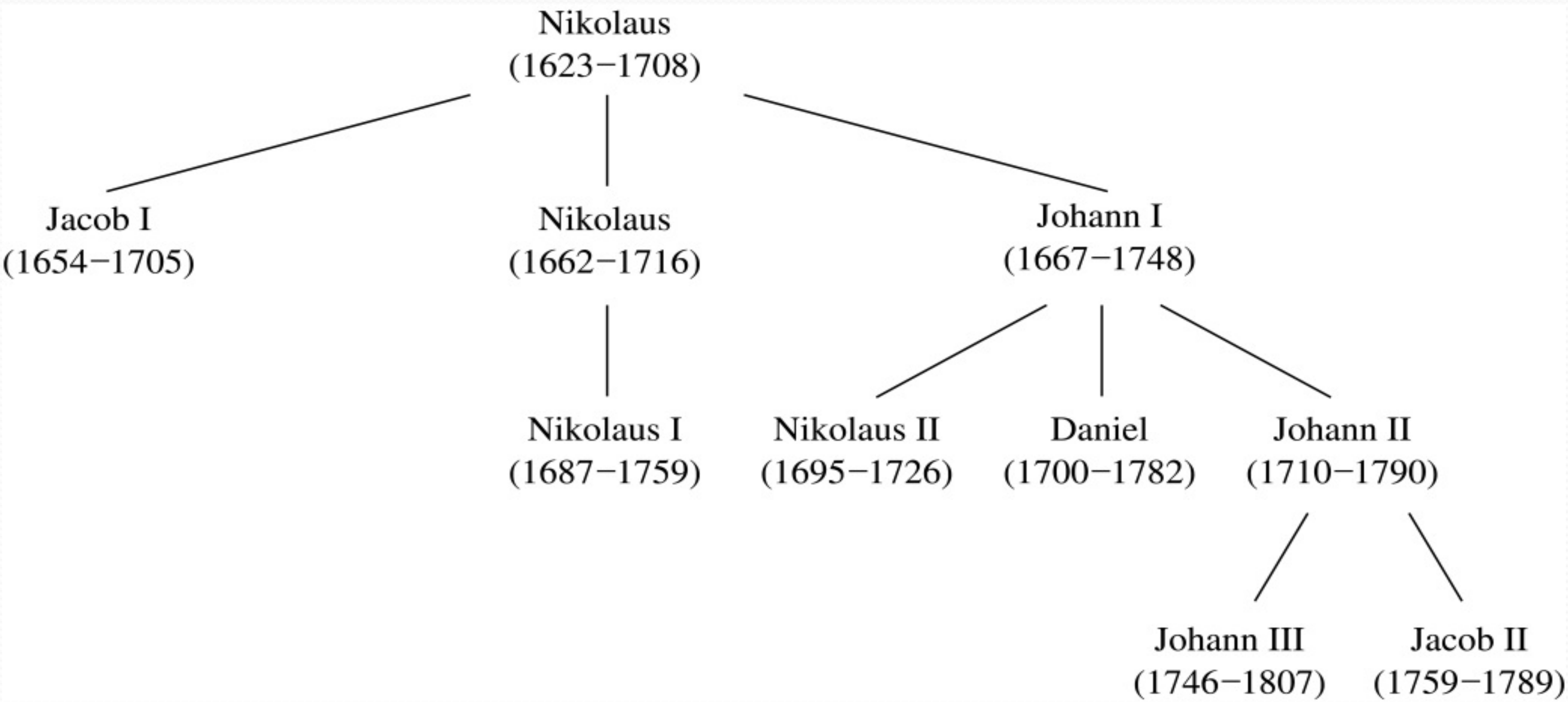
Definition: A *rooted tree* is a tree in which one vertex has been designated as the *root* and every edge is directed away from the root.

- An unrooted tree is converted into different rooted trees when different vertices are chosen as the root.



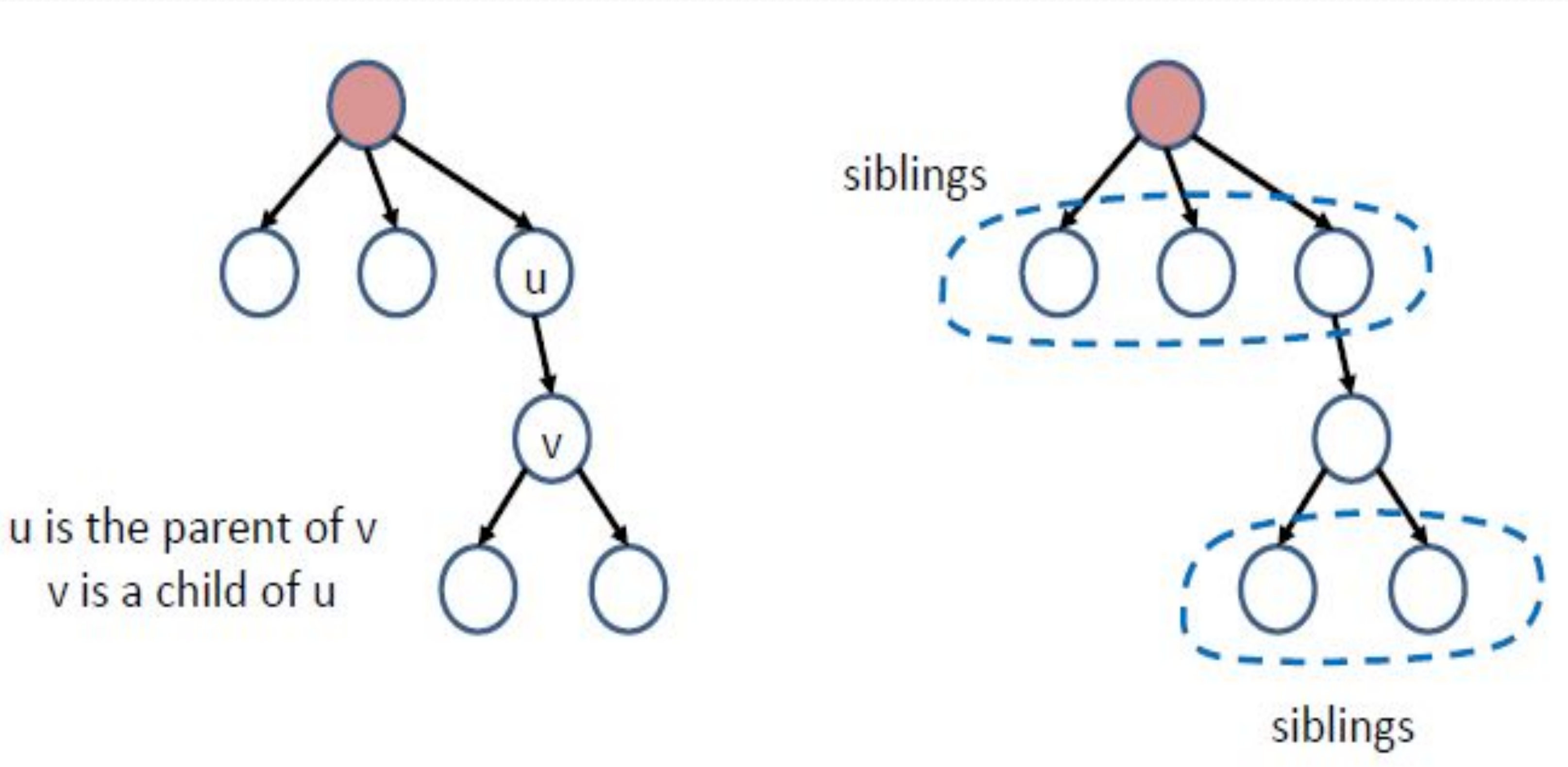
Rooted Tree Terminology

- Terminology for rooted trees is a mix from botany and genealogy (such as this family tree of the Bernoulli family of mathematicians).



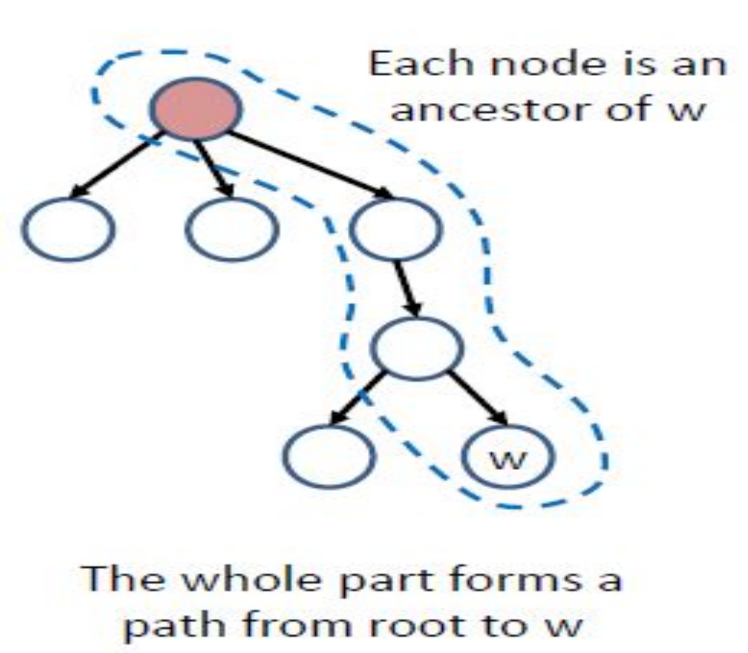
Rooted Tree Terminology

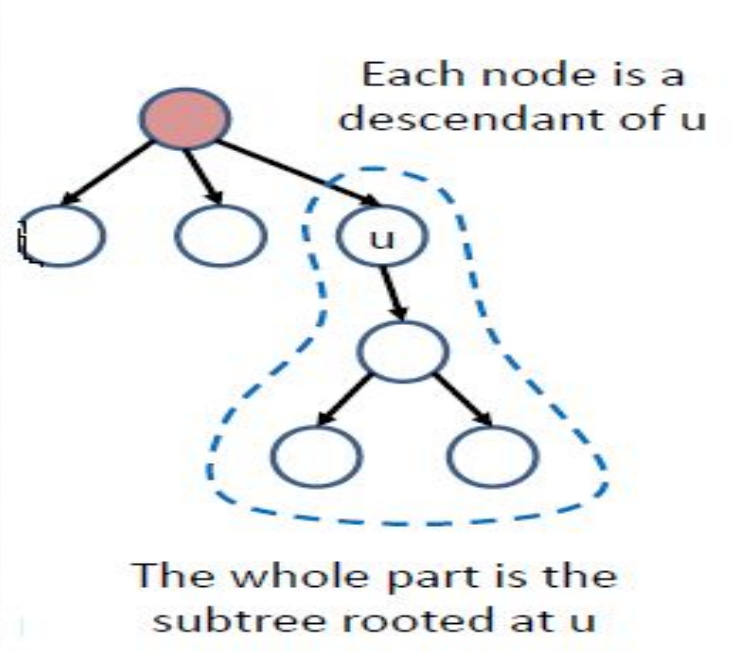
- If v is a vertex of a rooted tree other than the root, the *parent* of v is the unique vertex u such that there is a directed edge from u to v. When u is a parent of v, v is called a *child* of u. Vertices with the same parent are called *siblings*.



