

# Discrete Structures

Spring 2024 – Week 1



# Course Introduction

Lecture 1

---

# Course Details

- Instructor – Furqan Hussain Essani
- [fessani@cloud.neduet.edu.pk](mailto:fessani@cloud.neduet.edu.pk)
- Counseling Hours
  - Monday 14:30 – 16:00
  - Wednesday 14:30 – 16:00
- Or, with prior appointment via email

# Class Details

- Three sessions a week – 15 week semester including the mid-term week
- Course Learning Outcomes
  1. **Comprehend** the key concepts of discrete structures
  2. **Apply** logical reasoning to real-world computing problems
  3. **Analyze** discrete structures in the context of computer science

# Resources

- **Text Book**

- Discrete Mathematics and Its Applications, Kenneth H. Rosen, 8/e, McGraw Hill Publication

- **References**

- Discrete Mathematics 8th Edition by Richard Johnsonbaugh
  - Essential Discrete Mathematics for Computer Science by Harry Lewis and Rachel Zax
  - Discrete Mathematics for Computer Science 1st Edition by Jon Pierre Fortney

- **Google Classroom**

- Lecture Slides, Supplementary Material

# Marks Distribution

- Sessional 40%
  - Quizzes – 10%
  - Assignments – 10%
  - Mid-Term Examination – 20%
- Final Examination 60%

# What is a Discrete Structure?

- A discrete structure is a set of distinct elements, where the elements can be enumerated and counted
- Discrete structures are used to model and solve problems that involve discrete objects or events
- Study of discrete structures is formally known as Discrete Mathematics, opposed is Calculus that deals with continuous objects
- Examples of discrete objects: integers, steps taken by a computer program, distinct paths to travel from point A to point B on a map along a road network, ways to pick a winning set of numbers in a lottery

# Kinds of Problems being solved using Discrete Mathematics

- How many ways can a password be chosen following specific rules?
- How many valid Internet addresses are there?
- What is the probability of winning a particular lottery?
- Is there a link between two computers in a network?
- How can I identify spam email messages?
- How can I encrypt a message so that no unintended recipient can read it?
- How can we build a circuit that adds two integers?



# Kinds of Problems being solved using Discrete Mathematics

- What is the shortest path between two cities using a transportation system?
- Find the shortest tour that visits each of a group of cities only once and then ends in the starting city.
- How can we represent English sentences so that a computer can reason with them?
- How can we prove that there are infinitely many prime numbers?
- How can a list of integers be sorted so that the integers are in increasing order?
- How many steps are required to do such a sorting?
- How can it be proved that a sorting algorithm always correctly sorts a list?

# Learning Outcome

- **Discrete Structures:** Abstract mathematical structures that represent objects and the relationships between them. Examples are sets, permutations, relations, graphs, trees, and finite state machines
- **Combinatorial Analysis:** Techniques for counting objects of different kinds
- **Mathematical Reasoning:** Ability to read, understand, and construct mathematical arguments and proofs

# Sets And Functions

## Lecture 2

---

# Outline

- Introduction – Set Definition
- Describing Sets – Roster Notation, Set-Builder Notation, Venn Diagram
- Subsets, Power Sets, Cartesian Products
- Set Operations
- Function
- Describing Function

# Introduction

- Sets are one of the basic building blocks for the types of objects considered in discrete mathematics
- Set theory is an important branch of mathematics
- Many important discrete structures are built using sets
- Among them are combinations (unordered collections of objects used extensively in counting), relations (sets of ordered pairs that represent relationships between objects) graphs (sets of vertices and edges that connect vertices), and finite state machines (used to model computing machines)

# Set – Definition

- A *set* is an unordered collection of objects.
  - the students in this class
  - the chairs in this room
- The objects in a set are called the *elements*, or *members* of the set. A set is said to *contain* its elements.
- The notation  $a \in A$  denotes that  $a$  is an element of the set  $A$ .
- If  $a$  is not a member of  $A$ , write  $a \notin A$

# Set Description – Roster Method

- $S = \{a, b, c, d\}$
- Order not important

$$S = \{a, b, c, d\} = \{b, c, a, d\}$$

- Each distinct object is either a member or not; listing more than once does not change the set.

$$S = \{a, b, c, d\} = \{a, b, c, b, c, d\}$$

- Ellipses (...) may be used to describe a set without listing all of the members when the pattern is clear.

$$S = \{a, b, c, d, \dots, z\}$$

# Set Description – Roster Method

- Set of all vowels in the English alphabet:

$$V = \{a, e, i, o, u\}$$

- Set of all odd positive integers less than 10:

$$O = \{1, 3, 5, 7, 9\}$$

- Set of all positive integers less than 100:

$$S = \{1, 2, 3, \dots, 99\}$$

- Set of all integers less than 0:

$$S = \{\dots, -3, -2, -1\}$$



# Some Important Sets

- $\mathbf{N}$  = *natural numbers* =  $\{0,1,2,3,\dots\}$
- $\mathbf{Z}$  = *integers* =  $\{\dots,-3,-2,-1,0,1,2,3,\dots\}$
- $\mathbf{Z}^+$  = *positive integers* =  $\{1,2,3,\dots\}$
- $\mathbf{R}$  = set of *real numbers*
- $\mathbf{R}^+$  = set of *positive real numbers*
- $\mathbf{C}$  = set of *complex numbers*.
- $\mathbf{Q}$  = set of *rational numbers*

# Set Description – Set Builder Notation

- Specify the property or properties that all members must satisfy:

$$S = \{x \mid x \text{ is a positive integer less than } 100\}$$

$$O = \{x \mid x \text{ is an odd positive integer less than } 10\}$$

$$O = \{x \in \mathbf{Z}^+ \mid x \text{ is odd and } x < 10\}$$

- A predicate may be used:

$$S = \{x \mid P(x)\}$$

- Example:  $S = \{x \mid \text{Prime}(x)\}$

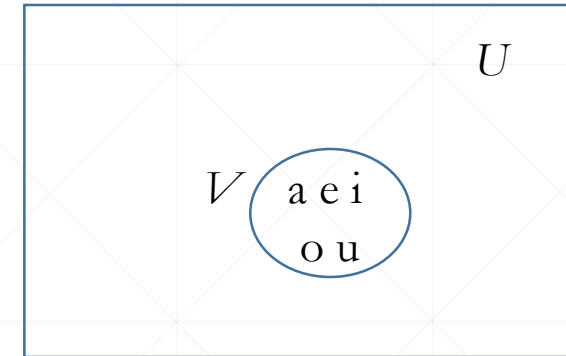
# Set Description – Interval Notation

- $[a,b] = \{x \mid a \leq x \leq b\}$
- $[a,b) = \{x \mid a \leq x < b\}$
- $(a,b] = \{x \mid a < x \leq b\}$
- $(a,b) = \{x \mid a < x < b\}$
- *closed interval*  $[a,b]$
- *open interval*  $(a,b)$

# Universal Set and Empty Set

- The *universal set*  $U$  is the set containing everything currently under consideration.
  - Sometimes implicit
  - Sometimes explicitly stated.
  - Contents depend on the context.
- The empty set is the set with no elements.
- Symbolized  $\emptyset$ , but  $\{ \}$  also used.

Venn Diagram



# Set Equality

- **Definition:** Two sets are *equal* if and only if they have the same elements.
- Therefore if  $A$  and  $B$  are sets, then  $A$  and  $B$  are equal if and only if

$$\forall x (x \in A \leftrightarrow x \in B)$$

- We write  $A = B$  if  $A$  and  $B$  are equal sets.

$$\{1, 3, 5\} = \{3, 5, 1\}$$

$$\{1, 5, 5, 5, 3, 3, 1\} = \{1, 3, 5\}$$

# Subsets

- **Definition:** The set  $A$  is a *subset* of  $B$ , if and only if every element of  $A$  is also an element of  $B$ .
- The notation  $A \subseteq B$  is used to indicate that  $A$  is a subset of the set  $B$ .
- $A \subseteq B$  holds if and only if  $\forall x(x \in A \rightarrow x \in B)$  is true.
- Every non-empty set  $S$  is guaranteed to have at least two subsets, the empty set and the set itself
- $\emptyset \subseteq S$ , for every set  $S$ .
- $S \subseteq S$ , for every set  $S$ .

# Showing a Set is or is not a Subset of Another Set

- **Showing that A is a Subset of B:** To show that  $A \subseteq B$ , show that if  $x$  belongs to  $A$ , then  $x$  also belongs to  $B$ .
- **Showing that A is not a Subset of B:** To show that  $A$  is not a subset of  $B$ ,  $A \not\subseteq B$ , find an element  $x \in A$  with  $x \notin B$ . (Such an  $x$  is a counterexample to the claim that  $x \in A$  implies  $x \in B$ .)

## Examples:

1. The set of all computer science majors at your school is a subset of all students at your school.
2. The set of integers with squares less than 100 is not a subset of the set of nonnegative integers.

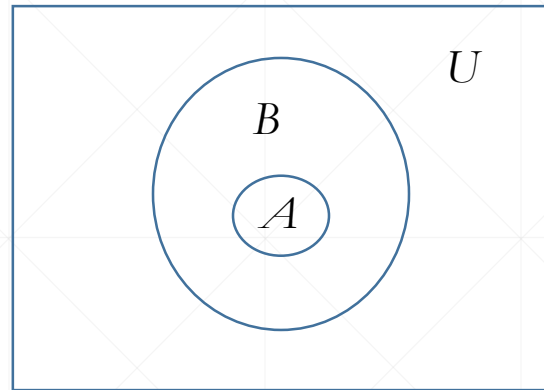
# Proper Subsets

- **Definition:** If  $A \subseteq B$ , but  $A \neq B$ , then we say  $A$  is a *proper subset* of  $B$ , denoted by  $A \subset B$ . If  $A \subset B$ , then

$$\forall x(x \in A \rightarrow x \in B) \wedge \exists x(x \in B \wedge x \notin A)$$

is true.

Venn Diagram





# Set Cardinality

- **Definition:** If there are exactly  $n$  distinct elements in  $S$  where  $n$  is a nonnegative integer, we say that  $S$  is *finite*. Otherwise it is *infinite*.
- **Definition:** The *cardinality* of a finite set  $A$ , denoted by  $|A|$ , is the number of (distinct) elements of  $A$ .

## Examples:

1.  $|\emptyset| = 0$
2. Let  $S$  be the letters of the English alphabet. Then  $|S| = 26$
3.  $|\{1,2,3\}| = 3$
4.  $|\{\emptyset\}| = 1$
5. The set of integers is infinite.

# Power Sets

- **Definition:** The set of all subsets of a set  $A$ , denoted  $\mathcal{P}(A)$ , is called the *power set* of  $A$ .
- **Example:** If  $A = \{a,b\}$  then
$$\mathcal{P}(A) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$$
- If a set has  $n$  elements, then the cardinality of the power set is  $2^n$ .

# Tuples

- The *ordered n-tuple*  $(a_1, a_2, \dots, a_n)$  is the ordered collection that has  $a_1$  as its first element and  $a_2$  as its second element and so on until  $a_n$  as its last element.
- Two n-tuples are equal if and only if their corresponding elements are equal.
- 2-tuples are called *ordered pairs*.
- The ordered pairs  $(a, b)$  and  $(c, d)$  are equal if and only if  $a = c$  and  $b = d$ .

# Cartesian Product

- **Definition:** The *Cartesian Product* of two sets  $A$  and  $B$ , denoted by  $A \times B$  is the set of ordered pairs  $(a,b)$  where  $a \in A$  and  $b \in B$ .

$$A \times B = \{(a, b) | a \in A \wedge b \in B\}$$

**Example:**

$$A = \{a, b\} \quad B = \{1, 2, 3\}$$

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

# Cartesian Product

**Definition:** The cartesian products of the sets  $A_1, A_2, \dots, A_n$ , denoted by  $A_1 \times A_2 \times \dots \times A_n$ , is the set of ordered  $n$ -tuples  $(a_1, a_2, \dots, a_n)$  where  $a_i$  belongs to  $A_i$  for  $i = 1, \dots, n$ .

$$A_1 \times A_2 \times \dots \times A_n = \{(a_1, a_2, \dots, a_n) | a_i \in A_i \text{ for } i = 1, 2, \dots, n\}$$

**Example:** What is  $A \times B \times C$  where  $A = \{0,1\}$ ,  $B = \{1,2\}$  and  $C = \{0,1,2\}$

**Solution:**  $A \times B \times C = \{(0,1,0), (0,1,1), (0,1,2), (0,2,0), (0,2,1), (0,2,2), (1,1,0), (1,1,1), (1,1,2), (1,2,0), (1,2,1), (1,2,2)\}$

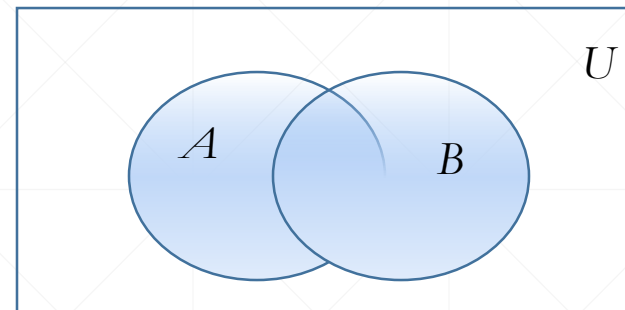
# Set Operation – Union

- **Definition:** Let  $A$  and  $B$  be sets. The union of the sets  $A$  and  $B$ , denoted by  $A \cup B$ , is the set:

$$\{x | x \in A \vee x \in B\}$$

- Example: What is  $\{1,2,3\} \cup \{3,4,5\}$ ?  
Solution:  $\{1,2,3,4,5\}$

Venn Diagram for  $A \cup B$



# Set Operation – Intersection

- **Definition:** The *intersection* of sets  $A$  and  $B$ , denoted by  $A \cap B$ , is

$$\{x | x \in A \wedge x \in B\}$$

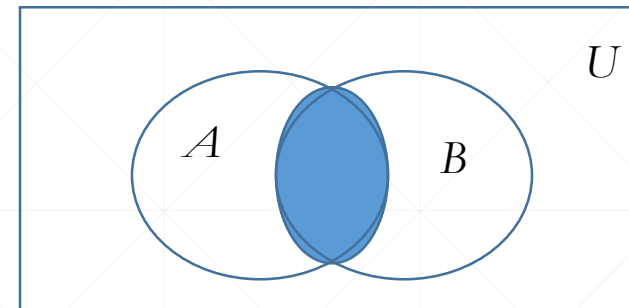
- Note if the intersection is empty, then  $A$  and  $B$  are said to be *disjoint*.
- **Example:** What is?  $\{1,2,3\} \cap \{3,4,5\}$  ?

**Solution:**  $\{3\}$

- **Example:** What is?  $\{1,2,3\} \cap \{4,5,6\}$  ?

**Solution:**  $\emptyset$

Venn Diagram for  $A \cap B$



# Set Operation – Complement

**Definition:** If  $A$  is a set, then the complement of the  $A$  (with respect to  $U$ ), denoted by  $\bar{A}$  is the set  $U - A$

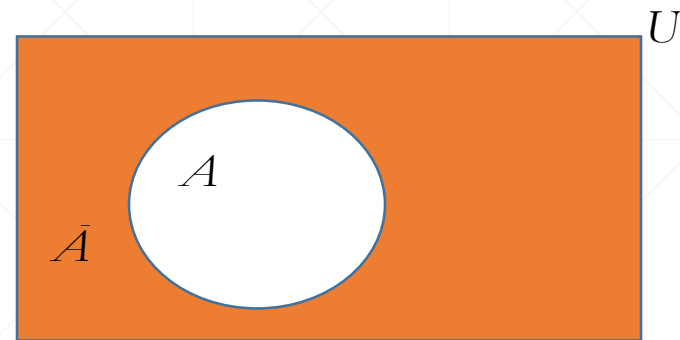
$$\bar{A} = \{x \in U \mid x \notin A\}$$

(The complement of  $A$  is sometimes denoted by  $A^c$ .)

**Example:** If  $U$  is the positive integers less than 100, what is the complement of  $\{x \mid x > 70\}$

**Solution:**  $\{x \mid x \leq 70\}$

Venn Diagram for Complement

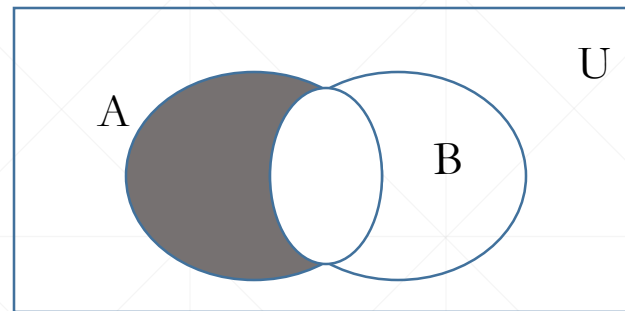




# Set Operation – Difference

- **Definition:** Let  $A$  and  $B$  be sets. The *difference* of  $A$  and  $B$ , denoted by  $A - B$ , is the set containing the elements of  $A$  that are not in  $B$ . The difference of  $A$  and  $B$  is also called the complement of  $B$  with respect to  $A$ .

$$A - B = \{x \mid x \in A \wedge x \notin B\} = A \cap \bar{B}$$



Venn Diagram for  $A - B$

# Set Operation – Symmetric Difference

**Definition:** The *symmetric difference* of **A** and **B**, denoted by  $A \oplus B$  is the set

$$(A - B) \cup (B - A)$$

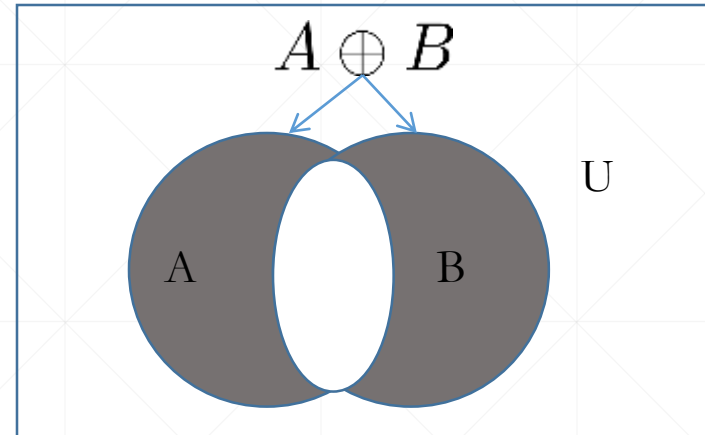
**Example:**

$$U = \{0,1,2,3,4,5,6,7,8,9,10\}$$

$$A = \{1,2,3,4,5\} \quad B = \{4,5,6,7,8\}$$

What is  $A \oplus B$  :

- **Solution:**  $\{1,2,3,6,7,8\}$



Venn Diagram

# Review Questions

**Example:**  $U = \{0,1,2,3,4,5,6,7,8,9,10\}$   $A = \{1,2,3,4,5\}$ ,  $B = \{4,5,6,7,8\}$

1.  $A \cup B$

**Solution:**  $\{1,2,3,4,5,6,7,8\}$

2.  $A \cap B$

**Solution:**  $\{4,5\}$

3.  $\bar{A}$

**Solution:**  $\{0,6,7,8,9,10\}$

# Review Questions

**Example:**  $U = \{0,1,2,3,4,5,6,7,8,9,10\}$   $A = \{1,2,3,4,5\}$ ,  $B = \{4,5,6,7,8\}$

4.  $\bar{B}$

**Solution:**  $\{0,1,2,3,9,10\}$

5.  $A - B$

**Solution:**  $\{1,2,3\}$

6.  $B - A$

**Solution:**  $\{6,7,8\}$

# Review Questions

- Determine whether each of these statements is true or false.
- a)  $x \in \{x\}$
- b)  $\{x\} \subseteq \{x\}$
- c)  $\{x\} \in \{x\}$
- d)  $\{x\} \in \{\{x\}\}$
- e)  $\emptyset \subseteq \{x\}$
- f)  $\emptyset \in \{x\}$
- a) True
- b) True
- c) False
- d) True
- e) True
- f) False

# Review Questions

- Let  $A$  be the set of students who live within one mile of school and let  $B$  be the set of students who walk to classes. Describe the students in each of these sets.
- a)  $A \cap B$
- **Solution:** Set of students who live within one mile of school AND who walk to classes
- b)  $A - B$
- **Solution:** Set of students who live within one mile of school AND DO NOT walk to classes

# Review Questions

- Suppose that  $A$  is the set of sophomores at your school and  $B$  is the set of students in discrete mathematics at your school. Express each of these sets in terms of  $A$  and  $B$ .
- **a)** the set of students at your school who either are sophomores or are taking discrete mathematics
- **b)** the set of students at your school who either are not sophomores or are not taking discrete mathematics

# Functions

## Lecture 3

---

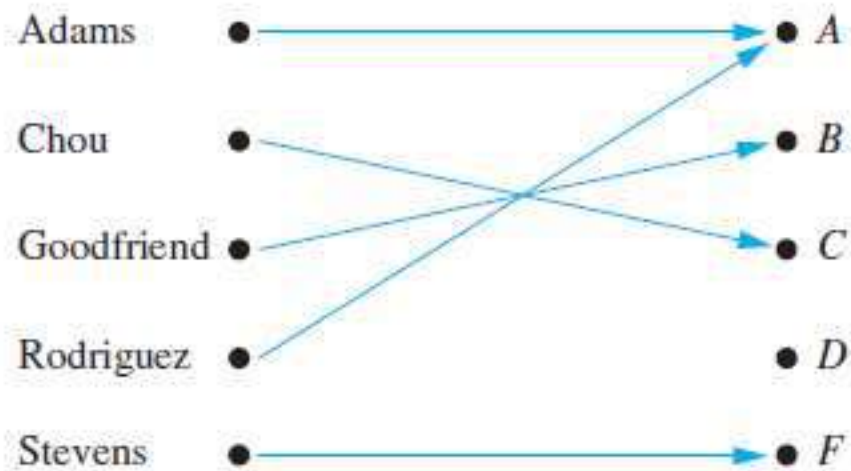


# Functions

- **Definition:** Let  $A$  and  $B$  be nonempty sets. A *function*  $f$  from  $A$  to  $B$ , denoted  $f: A \rightarrow B$  is an assignment of each element of  $A$  to exactly one element of  $B$ .
- We write  $f(a) = b$  if  $b$  is the unique element of  $B$  assigned by the function  $f$  to the element  $a$  of  $A$ .