Rooted Tree Terminology

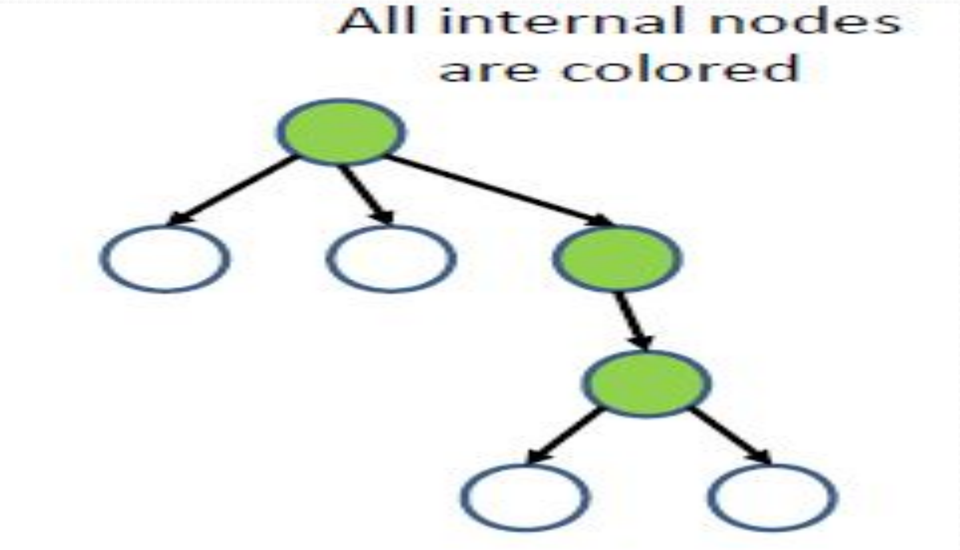
- The *ancestors* of a vertex are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root.
- The *descendants* of a vertex *v* are those vertices that have *v* as an ancestor. The subtree rooted at u includes all the descendants of u, and all edges that connect between them.





Rooted Tree Terminology

- A vertex of a rooted tree with no children is called a *leaf*. Vertices that have children are called *internal vertices*.



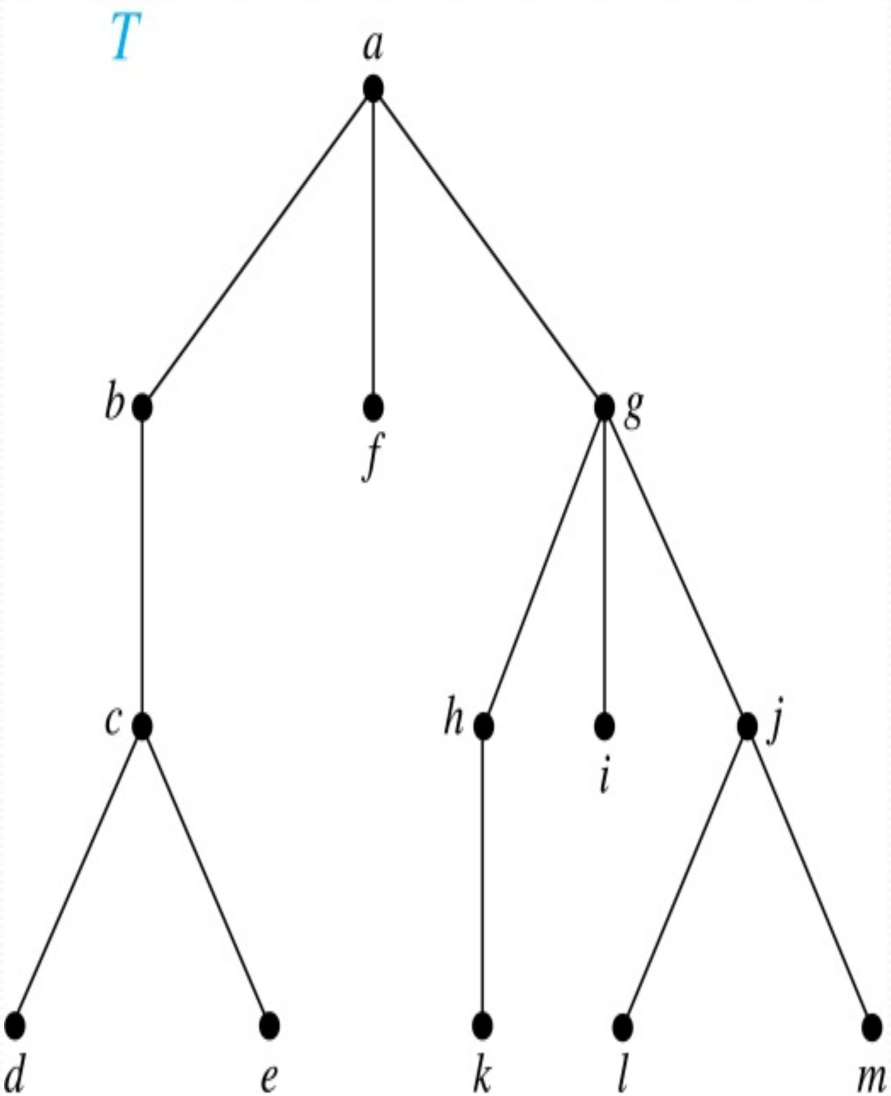
Terminology for Rooted Trees

Example: In the rooted tree *T* (with root *a*):

(i) Find the parent of *c*, the children of *g*, the siblings of *h*, the ancestors of *e*, and the descendants of *b*.

Solution:

(i) The parent of *c* is *b*. The children of *g* are *h*, *i*, and *j*. The siblings of *h* are *i* and *j*. The ancestors of *e* are c, *b*, and *a*. The descendants of *b* are *c*, *d*, and *e*.



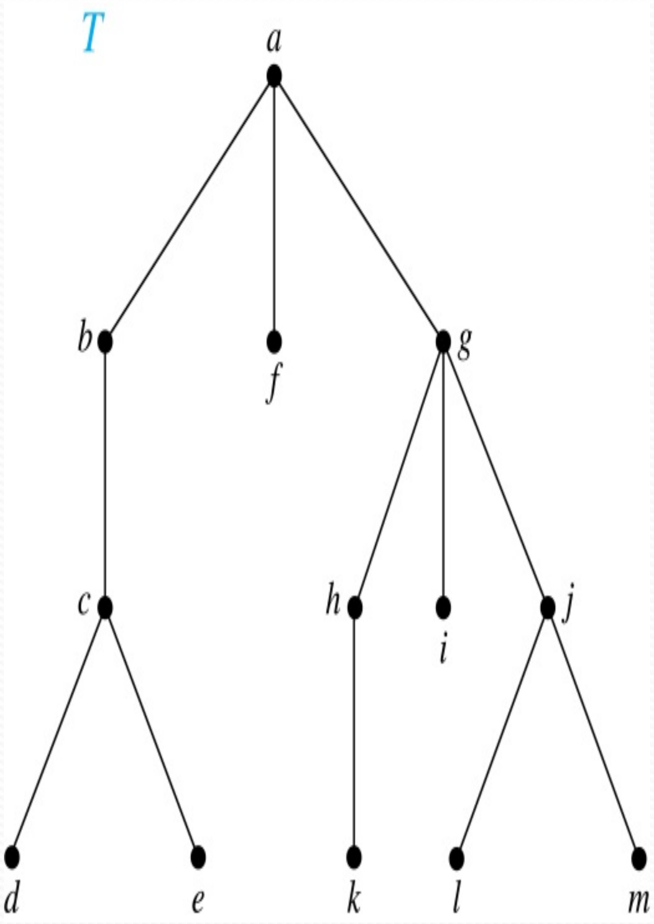
Terminology for Rooted Trees

Example: In the rooted tree T (with root a):

(i) Find all internal vertices and all leaves.

Solution:

(i) The internal vertices are a, b, c, g, h, and j. The leaves are d, e, f, i, k, l, and m.

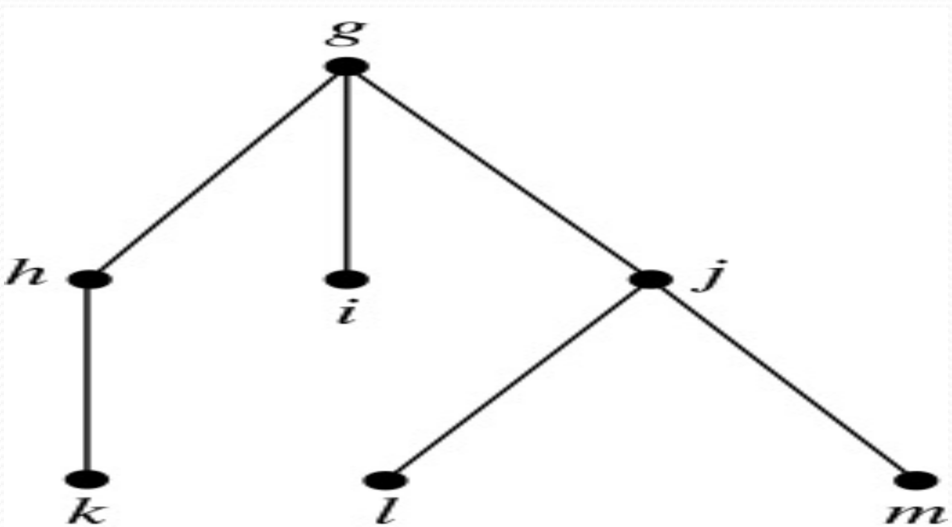


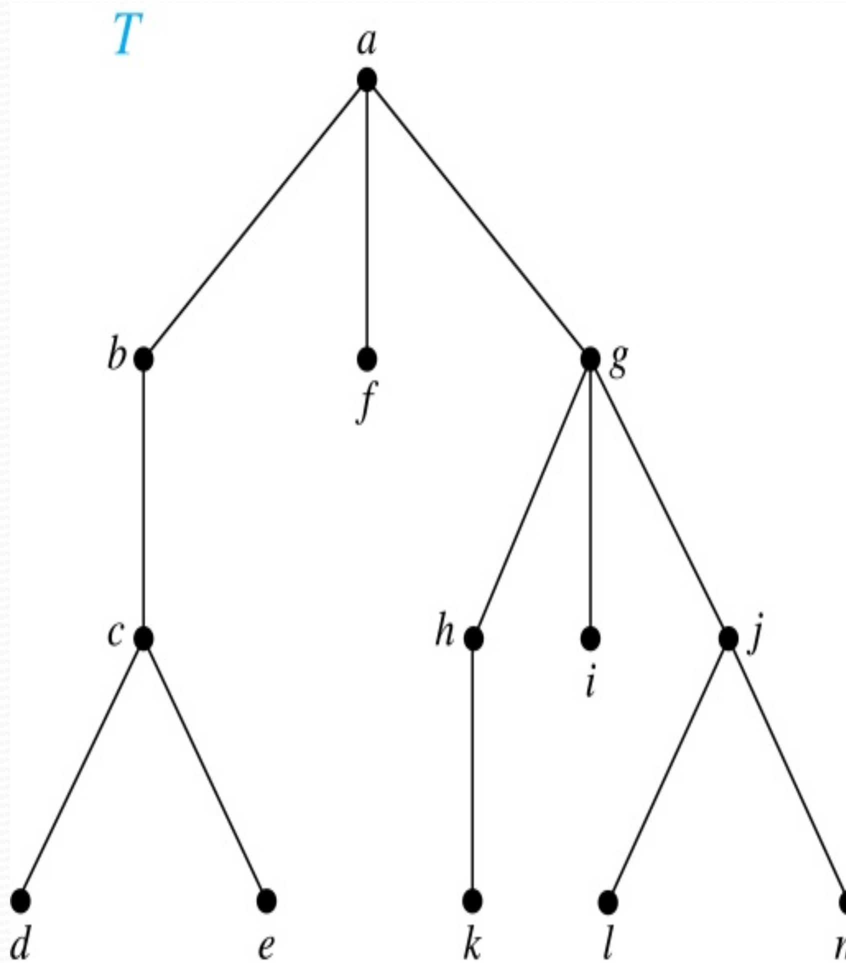
Terminology for Rooted Trees

(i) What is the subtree rooted at g?