# Functions

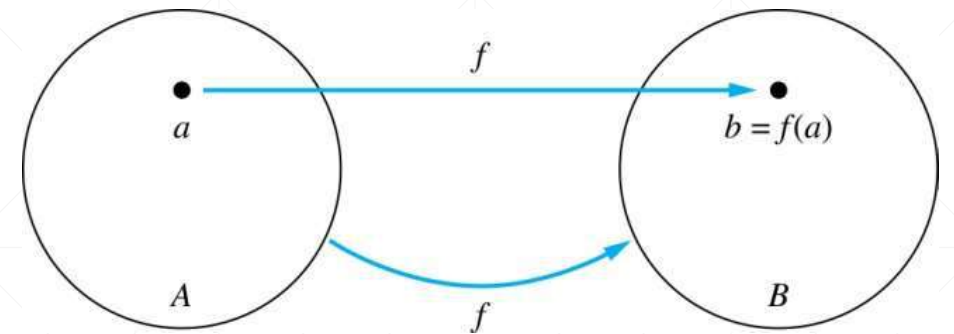
- Functions are sometimes called *mappings* or *transformations*.
- Assignment of grades in a discrete mathematics class



# Functions

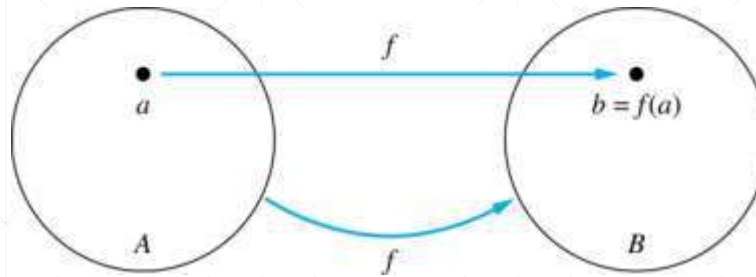
Given a function  $f: A \rightarrow B$ :

- We say  $f$  maps  $A$  to  $B$  or  $f$  is a *mapping* from  $A$  to  $B$ .
- $A$  is called the *domain* of  $f$  and  $B$  is called the *codomain* of  $f$ .
- If  $f(a) = b$ ,
  - then  $b$  is called the *image* of  $a$  under  $f$ .
  - $a$  is called the *preimage* of  $b$ .



# Functions

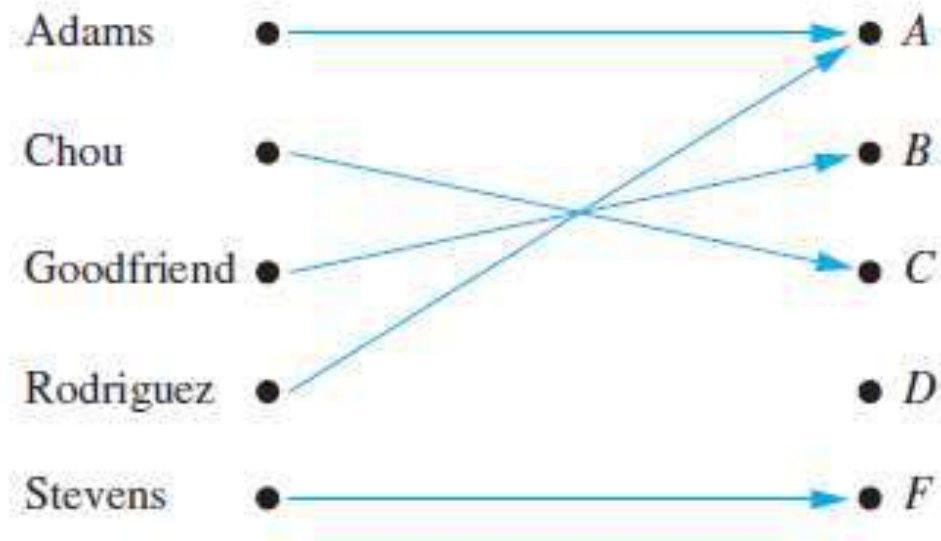
- The range of  $f$  is the set of all images of points in  $\mathbf{A}$  under  $f$
- We denote it by  $f(\mathbf{A})$



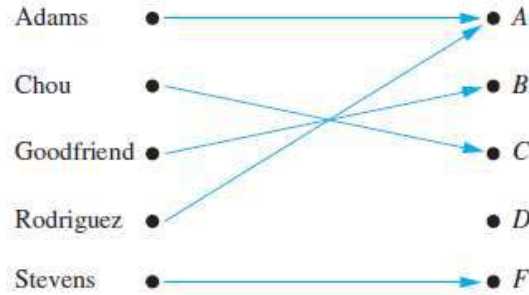
- Two functions are *equal* when they have the same domain, the same codomain and map each element of the domain to the same element of the codomain.

# Review Question

- What are the domain, codomain, and range of the function that assigns grades to students?



# Answer!



- Let  $F$  be the function mapping students to grades
- The domain of  $F$  is the set  $\{\text{Adams, Chou, Goodfriend, Rodriguez, Stevens}\}$
- Codomain is the set  $\{A, B, C, D, F\}$
- Range of  $F$  is the set  $\{A, B, C, F\}$

# Review Questions

$f(a) = ?$        $z$

The image of  $d$  is ?       $z$

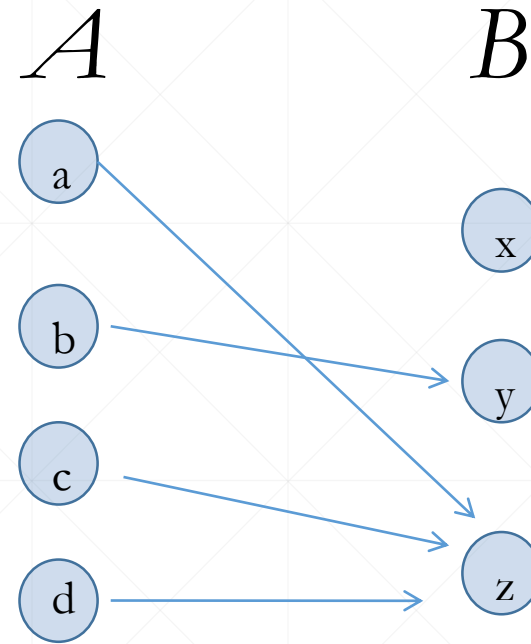
The domain of  $f$  is ?       $A$

The codomain of  $f$  is ?       $B$

The preimage of  $y$  is ?       $b$

$f(A) = ?$        $\{y, z\}$

The preimage(s) of  $z$  is (are) ?       $\{a, c, d\}$



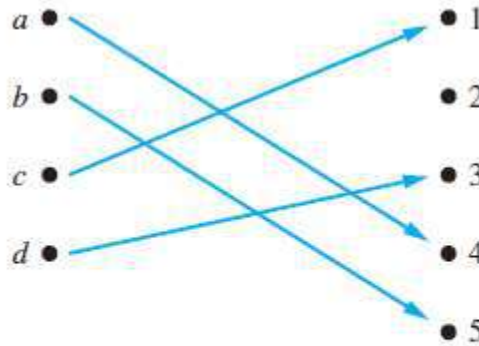
# Representing Functions

- Functions may be specified in different ways:
  1. An explicit statement of the assignment: Students and grades example
  2. A formula:  $f(x) = x + 1$
  3. A computer program: A C++/Java program that when given an integer  $n$ , produces the  $n$ th Fibonacci Number



# One-To-One Function – Injections

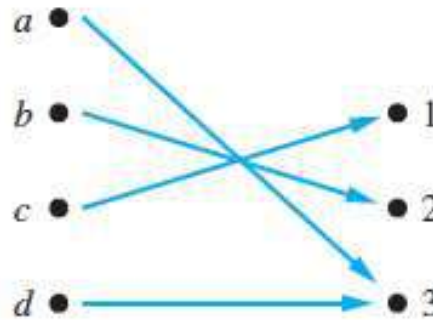
- **Definition:** A function  $f$  is said to be *one-to-one*, or *injective*, if and only if  $f(a) = f(b)$  implies that  $a = b$  for all  $a$  and  $b$  in the domain of  $f$ .
- A function is said to be an *injection* if it is one-to-one.



**FIGURE 3** A one-to-one function.

# Onto Function – Surjections

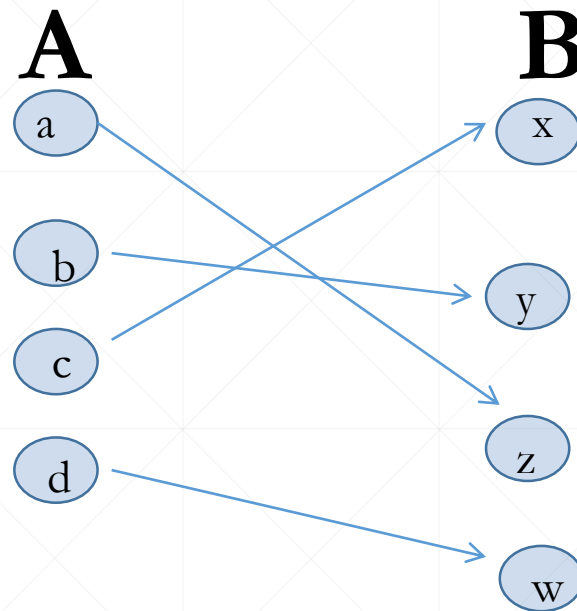
- **Definition:** A function  $f$  from  $A$  to  $B$  is called *onto* or *surjective*, if and only if for every element  $b \in B$  there is an element  $a \in A$  with  $f(a) = b$ .
- A function  $f$  is called a *surjection* if it is *onto*.



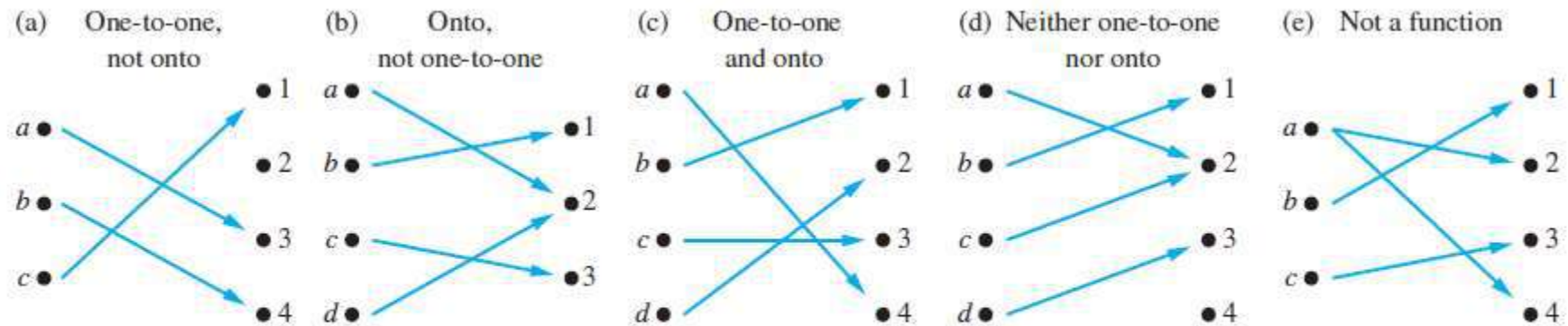
**FIGURE 4** An onto function.

# Bijections

- **Definition:** A function  $f$  is a *one-to-one correspondence*, or a *bijection*, if it is both one-to-one and onto (surjective and injective).

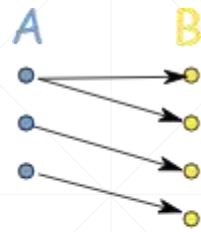


# Illustration I



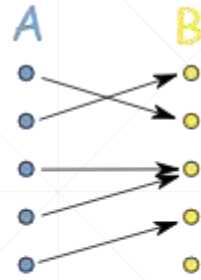
**FIGURE 5** Examples of different types of correspondences.

# Illustration II



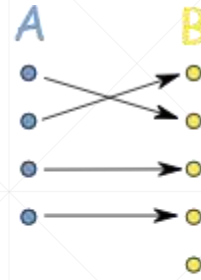
NOT a  
Function

*A has many B*



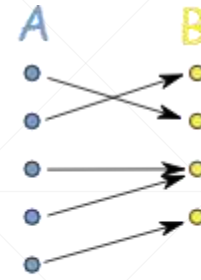
General  
Function

*B can have many A*



Injective  
(not surjective)

*B can't have many A*



Surjective  
(not injective)

*Every B has some A*



Bijjective  
(injective, surjective)

*A to B, perfectly*

# Showing that $f$ is one-to-one or onto

Suppose that  $f : A \rightarrow B$ .

*To show that  $f$  is injective* Show that if  $f(x) = f(y)$  for arbitrary  $x, y \in A$  with  $x \neq y$ , then  $x = y$ .

*To show that  $f$  is not injective* Find particular elements  $x, y \in A$  such that  $x \neq y$  and  $f(x) = f(y)$ .

*To show that  $f$  is surjective* Consider an arbitrary element  $y \in B$  and find an element  $x \in A$  such that  $f(x) = y$ .

*To show that  $f$  is not surjective* Find a particular  $y \in B$  such that  $f(x) \neq y$  for all  $x \in A$ .

# Showing that $f$ is one-to-one or onto

- **Example 1:** Let  $f$  be the function from  $\{a,b,c,d\}$  to  $\{1,2,3\}$  defined by  $f(a) = 3, f(b) = 2, f(c) = 1$ , and  $f(d) = 3$ . Is  $f$  an onto function?
- **Solution:** Yes,  $f$  is onto since all three elements of the codomain are images of elements in the domain. If the codomain were changed to  $\{1,2,3,4\}$ ,  $f$  would not be onto.

# Showing that $f$ is one-to-one or onto

- **Example 2:** Is the function  $f(x) = x^2$  from the set of integers to the set of integers onto?
- **Solution:** No,  $f$  is not onto because there is no integer  $x$  with  $x^2 = -1$ , for example.



# Practice Questions

- Section 2.1 Exercises – Page 131
- Questions 5, 11, 34
- Section 2.2 Exercises – Page 144
- Questions 1, 2, 3
- Section 2.3 Exercises – Page 161
- Questions 5, 10, 11, 17

# Discrete Structures

Spring 2024 – Week 2



# Sequence And Sums

## Lecture 1

---

# Introduction

- A discrete structure to represent an ordered lists of elements.
  - 1, 2, 3, 5, 8
  - 1, 3, 9, 27, 81, .....
- 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

**Definition:** A *sequence* is a function from a subset of the integers (usually either the set  $\{0, 1, 2, 3, 4, \dots\}$  or  $\{1, 2, 3, 4, \dots\}$ ) 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  $\{0, 1, 2, \dots\}$  to  $S$ .
- We call  $a_n$  a *term* of the sequence

# Sequences

**Example:** Consider the sequence  $\{a_n\}$  where  $a_n = \frac{1}{n}$

- The list of the terms of this sequence, beginning with  $a_1$  would be

$$\{a_n\} = \{a_1, a_2, a_3, \dots\}$$

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

# Arithmetic Progression

**Definition:** A *arithmetic progression* is a sequence of the form:

$$a, a + d, a + 2d, \dots, a + nd, \dots$$

where the *initial term*  $a$  and the *common difference*  $d$  are real numbers.

**Examples:**

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

$$\{s_n\} = \{s_0, s_1, s_2, s_3, s_4, \dots\} = \{-1, 3, 7, 11, 15, \dots\}$$

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

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

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

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

# Geometric Progression

**Definition:** A *geometric progression* is a sequence of the form:

$$a, ar, ar^2, \dots, ar^n, \dots$$

where the *initial term*  $a$  and the *common ratio*  $r$  are real numbers.

## Examples:

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

$$\{b_n\} = \{b_0, b_1, b_2, b_3, b_4, \dots\} = \{1, -1, 1, -1, 1, \dots\}$$

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

$$\{c_n\} = \{c_0, c_1, c_2, c_3, c_4, \dots\} = \{2, 10, 50, 250, 1250, \dots\}$$

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

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



# Recurrence Relations

- In previous examples, sequences are specified by providing explicit formulas for their terms
- Another way to specify a sequence is to provide one or more initial terms together with a rule for determining subsequent terms from those that precede them

# 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,  $a_0, a_1, \dots, a_{n-1}$ , 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  $n = 1, 2, 3, 4, \dots$  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

$$a_1 = a_0 + 3 = 2 + 3 = 5$$

$$a_2 = a_1 + 3 = 5 + 3 = 8$$

$$a_3 = a_2 + 3 = 8 + 3 = 11$$

# 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  $n = 2, 3, 4, \dots$  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
- $a_2 = a_1 - a_0 = 5 - 3 = 2$
- $a_3 = a_2 - a_1 = 2 - 5 = -3$

# Fibonacci Sequence

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

$$f_2 = f_1 + f_0 = 1 + 0 = 1$$

$$f_3 = f_2 + f_1 = 1 + 1 = 2$$

$$f_4 = f_3 + f_2 = 2 + 1 = 3$$

$$f_5 = f_4 + f_3 = 3 + 2 = 5$$

$$f_6 = f_5 + f_4 = 5 + 3 = 8$$

# 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*.

# Solving Recurrence Relations

- **Example 1:** Let  $\{a_n\}$  be a sequence that satisfies the recurrence relation  $a_n = a_{n-1} + 3$  for  $n = 1, 2, 3, 4, \dots$  and suppose that  $a_0 = 2$ . Solve the recurrence relation and initial condition.

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

# 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  $n = 2, 3, 4, \dots$  and suppose that  $a_1 = 2$ .

$$a_2 = 2 + 3$$

$$a_3 = (2 + 3) + 3 = 2 + 3 \cdot 2$$

$$a_4 = (2 + 2 \cdot 3) + 3 = 2 + 3 \cdot 3$$

.

.

.

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



# 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  $n = 2, 3, 4, \dots$  and suppose that  $a_1 = 2$ .

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

# Summations

- Sum of the terms  $a_m, a_{m+1}, \dots, a_n$  from the sequence  $\{a_n\}$
- The notation:

$$\sum_{j=m}^n a_j \quad \sum_{j=m}^n a_j \quad \sum_{m \leq j \leq n} a_j$$

represents

$$a_m + a_{m+1} + \dots + a_n$$

- The variable  $j$  is called the *index of summation*. It runs through all the integers starting with its *lower limit*  $m$  and ending with its *upper limit*  $n$ .

# Summations

- More generally for a set  $S$ :

$$\sum_{j \in S} a_j$$

- **Examples:**

$$r^0 + r^1 + r^2 + r^3 + \dots + r^n = \sum_0^n r^j$$

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots = \sum_1^{\infty} \frac{1}{i}$$

$$\text{If } S = \{2, 5, 7, 10\} \text{ then } \sum_{j \in S} a_j = a_2 + a_5 + a_7 + a_{10}$$

# Some Useful Summation Formulae

**TABLE 2** Some Useful Summation Formulae.

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

Geometric Series: We just proved this.

Later we will prove some of these by induction.

Proof in text (requires calculus)

# Propositional Logic

## Lecture 2

---

# Section Summary

- Propositions
- Connectives
  - Negation, Conjunction, Disjunction, Implication
  - Contrapositive, Inverse, Converse
  - Bi-conditional
- Truth Tables

# Propositions

- A *proposition* is a declarative sentence that is either true or false.
- Examples of propositions:
  - a) The Moon is made of green cheese.
  - b) Trenton is the capital of New Jersey.
  - c) Toronto is the capital of Canada.
  - d)  $1 + 0 = 1$
  - e)  $0 + 0 = 2$

# Propositions

- Examples that are not propositions.
  - a) Sit down!
  - b) What time is it?
  - c)  $x + 1 = 2$
  - d)  $x + y = z$



# Propositional Logic

- Constructing Propositions
- Propositional Variables:  $p, q, r, s, \dots$
- The proposition that is always true is denoted by **T** and the proposition that is always false is denoted by **F**.
- Compound Propositions – constructed from logical connectives and other propositions
- Negation  $\neg$ , Conjunction  $\wedge$ , Disjunction  $\vee$ , Implication  $\rightarrow$ , Biconditional  $\leftrightarrow$

# Compound Propositions: Negation

- The *negation* of a proposition  $p$  is denoted by  $\neg p$  and has this truth table:

$p$	$\neg p$
T	F
F	T

- Example:** If  $p$  denotes “The earth is round.”, then  $\neg p$  denotes “It is not the case that the earth is round,” or more simply “The earth is not round.”