Solution:

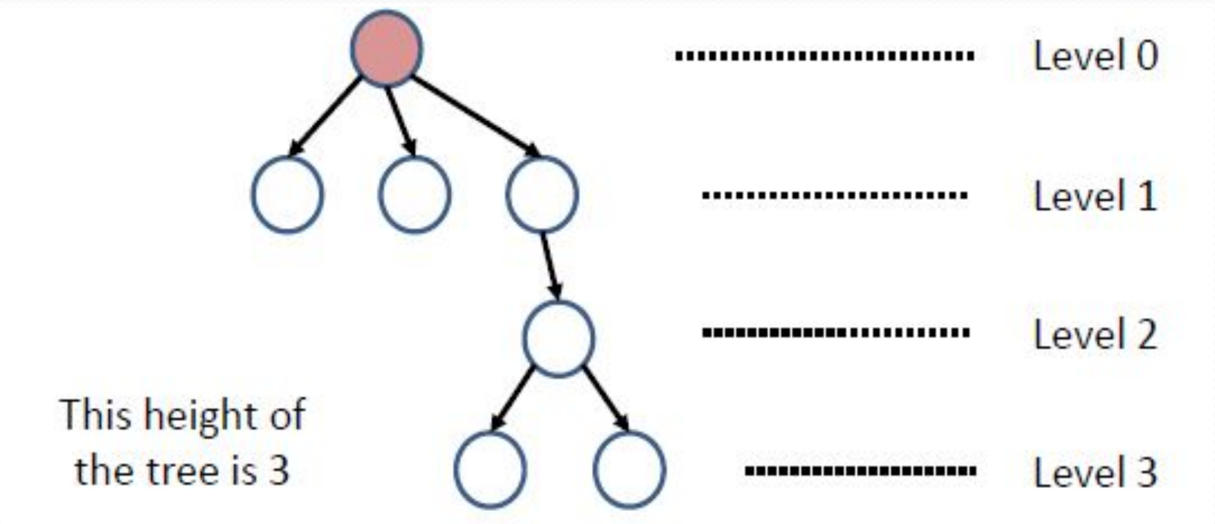
(i) We display the subtree rooted at g.





Level of vertices and height of trees

- When working with trees, we often want to have rooted trees where the sub trees at each vertex contain paths of approximately the same length.
- To make this idea precise we need some definitions:
  - The *level* of a vertex *v* in a rooted tree is the length of the unique path from the root to this vertex.
  - The *height* of a rooted tree is the maximum of the levels of the vertices.

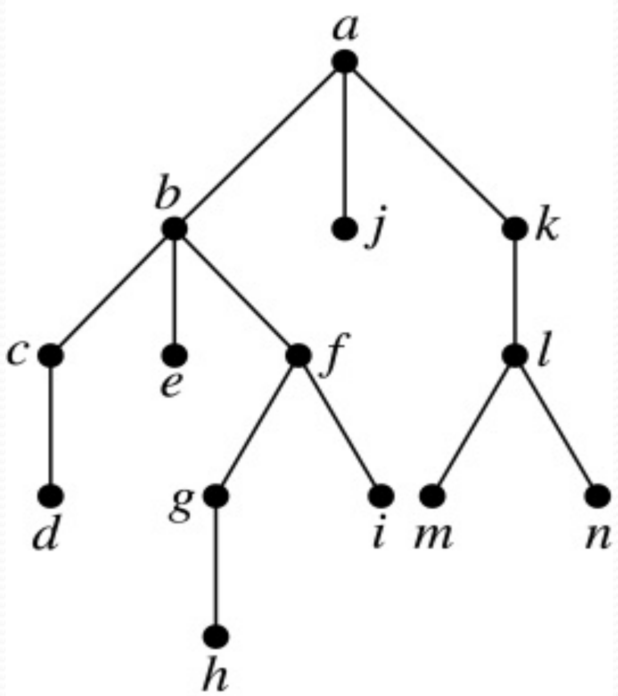


Level of vertices and height of trees

Example:

(i) Find the level of each vertex in the tree to the right.

(ii) What is the height of the tree?



Solution:

(i) The root a is at level 0.

Vertices b, j, and k are at level 1.

Vertices c, e, f, and l are at level 2.

Vertices d, g, i, m, and n are at level 3.

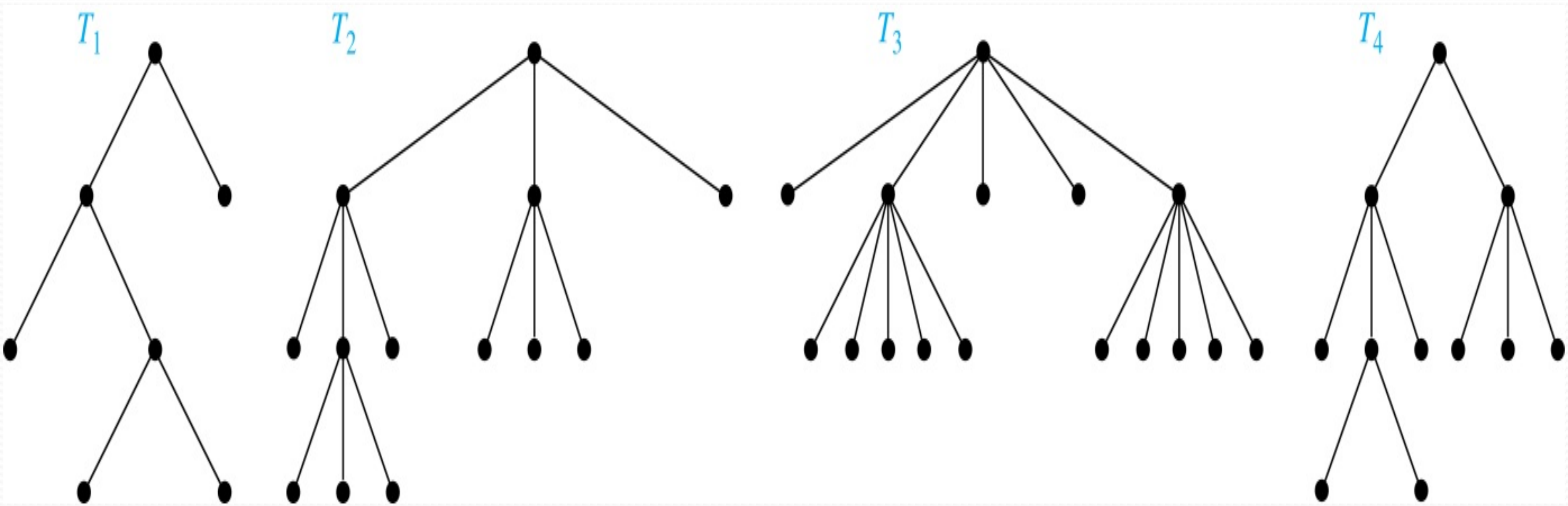
Vertex h is at level 4.

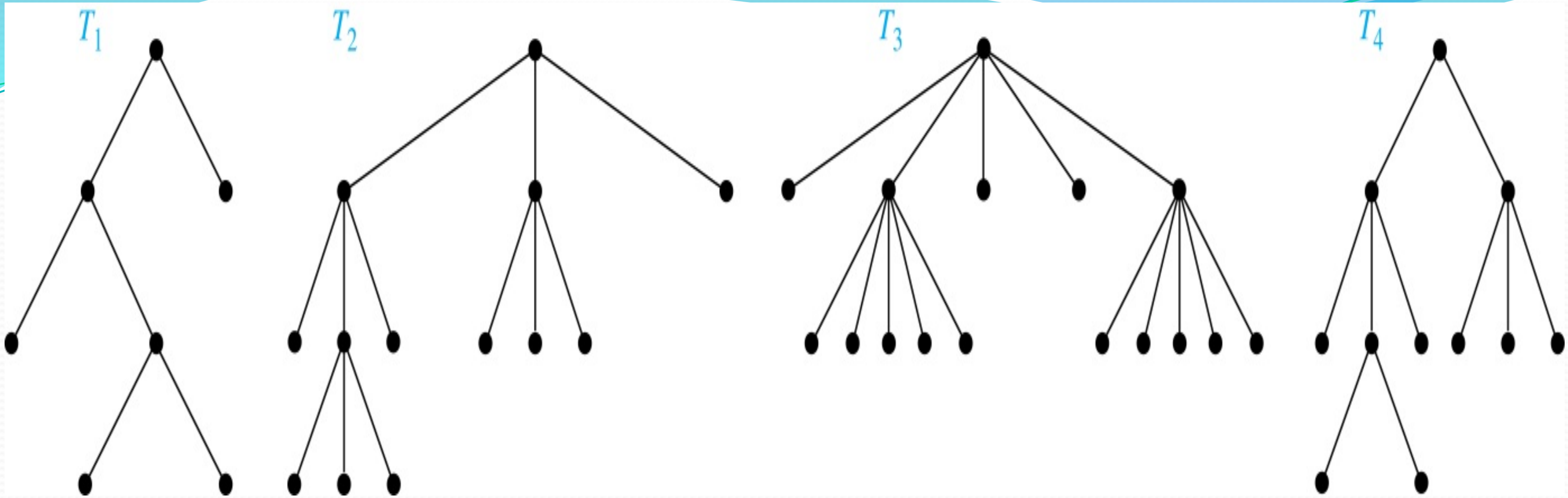
(ii) The height is 4, since 4 is the largest level of any vertex.

m-ary Rooted Trees

Definition: A rooted tree is called an *m-ary tree* if every internal vertex has no more than *m* children. The tree is called a *full m-ary tree* if every internal vertex has exactly *m* children. An *m*-ary tree with $m = 2$ is called a *binary* tree.

Example: Are the following rooted trees full *m*-ary trees for some positive integer *m*?





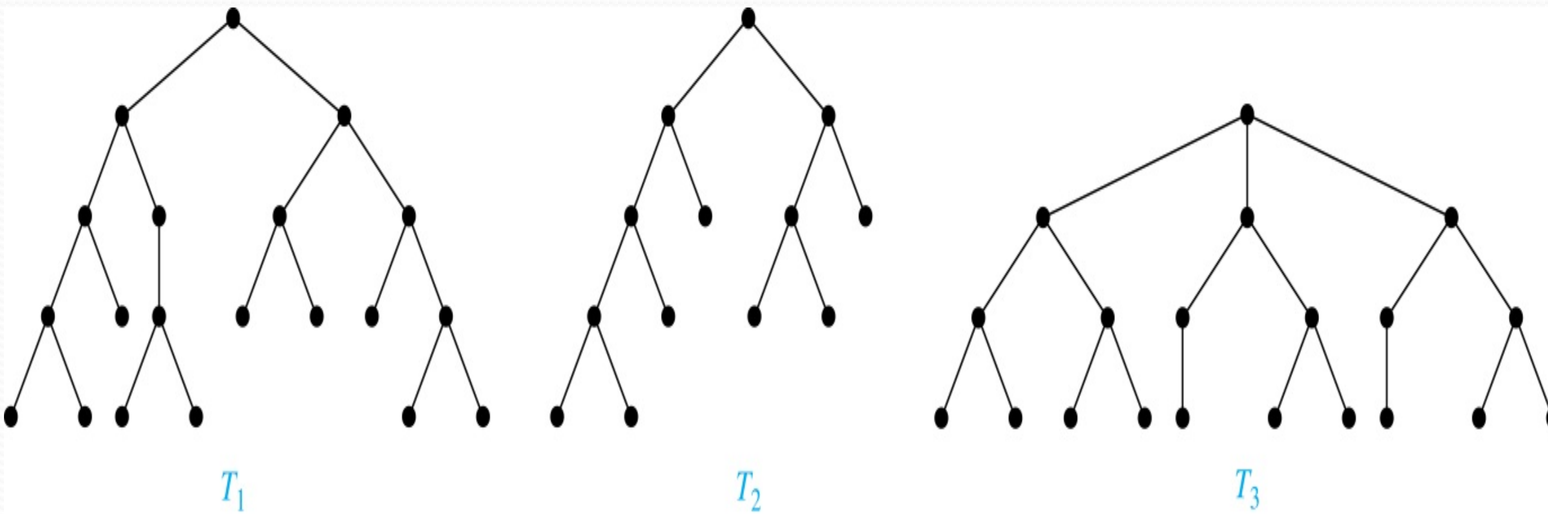
Solution:

- T_1 is a full binary tree because each of its internal vertices has two children.
- T_2 is a full 3-ary tree because each of its internal vertices has three children.
- In T_3 each internal vertex has five children, so T_3 is a full 5-ary tree.
- T_4 is not a full *m*-ary tree for any m because some of its internal vertices have two children and others have three children.