# Compound Propositions: Conjunction

- The *conjunction* of propositions  $p$  and  $q$  is denoted by  $p \wedge q$  and has this truth table:

$p$	$q$	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

- Example:** If  $p$  denotes “I am at home.” and  $q$  denotes “It is raining.” then  $p \wedge q$  denotes “I am at home and it is raining.”

# Compound Propositions: Disjunction

- The *disjunction* of propositions  $p$  and  $q$  is denoted by  $p \vee q$  and has this truth table:

$p$	$q$	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

- Example:** If  $p$  denotes “I am at home.” and  $q$  denotes “It is raining.” then  $p \vee q$  denotes “I am at home or it is raining.”

# The Connective Or in English

- In English “or” has two distinct meanings.
- “Inclusive Or” - In the sentence “Students who have taken CS202 or Math120 may take this class,” we assume that students need to have taken one of the prerequisites, but may have taken both. This is the meaning of disjunction. For  $p \vee q$  to be true, either one or both of  $p$  and  $q$  must be true.
- “Exclusive Or” - When reading the sentence “Soup or salad comes with this entrée,” we do not expect to be able to get both soup and salad. This is the meaning of Exclusive Or (Xor). In  $p \oplus q$ , one of  $p$  and  $q$  must be true, but not both. The truth table for  $\oplus$  is:

# The Connective Or in English

$p$	$q$	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

# Compound Propositions: Implication

- If  $p$  and  $q$  are propositions, then  $p \rightarrow q$  is a *conditional statement* or *implication* which is read as “if  $p$ , then  $q$ ” and has this truth table:

$p$	$q$	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

# Compound Propositions: Implication

- **Example:** If  $p$  denotes “I am at home.” and  $q$  denotes “It is raining.” then  $p \rightarrow q$  denotes “If I am at home then it is raining.”
- In  $p \rightarrow q$ ,  $p$  is the *hypothesis* (*antecedent* or *premise*) and  $q$  is the *conclusion* (or *consequence*).



# Understanding Implication

- In  $p \rightarrow q$  there does not need to be any connection between the antecedent or the consequent. The “meaning” of  $p \rightarrow q$  depends only on the truth values of  $p$  and  $q$ .
- These implications are perfectly fine, but would not be used in ordinary English.
- *“If the moon is made of green cheese, then I have more money than Bill Gates.”*
- *“If the moon is made of green cheese then I’m on welfare.”*
- *“If  $1 + 1 = 3$ , then your grandma wears combat boots.”*

# Understanding Implication (cont)

- One way to view the logical conditional is to think of an obligation or contract.
- *“If I am elected, then I will lower taxes.”*
- *“If you get 100% on the final, then you will get an A.”*
- If the politician is elected and does not lower taxes, then the voters can say that he or she has broken the campaign pledge. Something similar holds for the professor. This corresponds to the case where  $p$  is true and  $q$  is false.

# Different Ways of Expressing $p \rightarrow q$

- if  $p$ , then  $q$
- if  $p$ ,  $q$
- $q$  unless  $\neg p$
- $q$  if  $p$
- $q$  whenever  $p$
- $q$  follows from  $p$

$p$  implies  $q$

$p$  only if  $q$

$q$  when  $p$

$p$  is sufficient for  $q$

$q$  is necessary for  $p$

a necessary condition for  $p$  is  $q$

a sufficient condition for  $q$  is  $p$

# Converse, Contra-Positive, and Inverse

- From  $p \rightarrow q$  we can form new conditional statements
- $q \rightarrow p$  is the **converse** of  $p \rightarrow q$
- $\neg q \rightarrow \neg p$  is the **contrapositive** of  $p \rightarrow q$
- $\neg p \rightarrow \neg q$  is the **inverse** of  $p \rightarrow q$

# Converse, Contra-Positive, and Inverse

- **Example:** Find the converse, inverse, and contra-positive of “It raining is a sufficient condition for my not going to town.”
- **Solution:**
- **converse:** If I do not go to town, then it is raining.
- **inverse:** If it is not raining, then I will go to town.
- **contrapositive:** If I go to town, then it is not raining.

# Bi-Conditional

- If  $p$  and  $q$  are propositions, then we can form the *biconditional* proposition  $p \leftrightarrow q$ , read as “ $p$  if and only if  $q$ .” The biconditional  $p \leftrightarrow q$  denotes the proposition with this truth table:

$p$	$q$	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

# Bi-Conditional

- If  $p$  denotes “I am at home.” and  $q$  denotes “It is raining.”
- Then  $p \leftrightarrow q$  denotes “I am at home if and only if it is raining.”

# Expressing the Bi-Conditional

- Some alternative ways “ $p$  if and only if  $q$ ” is expressed in English:
  - $p$  is necessary and sufficient for  $q$
  - if  $p$  then  $q$  , and conversely
  - $p$  iff  $q$



# Truth Tables For Compound Propositions

- Construction of a truth table:
- Rows – need a row for every possible combination of values for the atomic propositions.
- Columns - need a column for the compound proposition (usually at far right). Need a column for the truth value of each expression that occurs in the compound proposition as it is built up. This includes the atomic propositions

# Example Truth Table

- Construct a truth table for

$$p \vee q \rightarrow \neg r$$

$p$	$q$	$r$	$\neg r$	$p \vee q$	$p \vee q \rightarrow \neg r$
T	T	T	F	T	F
T	T	F	T	T	T
T	F	T	F	T	F
T	F	F	T	T	T
F	T	T	F	T	F
F	T	F	T	T	T
F	F	T	F	F	T
F	F	F	T	F	T

# Equivalent Propositions

- Two propositions are *equivalent* if they always have the same truth value.
- **Example:** Show using a truth table that the conditional is equivalent to the contra-positive.
- **Solution:**

$p$	$q$	$\neg p$	$\neg q$	$p \rightarrow q$	$\neg q \rightarrow \neg p$
T	T	F	F	T	T
T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	T

# Using a Truth Table to Show Non-Equivalence

- **Example:** Show using truth tables that neither the converse nor inverse of an implication are not equivalent to the implication.
- **Solution:**

$p$	$q$	$\neg p$	$\neg q$	$p \rightarrow q$	$\neg p \rightarrow \neg q$	$q \rightarrow p$
T	T	F	F	T	T	T
T	F	F	T	F	T	T
F	T	T	F	T	F	F
F	F	T	T	T	T	T

# Problem

- How many rows are there in a truth table with  $n$  propositional variables?
- **Solution:**  $2^n$
- Note that this means that with  $n$  propositional variables, we can construct  $2^n$  distinct (i.e., not equivalent) propositions.

# Precedence of Logical Operators

Operator	Precedence
$\neg$	1
$\wedge$	2
$\vee$	3
$\rightarrow$	4
$\leftrightarrow$	5

- $p \vee q \rightarrow \neg r$  is equivalent to  $(p \vee q) \rightarrow \neg r$
- If the intended meaning is  $p \vee (q \rightarrow \neg r)$  then parentheses must be used.

# Applications of Propositional Logic

Lecture 3

---

# Applications of Propositional Logic

1. Translating English to Propositional Logic
2. System Specifications
3. Logic Circuits



# Translating English Sentences

- Steps to convert an English sentence to a statement in propositional logic
  - ✓ Identify atomic propositions and represent using propositional variables.
  - ✓ Determine appropriate logical connectives

# Translating English Sentences – Example I

- “If I go to Harry’s or to the country, I will not go shopping.”
- $p$ : I go to Harry’s
- $q$ : I go to the country.
- $r$ : I will go shopping.

If  $p$  or  $q$  then not  $r$ .

$$(p \vee q) \rightarrow \neg r$$

# Translating English Sentences – Example II

- Translate the following sentence into propositional logic:
- “You can access the Internet from campus only if you are a computer science major or you are not a freshman.”
- Let  $a$ ,  $c$ , and  $f$  represent respectively “You can access the internet from campus,” “You are a computer science major,” and “You are a freshman.”

- $$a \rightarrow (c \vee \neg f)$$

# System Specifications

- System and Software engineers take requirements in English and express them in a precise specification language based on logic.
- **Example:** Express in propositional logic:  
“The automated reply cannot be sent when the file system is full”
- **Solution:** One possible solution: Let  $p$  denote “The automated reply can be sent” and  $q$  denote “The file system is full.”

$$q \rightarrow \neg p$$

# Consistent System Specifications

- **Definition:** A list of propositions is *consistent* if it is possible to assign truth values to the proposition variables so that each proposition is true

# Consistent System Specifications – Exercise

- Are these specifications consistent?
- “The diagnostic message is stored in the buffer or it is retransmitted.”
- “The diagnostic message is not stored in the buffer.”
- “If the diagnostic message is stored in the buffer, then it is retransmitted.”

# Consistent System Specifications – Solution

- Let  $p$  denote “The diagnostic message is stored in the buffer.”
- Let  $q$  denote “The diagnostic message is retransmitted”
- The specification can be written as:  $p \vee q, \neg p, p \rightarrow q$ .
- When  $p$  is false and  $q$  is true all three statements are true, hence the specification is consistent
- Could come to same conclusion by use of a truth table **(HW)**

# Consistent System Specifications

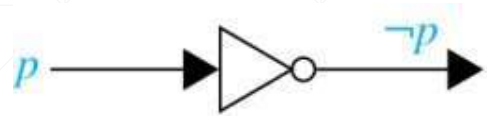
- What if another proposition “The diagnostic message is not retransmitted is added.”
- Added  $\neg q$
- There is no satisfying assignment, so the specification is not consistent.



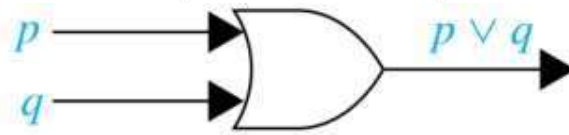
# Logic Gates

- Electronic circuits; each input/output signal can be viewed as a 0 or 1
- 0 represents False, 1 represents True
- Circuits are constructed from three basic circuits called gates
- The inverter (NOT gate) takes an input bit and produces the negation of that bit. The OR gate takes two input bits and produces the value equivalent to the disjunction of the two bits. The AND gate takes two input bits and produces the value equivalent to the conjunction of the two bits.

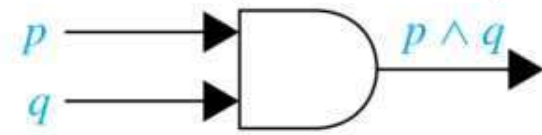
# Logic Gates



Inverter



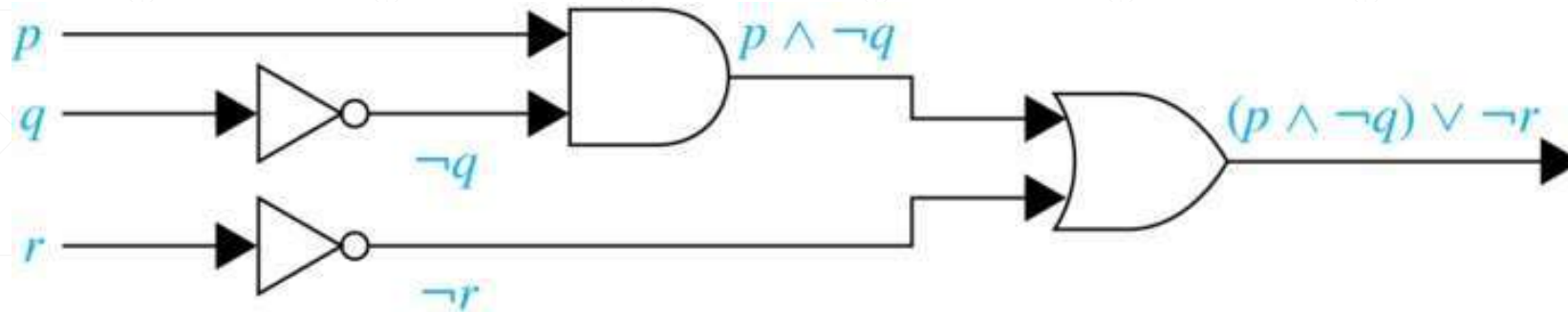
OR gate



AND gate

# Logic Gates

- More complicated digital circuits can be constructed by combining these basic circuits to produce the desired output given the input signals by building a circuit for each piece of the output expression and then combining them. For example:



# Practice Exercises

- Section 2.4 Page 177
- Questions 4, 7, 17 (a, b, c), 29
- Section 1.1 Page 13
- Questions 11, 14, 29, 34 (a, b, c)
- Section 1.2 Page
- Questions 5, 10, 47

# Discrete Structures

Spring 2024 – Week 3



# Predicates

## Lecture 1

---

# Introduction

- Propositional logic cannot adequately express the meaning of all statements in mathematics and in natural language
- Suppose if we have: “All men are mortal.”, there is no rule of propositional logic allow to conclude the truth of the statement, “Socrates is a man.”
- Need a language that talks about objects, their properties, and their relations

# Predicate Logic

- Predicate logic (First Order Logic) – more powerful type of logic
- Used to express the meaning of a wide range of statements in mathematics and computer science in ways that permit us to ***reason*** and ***explore relationships*** between objects
- Predicate logic uses the following new features:
  - Variables:  $x, y, z$
  - Predicates:  $P(x), M(x)$
  - Quantifiers (to be covered later)



# Predicates

- Statements involving variables, such as,  $x > 3$  are neither true nor false when the values of the variables are not specified
- The statement “x is greater than 3” has two parts
  1. Variable x, the subject of the statement
  2. “is greater than 3” – property that the subject of the statement can have – a **predicate**

# Predicates

- The statement “ $x$  is greater than 3” can be denoted by  $P(x)$ , where  $P$  denotes the predicate “is greater than 3” and  $x$  is the variable
- The statement  $P(x)$  is also said to be the value of the **propositional function**  $P$  at  $x$

# Predicate / Propositional Functions

- Predicate / Propositional Function are a generalization of propositions
- Propositional functions become propositions (and have truth values) when their variables are each replaced by a value from the domain (or bound by a quantifier)

# Predicate / Propositional Functions

- Let  $P(x)$  denote “ $x > 0$ ” and the domain be the integers. Then find:
- $P(-3)$
- $P(0)$
- $P(3)$

# Predicate / Propositional Functions

- Let  $P(x)$  denote “ $x > 0$ ” and the domain be the integers. Then:
- $P(-3)$             false
- $P(0)$             false
- $P(3)$             true
  
- Often the domain is denoted by  $U$ .
- In this example  $U$  is the integers

# Example I

- Let  $P(x)$  denote the statement “ $x > 3$ .” What are the truth values of  $P(4)$  and  $P(2)$ ?
- Solution:
- Obtain the statement  $P(4)$  by setting  $x = 4$  in the statement “ $x > 3$ .” Hence,  $P(4)$ , which is the statement “ $4 > 3$ ,” is true.
- However,  $P(2)$ , which is the statement “ $2 > 3$ ,” is false

# Example II

- Let “ $x + y = z$ ” be denoted by  $R(x, y, z)$  and  $U$  (for all three variables) be the integers. Find the following truth values:
  1.  $R(2, -1, 5)$
  2.  $R(3, 4, 7)$
  3.  $R(x, 3, z)$

# Example II

- Let “ $x + y = z$ ” be denoted by  $R(x, y, z)$  and  $U$  (for all three variables) be the integers. Find the following truth values:

1.  $R(2, -1, 5)$       Solution: F
2.  $R(3, 4, 7)$       Solution: T
3.  $R(x, 3, z)$       Solution: Not a Proposition



# Example III

- Let “ $x - y = z$ ” be denoted by  $Q(x, y, z)$ , with  $U$  as the integers. Find these truth values:
  1.  $Q(2, -1, 3)$
  2.  $Q(3, 4, 7)$
  3.  $Q(x, 3, z)$

# Compound Expressions

- Connectives from propositional logic carry over to predicate logic.
- Disjunction  $\vee$
- Conjunction  $\wedge$
- Implication  $\rightarrow$
- Bi-Conditional  $\leftrightarrow$

# Compound Expressions

- Connectives from propositional logic carry over to predicate logic.
- If  $P(x)$  denotes “ $x > 0$ ,” find these truth values:
- $P(3) \vee P(-1)$
- $P(3) \wedge P(-1)$
- $P(3) \rightarrow P(-1)$
- $P(3) \rightarrow \neg P(-1)$

# Compound Expressions

- If  $P(x)$  denotes “ $x > 0$ ,” find these truth values:

- $P(3) \vee P(-1)$

Solution: T

- $P(3) \wedge P(-1)$

Solution: F

- $P(3) \rightarrow P(-1)$

Solution: F

- $P(3) \rightarrow \neg P(-1)$

Solution: T

# Quantifiers

## Lecture 2

---

# Quantifiers

- We need quantifiers to express the meaning of English words including *all* and *some*
  - “All men are Mortal.”
  - “Some cats do not have fur.”
- The two most important quantifiers are:
  - *Universal Quantifier*, “For all,” symbol:  $\forall$
  - *Existential Quantifier*, “There exists,” symbol:  $\exists$

# Quantifiers

- Quantifiers are used to write as in  $\forall x P(x)$  and  $\exists x P(x)$
- $\forall x P(x)$  asserts  $P(x)$  is true for every  $x$  in the *domain*.
- $\exists x P(x)$  asserts  $P(x)$  is true for some  $x$  in the *domain*.
- The quantifiers are said to bind the variable  $x$  in these expressions.

# Universal Quantifier

- $\forall x P(x)$  is read as “For all  $x$ ,  $P(x)$ ” or “For every  $x$ ,  $P(x)$ ”
- *Examples:*
- If  $P(x)$  denotes “ $x > 0$ ” and  $U$  is integers, then  $\forall x P(x)$  is false
- If  $P(x)$  denotes “ $x > 0$ ” and  $U$  is positive integers, then  $\forall x P(x)$  is true.
- If  $P(x)$  denotes “ $x$  is even” and  $U$  is integers, then  $\forall x P(x)$  is false



# Existential Quantifier

- $\exists x P(x)$  is read as “For some  $x$ ,  $P(x)$ ”, or as “There is an  $x$  such that  $P(x)$ ,” or “For at least one  $x$ ,  $P(x)$ .”
- *Examples*
- If  $P(x)$  denotes “ $x > 0$ ” and  $U$  is the integers, then  $\exists x P(x)$  is true. It is also true if  $U$  is the positive integers
- If  $P(x)$  denotes “ $x < 0$ ” and  $U$  is the positive integers, then  $\exists x P(x)$  is false
- If  $P(x)$  denotes “ $x$  is even” and  $U$  is the integers, then  $\exists x P(x)$  is true

# Thinking About Quantifiers

- When the domain of discourse is finite, we can think of quantification as looping through the elements of the domain
- To evaluate  $\forall x P(x)$  loop through all  $x$  in the domain
  - If at every step  $P(x)$  is true, then  $\forall x P(x)$  is true.
  - If at a step  $P(x)$  is false, then  $\forall x P(x)$  is false and the loop terminates
- To evaluate  $\exists x P(x)$  loop through all  $x$  in the domain
  - If at some step,  $P(x)$  is true, then  $\exists x P(x)$  is true and the loop terminates.
  - If the loop ends without finding an  $x$  for which  $P(x)$  is true, then  $\exists x P(x)$  is false

# Properties of Quantifiers

- The truth value of  $\exists x P(x)$  and  $\forall x P(x)$  depend on both the propositional function  $P(x)$  and on the domain  $U$ 
  1. If  $U$  is the positive integers and  $P(x)$  is the statement “ $x < 2$ ”, then  $\exists x P(x)$  is true, but  $\forall x P(x)$  is false
  2. If  $U$  is the negative integers and  $P(x)$  is the statement “ $x < 2$ ”, then both  $\exists x P(x)$  and  $\forall x P(x)$  are true
  3. If  $U$  consists of 3, 4, and 5, and  $P(x)$  is the statement “ $x > 2$ ”, then both  $\exists x P(x)$  and  $\forall x P(x)$  are true. But if  $P(x)$  is the statement “ $x < 2$ ”, then both  $\exists x P(x)$  and  $\forall x P(x)$  are false

# Precedence of Quantifiers

- The quantifiers  $\forall$  and  $\exists$  have higher precedence than all the logical operators
- For example,  $\forall x P(x) \vee Q(x)$  means  $(\forall x P(x)) \vee Q(x)$
- $\forall x (P(x) \vee Q(x))$  means something different
- Unfortunately, often people write  $\forall x P(x) \vee Q(x)$  when they mean  $\forall x (P(x) \vee Q(x))$ .