Balanced *m*-Ary Trees

Definition: A rooted m-ary tree of height h is *balanced* if all leaves are at levels h or $h - 1$.

Example: Which of the rooted trees shown below is balanced?

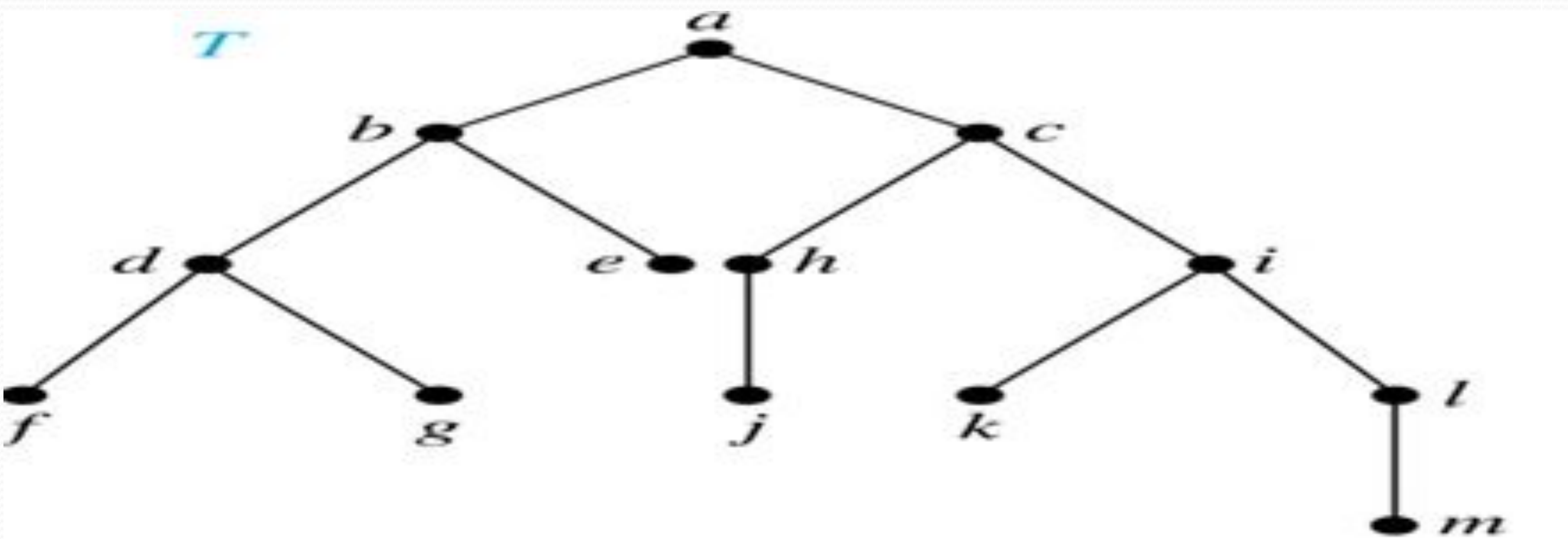


Solution: T_1 and T_3 are balanced, but T_2 is not because it has leaves at levels 2, 3, and 4.

Ordered Rooted Trees

Definition: An *ordered rooted tree* is a rooted tree where the children of each internal vertex are ordered.

- We draw ordered rooted trees so that the children of each internal vertex are shown in order from left to right.



Binary Trees

Definition: A *binary tree* is an ordered rooted where each internal vertex has at most two children. If an internal vertex of a binary tree has two children, the first is called the *left child* and the second the *right child*. The tree rooted at the left child of a vertex is called the *left subtree* of this vertex, and the tree rooted at the right child of a vertex is called the *right subtree* of this vertex.

Example:

Consider the binary tree T.

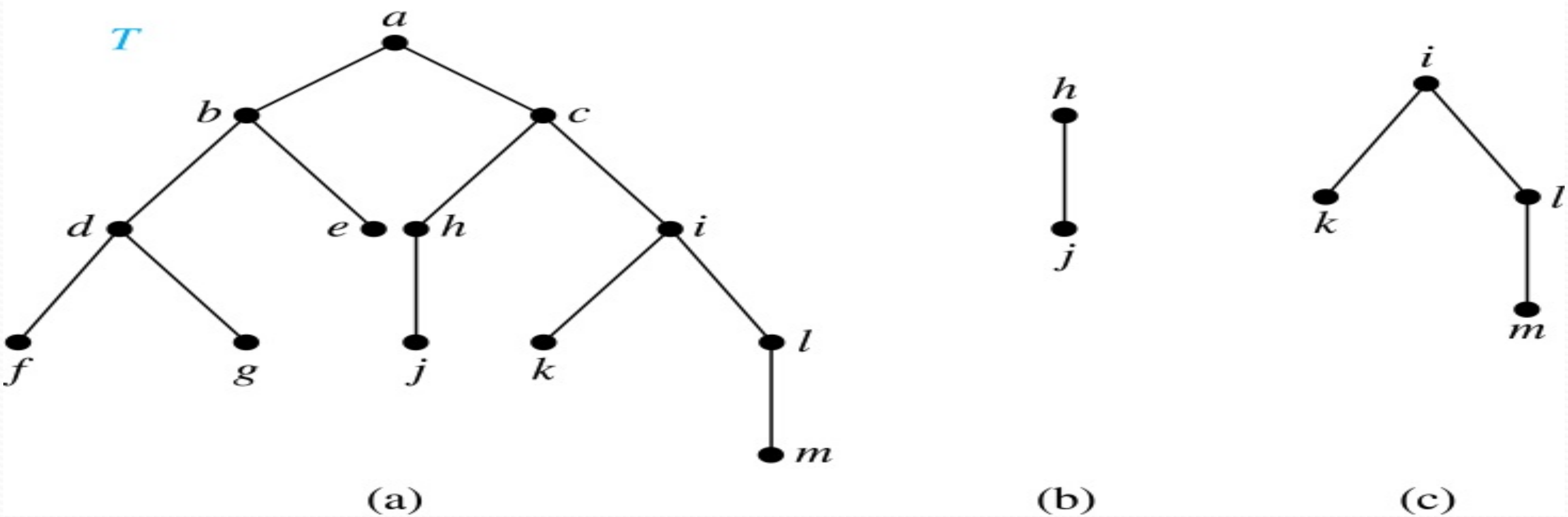
(i) What are the left and right children of d?

(ii) What are the left and right sub trees of c?

Solution:

(i) The left child of d is f and the right child is g.

(ii) The left and right subtrees of c are displayed in (b) and (c).



Properties of Trees

- A tree with n vertices has $n-1$ edges.
- A full m-ary tree with i internal vertices has $n = mi + 1$ vertices.
- A full m-ary tree with:

  (i) *n vertices has $i = (n - 1)/m$ internal vertices and $l = [(m - 1)n + 1]/m$ leaves,*

  (ii) i internal vertices has $n = mi + 1$ vertices and $l = (m - 1)i + 1$ leaves,

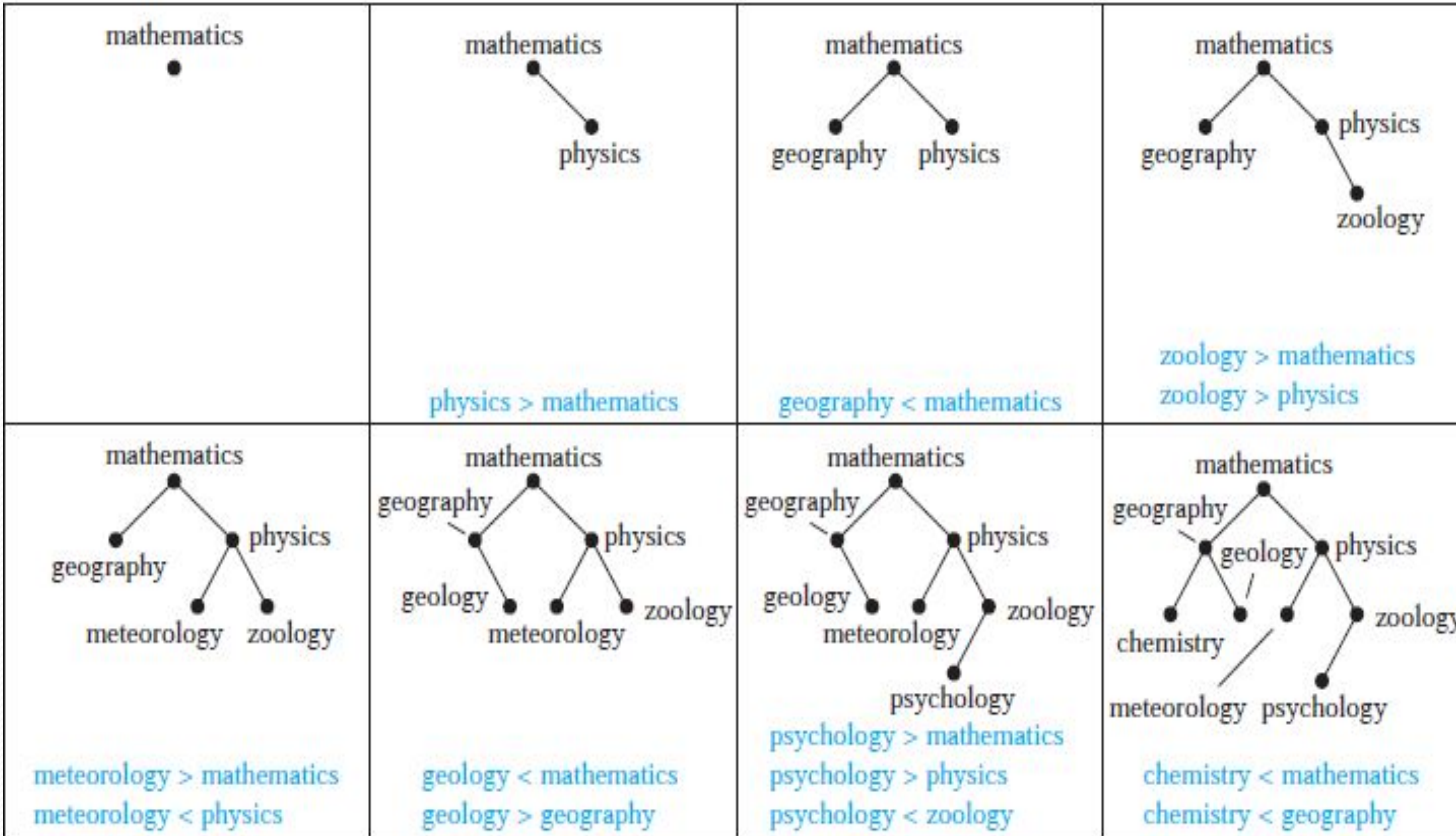
  (iii) l leaves has $n = (ml - 1)/(m - 1)$ vertices and $i = (l - 1)/(m - 1)$ internal vertices.
- There are at most m^h leaves in an m-ary tree of height h.

Binary Search Tree

Definition: A binary tree in which the vertices are labeled with items so that a label of a vertex is greater than the labels of all vertices in the left subtree of this vertex and is less than the labels of all vertices in the right subtree of this vertex.

- Searching for items in a list is one of the most important tasks that arises in computer science.
- Our primary goal is to implement a searching algorithm that finds items efficiently when the items are totally ordered. This can be accomplished through the use of a binary search tree.

Example : Form a binary search tree for the words mathematics, physics, geography, zoology, meteorology, geology, psychology, and chemistry (using alphabetical order).



Tree Traversal

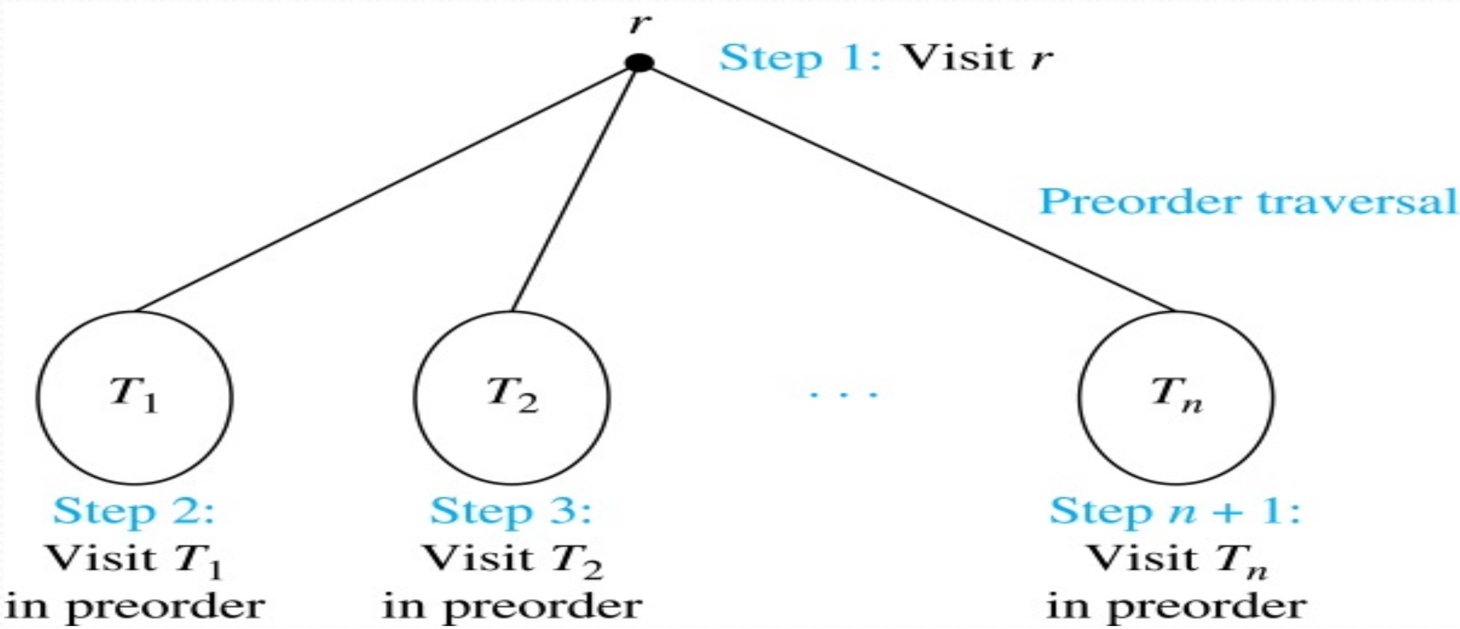
Section 11.3

Tree Traversal

- Procedures for systematically visiting every vertex of an ordered tree are called *traversals*.
- The three most commonly used *traversals* are *preorder traversal*, *inorder traversal*, and *postorder traversal*.

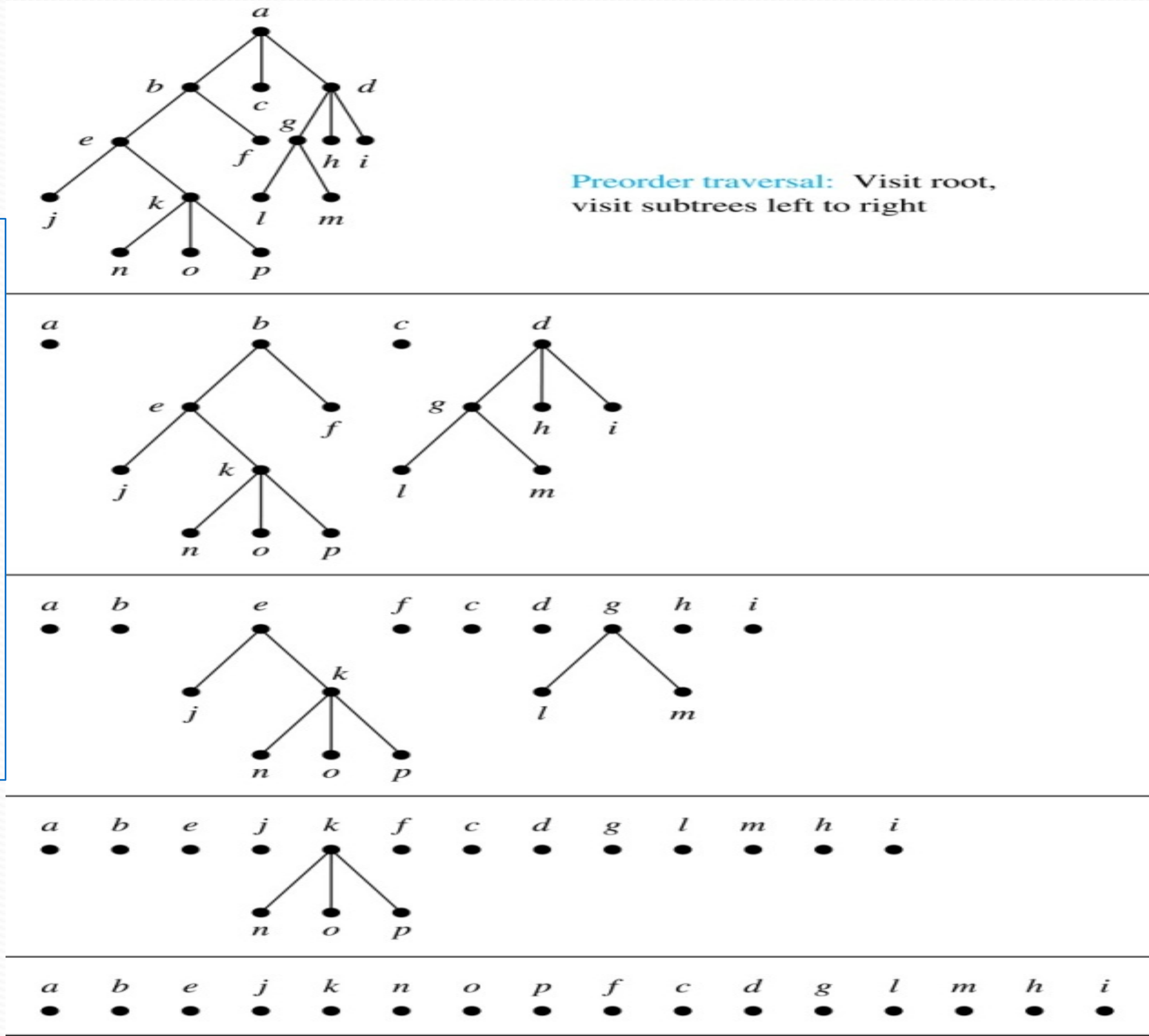
Preorder Traversal

Definition: Let T be an ordered rooted tree with root r. If T consists only of r, then r is the *preorder traversal* of T. Otherwise, suppose that $T_1, T_2, \ldots, T_n$ are the subtrees of r from left to right in T. The preorder traversal begins by visiting r, and continues by traversing T_1 in preorder, then T_2 in preorder, and so on, until T_n is traversed in preorder.



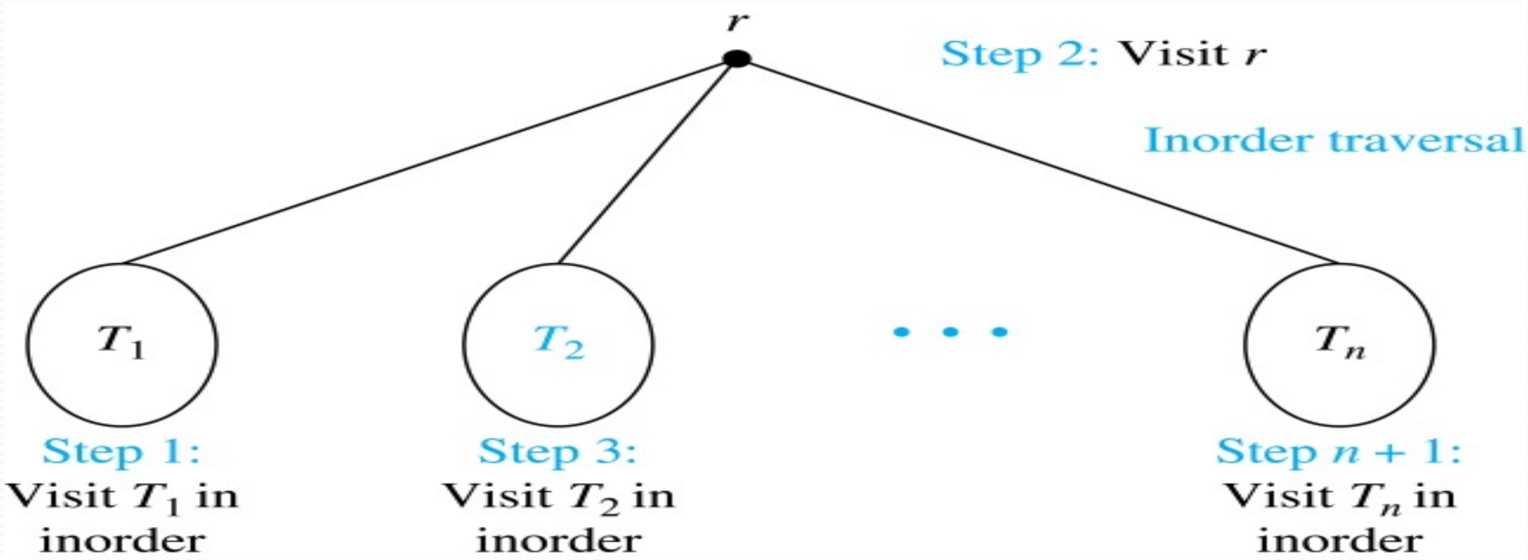
Preorder Traversal (*continued*)

procedure *preorder* (T: ordered rooted tree)
r := root of T
list r
for each child c of r from left to right
    $T(c)$:= subtree with c as root
    preorder($T(c)$)