# Equivalences in Predicate Logic

- Statements involving predicates and quantifiers are *logically equivalent* if and only if they have the same truth value
- For every predicate substituted into these statements, and
- For every domain of discourse used for the variables in the expressions.
- The notation  $S \equiv T$  indicates that  $S$  and  $T$  are logically equivalent

# Quantifiers as Conjunctions and Disjunctions

- If the domain is finite,  $U$  consists of 1, 2, and 3
- Universally quantified proposition is equivalent to a conjunction of propositions without quantifiers

$$\forall x P(x) \equiv P(1) \wedge P(2) \wedge P(3)$$

- Existentially quantified proposition is equivalent to a disjunction of propositions without quantifiers

$$\exists x P(x) \equiv P(1) \vee P(2) \vee P(3)$$

# Negating Quantified Expressions – Universal Quantifier

- Consider  $\forall x J(x)$  “Every student in your class has taken a course in Java.”  
Here  $J(x)$  is “ $x$  has taken a course in Java”
- Negating the original statement gives
- “It is not the case that every student in your class has taken Java.”
- This implies that “There is a student in your class who has not taken Java.”
- Symbolically  $\neg \forall x J(x)$  and  $\exists x \neg J(x)$  are equivalent

# Negating Quantified Expressions – Existential Quantifier

- Consider  $\exists x J(x)$  “There is a student in this class who has taken a course in Java.” Where  $J(x)$  is “ $x$  has taken a course in Java.”
- Negating the original statement gives
- “It is not the case that there is a student in this class who has taken Java.”
- This implies that “Every student in this class has not taken Java”
- Symbolically  $\neg \exists x J(x)$  and  $\forall x \neg J(x)$  are equivalent



# De Morgan's Law for Quantifiers

- The rules for negating quantifiers are:

TABLE 2 De Morgan's Laws for Quantifiers.			
<i>Negation</i>	<i>Equivalent Statement</i>	<i>When Is Negation True?</i>	<i>When False?</i>
$\neg\exists x P(x)$	$\forall x \neg P(x)$	For every $x$ , $P(x)$ is false.	There is an $x$ for which $P(x)$ is true.
$\neg\forall x P(x)$	$\exists x \neg P(x)$	There is an $x$ for which $P(x)$ is false.	$P(x)$ is true for every $x$ .

- The reasoning in the table shows that:

$$\neg\exists x P(x) \equiv \forall x \neg P(x)$$

$$\neg\forall x P(x) \equiv \exists x \neg P(x)$$

# Using Predicate Logic

## Lecture 3

---

# Review Question I

- Let  $P(x)$  denote the statement “ $x \leq 4$ .” What are these truth values?
- **a)**  $P(0)$
- **b)**  $P(4)$
- **c)**  $P(6)$

# Review Question II

- Let  $P(x)$  be the statement “ $x$  spends more than five hours every weekday in class,” where the domain for  $x$  consists of all students. Express each of these quantifications in English.
- a)  $\exists x P(x)$
- b)  $\forall x P(x)$

# Review Question III

- Determine the truth value of each of these statements if the domain for all variables consists of all integers.
- a)  $\forall n (n^2 \geq 0)$
- b)  $\exists n (n^2 == 2)$
- c)  $\forall n (n^2 \geq n)$
- d)  $\exists n (n^2 < 0)$

# Translating from English to Predicate Logic

1. Decide on  $U$  (Domain)
2. Define Propositional Function  $P(x)$
3. Use Appropriate Quantifiers – Universal / Existential

# Translating from English to Predicate Logic – I

- “Every student in this class has taken a course in Java.”
  1. Decide on  $U$  (domain):  $U$  all students in this class
  2. Define Propositional Function:  $J(x)$  denoting, “ $x$  has taken a course in Java”
  3. Use Appropriate Quantifiers
- Answer
- $\forall x J(x)$

# Translating from English to Predicate Logic – II

- “Every student in this class has taken a course in Java.”
  1. Decide on  $U$  (domain):  $U$  all people
  2. Define Propositional Function:  $J(x)$  denoting, “ $x$  has taken a course in Java”,  $S(x)$  denoting, “ $x$  is a student in this class”
  3. Use Appropriate Quantifiers
- Answer
  - $\forall x (S(x) \rightarrow J(x))$



# Translating from English to Predicate Logic – III

- “Some student in this class has taken a course in Java.”
  - If  $U$  is all students in this class
  - $J(x)$  denoting, “ $x$  has taken a course in Java”
  - Answer:  $\exists x J(x)$
- 
- If  $U$  is all people
  - $S(x)$  denoting, “ $x$  is a student in this class”,
  - Answer:  $\exists x (S(x) \wedge J(x))$

# Translating from English to Predicate Logic – IV

1. “Some student in this class has visited Mexico.”

- Let  $U$  be all people,  $M(x)$  denote “ $x$  has visited Mexico” and  $S(x)$  denote “ $x$  is a student in this class”

$$\exists x (S(x) \wedge M(x))$$

2. “Every student in this class has visited Canada or Mexico.”

- Add  $C(x)$  denoting “ $x$  has visited Canada.”

$$\forall x (S(x) \rightarrow (M(x) \vee C(x)))$$

# Review Question IV

- Let  $P(x)$  be the statement “ $x$  can speak Russian” and let  $Q(x)$  be the statement “ $x$  knows the computer language C++.” Express each of these sentences in terms of  $P(x)$ ,  $Q(x)$ , quantifiers, and logical connectives. The domain for quantifiers consists of all students at your school.
- **a)** There is a student at your school who can speak Russian and who knows C++.
- **b)** There is a student at your school who can speak Russian but who doesn't know C++.

# Review Question IV (contd.)

- **c)** Every student at your school either can speak Russian or knows C++.
- **d)** No student at your school can speak Russian or knows C++.

# Review Question V

- Translate each of these statements into logical expressions using predicates, quantifiers, and logical connectives.
- **a)** No one is perfect.
- **b)** Not everyone is perfect.
- **c)** All your friends are perfect.
- **d)** At least one of your friends is perfect

# Review Question V (contd.)

- e) Everyone is your friend and is perfect.
- f ) Not everybody is your friend or someone is not perfect.

# Practice

- $U = \{\text{fleegles, snurds, thingamabobs}\}$
- $F(x)$ :  $x$  is a fleegle
- $S(x)$ :  $x$  is a snurd
- $T(x)$ :  $x$  is a thingamabob

1. “Everything is a fleegle”

- **Solution:**  $\forall x F(x)$

# Practice

- $U = \{\text{fleegles, snurds, thingamabobs}\}$
- $F(x)$ :  $x$  is a fleegle
- $S(x)$ :  $x$  is a snurd
- $T(x)$ :  $x$  is a thingamabob

2. “Nothing is a snurd.”

- **Solution:**  $\neg \exists x S(x)$  or  $\forall x \neg S(x)$



# Practice

- $U = \{\text{fleegles, snurds, thingamabobs}\}$
- $F(x)$ :  $x$  is a fleegle
- $S(x)$ :  $x$  is a snurd
- $T(x)$ :  $x$  is a thingamabob

3. “All fleegles are snurds.”

- **Solution:**  $\forall x (F(x) \rightarrow S(x))$

# Practice

- $U = \{\text{fleegles, snurds, thingamabobs}\}$
- $F(x)$ :  $x$  is a fleegle
- $S(x)$ :  $x$  is a snurd
- $T(x)$ :  $x$  is a thingamabob

4. “Some fleegles are thingamabobs.”

- **Solution:**  $\exists x (F(x) \wedge T(x))$

# Practice

- $U = \{\text{fleegles, snurds, thingamabobs}\}$
- $F(x)$ :  $x$  is a fleegle
- $S(x)$ :  $x$  is a snurd
- $T(x)$ :  $x$  is a thingamabob

5. “No snurd is a thingamabob.”

- **Solution:**  $\neg \exists x (S(x) \wedge T(x))$  or  $\forall x (\neg S(x) \vee \neg T(x))$

# Practice

- $U = \{\text{fleegles, snurds, thingamabobs}\}$
- $F(x)$ :  $x$  is a fleegle
- $S(x)$ :  $x$  is a snurd
- $T(x)$ :  $x$  is a thingamabob

6. “If any fleegle is a snurd then it is also a thingamabob.”

- **Solution:**  $\forall x ((F(x) \wedge S(x)) \rightarrow T(x))$

# Discrete Structures

Spring 2024 – Week 4



# Valid Arguments

## Lecture 1

---

# Preliminary

- A compound proposition that is always true, no matter what the truth values of the propositional variables that occur in it, is called a *tautology*.
- A compound proposition that is always false is called a *contradiction*.
- A compound proposition that is neither a tautology nor a contradiction is called a *contingency*

# Tautology, Contradiction, Contingency – Examples

- Either it will rain tomorrow, or it will not rain tomorrow – a *tautology*
- It is raining now, and it is not raining now – a *contradiction*
- If we go to store, we will buy some apples – a *contingency*



# Tautology, Contradiction, Contingency – Using Truth Table

- Either it will rain tomorrow, or it will not rain tomorrow
- Let  $p$  denote “it will rain tomorrow”
- $p \vee \neg p$

$p$	$\neg p$	$p \vee \neg p$
T	F	T
F	T	T

# Tautology, Contradiction, Contingency – Using Truth Table

- It is raining now, and it is not raining now
- Let  $p$  denote “it is raining now”
- $p \wedge \neg p$

$p$	$\neg p$	$p \wedge \neg p$
T	F	F
F	T	F

# Tautology, Contradiction, Contingency – Using Truth Table

- If we go to store, we will buy some apples
- Let  $p$  denote “we go to store”, and  $q$  denote “we buy apples”
- $p \rightarrow q$

$p$	$q$	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

# Tautology, Contradiction, Contingency – Exercise

- Construct truth tables to test the following sentences for tautology, contradiction and contingency

1.  $\neg (p \wedge \neg p)$

2.  $p \vee (q \rightarrow p)$

3.  $\neg p \rightarrow p$

# Argument

- An argument in propositional logic is a sequence of propositions
- All but the final proposition in the argument are called ***premises / hypotheses***
- The final proposition is called the ***conclusion***

# Argument – Example

- Consider the following argument (sequence of propositions)
  - *“If you have a current password, then you can log onto the network.”*
  - *“You have a current password.”*
  - *Therefore, “You can log onto the network.”*
- 
- To determine if it is a valid argument?

# Validity of an Argument

- A *valid argument* is an argument in which the conclusion must be true whenever the premises are true
- An argument is **valid** if and only if it is impossible for all the premises to be true and the conclusion to be false

# Argument Form

- Using  $p$  to represent “You have a current password” and  $q$  to represent “You can log onto the network.”

- Then, the argument has the form

- $$\begin{array}{c} p \rightarrow q \\ \\ p \\ \hline \therefore q \end{array}$$

- Symbol  $\therefore$  denotes “therefore.”



# Validity of an Argument

- In this example, when both  $p \rightarrow q$  and  $p$  are true,  $q$  must also be true
- This form of argument is **valid** because whenever all its premises are true, the conclusion must also be true

# Validity of Argument

P	Q	$P \rightarrow Q$	$(P \rightarrow Q) \wedge P$
T	T	T	T
T	F	F	F
F	T	T	F
F	F	T	F

# Validity of an Argument

- An argument is **valid** if the truth of all its premises implies that the conclusion is true.
- For  $p$  and  $q$  being propositional variables, the statement  $((p \rightarrow q) \wedge p) \rightarrow q$  is a *tautology*
- In the case of a valid argument we say the conclusion follows from the premises

# Validity of Argument

P	Q	$P \rightarrow Q$	$(P \rightarrow Q) \wedge P$	$((P \rightarrow Q) \wedge P) \rightarrow Q$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

# Invalid Argument – Example

- If it is raining, then the streets are wet. The streets are wet. Therefore, it is raining.”

# Another Example

- Consider the following argument (sequence of propositions)
  - *“If it is raining, then the streets are wet”*
  - *“The streets are wet”*
  - *Therefore, “It is raining”*
- Determine if it is a valid argument?

# Argument Form

- Using  $p$  to represent “It is raining” and  $q$  to represent “Streets are wet”
- Then, the argument has the form

- $$\begin{array}{c} p \rightarrow q \\ q \\ \hline \therefore p \end{array}$$

# Validity of an Argument

P	Q	$P \rightarrow Q$	$(P \rightarrow Q) \wedge Q$
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	F



# Validity of Argument – Not a Tautology

P	Q	$P \rightarrow Q$	$(P \rightarrow Q) \wedge Q$	$((P \rightarrow Q) \wedge Q) \rightarrow P$
T	T	T	T	T
T	F	F	F	T
F	T	T	T	F
F	F	T	F	T

# Validity of an Argument – Important Note!

- Important to distinguish between the notions of the truth and the validity
- While individual statements / propositions may be either true or false, arguments cannot.
- Similarly, arguments may be described as valid or invalid, but statements / propositions cannot

# Validity of an Argument – Important Note!

- It is important to understand that the premises of an argument do not have *actually to be true* in order for the argument to be valid
- It is possible for valid arguments to contain either true or false premises / hypotheses
- An argument is valid if the premises and conclusion are related to each other in the right way so that if the premises *were* true, then the conclusion would have to be true as well

# Valid Argument with false premises

- All numbers are positive
- All positive numbers are larger than 5
- Therefore, all numbers are larger than 5

# Valid Argument with False Premises

- Consider the following argument
  - *“If I teach Discrete Mathematics, I am a Superman”*
  - *“I teach Discrete Mathematics”*
  - *Therefore, “I am a Superman”*
- A Valid Argument with a False Premise

# Sound Argument

- An argument is *sound* if and only if it is both ***valid***, and all of its premises are ***actually true***.
- Otherwise, a deductive argument is *unsound*.

# Validity of an Argument

- The key to showing that an argument in propositional logic is valid is to show that its *argument form* is valid
- Need to know the techniques to show that argument forms are valid

# Rules of Inference

## Lecture 2

---



# Rules of Inference for Propositional Logic

- A truth table can be used to show that an argument form is valid by showing that whenever the premises are true, the conclusion must also be true.
- However, it's a tedious approach.
- If an argument form involves 10 different propositional variables, to use a truth table to show this argument form is valid requires  $2^{10} = 1024$  different rows.

# Rules of Inference

- **Rules of inference** – simple argument forms
- These rules of inference can be used as building blocks to construct more complicated valid argument forms

# Rules of Inference

<i>Rule of Inference</i>	<i>Tautology</i>	<i>Name</i>
$\begin{array}{l} p \\ p \rightarrow q \\ \hline \therefore q \end{array}$	$(p \wedge (p \rightarrow q)) \rightarrow q$	Modus ponens
$\begin{array}{l} \neg q \\ p \rightarrow q \\ \hline \therefore \neg p \end{array}$	$(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$	Modus tollens
$\begin{array}{l} p \rightarrow q \\ q \rightarrow r \\ \hline \therefore p \rightarrow r \end{array}$	$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$	Hypothetical syllogism
$\begin{array}{l} p \vee q \\ \neg p \\ \hline \therefore q \end{array}$	$((p \vee q) \wedge \neg p) \rightarrow q$	Disjunctive syllogism

# Rules of Inference (contd.)

$\frac{p}{\therefore p \vee q}$	$p \rightarrow (p \vee q)$	Addition
$\frac{p \wedge q}{\therefore p}$	$(p \wedge q) \rightarrow p$	Simplification
$\frac{p}{q} \therefore p \wedge q$	$((p) \wedge (q)) \rightarrow (p \wedge q)$	Conjunction
$\frac{p \vee q}{\neg p \vee r} \therefore q \vee r$	$((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$	Resolution

# 1. Modus Ponens

$$\frac{p \rightarrow q \quad p}{\therefore q}$$

**Corresponding Tautology:**

$$(p \wedge (p \rightarrow q)) \rightarrow q$$

**Example:**

Let  $p$  be “It is snowing.”

Let  $q$  be “I will study discrete math.”

“If it is snowing, then I will study discrete math.”

“It is snowing.”

“Therefore, I will study discrete math.”

## 2. Modus Tollens

$$\frac{p \rightarrow q \quad \neg q}{\therefore \neg p}$$

**Corresponding Tautology:**

$$(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$$

**Example:**

Let  $p$  be “it is snowing.”

Let  $q$  be “I will study discrete math.”

“If it is snowing, then I will study discrete math.”

“I will not study discrete math.”

“Therefore, it is not snowing.”

### 3. Hypothetical Syllogism

$$\frac{p \rightarrow q \quad q \rightarrow r}{\therefore p \rightarrow r}$$

**Corresponding Tautology:**

$$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$$

**Example:**

Let  $p$  be “it snows.”

Let  $q$  be “I will study discrete math.”

Let  $r$  be “I will get an A.”

“If it snows, then I will study discrete math.”

“If I study discrete math, I will get an A.”

“Therefore , If it snows, I will get an A.”

## 4. Disjunctive Syllogism

$$\frac{p \vee q \quad \neg p}{\therefore q}$$

**Corresponding Tautology:**

$$(\neg p \wedge (p \vee q)) \rightarrow q$$

**Example:**

Let  $p$  be “I will study discrete math.”

Let  $q$  be “I will study English literature.”

“I will study discrete math or I will study English literature.”

“I will not study discrete math.”

“Therefore , I will study English literature.”



## 5. Addition

$$\frac{p}{\therefore p \vee q}$$

**Corresponding Tautology:**

$$p \rightarrow (p \vee q)$$

**Example:**

Let  $p$  be “I will study discrete math.”

Let  $q$  be “I will visit Las Vegas.”

“I will study discrete math.”

“Therefore, I will study discrete math or I will visit Las Vegas.”

## 6. Simplification

$$\frac{p \wedge q}{\therefore q}$$

**Corresponding Tautology:**  
 $(p \wedge q) \rightarrow p$

**Example:**

Let  $p$  be “I will study discrete math.”

Let  $q$  be “I will study English literature.”

“I will study discrete math and English literature”

“Therefore, I will study discrete math.”

# 7. Conjunction

$$\frac{p \quad q}{\therefore p \wedge q}$$

**Corresponding Tautology:**

$$((p) \wedge (q)) \rightarrow (p \wedge q)$$

**Example:**

Let  $p$  be “I will study discrete math.”

Let  $q$  be “I will study English literature.”

“I will study discrete math.”

“I will study English literature.”

“Therefore, I will study discrete math and I will study English literature.”

## 8. Resolution

$$\frac{\neg p \vee r \quad p \vee q}{\therefore q \vee r}$$

**Corresponding Tautology:**

$$((\neg p \vee r) \wedge (p \vee q)) \rightarrow (q \vee r)$$

**Example:**

Let  $p$  be “I will study discrete math.”

Let  $r$  be “I will study English literature.”

Let  $q$  be “I will study databases.”

“I will not study discrete math or I will study English literature.”

“I will study discrete math or I will study databases.”

“Therefore, I will study databases or I will study English literature.”

# Using the Rules of Inference to Build Valid Arguments

- A valid argument is a sequence of statements. Each statement is either a premise or follows from previous statements by rules of inference.
- The last statement is called conclusion.
- A valid argument takes the following form:
  - S1
  - S2
  - .
  - .
  - S<sub>n</sub>
  - $\therefore C$

# Valid Arguments – Example I

- From the single proposition  $p \wedge (p \rightarrow q)$
- Show that  $q$  is a conclusion
- **Solution:**

Step	Reason
1. $p \wedge (p \rightarrow q)$	Premise
2. $p$	Simplification using (1)
3. $p \rightarrow q$	Simplification using (1)
4. $q$	Modus Ponens using (2) and (3)

# Valid Arguments – Example II

- With the following hypotheses:
- *“It is not sunny this afternoon and it is colder than yesterday.”*
- *“We will go swimming only if it is sunny.”*
- *“If we do not go swimming, then we will take a canoe trip.”*
- *“If we take a canoe trip, then we will be home by sunset.”*
- Using the inference rules, construct a valid argument for the conclusion:
- *“We will be home by sunset.”*

# Valid Arguments – Example II

- **Solution:**
- Choose propositional variables
- **p**: “It is sunny this afternoon.” **r**: “We will go swimming.” **t**: “We will be home by sunset.” **q**: “It is colder than yesterday.” **s**: “We will take a canoe trip.”
- Translation into propositional logic:

Hypotheses:  $\neg p \wedge q, r \rightarrow p, \neg r \rightarrow s, s \rightarrow t$

Conclusion:  $t$



# Valid Arguments – Example II

- Construct the Valid Argument

Step	Reason
1. $\neg p \wedge q$	Premise
2. $\neg p$	Simplification using (1)
3. $r \rightarrow p$	Premise
4. $\neg r$	Modus tollens using (2) and (3)
5. $\neg r \rightarrow s$	Premise
6. $s$	Modus ponens using (4) and (5)
7. $s \rightarrow t$	Premise
8. $t$	Modus ponens using (6) and (7)

# Exercises on Rules of Inference

Lecture 3

---

# What Rule of Inference is used in the Argument?

- Alice is a mathematics major.
- Therefore, Alice is either a mathematics major or a computer science major.

# What Rule of Inference is used in the Argument?

- Jerry is a mathematics major and a computer science major.
- Therefore, Jerry is a mathematics major.

# What Rule of Inference is used in the Argument?

- If it is rainy, then the pool will be closed.
- It is rainy.
- Therefore, the pool is closed.

# What Rule of Inference is used in the Argument?

- If it snows today, the university will close.
- The university is not closed today.
- Therefore, it did not snow today.

# What Rule of Inference is used in the Argument?

- If I go swimming, then I will stay in the sun too long.
- If I stay in the sun too long, then I will sunburn.
- Therefore, if I go swimming, then I will sunburn.