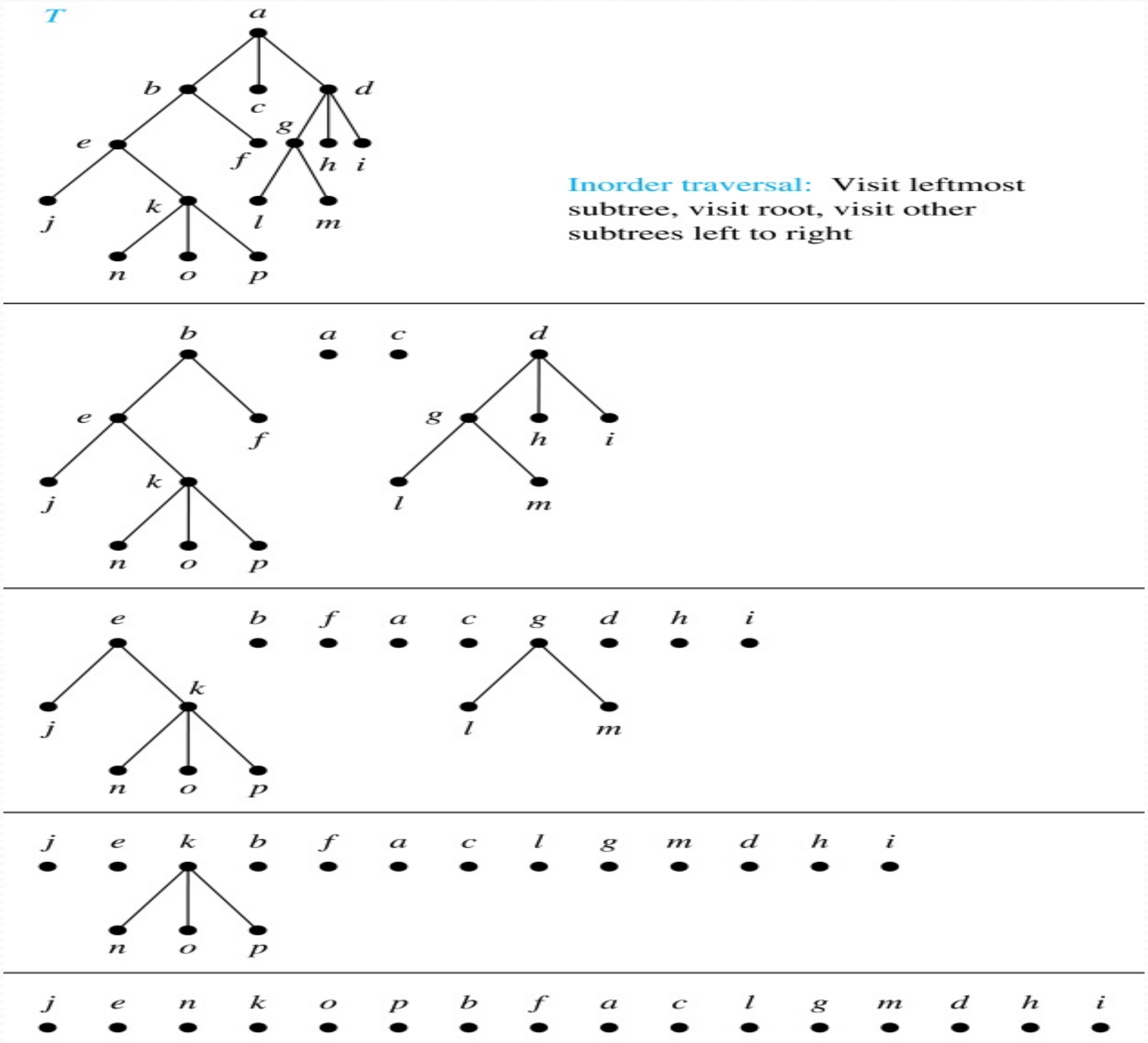
Inorder Traversal

Definition: Let T be an ordered rooted tree with root r. If T consists only of r, then r is the *inorder traversal* of T. Otherwise, suppose that $T_1, T_2, \ldots, T_n$ are the subtrees of r from left to right in T. The inorder traversal begins by traversing T_1 in inorder, then visiting r, and continues by traversing T_2 in inorder, and so on, until T_n is traversed in inorder.



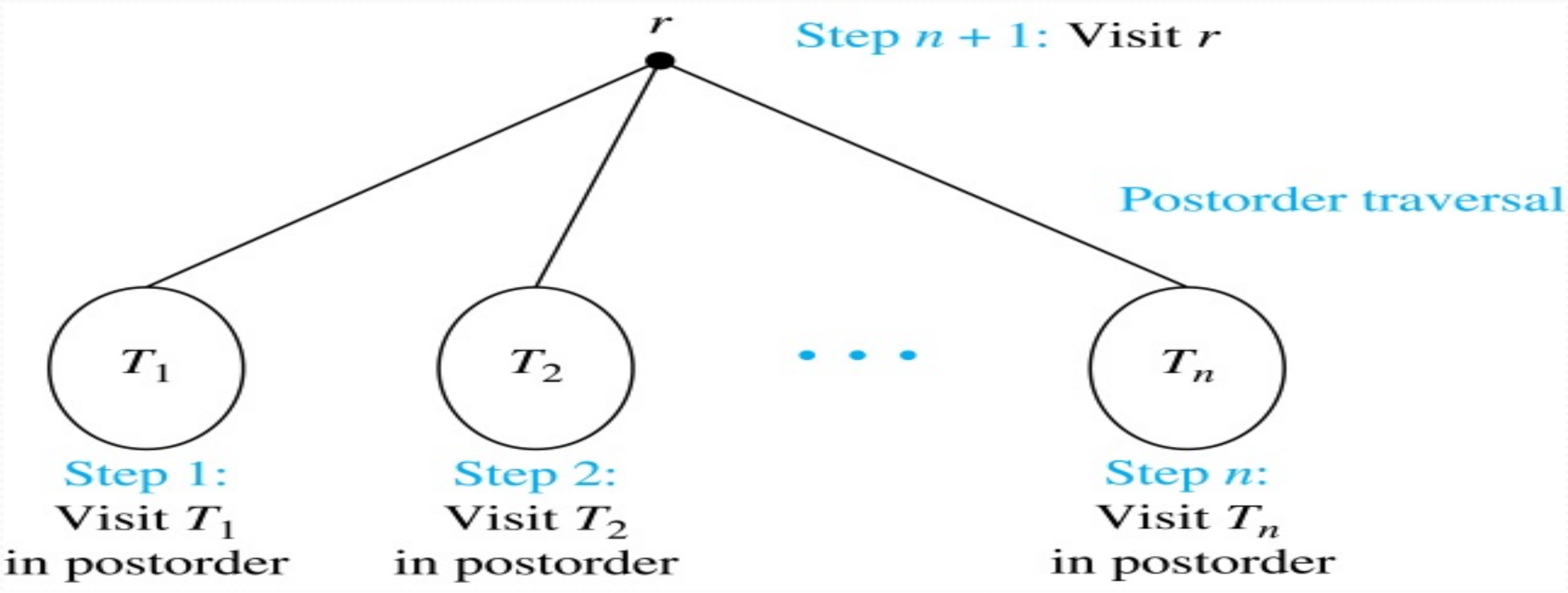
Inorder Traversal (*continued*)

procedure *inorder* (T: ordered rooted tree)
r := root of T
if r is a leaf **then** list r
else
  l := first child of r from left to right
  $T(l)$:= subtree with l as its root
  inorder($T(l)$)
  list(r)
  for each child c of r from left to right
    $T(c)$:= subtree with c as root
    inorder($T(c)$)



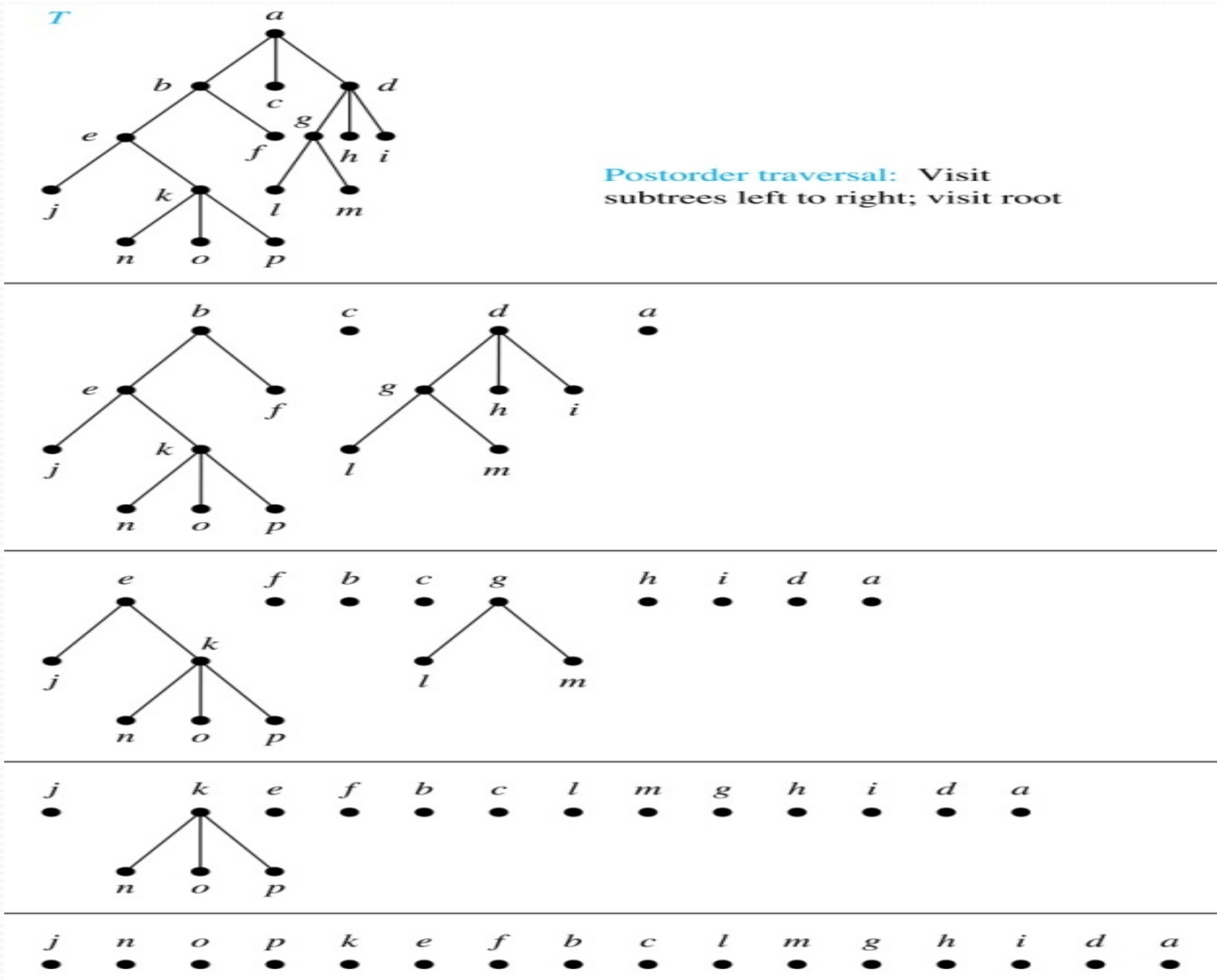
Postorder Traversal

Definition: Let T be an ordered rooted tree with root r. If T consists only of r, then r is the *postorder traversal* of T. Otherwise, suppose that $T_1, T_2, \ldots, T_n$ are the subtrees of r from left to right in T. The postorder traversal begins by traversing T_1 in postorder, then T_2 in postorder, and so on, after T_n is traversed in postorder, r is visited.