# Using Rules of Inferences

- Use rules of inference to show that the hypotheses:
- *“If it does not rain or if it is not foggy, then the sailing race will be held and the lifesaving demonstration will go on,”*
- *“If the sailing race is held, then the trophy will be awarded,”* and
- *“The trophy was not awarded”*
- 
- Imply the conclusion *“It rained.”*



# Using Rules of Inferences

- Use rules of inference to show that the hypotheses
- *“John works hard,”*
- *“If John works hard, then he is a dull boy,”* and
- *“If John is a dull boy, then he will not get the job”*
- Imply the conclusion *“John will not get the job.”*

# Using Rules of Inferences

- Show that the premises
- *“If you send me an e-mail message, then I will finish writing the program,”*
- *“If you do not send me an e-mail message, then I will go to sleep early,”* and
- *“If I go to sleep early, then I will wake up feeling refreshed”*
- Lead to the conclusion *“If I do not finish writing the program, then I will wake up feeling refreshed.”*

# Using Rules of Inference

- From the given premises, what relevant conclusion or conclusions can be drawn? Explain the rules of inference used to obtain each conclusion
- *“I am either clever or lucky.”*
- *“I am not lucky.”*
- *“If I am lucky, then I will win the lottery.”*

# Using Rules of Inference

- From the given premises, what relevant conclusion or conclusions can be drawn? Explain the rules of inference used to obtain each conclusion
- *“If I eat spicy foods, then I have strange dreams.”*
- *“I have strange dreams if there is thunder while I sleep.”*
- *“I did not have strange dreams.”*

# Discrete Structures

Spring 2024 – Week 5



# Algorithms

## Lecture 1

---

# Problems and Algorithms

- In many domains there are key general problems that ask for output with specific properties when given valid input.
- The first step is to precisely state the problem, using the appropriate structures to specify the input and the desired output.
- We then solve the general problem by specifying the steps of a procedure that takes a valid input and produces the desired output.
- This procedure is called an algorithm

# Algorithms



Abu Ja'far Mohammed Ibin Musa Al-Khowarizmi  
(780-850)

- *Definition:* An algorithm is a finite set of precise instructions for performing a computation or for solving a problem.
- *Example:* Describe an algorithm for finding the maximum value in a finite sequence of integers.



# Algorithms

- Solution: Perform the following steps:
  1. Set the temporary maximum equal to the first integer in the sequence.
  2. Compare the next integer in the sequence to the temporary maximum.
  3. If it is larger than the temporary maximum, set the temporary maximum equal to this integer.
  4. Repeat the previous step if there are more integers. If not, stop.
- When the algorithm terminates, the temporary maximum is the largest integer in the sequence.

# Specifying Algorithms

- Algorithms can be specified in different ways. Their steps can be described in English or in pseudocode.
- Pseudocode is an intermediate step between an English language description of the steps and a coding of these steps using a programming language.
- Programmers can use the description of an algorithm in pseudocode to construct a program in a particular language.
- Pseudocode helps us analyze the time required to solve a problem using an algorithm, independent of the actual programming language used to implement algorithm.

# Properties of Algorithms

There are several properties that algorithms generally share. They are useful to keep in mind when algorithms are described. These properties are:

- *Input*: An algorithm has input values from a specified set.
- *Output*: From the input values, the algorithm produces the output values from a specified set. The output values are the solution.
- *Correctness*: An algorithm should produce the correct output values for each set of input values.
- *Finiteness*: An algorithm should produce the output after a finite number of steps for any input.
- *Effectiveness*: It must be possible to perform each step of the algorithm correctly and in a finite amount of time.
- *Generality*: The algorithm should work for all problems of the desired form.

# Finding the Maximum Element in a Finite Sequence

- The algorithm in pseudocode:

```
procedure  $\text{max}(a_1, a_2, \dots, a_n: \text{integers})$   
   $\text{max} := a_1$   
  for  $i := 2$  to  $n$   
    if  $\text{max} < a_i$  then  $\text{max} := a_i$   
  return  $\text{max}$  { $\text{max}$  is the largest element}
```

- Does this algorithm have all the properties listed on the previous slide?

# Exercise

- Determine which characteristics of an algorithm the following procedures have and which they lack.
- **procedure** *double*(*n*: positive integer)
  - **while**  $n > 0$
  - $n := 2n$

# Some Example Algorithm Problems

- Classes of problems to be studied in this section.
  1. *Searching Problems*: finding the position of a particular element in a list.
  2. *Sorting problems*: putting the elements of a list into increasing order.

# Searching Problems

- The general searching problem is to locate an element  $x$  in the list of distinct elements  $a_1, a_2, \dots, a_n$ , or determine that it is not in the list.
- The solution to a searching problem is the location of the term in the list that equals  $x$  (that is,  $i$  is the solution if  $x = a_i$ ) or 0 if  $x$  is not in the list.

# 1. Linear Search Algorithm

- The linear search algorithm locates an item in a list by examining elements in the sequence one at a time, starting at the beginning.
- First compare  $x$  with  $a_1$ . If they are equal, return the position 1.
- If not, try  $a_2$ . If  $x = a_2$ , return the position 2.
- Keep going, and if no match is found when the entire list is scanned, return 0.



# 1. Linear Search Algorithm

```
procedure linear search( $x$ :integer,  
                         $a_1, a_2, \dots, a_n$ : distinct integers)  
 $i := 1$   
while ( $i \leq n$  and  $x \neq a_i$ )  
     $i := i + 1$   
if  $i \leq n$  then  $location := i$   
else  $location := 0$   
return  $location$  { $location$  is the subscript of the term that equals  $x$ , or is 0 if  $x$  is not  
found}
```

## 2. Binary Search Algorithm

- Assume the input is a list of items in increasing order.
- The algorithm begins by comparing the element to be found with the middle element.
  - If the middle element is lower, the search proceeds with the upper half of the list.
  - If it is not lower, the search proceeds with the lower half of the list (through the middle position).
- Repeat this process until we have a list of size 1.
  - If the element we are looking for is equal to the element in the list, the position is returned.
  - Otherwise, 0 is returned to indicate that the element was not found.

## 2. Binary Search Algorithm

```
procedure binary search( $x$ : integer,  $a_1, a_2, \dots, a_n$ : increasing integers)
   $i := 1$  { $i$  is the left endpoint of interval}
   $j := n$  { $j$  is right endpoint of interval}
  while  $i < j$ 
     $m := \lfloor (i + j) / 2 \rfloor$ 
    if  $x > a_m$  then  $i := m + 1$ 
    else  $j := m$ 
  if  $x = a_i$  then  $location := i$ 
  else  $location := 0$ 
  return  $location$  {location is the subscript  $i$  of the term  $a_i$  equal to  $x$ ,
    or 0 if  $x$  is not found}
```

# Binary Search Algorithm

**Example:** The steps taken by a binary search for 19 in the list:

1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

1. The list has 16 elements, so the midpoint is 8. The value in the 8<sup>th</sup> position is 10. Since  $19 > 10$ , further search is restricted to positions 9 through 16.

1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

2. The midpoint of the list (positions 9 through 16) is now the 12<sup>th</sup> position with a value of 16. Since  $19 > 16$ , further search is restricted to the 13<sup>th</sup> position and above.

1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

# Binary Search Algorithm

3. The midpoint of the current list is now the 14<sup>th</sup> position with a value of 19. Since  $19 \neq 19$ , further search is restricted to the portion from the 13<sup>th</sup> through the 14<sup>th</sup> positions.

1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

4. The midpoint of the current list is now the 13<sup>th</sup> position with a value of 18. Since  $19 > 18$ , search is restricted to the portion from the 14<sup>th</sup> position through the 14<sup>th</sup>.

1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

5. Now the list has a single element and the loop ends. Since  $19 = 19$ , the location 14 is returned.

# Sorting Algorithms

## Lecture 2

---

# Sorting

- To *sort* the elements of a list is to put them in increasing order (numerical order, alphabetic, and so on).
- Sorting is an important problem because:
- Lot of computing resources are spent to sorting different kinds of lists, especially applications involving large databases of information that need to be presented in a particular order (e.g., by customer, part number etc.).

# Bubble Sort

- *Bubble sort* makes multiple passes through a list. Every pair of elements that are found to be out of order are interchanged.

```
procedure bubblesort( $a_1, \dots, a_n$ : real numbers  
                    with  $n \geq 2$ )  
  for  $i := 1$  to  $n - 1$   
    for  $j := 1$  to  $n - i$   
      if  $a_j > a_{j+1}$  then interchange  $a_j$  and  $a_{j+1}$   
  { $a_1, \dots, a_n$  is now in increasing order}
```

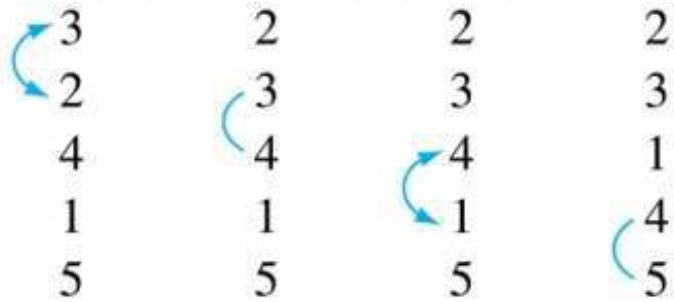


# Bubble Sort

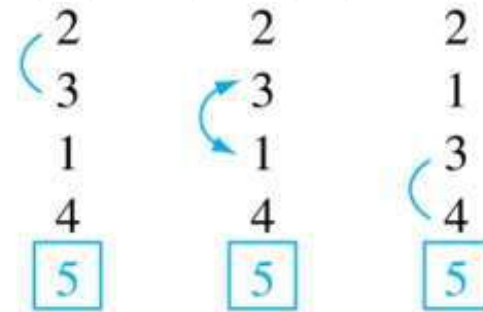
- Example: Show the steps of bubble sort with 3 2 4 1 5
- At the first pass the largest element has been put into the correct position
- At the end of the second pass, the 2nd largest element has been put into the correct position.
- In each subsequent pass, an additional element is put in the correct position.

# Bubble Sort

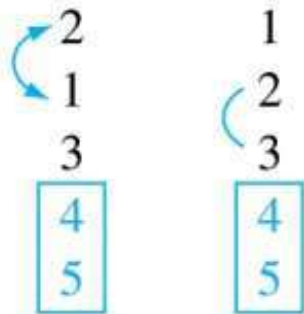
First pass



Second pass





Third pass



Fourth pass



 : an interchange

 : pair in correct order

numbers in color

guaranteed to be in correct order

# Bubble Sort – Illustration

6 5 3 1 8 7 2 4

[https://en.wikipedia.org/wiki/Bubble\\_sort#/media/File:Bubble-sort-example-300px.gif](https://en.wikipedia.org/wiki/Bubble_sort#/media/File:Bubble-sort-example-300px.gif)

---

# Insertion Sort

- Insertion sort begins with the 2nd element. It compares the 2nd element with the 1st and puts it before the first if it is not larger.
- Next the 3<sup>rd</sup> element is put into the correct position among the first 3 elements.
- In each subsequent pass, the  $n+1^{\text{st}}$  element is put into its correct position among the first  $n+1$  elements.
- Linear search is used to find the correct position.

# Insertion Sort

- **Example:** Show all the steps of insertion sort with the input:

- 3 2 4 1 5

i.    2 3 4 1 5   *(first two positions are interchanged)*

ii.   2 3 4 1 5   *(third element remains in its position)*

iii.  1 2 3 4 5   *(fourth is placed at beginning)*

iv.   1 2 3 4 5   *(fifth element remains in its position)*

# Insertion Sort – Illustration

6 5 3 1 8 7 2 4

[https://en.wikipedia.org/wiki/Insertion\\_sort#/media/File:Insertion-sort-example-300px.gif](https://en.wikipedia.org/wiki/Insertion_sort#/media/File:Insertion-sort-example-300px.gif)

# Insertion Sort

```
procedure insertion sort
  ( $a_1, \dots, a_n$ :
    real numbers with  $n \geq 2$ )
  for  $j := 2$  to  $n$ 
     $i := 1$ 
    while  $a_j > a_i$ 
       $i := i + 1$ 
     $m := a_j$ 
    for  $k := 0$  to  $j - i - 1$ 
       $a_{j-k} := a_{j-k-1}$ 
     $a_i := m$ 
  {Now  $a_1, \dots, a_n$  is in increasing order}
```

# Complexity of Algorithms

## Lecture 3

---



# The Complexity of Algorithms

- Given an algorithm, how efficient is this algorithm for solving a problem given input of a particular size? To answer this question, we ask:
  1. How much time does this algorithm use to solve a problem?
  2. How much computer memory does this algorithm use to solve a problem?

# Time and Space Complexity

- When we analyze the time the algorithm uses to solve the problem given input of a particular size, we are studying the *time complexity* of the algorithm.
- When we analyze the computer memory the algorithm uses to solve the problem given input of a particular size, we are studying the *space complexity* of the algorithm.

# Time Complexity

- The focus would be on time complexity only
- Time complexity is measured in terms of the number of operations an algorithm uses and big- $O$  and big- $\Theta$  notation is used to estimate the time complexity.
- This analysis can be used to see whether it is practical to use this algorithm to solve problems with input of a particular size and to compare the efficiency of different algorithms for solving the same problem

# Time Complexity

- To analyze the time complexity of algorithms, we determine the number of operations, such as comparisons and arithmetic operations (addition, multiplication, etc.)
- We can estimate the time a computer may actually use to solve a problem using the amount of time required to do basic operations.
- We will focus on the *worst-case time* complexity of an algorithm. This provides an upper bound on the number of operations an algorithm uses to solve a problem with input of a particular size.

# Time Complexity – Finding Maximum Element in a Finite Sequence

- **procedure**  $\text{max}$  ( $a_1, a_2, \dots, a_n$ : integers)
- $\text{max} := a_1$
- **for**  $i := 2$  to  $n$
- if  $\text{max} < a_i$  then  $\text{max} := a_i$
- return  $\text{max}$  { $\text{max}$  is the largest element}

# Time Complexity – Finding Maximum Element in a Finite Sequence

- **Solution:** Count the number of comparisons.
- The  $\max < a_i$  comparison is made  $n - 1$  times.
- Each time  $i$  is incremented, a test is made to see if  $i \leq n$ .
- One last comparison determines that  $i > n$ .
- Exactly  $2(n - 1) + 1 = 2n - 1$  comparisons are made.
- Hence, the time complexity of the algorithm is  $\Theta(n)$ .

# Time Complexity – Linear Search

- **procedure** *linear search*( $x$ :integer,  $a_1, a_2, \dots, a_n$ : distinct integers)
- $i := 1$
- **while** ( $i \leq n$  and  $x \neq a_i$ )
- $i := i + 1$
- **if**  $i \leq n$  **then**  $location := i$
- **else**  $location := 0$
- **return**  $location$  {  $location$  is the subscript of the term that equals  $x$ , or is 0 if  $x$  is not found }

# Time Complexity – Linear Search

- **Solution:** Count the number of comparisons.
  - At each step two comparisons are made;  $i \leq n$  and  $x \neq a_i$ .
  - To end the loop, one comparison  $i \leq n$  is made.
  - After the loop, one more  $i \leq n$  comparison is made.
- If  $x = a_i$ ,  $2i + 1$  comparisons are used. If  $x$  is not on the list,  $2n + 1$  comparisons are made and then an additional comparison is used to exit the loop. So, in the worst case  $2n + 2$  comparisons are made. Hence, the complexity is  $\Theta(n)$ .



# Time Complexity – Binary Search

- **procedure** binary search( $x$ : integer,  $a_1, a_2, \dots, a_n$ : increasing integers)
- $i := 1$  { $i$  is the left endpoint of interval}
- $j := n$  { $j$  is right endpoint of interval}
- **while**  $i < j$
- $m := \lfloor (i + j) / 2 \rfloor$
- **if**  $x > a_m$  **then**  $i := m + 1$
- **else**  $j := m$
- **if**  $x = a_i$  **then**  $location := i$
- **else**  $location := 0$
- **return**  $location$  {location is the subscript  $i$  of the term  $a_i$  equal to  $x$ , or 0 if  $x$  is not found}

# Time Complexity – Binary Search

- **Solution:** Assume (for simplicity)  $n = 2^k$  elements. Note that  $k = \log n$ .
- Two comparisons are made at each stage;  $i < j$ , and  $x > a_m$
- At the first iteration the size of the list is  $2k$  and after the first iteration it is  $2k-1$  then  $2k-2$  and so on until the size of the list is  $2^1 = 2$
- At the last step, a comparison tells us that the size of the list is the size is  $2^0 = 1$  and the element is compared with the single remaining element.
- Hence, at most  $2k + 2 = 2 \log n + 2$  comparisons are made.
- Therefore, the time complexity is  $\Theta(\log n)$ , better than linear search.

# Time Complexity – Bubble Sort

```
procedure bubblesort( $a_1, \dots, a_n$ : real numbers  
                    with  $n \geq 2$ )  
  for  $i := 1$  to  $n - 1$   
    for  $j := 1$  to  $n - i$   
      if  $a_j > a_{j+1}$  then interchange  $a_j$  and  $a_{j+1}$   
  { $a_1, \dots, a_n$  is now in increasing order}
```

# Time Complexity – Bubble Sort

- **Solution:** A sequence of  $n-1$  passes is made through the list. On each pass  $n - i$  comparisons are made.

- The worst-case complexity of bubble sort is  $\Theta(n^2)$  since  $(n-1) + (n-2) + \dots + 2 + 1 = \frac{n(n-1)}{2}$

$$\frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

# Time Complexity – Insertion Sort

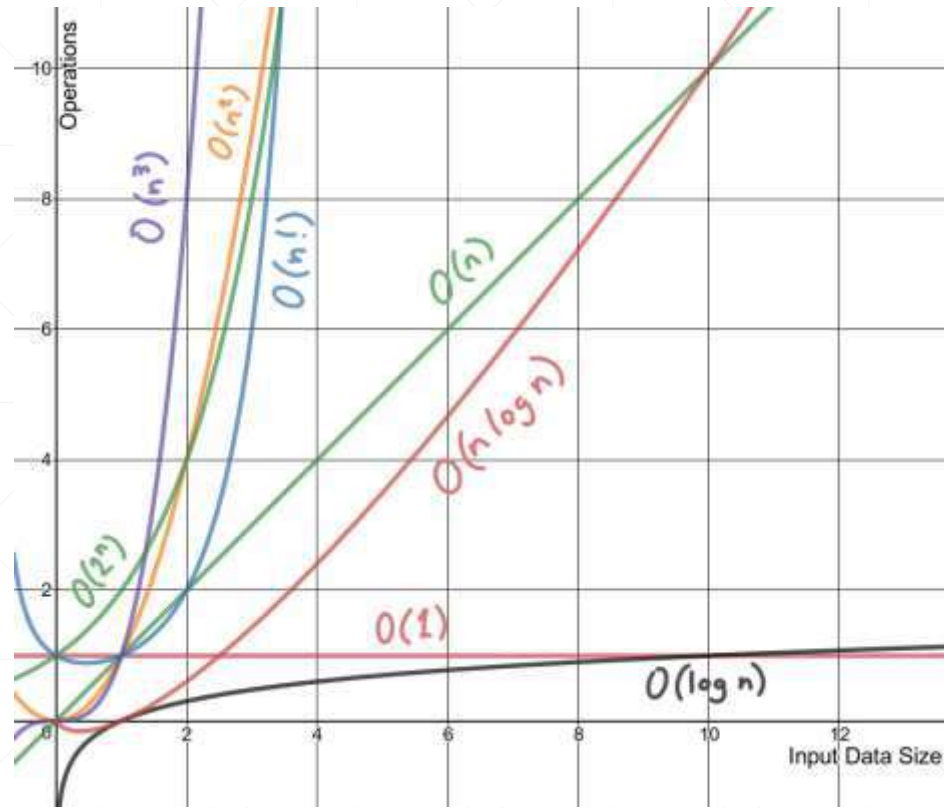
- **Solution:** The total number of comparisons are:

$$2 + 3 + \dots + n = \frac{n(n-1)}{2} - 1$$

- Therefore the complexity is  $\Theta(n^2)$

```
procedure insertion sort( $a_1, \dots, a_n$ :  
    real numbers with  $n \geq 2$ )  
  for  $j := 2$  to  $n$   
     $i := 1$   
    while  $a_j > a_i$   
       $i := i + 1$   
     $m := a_j$   
    for  $k := 0$  to  $j - i - 1$   
       $a_{j-k} := a_{j-k-1}$   
     $a_i := m$ 
```

# Comparison of Time Complexity



# Discrete Structures

Spring 2024 – Week 6



# Mathematical Induction

## Lecture 1

---



# Background

- Many mathematical statements assert that a property is true for all positive integers.
  1. For every positive integer  $n$ :  $n! \leq n^n$
  2.  $n^3 - n$  is divisible by 3
  3. The sum of the first  $n$  positive integers is  $n(n + 1)/2$ .

# Background

- Mathematical Induction is used to prove such type of results
- Mathematical induction can be used to prove statements that assert that  $P(n)$  is true for all positive integers  $n$ , where  $P(n)$  is a propositional function.

# Background

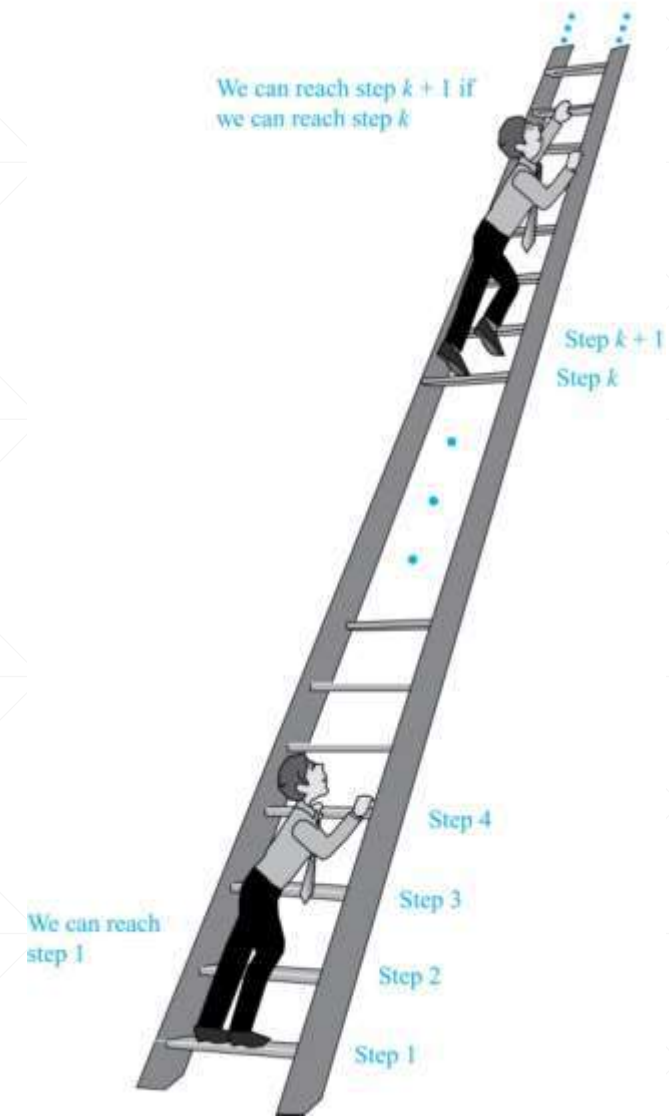
- Proofs using mathematical induction have two parts. First, they show that the statement holds for the positive integer 1. Second, they show that if the statement holds for a positive integer then it must also hold for the next larger integer.
- Mathematical induction is based on the rule of inference that tells us that if  $P(1)$  and  $\forall k(P(k) \rightarrow P(k + 1))$  are true for the domain of positive integers, then  $\forall n P(n)$  is true.

# Climbing an Infinite Ladder

- Suppose we have an infinite ladder:
  1. We can reach the first rung of the ladder.
  2. If we can reach a particular rung of the ladder, then we can reach the next rung.
- From (1), we can reach the first rung. Then by applying (2), we can reach the second rung. Applying (2) again, the third rung. And so on. We can apply (2) any number of times to reach any particular rung, no matter how high up.

# Climbing an Infinite Ladder

- This example motivates proof by mathematical induction.



# Principle of Mathematical Induction

- To prove that  $P(n)$  is true for all positive integers  $n$ , we complete these steps:
- Basis Step: Show that  $P(1)$  is true.
- Inductive Step: Show that  $P(k) \rightarrow P(k + 1)$  is true for all positive integers  $k$ .
- To complete the inductive step, assuming the inductive hypothesis that  $P(k)$  holds for an arbitrary integer  $k$ , show that  $P(k + 1)$  must be true.

# Principle of Mathematical Induction

- Climbing an Infinite Ladder Example:
- Basis Step: By (1), we can reach rung 1.
- Inductive Step: Assume the inductive hypothesis that we can reach rung  $k$ . Then by (2), we can reach rung  $k + 1$ .
- Hence,  $P(k) \rightarrow P(k + 1)$  is true for all positive integers  $k$ . We can reach every rung on the ladder.

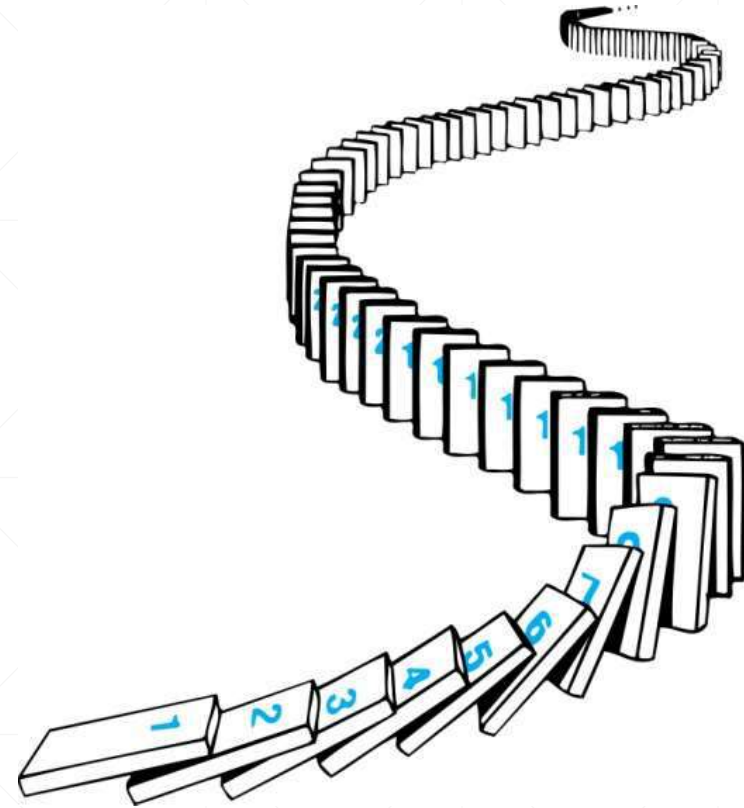
# Important Points About Using Mathematical Induction

- Mathematical induction can be expressed as the rule of inference
- $$(P(1) \wedge \forall k (P(k) \rightarrow P(k + 1))) \rightarrow \forall n P(n),$$
- where the domain is the set of positive integers.
- In a proof by mathematical induction, we don't assume that  $P(k)$  is true for all positive integers! We show that if we assume that  $P(k)$  is true, then  $P(k + 1)$  must also be true.
- Proofs by mathematical induction do not always start at the integer 1. In such a case, the basis step begins at a starting point  $b$  where  $b$  is an integer



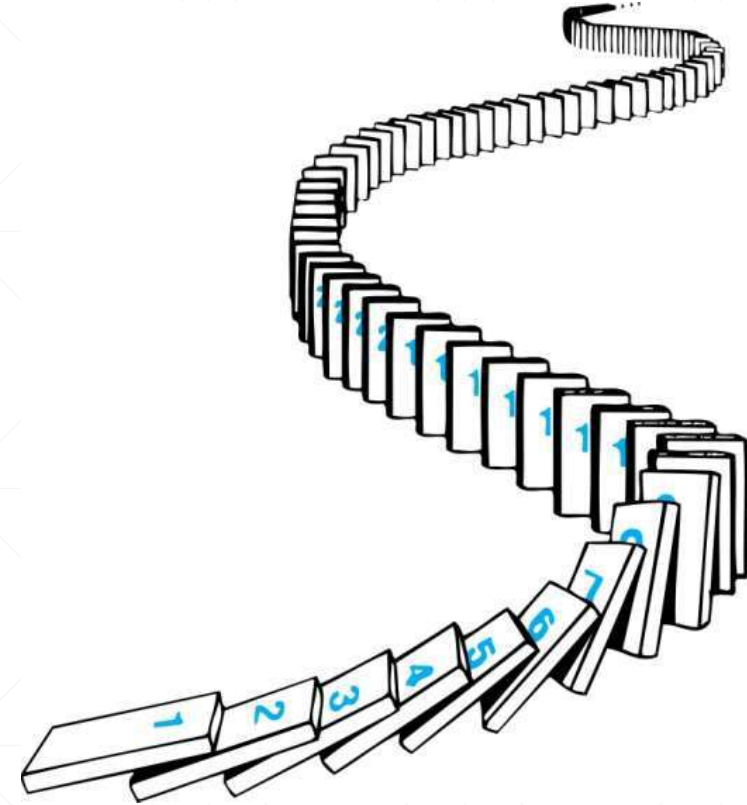
# How Mathematical Induction Works – Another Example

- Consider an infinite sequence of dominoes, labeled  $1, 2, 3, \dots$ , where each domino is standing
- Let  $P(n)$  be the proposition that the  $n$ th domino is knocked over
- We know that the first domino is knocked down, i.e.,  $P(1)$  is true .



# How Mathematical Induction Works – Another Example

- We also know that if whenever the  $k^{\text{th}}$  domino is knocked over, it knocks over the  $(k + 1)^{\text{st}}$  domino, i.e,  $P(k) \rightarrow P(k + 1)$  is true for all positive integers  $k$ .
- Hence, all dominos are knocked over.
- $P(n)$  is true for all positive integers  $n$ .



# Using Mathematical Induction

- Use mathematical induction to show that
- $1 + 2 + 2^2 + \dots + 2^n = 2^{n+1} - 1$ , for all nonnegative integers  $n$ .
- *Solution:* Let  $P(n)$  be the proposition that  $1 + 2 + 2^2 + \dots + 2^n = 2^{n+1} - 1$  for the integer  $n$
- *Basis Step:*  $P(0)$  is true because  $2^0 = 2^{0+1} - 1$
- *Inductive Hypothesis:* For the inductive hypothesis, we assume that  $P(k)$  is true for an arbitrary nonnegative integer  $k$ . That is, we assume that  $1 + 2 + 2^2 + \dots + 2^k = 2^{k+1} - 1$

# Using Mathematical Induction

- To carry out the inductive step using this assumption, we must show that when we assume that  $P(k)$  is true, then  $P(k + 1)$  is also true.
- That is, we must show that  $1 + 2 + 2^2 + \dots + 2^k + 2^{k+1} = 2^{k+2} - 1$ , assuming the inductive hypothesis  $P(k)$ .
- $1 + 2 + 2^2 + \dots + 2^k + 2^{k+1} = (1 + 2 + 2^2 + \dots + 2^k) + 2^{k+1}$
- $= (2^{k+1} - 1) + 2^{k+1}$  (inductive hypothesis)
- $= 2 \cdot 2^{k+1} - 1$
- $= 2^{k+2} - 1$

# Proving a Summation Formula by Mathematical Induction

- Example: Show that:  $\sum_{i=1}^n = \frac{n(n+1)}{2}$
- Basis Step:  $P(1)$  is true since  $1(1+1)/2 = 1$ .
- Inductive Step: Assume true for  $P(k)$ .
- The inductive hypothesis is  $\sum_{i=1}^k = \frac{k(k+1)}{2}$
- Under this assumption, 
$$\begin{aligned} 1 + 2 + \dots + k + (k+1) &= \frac{k(k+1)}{2} + (k+1) \\ &= \frac{k(k+1) + 2(k+1)}{2} \\ &= \frac{(k+1)(k+2)}{2} \end{aligned}$$

# Recursively Defined Functions

## Lecture 2

---

# Recursively Defined Functions

- Definition: A recursive or inductive definition of a function consists of two steps.
- Basis Step: Specify the value of the function at zero.
- Recursive Step: Give a rule for finding its value at an integer from its values at smaller integers.
- A function  $f(n)$  is the same as a sequence  $a_0, a_1, \dots$ , where  $a_i$ , where  $f(i) = a_i$

# Recursively Defined Functions

- Example: Suppose  $f$  is defined by:
- $f(0) = 3$ ,
- $f(n + 1) = 2f(n) + 3$ , Find  $f(1)$ ,  $f(2)$ ,  $f(3)$ ,  $f(4)$
- Solution:  $f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9$
- $f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21$
- $f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45$
- $f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93$



# Recursively Defined Functions

- Example: Give a recursive definition of the factorial function  $n!$
- Solution:
- $f(0) = 1$
- $f(n + 1) = (n + 1) \cdot f(n)$

# Recursively Defined Functions

- Example: Give a recursive definition of:

$$\sum_{k=0}^n a_k.$$

- Solution:

- The first part of the definition is

$$\sum_{k=0}^0 a_k = a_0.$$

- The second part is

$$\sum_{k=0}^{n+1} a_k = \left( \sum_{k=0}^n a_k \right) + a_{n+1}.$$

# Fibonacci Numbers

- Example : The Fibonacci numbers are defined as follows:
- $f_0 = 0, f_1 = 1, f_n = f_{n-1} + f_{n-2}$
- Find  $f_2, f_3, f_4, f_5$
- $f_2 = f_1 + f_0 = 1 + 0 = 1$
- $f_3 = f_2 + f_1 = 1 + 1 = 2$
- $f_4 = f_3 + f_2 = 2 + 1 = 3$
- $f_5 = f_4 + f_3 = 3 + 2 = 5$

# Discrete Structures

Spring 2024 – Week 7



# Basics Of Counting

Lecture 1

---

---

# Motivation

- Suppose that a password on a computer system consists of six, seven, or eight characters. Each of these characters must be a digit or a letter of the alphabet. Each password must contain at least one digit. How many such passwords are there?

# Introduction

- Combinatorics is the study of arrangements of objects – an important part of discrete mathematics
- An important part of Combinatorics is Enumeration – the counting of objects with certain properties
- Counting problems arise throughout mathematics and computer science. Counting successful outcomes of experiments and all the possible outcomes of these experiments to determine probabilities of discrete events. Or counting number of operations used by an algorithm to study its time complexity.

# Basic Counting Principle – The Product Rule

- The Product Rule – One of the two basic counting principles
- Used to solve many different counting problems.
- The product rule applies when a procedure is made up of separate tasks.



# Basic Counting Principle – The Product Rule

- Suppose that a procedure can be broken down into a sequence of two tasks
- If there are  $n_1$  ways to do the first task and for each of these ways of doing the first task, there are  $n_2$  ways to do the second task, then there are  $n_1 n_2$  ways to do the procedure

# Basic Counting Principle – The Product Rule

- An extended version of the product rule is often useful.
- Suppose that a procedure is carried out by performing the tasks  $T_1, T_2, \dots, T_m$  in sequence.
- If each task  $T_i$ ,  $i = 1, 2, \dots, n$ , can be done in  $n_i$  ways, regardless of how the previous tasks were done, then there are  $n_1 \cdot n_2 \cdot \dots \cdot n_m$  ways to carry out the procedure

# Basic Counting Principle – The Product Rule

- **Ex 1.** A new company with just two employees, Sanchez and Patel, rents a floor of a building with 12 offices. How many ways are there to assign different offices to these two employees?
- **Solution:** The procedure of assigning offices to these two employees consists of assigning an office to Sanchez, which can be done in **12 ways**, then assigning an office to Patel different from the office assigned to Sanchez, which can be done in **11 ways**. By the product rule, there are  **$12 \cdot 11 = 132$**  ways to assign offices to these two employees.

# Basic Counting Principle – The Product Rule

- ***Ex 2.*** The chairs of an auditorium are to be labeled with an uppercase English letter followed by a positive integer not exceeding 100. What is the largest number of chairs that can be labeled differently?
- ***Solution:*** The procedure of labeling a chair consists of two tasks, namely, assigning to the seat one of the **26 uppercase English letters**, and then assigning to it one of the **100 possible integers**. The product rule shows that there are  **$26 \cdot 100 = 2600$**  different ways that a chair can be labeled. Therefore, the largest number of chairs that can be labeled differently is 2600.

# Basic Counting Principle – The Product Rule

- **Ex 3.** There are 32 computers in a data center in the cloud. Each of these computers has 24 ports. How many different computer ports are there in this data center?
- **Solution:** The procedure of choosing a port consists of two tasks, first picking a computer and then picking a port on this computer. Because there are **32 ways to choose the computer** and **24 ways to choose the port** no matter which computer has been selected, the product rule shows that there are  **$32 \cdot 24 = 768$  ports**

# Basic Counting Principle – The Product Rule

- ***Ex 4.*** How many different bit strings of length seven are there?
- ***Solution:*** Each of the seven bits can be chosen in two ways, because each bit is either 0 or 1. Therefore, the product rule shows there are a total of  $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 128$  different bit strings of length seven.

# Basic Counting Principle – The Product Rule

- **Ex 5.** How many different license plates can be made if each plate contains a sequence of three uppercase English letters followed by three digits?
- **Solution:** There are 26 choices for each of the three uppercase English letters and 10 choices for each of the three digits. Hence, by the product rule there are a total of  $26 \cdot 26 \cdot 26 \cdot 10 \cdot 10 \cdot 10 = 17,576,000$  possible license plates.

# Basic Counting Principle – The Product Rule

- The product rule is phrased in terms of sets in the following way:
- If  $A_1, A_2, \dots, A_m$  are finite sets, then the number of elements in the Cartesian product of these sets is the product of the number of elements in each set

$$|A_1 \times A_2 \times \cdots \times A_m| = |A_1| \cdot |A_2| \cdot \cdots \cdot |A_m|.$$



# Basic Counting Principle – The Sum Rule

- If a task can be done either in one of  $n_1$  ways or in one of  $n_2$  ways, where none of the set of  $n_1$  ways is the same as any of the set of  $n_2$  ways, then there are  $n_1 + n_2$  ways to do the task.

# Basic Counting Principle – The Sum Rule

- **Ex 6.** Suppose that either a member of the mathematics faculty or a student who is a mathematics major is chosen as a representative to a university committee. How many different choices are there for this representative if there are 37 members of the mathematics faculty and 83 mathematics majors and no one is both a faculty member and a student?
- **Solution:** There are 37 ways to choose a member of the mathematics faculty and there are 83 ways to choose a student who is a mathematics major. Choosing a member of the mathematics faculty is never the same as choosing a student who is a mathematics major because no one is both a faculty member and a student. By the sum rule it follows that there are  $37 + 83 = 120$  possible ways to pick this representative.

# Basic Counting Principle – The Sum Rule

- **Ex 7.** A student can choose a computer project from one of three lists. The three lists contain 23, 15, and 19 possible projects, respectively. No project is on more than one list. How many possible projects are there to choose from?
- **Solution:** The student can choose a project by selecting a project from the first list, the second list, or the third list. Because no project is on more than one list, by the sum rule there are  $23 + 15 + 19 = 57$  ways to choose a project.

# Basic Counting Principle – The Sum Rule

- The sum rule can be phrased in terms of sets as:
- If  $A_1, A_2, \dots, A_m$  are pairwise disjoint finite sets, then the number of elements in the union of these sets is the sum of the numbers of elements in the sets.

$$|A_1 \cup A_2 \cup \dots \cup A_m| = |A_1| + |A_2| + \dots + |A_m| \text{ when } A_i \cap A_j = \emptyset \text{ for all } i, j.$$

# Complex Counting Problems

Lecture 2

---

---

# Complex Counting Problems

- Many counting problems cannot be solved using just the sum rule or just the product rule.
- However, many complicated counting problems can be solved using both of these rules in combination

# Complex Counting Problems – Total Variables

- ***Ex 1.*** Suppose statement labels in a programming language can be either a single letter or a letter followed by a digit. Find the number of possible labels
- ***Solution:*** Combining sum and the product rule
- $26 + 26 \cdot 10 = 286$

# Complex Counting Problems – Total No. of Passwords

- **Ex 2.** Each user on a computer system has a password, which is six to eight characters long, where each character is an uppercase letter or a digit. Each password must contain at least one digit. How many possible passwords are there?
- **Solution:** Let  $P$  be the total number of passwords, and let  $P_6$ ,  $P_7$ , and  $P_8$  be the passwords of length 6, 7, and 8. By the sum rule  $P = P_6 + P_7 + P_8$ . To find each of  $P_6$ ,  $P_7$ , and  $P_8$ , we find the number of passwords of the specified length composed of letters and digits and subtract the number composed only of letters



# Complex Counting Problems – Total No. of Passwords

- $P_6 = 36^6 - 26^6 = 2,176,782,336 - 308,915,776 = 1,867,866,560.$
- $P_7 = 36^7 - 26^7 = 78,364,164,096 - 8,031,810,176 = 70,332,353,920.$
- $P_8 = 36^8 - 26^8 = 2,821,109,907,456 - 208,827,064,576 = 2,612,282,842,880.$
- Consequently,  $P = P_6 + P_7 + P_8 = 2,684,483,063,360.$

# Complex Counting Problems – Total Internet Addresses

- Version 4 of the Internet Protocol (IPv4) uses 32 bits.
  - Class A Addresses: used for the largest networks, a 0, followed by a 7-bit netid and a 24-bit hostid
  - Class B Addresses: used for the medium-sized networks, a 10, followed by a 14-bit netid and a 16-bit hostid
  - Class C Addresses: used for the smallest networks, a 110, followed by a 21-bit netid and a 8-bit hostid
-

# Complex Counting Problems – Total Internet Addresses

Bit Number	0	1	2	3	4	8	16	24	31
Class A	0	netid				hostid			
Class B	1	0	netid				hostid		
Class C	1	1	0	netid				hostid	
Class D	1	1	1	0	Multicast Address				
Class E	1	1	1	1	0	Address			

- Neither Class D nor Class E addresses are assigned as the address of a computer on the internet. Only Classes A, B, and C are available.
- 1111111 is not available as the netid of a Class A network.
- Hostids consisting of all 0s and all 1s are not available in any network.

# Complex Counting Problems – Total Internet Addresses

- Let  $x$  be the number of available addresses, and let  $x_A$ ,  $x_B$ , and  $x_C$  denote the number of addresses for the respective classes.
- To find,  $x_A$ :
- $2^7 - 1 = 127$  netids.
- $2^{24} - 2 = 16,777,214$  hostids.
- $x_A = 127 \cdot 16,777,214 = 2,130,706,178$ .

# Complex Counting Problems – Total Internet Addresses

- To find,  $x_B$ :
- $2^{14} = 16,384$  netids.
- $2^{16} - 2 = 16,534$  hostids.
- $x_B = 16,384 \cdot 16,534 = 1,073,709,056$ .

# Complex Counting Problems – Total Internet Addresses

- To find,  $x_C$ :
- $2^{21} = 2,097,152$  netids.
- $2^8 - 2 = 254$  hostids.
- $x_C = 2,097,152 \cdot 254 = 532,676,608$ .