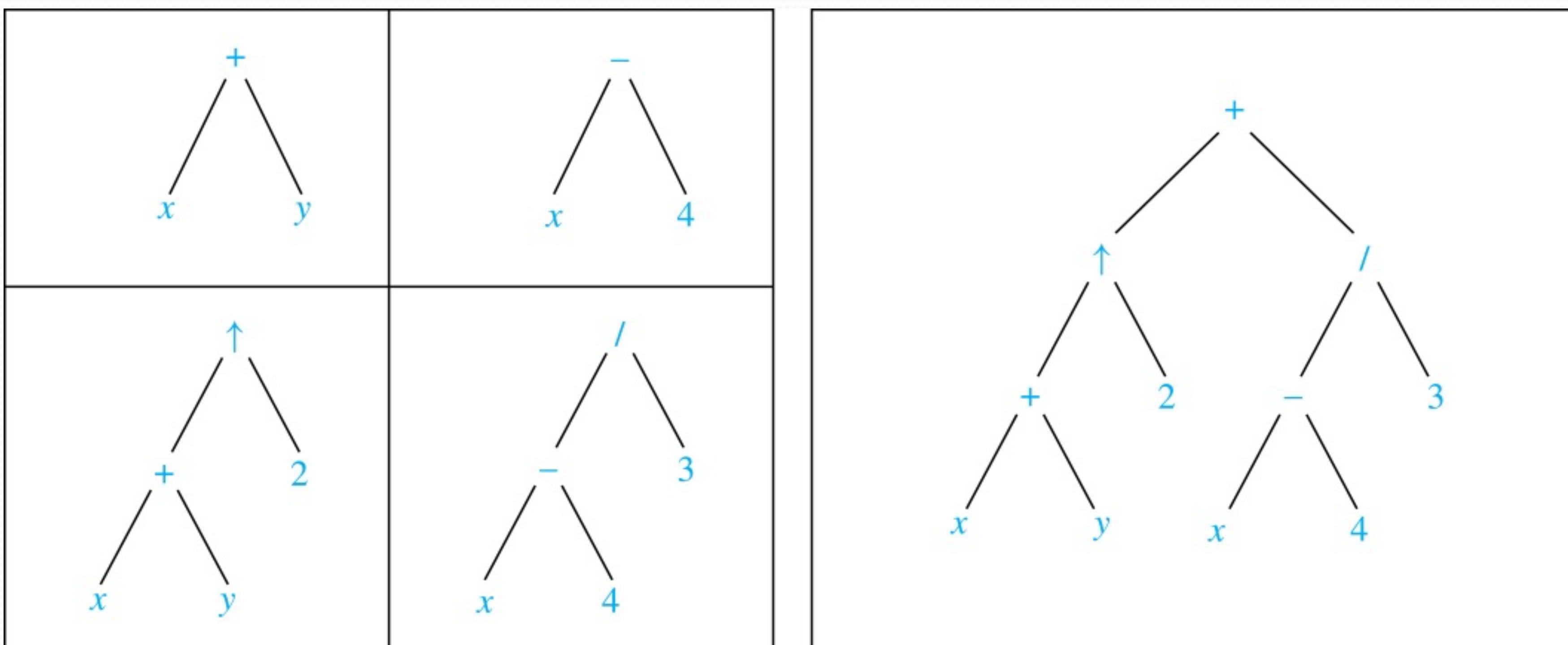
Post order Traversal (*continued*)

procedure *postordered* (*T*: ordered rooted tree)
r := root of *T*
for each child *c* of *r* from left to right
    T(*c*) := subtree with *c* as root
    postorder(*T*(*c*))
list *r*



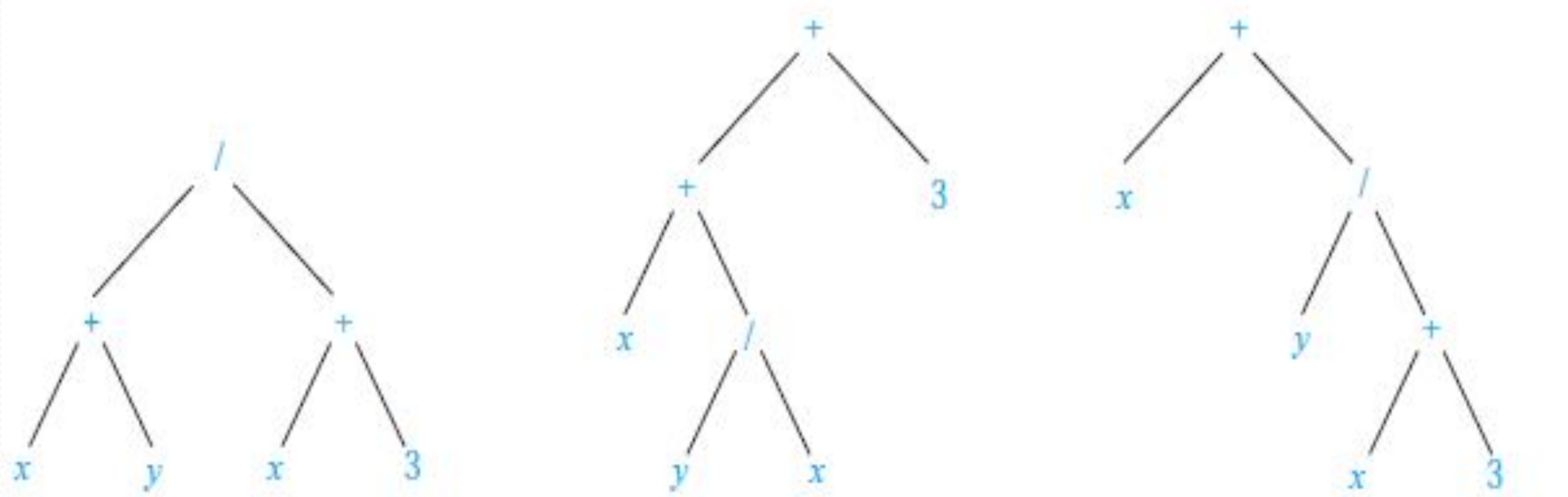
Expression Trees

- Complex expressions can be represented using ordered rooted trees.
- Consider the expression $((x + y) \uparrow 2\ ) + ((x - 4)/3)$.
- A binary tree for the expression can be built from the bottom up, as is illustrated here.

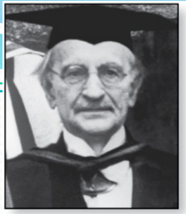


Infix Notation

- An inorder traversal of the tree representing an expression produces the original expression when parentheses are included except for unary operations, which now immediately follow their operands.
- We illustrate why parentheses are needed with an example that displays three trees all yield the same infix representation.



Rooted Trees Representing $(x+y)/(x+3)$, $(x+(y/x))+3$, and $x+(y/(x+3))$.



Jan Łukasiewicz
(1878-1956)

Prefix Notation

- When we traverse the rooted tree representation of an expression in preorder, we obtain the *prefix* form of the expression. Expressions in prefix form are said to be in *Polish notation*, named after the Polish logician Jan Łukasiewicz.
- Operators precede their operands in the prefix form of an expression. Parentheses are not needed as the representation is unambiguous.
- The prefix form of $((x + y) \uparrow 2\,) + ((x - 4)/3)$ is $+ \uparrow + \, x \, y \; 2 \, / - x \; 4 \; 3$.
- Prefix expressions are evaluated by working from right to left. When we encounter an operator, we perform the corresponding operation with the two operations to the right.

Prefix Notation

- **Example**: We show the steps used to evaluate a particular prefix expression:

$$+ \quad - \quad * \quad 2 \quad 3 \quad 5 \quad / \quad \underbrace{\uparrow \quad 2 \quad 3}_{2 \uparrow 3 = 8} \quad 4$$

$$+ \quad - \quad * \quad 2 \quad 3 \quad 5 \quad \underbrace{/ \quad 8 \quad 4}_{8 / 4 = 2}$$

$$+ \quad - \quad \underbrace{* \quad 2 \quad 3}_{2 * 3 = 6} \quad 5 \quad 2$$

$$+ \quad \underbrace{- \quad 6 \quad 5}_{6 - 5 = 1} \quad 2$$

$$\underbrace{+ \quad 1 \quad 2}_{1 + 2 = 3}$$

Value of expression: 3

Postfix Notation

- We obtain the *postfix form* of an expression by traversing its binary trees in postorder. Expressions written in postfix form are said to be in *reverse Polish notation.*
- Parentheses are not needed as the postfix form is unambiguous.
- $x\ y + 2 \uparrow x\ 4 - 3\ / +$ is the postfix form of $((x + y) \uparrow 2\ ) + ((x - 4)/3)$.
- A binary operator follows its two operands. So, to evaluate an expression one works from left to right, carrying out an operation represented by an operator on its preceding operands.

Postfix Notation

- **Example**: We show the steps used to evaluate a particular postfix expression.

$$7\quad \underbrace{2\quad 3\quad *}_{2 * 3 = 6}\quad -\quad 4\quad \uparrow\quad 9\quad 3\quad /\quad +$$

$$\underbrace{7\quad 6\quad -}_{7 - 6 = 1}\quad 4\quad \uparrow\quad 9\quad 3\quad /\quad +$$

$$\underbrace{1\quad 4\quad \uparrow}_{1^4 = 1}\quad 9\quad 3\quad /\quad +$$

$$1\quad \underbrace{9\quad 3\quad /}_{9 / 3 = 3}\quad +$$

$$\underbrace{1\quad 3\quad +}_{1 + 3 = 4}$$

Value of expression: 4

Spanning Trees

Section 11.4

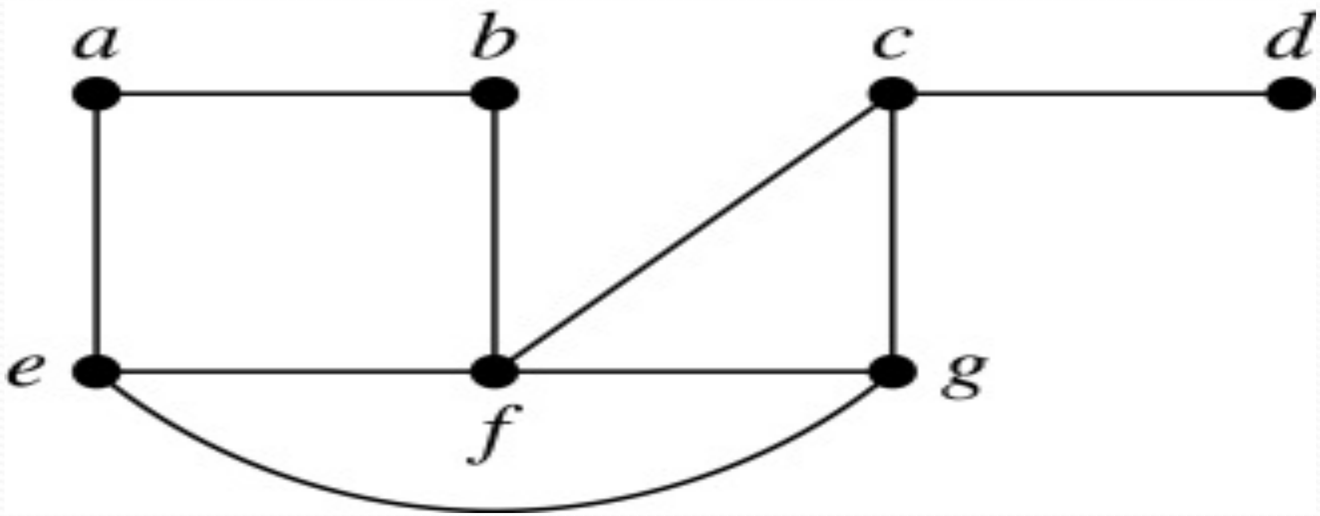
Section Summary

- Spanning Trees
- Prim’s Algorithm
- Kruskal Algorithm

Spanning Trees

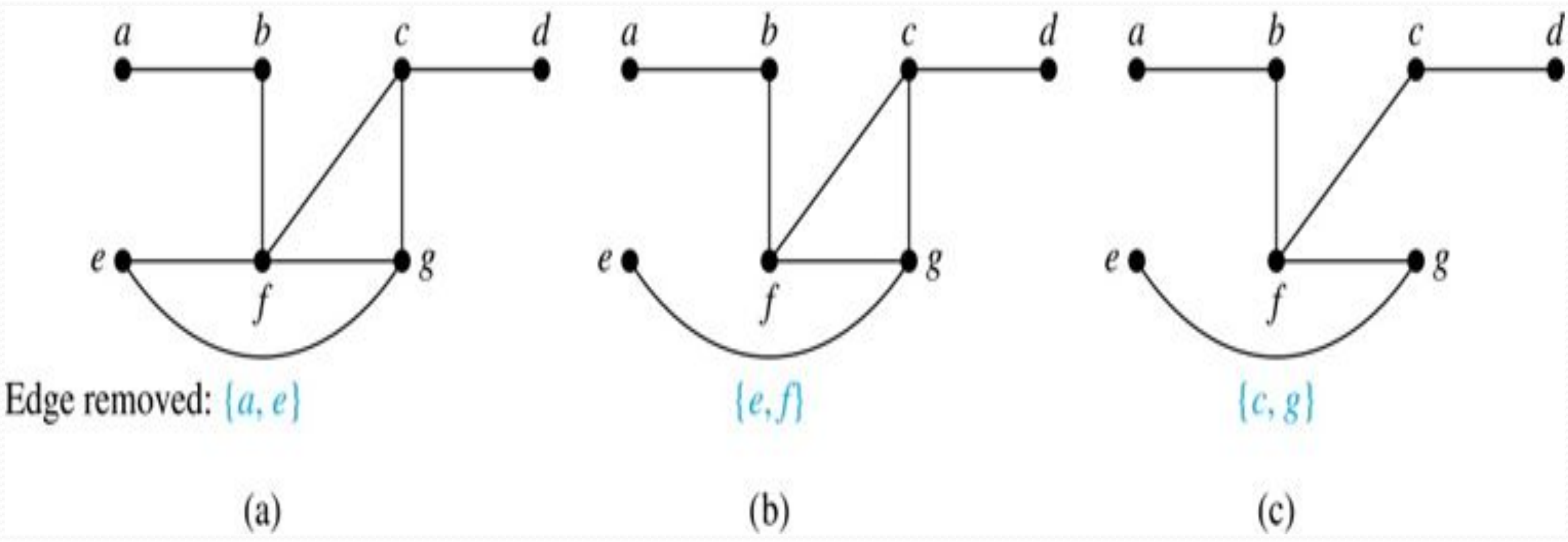
Definition: Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G.