# Complex Counting Problems – Total Internet Addresses

- Hence, the total number of available IPv4 addresses is
- $x = x_A + x_B + x_C$
- $= 2,130,706,178 + 1,073,709,056 + 532,676,608$
- $= 3,737,091,842.$

# Basic Counting Principle – Subtraction Rule

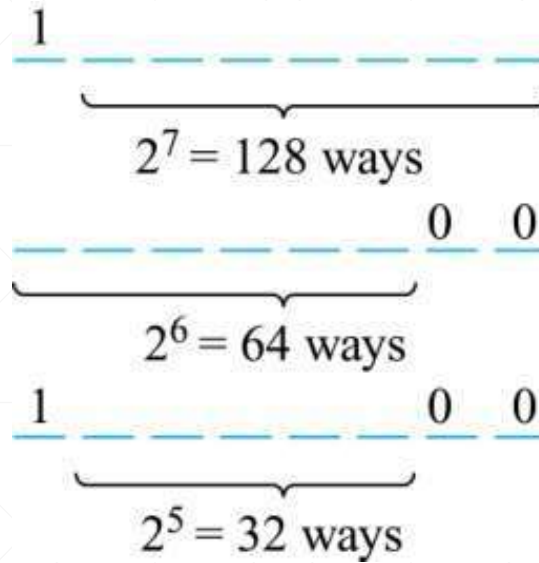
- If a task can be done either in one of  $n_1$  ways or in one of  $n_2$  ways, then the total number of ways to do the task is  $n_1 + n_2$  minus the number of ways to do the task that are common to the two different ways.
- Also known as, the *principle of inclusion-exclusion*

$$|A \cup B| = |A| + |B| - |A \cap B|$$



# Subtraction Rule – Counting Bit Strings

- **Ex 4.** How many bit strings of length eight either start with a 1 bit or end with the two bits 00?



# Subtraction Rule – Counting Bit Strings

- *Solution:*
- Number of bit strings of length eight that start with a 1 bit:  $2^7 = 128$
- Number of bit strings of length eight that end with bits 00:  $2^6 = 64$
- Number of bit strings of length eight that start with a 1 bit and end with bits 00 :  $2^5 = 32$
- Therefore, the number is
- $128 + 64 - 32 = 160$

# Basic Counting Principle – Division Rule

- The division rule comes in handy when it appears that a task can be done in  $n$  different ways, but it turns out that for each way of doing the task, there are  $d$  equivalent ways of doing it.
- Under these circumstances, we can conclude that there are  $n/d$  equivalent ways of doing the task.

# Basic Counting Principle – Division Rule

- **Example:** How many ways are there to seat four people around a circular table, where two seating are considered the same when each person has the same left and right neighbor?
- **Solution:** Number the seats around the table from 1 to 4 proceeding clockwise. There are four ways to select the person for seat 1, 3 for seat 2, 2, for seat 3, and one way for seat 4. Thus there are  $4! = 24$  ways to order the four people. But since two seating are the same when each person has the same left and right neighbor, for every choice for seat 1, we get the same seating. Therefore, by the division rule, there are  $24/4 = 6$  different seating arrangements.

# The Pigeonhole Principle

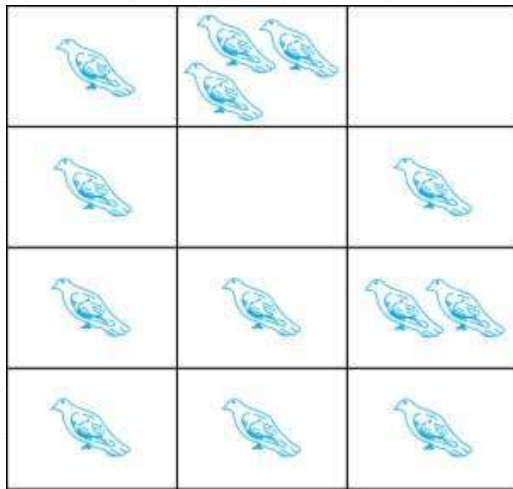
Lecture 3

---

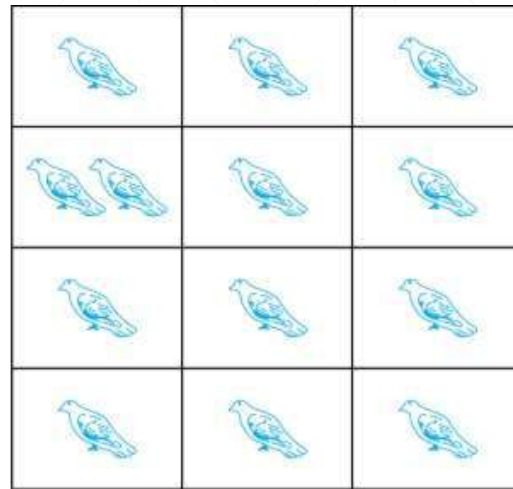
---

# The Pigeonhole Principle

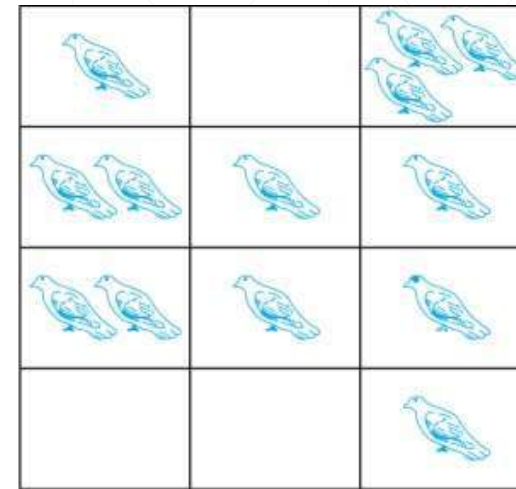
- If a flock of 20 pigeons roosts in a set of 19 pigeonholes, one of the pigeonholes must have more than 1 pigeon



(a)



(b)



(c)

# The Pigeonhole Principle

- If  $k$  is a positive integer and  $k + 1$  objects are placed into  $k$  boxes, then at least one box contains two or more objects.
- **Proof:** We use a proof by contraposition.
- Suppose none of the  $k$  boxes has more than one object.
- Then the total number of objects would be at most  $k$ .
- This contradicts the statement that we have  $k + 1$  objects

# The Pigeonhole Principle

- **Example:** Among any group of 367 people, there must be at least two with the same birthday, because there are only 366 possible birthdays



# Generalized Pigeonhole Principle

- If  $N$  objects are placed into  $k$  boxes, then there is at least one box containing at least  $\lceil N/k \rceil$  objects.
- **Example:** Among 100 people there are at least  $\lceil 100/12 \rceil = 9$  who were born in the same month

# Generalized Pigeonhole Principle

- **Example:** How many cards must be selected from a standard deck of 52 cards to guarantee that at least three cards of the same suit are chosen?
- **Solution:** We assume four boxes; one for each suit. Using the generalized pigeonhole principle, at least one box contains at least  $\lceil N/4 \rceil$  cards.
- At least three cards of one suit are selected if  $\lceil N/4 \rceil \geq 3$
- The smallest integer  $N$  such that  $\lceil N/4 \rceil \geq 3$  is
- $N = 2 \cdot 4 + 1 = 9$ .

# Practice

- What is the least number of 16 pigeons occupying 5 holes?
- **Solution**
- $N = 16, k = 5,$
- Therefore minimum number of pigeons in a hole  $\lceil 16/5 \rceil = 4$

# Discrete Structures

Spring 2024 – Week 9



# Permutations

## Lecture 1

---

# Introduction

- Many counting problems can be solved by finding the number of ways to arrange a specified number of distinct elements of a set of a particular size, where *the order of these elements matters*

# Example

- **In how many ways can we select three students from a group of five students to stand in line for a picture?**
- By the product rule, there are  $5 \cdot 4 \cdot 3 = 60$  ways to select three students from a group of five students to stand in line for a picture.
- **In how many ways can we arrange all five of these students in a line for a picture?**
- Consequently, there are  $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$  ways to arrange all five students in a line for a picture.

# Permutations

- **Definition:** A *permutation* of a set of distinct objects is an ordered arrangement of these objects.
- An ordered arrangement of  $r$  elements of a set is called an  *$r$ -permutation*.
- **Example:** Let  $S = \{1,2,3\}$
- The ordered arrangement 3,1,2 is a permutation of  $S$ .
- The ordered arrangement 3,2 is a 2-permutation of  $S$ .



# Permutations

- The number of  $r$ -permutations of a set with  $n$  elements is denoted by  $P(n,r)$ .
- The 2-permutations of  $S = \{1,2,3\}$  are
- 1,2; 1,3; 2,1; 2,3; 3,1; and 3,2.
- Hence,  $P(3,2) = 6$ .

# Formula for Number of Permutations

- **Theorem 1:** If  $n$  is a positive integer and  $r$  is an integer with  $1 \leq r \leq n$ , then there are
- $P(n, r) = n(n - 1)(n - 2) \cdots (n - (r - 1))$
- $r$ -permutations of a set with  $n$  distinct elements
- **Proof:** Use the product rule. The first element can be chosen in  $n$  ways. The second in  $n - 1$  ways, and so on until there are  $(n - (r - 1))$  ways to choose the last element.

# Formula for Number of Permutations

- Note that  $P(n,0) = 1$ , since there is only one way to order zero elements
- **Corollary 1:** If  $n$  and  $r$  are integers with  $1 \leq r \leq n$ , then

$$P(n, r) = \frac{n!}{(n-r)!}$$

# Solving Counting Problems by Counting Permutations

- **Example 1:** How many ways are there to select a first-prize winner, a second prize winner, and a third-prize winner from 100 different people who have entered a contest?
- **Solution:**
- $P(100,3) = 100 \cdot 99 \cdot 98 = 970,200$

# Solving Counting Problems by Counting Permutations

- **Example 2:** Suppose that there are eight runners in a race. The winner receives a gold medal, the second place finisher receives a silver medal, and the third-place finisher receives a bronze medal. How many different ways are there to award these medals, if all possible outcomes of the race can occur and there are no ties?
- **Solution:**
- The number of different ways to award the medals is the number of 3-permutations of a set with eight elements. Hence, there are  $P(8, 3) = 8 \cdot 7 \cdot 6 = 336$  possible ways to award the medals.

# Solving Counting Problems by Counting Permutations

- **Example 3:** Suppose that a saleswoman has to visit eight different cities. She must begin her trip in a specified city, but she can visit the other seven cities in any order she wishes. How many possible orders can the saleswoman use when visiting these cities?
- **Solution:** The first city is chosen, and the rest are ordered arbitrarily. Hence the orders are:
- $7! = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5040$
- If she wants to find the tour with the shortest path that visits all the cities, she must consider 5040 paths!

# Solving Counting Problems by Counting Permutations

- **Example 4:** How many permutations of the letters  $ABCDEFGH$  contain the string  $ABC$ ?
- **Solution:** We solve this problem by counting the permutations of six objects,  $ABC$ ,  $D$ ,  $E$ ,  $F$ ,  $G$ , and  $H$ .
- $6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720$

# Combinations

## Lecture 2

---



# Introduction

- Many other counting problems can be solved by finding the number of ways to select a particular number of elements from a set of a particular size, where *the order of the elements selected does not matter*.

# Example

- How many different committees of three students can be formed from a group of four students?
- To answer this question, we need only find the number of subsets with three elements from the set containing the four students. We see that there are four such subsets, one for each of the four students, because choosing three students is the same as choosing one of the four students to leave out of the group.

# Combinations

- **Definition:** An *r-combination* of elements of a set is an unordered selection of  $r$  elements from the set.
- Thus, an  $r$ -combination is simply a subset of the set with  $r$  elements.
- The number of  $r$ -combinations of a set with  $n$  distinct elements is denoted by  $C(n, r)$ .
- The notation  $\binom{n}{r}$  is also used and is called a *binomial coefficient*

# Combinations

- **Example:** Let  $S$  be the set  $\{a, b, c, d\}$
- Then  $\{a, c, d\}$  is a 3-combination from  $S$ . It is the same as  $\{d, c, a\}$  since the order listed does not matter.
- $C(4,2) = 6$  because the 2-combinations of  $\{a, b, c, d\}$  are the six subsets  $\{a, b\}$ ,  $\{a, c\}$ ,  $\{a, d\}$ ,  $\{b, c\}$ ,  $\{b, d\}$ , and  $\{c, d\}$ .

# Combinations

- **Theorem 2:** The number of  $r$ -combinations of a set with  $n$  elements, where  $n \geq r \geq 0$ , equals

$$C(n, r) = \frac{n!}{(n-r)!r!}.$$

- **Proof:** By the product rule  $P(n, r) = C(n, r) \cdot P(r, r)$ .
- Therefore,

$$C(n, r) = \frac{P(n, r)}{P(r, r)} = \frac{n!/(n-r)!}{r!/(r-r)!} = \frac{n!}{(n-r)!r!}.$$

# Combinations

- **Example:** How many poker hands of five cards can be dealt from a standard deck of 52 cards? Also, how many ways are there to select 47 cards from a deck of 52 cards?
- **Solution:** Since the order in which the cards are dealt does not matter, the number of five card hands is:

$$\begin{aligned} C(52, 5) &= \frac{52!}{5!47!} \\ &= \frac{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48}{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 26 \cdot 17 \cdot 10 \cdot 49 \cdot 12 = 2,598,960 \end{aligned}$$

- The different ways to select 47 cards from 52 is

$$C(52, 47) = \frac{52!}{47!5!} = C(52, 5) = 2,598,960.$$

# Combinations

- **Corollary 2:** Let  $n$  and  $r$  be nonnegative integers with  $r \leq n$ .
- Then  $C(n, r) = C(n, n - r)$ .
- **Proof:** From Theorem 2, it follows that

$$C(n, r) = \frac{n!}{(n-r)!r!}$$

and

Hence,  $C(n, r) = C(n, n - r)$

$$C(n, n - r) = \frac{n!}{(n-r)![n-(n-r)]!} = \frac{n!}{(n-r)!r!} .$$

# Combinations

- **Example:** How many ways are there to select five players from a 10-member tennis team to make a trip to a match at another school.
- **Solution:** By Theorem 2, the number of combinations is

$$C(10, 5) = \frac{10!}{5!5!} = 252.$$



# Combinations

- **Example:** A group of 30 people have been trained as astronauts to go on the first mission to Mars. How many ways are there to select a crew of six people to go on this mission?
- **Solution:** By Theorem 2, the number of possible crews is

$$C(30, 6) = \frac{30!}{6!24!} = \frac{30 \cdot 29 \cdot 28 \cdot 27 \cdot 26 \cdot 25}{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 593,775 .$$

# Combinations

- Suppose that there are 9 faculty members in the mathematics department and 11 in the computer science department. How many ways are there to select a committee to develop a discrete mathematics course at a school if the committee is to consist of three faculty members from the mathematics department and four from the computer science department?

# Combinations

- By the product rule, the answer is the product of the number of 3-combinations of a set with 9 elements and the number of 4-combinations of a set with 11 elements.
- By Theorem 2, the number of ways to select the committee is

$$C(9, 3) \cdot C(11, 4) = \frac{9!}{3!6!} \cdot \frac{11!}{4!7!} = 84 \cdot 330 = 27,720.$$

# Binomial Coefficients

## Lecture 3

---

# Powers of Binomial Expressions

- **Definition:** A *binomial* expression is the sum of two terms, such as  $x + y$ . (More generally, these terms can be products of constants and variables.)
- We can use counting principles to find the coefficients in the expansion of  $(x + y)^n$  where  $n$  is a positive integer.
- To illustrate this idea, we first look at the process of expanding  $(x + y)^3$ .
- $(x + y)(x + y)(x + y)$  expands into a sum of terms that are the product of a term from each of the three sums.
- Terms of the form  $x^3$ ,  $x^2y$ ,  $xy^2$ ,  $y^3$  arise. The question is what are the coefficients?

# Powers of Binomial Expressions

- To obtain  $x^3$ , an  $x$  must be chosen from each of the sums. There is only one way to do this. So, the coefficient of  $x^3$  is 1.
- To obtain  $x^2y$ , an  $x$  must be chosen from two of the sums and a  $y$  from the other. There are  $\binom{3}{2}$  ways to do this and so the coefficient of  $x^2y$  is 3
- To obtain  $xy^2$ , an  $x$  must be chosen from any of the sums and a  $y$  from the other two. There are  $\binom{3}{1}$  ways to do this and so the coefficient of  $xy^2$  is 3
- To obtain  $y^3$ , a  $y$  must be chosen from each of the sums. There is only one way to do this. So, the coefficient of  $y^3$  is 1.
- We have used a counting argument to show that  $(x + y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$
- Next we present the binomial theorem gives the coefficients of the terms in the expansion of  $(x + y)^n$

# Binomial Theorem

- **Binomial Theorem:** Let  $x$  and  $y$  be variables, and  $n$  a nonnegative integer. Then:

$$(x+y)^n = \sum_{j=0}^n \binom{n}{j} x^{n-j} y^j = \binom{n}{0} x^n + \binom{n}{1} x^{n-1} y + \cdots + \binom{n}{n-1} x y^{n-1} + \binom{n}{n} y^n.$$

- **Proof:** We use combinatorial reasoning . The terms in the expansion of  $(x+y)^n$  are of the form  $x^{n-j} y^j$  for  $j = 0, 1, 2, \dots, n$ .
- To form the term  $x^{n-j} y^j$ , it is necessary to choose  $n-j$   $x$ s from the  $n$  sums. Therefore, the coefficient of  $x^{n-j} y^j$  is  $\binom{n}{n-j}$  which equals  $\binom{n}{j}$ .



# Using the Binomial Theorem

- **Example:** What is the coefficient of  $x^{12}y^{13}$  in the expansion of  $(2x - 3y)^{25}$ ?
- **Solution:** We view the expression as  $(2x + (-3y))^{25}$ . By the binomial theorem

$$(2x + (-3y))^{25} = \sum_{j=0}^{25} \binom{25}{j} (2x)^{25-j} (-3y)^j.$$

Consequently, the coefficient of  $x^{12}y^{13}$  in the expansion is obtained when  $j = 13$ .

$$\binom{25}{13} 2^{12} (-3)^{13} = -\frac{25!}{13!12!} 2^{12} 3^{13}.$$



# Practice Problems

---

# Practice Problems

- In how many different orders can five runners finish a race if no ties are allowed?
- Permutations/Combination?
- $N = ?$
- $R = ?$

# Practice Problems

- How many ways are there for four men and five women to stand in a line so that
  - a. All men stand together?
  - b. All women stand together?

Permutation/Combination?

$N=?$ ,  $R=?$

Any special case?

# Practice Problems

- A club has 25 members.
  - a. How many ways are there to choose four members of the club to serve on an executive committee?
  - b. How many ways are there to choose a president, vice president, secretary, and treasurer of the club, where no person can hold more than one office?

Permutation/Combination?

$N=?$   $R=?$

# Practice Problems

- Suppose that a department contains 10 men and 15 women. How many ways are there to form a committee with six members if it must have the same number of men and women?

# Discrete Structures

Spring 2024 – Week 10



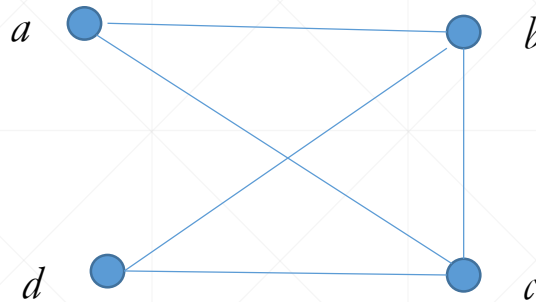
# Graphs

## Lecture 1

---

# Graphs

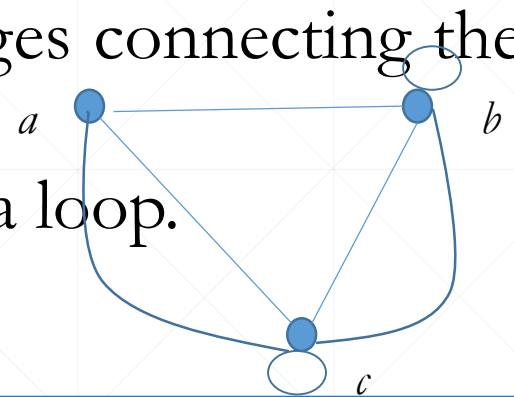
- **Definition:** A *graph*  $G = (V, E)$  consists of a nonempty set  $V$  of *vertices* (or *nodes*) and a set  $E$  of *edges*. Each edge has either one or two vertices associated with it, called its *endpoints*. An edge is said to connect its endpoints.
- Example: A graph with four vertices and five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





# 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.
- 

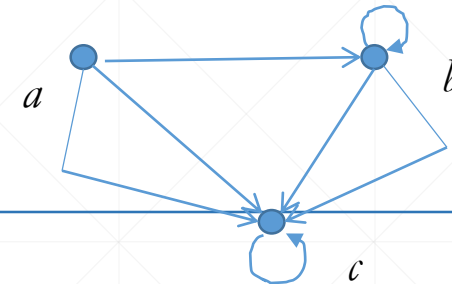
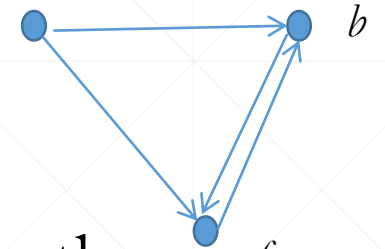


# Directed Graphs

- **Definition:** A *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.