Example: Find the spanning tree of the simple graph:



Spanning Trees

Solution: The graph is connected, but is not a tree because it contains simple circuits. Remove the edge $\{a, e\}$. Now one simple circuit is gone, but the remaining subgraph still has a simple circuit. Remove the edge $\{e, f\}$ and then the edge $\{c, g\}$ to produce a simple graph with no simple circuits. It is a spanning tree, because it contains every vertex of the original graph.



Minimum Spanning

Section 11.5

Minimum spanning tree

- A *minimum spanning tree* in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.
- **Example:** A company plans to build a communications network connecting its five computer centers. Any pair of these centers can be linked with a leased telephone line. Which links should be made to ensure that there is a path between any two computer centers so that the total cost of the network is minimized?

Minimum spanning tree

- **Solution:** We can model this problem using the weighted graph shown in Figure 1, where vertices represent computer centers, edges represent possible leased lines, and the weights on edges are the monthly lease rates of the lines represented by the edges. We can solve this problem by finding a spanning tree so that the sum of the weights of the edges of the tree is minimized. Such a spanning tree is called a **minimum spanning tree**.

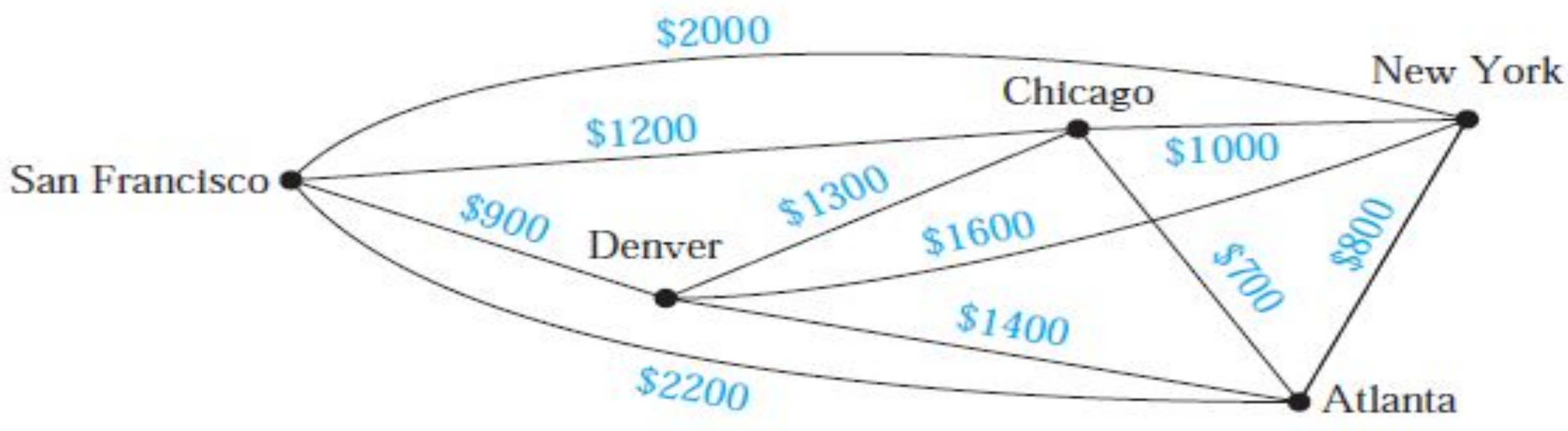


FIGURE 1 A Weighted Graph Showing Monthly Lease Costs for Lines in a Computer Network.

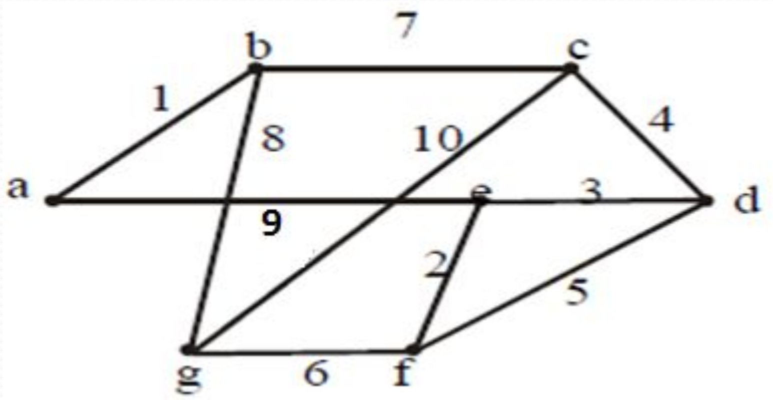
PRIM'S ALGORITHM

ALGORITHM 1 Prim's Algorithm.

procedure *Prim*(G: weighted connected undirected graph with n vertices)
T := a minimum-weight edge
for $i := 1$ **to** $n - 2$
  e := an edge of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T
  $T := T$ with e added
return T {T is a minimum spanning tree of G}

Minimal spanning tree (MST)

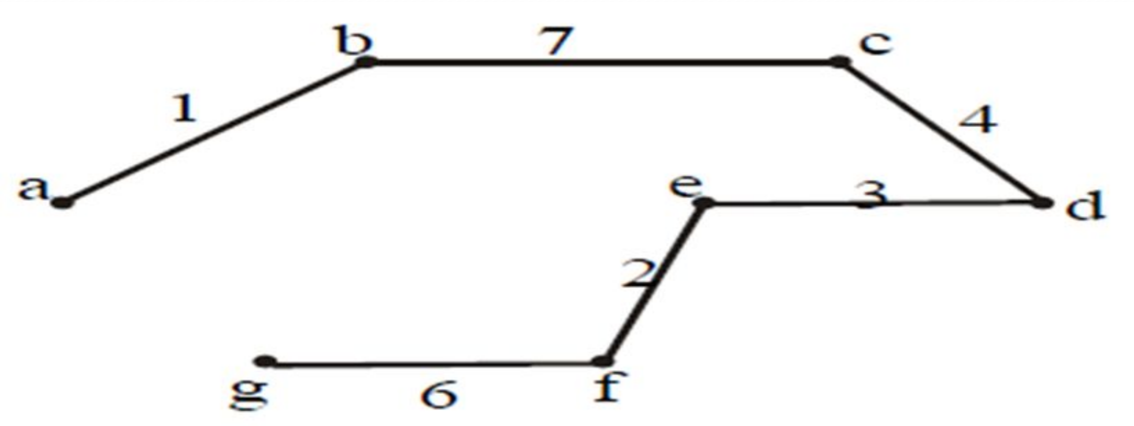
Example: Use Prims algorithm to find a minimal spanning tree for the graph below. Indicate the order in which edges are added to form the tree.



Order of adding the edges:

{a , b}, {b , c}, {c , d}, {d , e}, {e , f}, {f , g}

MST COST = 23



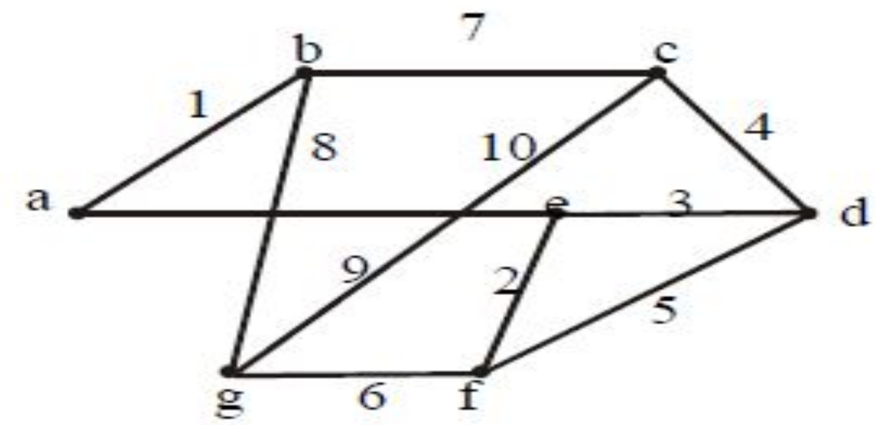
KRUSKAL'S ALGORITHM

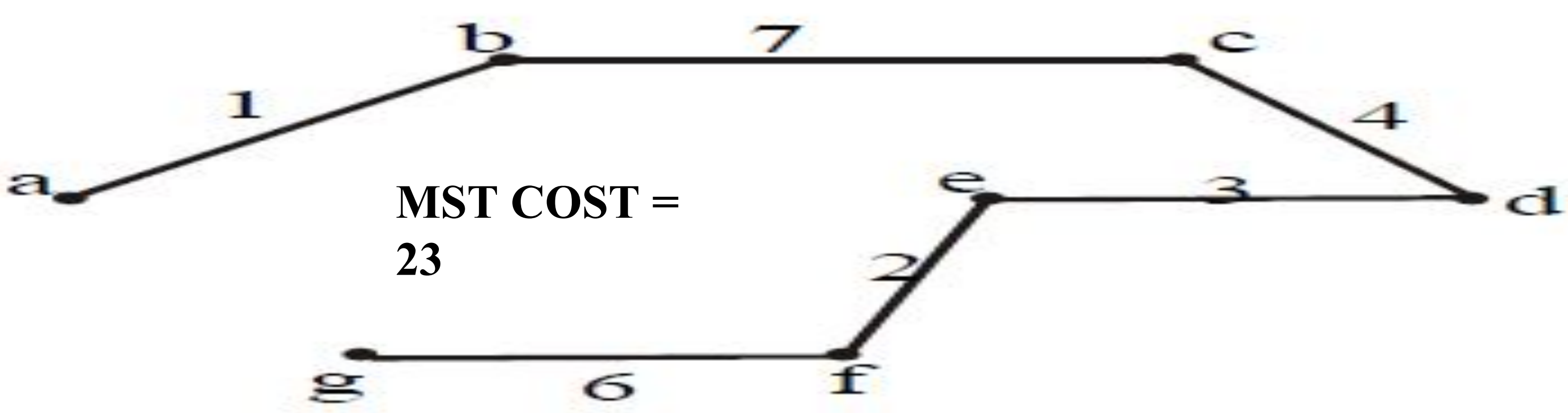
ALGORITHM 2 Kruskal's Algorithm.

procedure *Kruskal*(G: weighted connected undirected graph with n vertices)
T := empty graph
for $i := 1$ **to** $n - 1$
    e := any edge in G with smallest weight that does not form a simple circuit
        when added to T
    $T := T$ with e added
return T {T is a minimum spanning tree of G}

Minimal spanning tree (MST)

Example: Use Kruskal's algorithm to find a minimal spanning tree for the graph below. Indicate the order in which edges are added to form the tree.





MST COST = 23

Order of adding the edges:
{a, b}, {e, f}, {e, d}, {c, d}, {g, f}, {b, c}