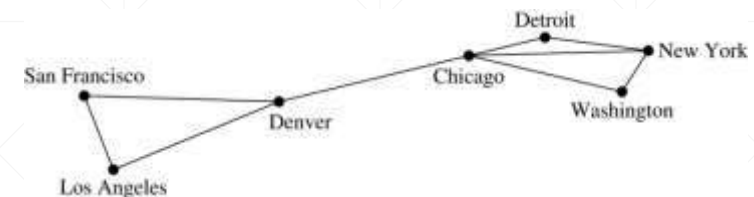
# Some Terminology (continued)

- A *simple directed graph* has no loops and no multiple edges.
- Example: 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: Directed multigraph the multiplicity of  $(a,b)$  is 1 and the multiplicity of  $(b,c)$  is 2.



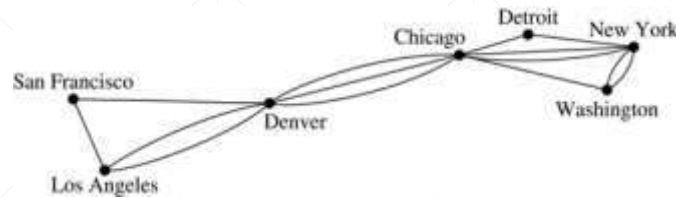
# Graph Models: Computer Networks

- When we build a graph model, we use the appropriate type of graph to capture the important features of the application.
- 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.

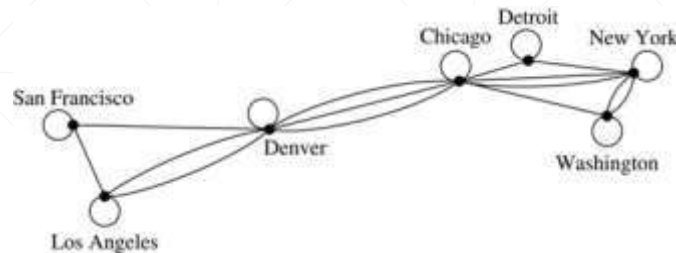


# Graph Models: Computer Networks

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

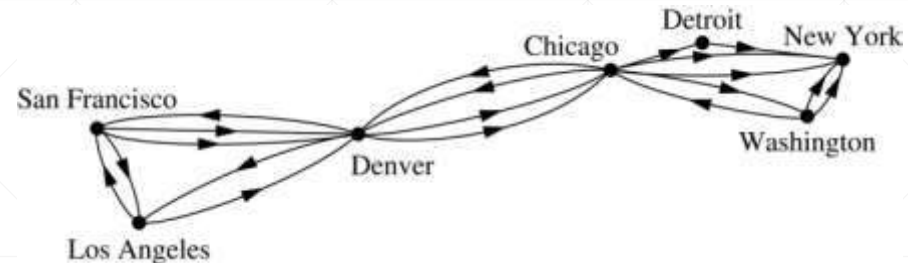


- To model a computer network with diagnostic links at data centers, we use a *pseudograph*, as loops are needed.



# 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 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

- Graph theory can be used in models of:
- Social networks
- Communications networks
- Information networks
- Software design
- Transportation networks
- Biological networks



# Graph Models: Social Networks

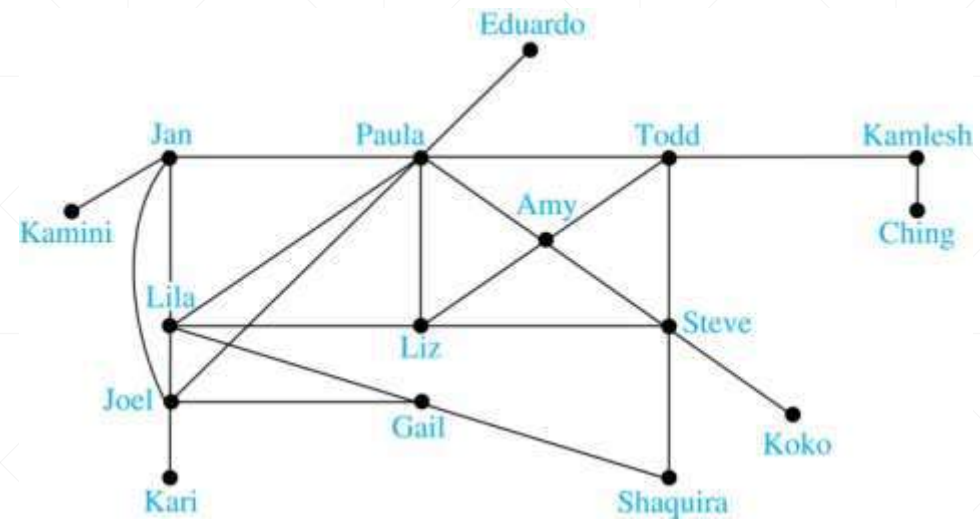
- Graphs can be used to model social structures based on different kinds of relationships between people or groups.
- In a social network, *vertices* represent individuals or organizations and *edges* represent relationships between them.

# Graph Models: Social Networks

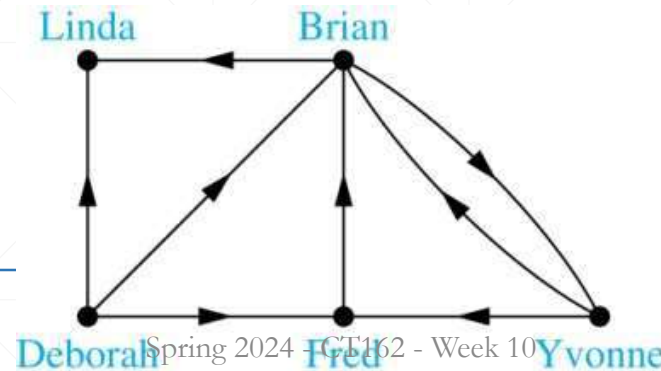
- Useful graph models of social networks include:
- **Friendship graphs** – undirected graphs where two people are connected if they are friends (in the real world, on Facebook, or in a particular virtual world, and so on.)
- **Influence graphs** – directed graphs where there is an edge from one person to another if the first person can influence the second person
- **Collaboration graphs** – undirected graphs where two people are connected if they collaborate in a specific way

# Graph Models: Social Networks (continued)

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



- **Example:** An influence graph



# 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,  $N(A) = \bigcup_{v \in A} N(v)$ .

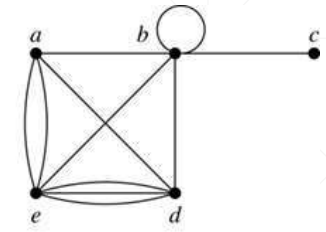
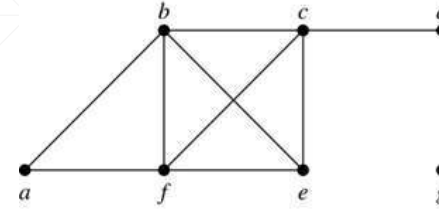
# Basic Terminology

- **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 graph  $G$ ?

- **Solution**



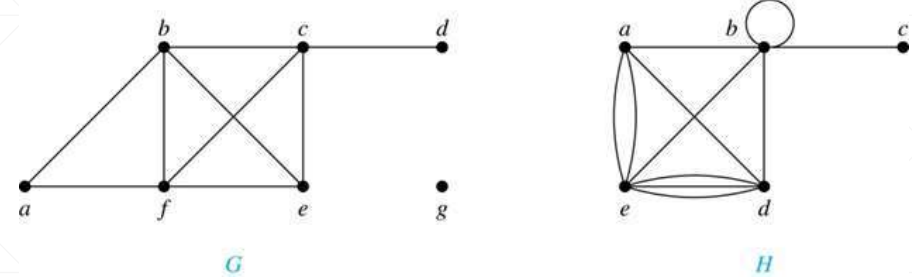
$G$ :  $\deg(a) = 2, \deg(b) = \deg(c) = \deg(f) = 4, \deg(d) = 1,$   
 $\deg(e) = 3, \deg(g) = 0.$

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

# Degrees and Neighborhoods of Vertices

- **Example:** What are the degrees and neighborhoods of the vertices in the graph  $H$ ?

- **Solution**



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

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

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

# Handshaking Theorem – Degrees of Vertices

- **Theorem 1 (*Handshaking Theorem*):** If  $G = (V, E)$  is an undirected graph with  $m$  edges, then  $2m = \sum_{v \in V} \deg(v)$
- ***Proof.***
- Each edge contributes twice to the degree count of all vertices. Hence, both the left-hand and right-hand sides of this equation equal twice the number of edges.
- *Think about the graph where vertices represent the people at a party and an edge connects two people who have shaken hands.*



# Handshaking Theorem

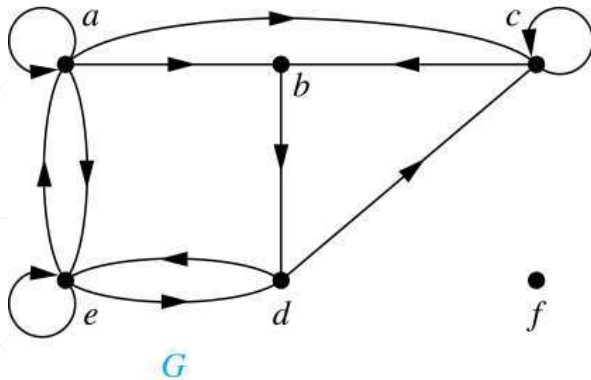
- **Example:** How many edges are there in a graph with 10 vertices of degree six?
- **Solution:**
- Because the sum of the degrees of the vertices is  $6 \cdot 10 = 60$ , the handshaking theorem tells us that  $2m = 60$ . So the number of edges  $m = 30$

# Directed Graphs

- **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

- **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



$$\begin{aligned} \deg^-(a) &= 2, \deg^-(b) = 2, \deg^-(c) = 3, \deg^-(d) = 2, \deg^-(e) = 3, \deg^-(f) = 0. \\ \deg^+(a) &= 4, \deg^+(b) = 1, \deg^+(c) = 2, \deg^+(d) = 2, \deg^+(e) = 3, \deg^+(f) = 0. \end{aligned}$$

# Directed Graphs

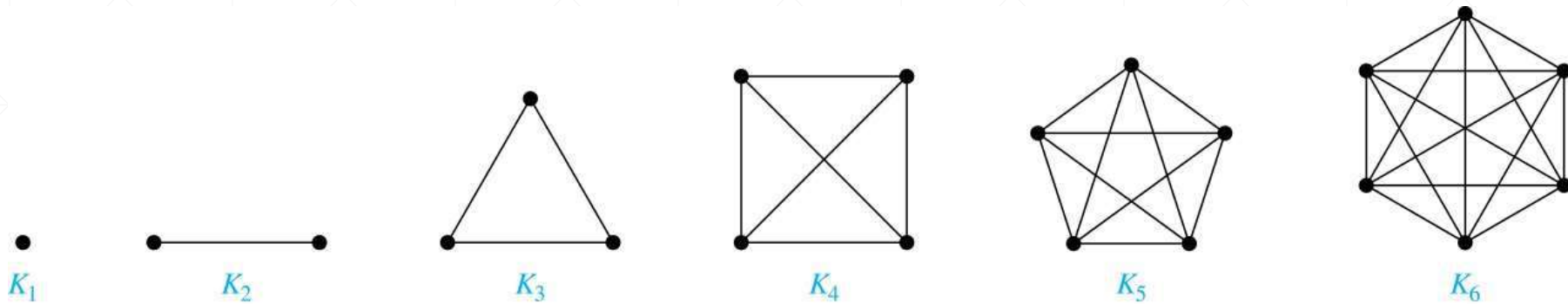
- **Theorem 3:** Let  $G = (V, E)$  be a graph with directed edges. Then:

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

- ***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.

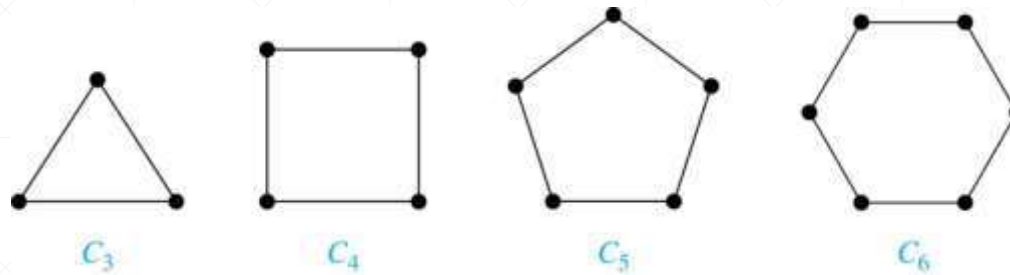
# Special Types of Simple Graphs: Complete Graphs

- 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.



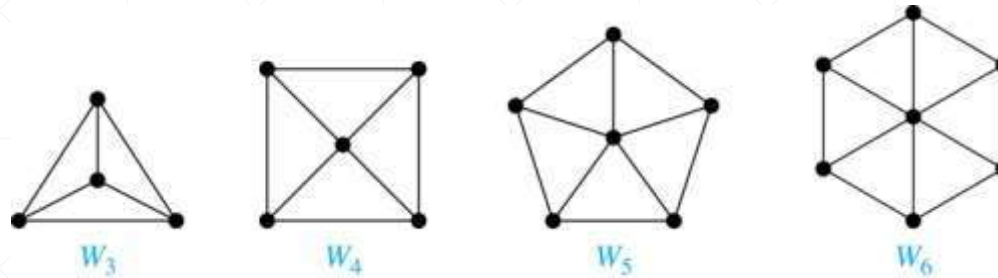
# Special Types of Simple Graphs: Cycles

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



# Special Types of Simple Graphs: Wheels

- 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.



# Bipartite Graphs

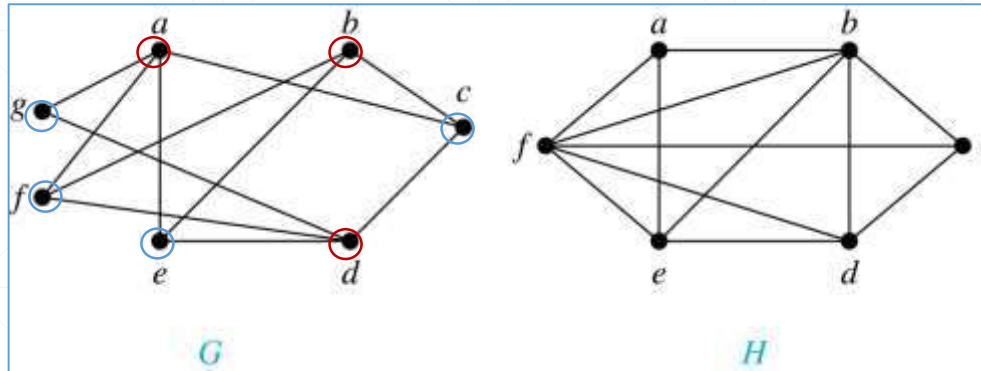
**Definition:** A simple graph  $G$  is bipartite if  $V$  can be partitioned into two disjoint subsets  $V_1$  and  $V_2$  such that every edge connects a vertex in  $V_1$  and a vertex in  $V_2$ . In other words, there are no edges which connect two vertices in  $V_1$  or in  $V_2$ .



# Bipartite Graphs

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.

$G$  is  
bipartite

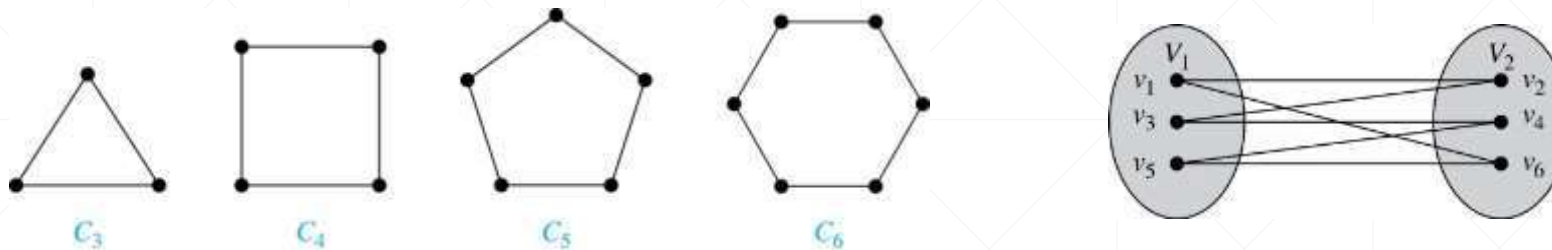


$H$  is not bipartite  
since if we color  $a$   
red, then the  
adjacent vertices  $f$   
and  $b$  must both be  
blue.

# Bipartite Graphs (*continued*)

**Example:** Show that  $C_6$  is bipartite.

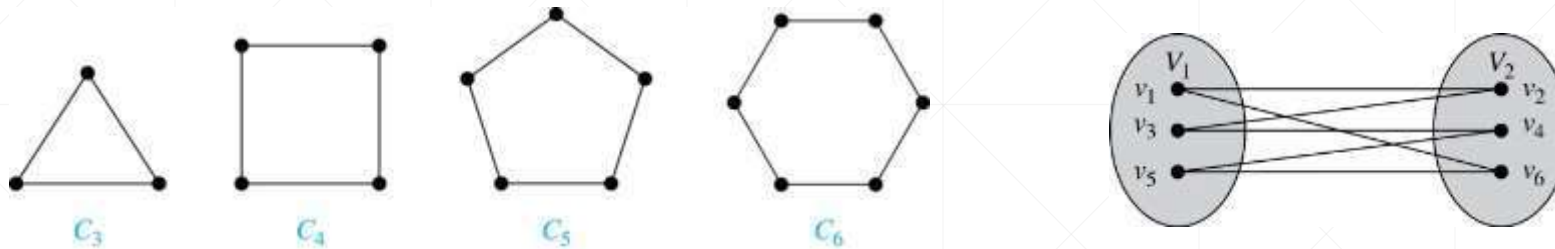
**Solution:** We can partition the vertex set into  $V_1 = \{v_1, v_3, v_5\}$  and  $V_2 = \{v_2, v_4, v_6\}$  so that every edge of  $C_6$  connects a vertex in  $V_1$  and  $V_2$ .



# Bipartite Graphs (*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.



# Representing Graphs

## Lecture 2

---

# Representing Graphs

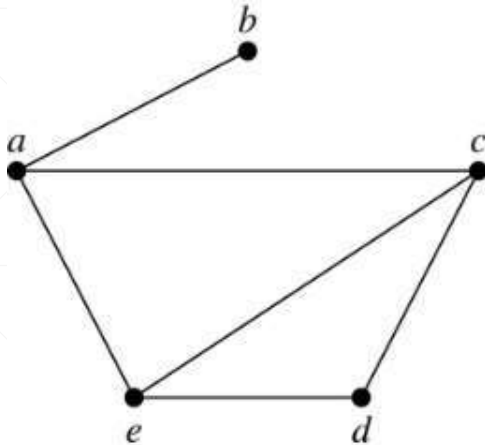
1. Adjacency List
2. Adjacency Matrices
3. Incidence Matrices

# 1. 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.

# 1. Adjacency Lists

Example

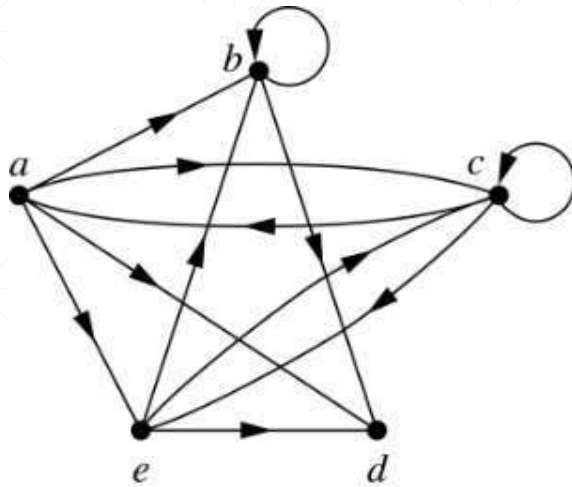


**TABLE 1** An Adjacency List  
for a Simple Graph.

<i>Vertex</i>	<i>Adjacent Vertices</i>
<i>a</i>	<i>b, c, e</i>
<i>b</i>	<i>a</i>
<i>c</i>	<i>a, d, e</i>
<i>d</i>	<i>c, e</i>
<i>e</i>	<i>a, c, d</i>

# 1. Adjacency Lists

Example



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

<i>Initial Vertex</i>	<i>Terminal Vertices</i>
<i>a</i>	<i>b, c, d, e</i>
<i>b</i>	<i>b, d</i>
<i>c</i>	<i>a, c, e</i>
<i>d</i>	
<i>e</i>	<i>b, c, d</i>



## 2. Adjacency Matrices

- **Definition:** Suppose that  $G = (V, E)$  is a simple graph where  $|V| = n$ . Arbitrarily list the vertices of  $G$  as  $v_1, v_2, \dots, v_n$ .

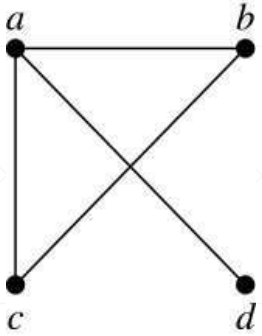
The *adjacency matrix*  $\mathbf{A}_G$  of  $G$ , with respect to the listing of vertices, is the  $n \times n$  zero-one matrix with 1 as its  $(i, j)$ th entry when  $v_i$  and  $v_j$  are adjacent, and 0 as its  $(i, j)$ th entry when they are not adjacent.

In other words, if the graphs adjacency matrix is  $\mathbf{A}_G = [a_{ij}]$ , then

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

## 2. Adjacency Matrices

- Example



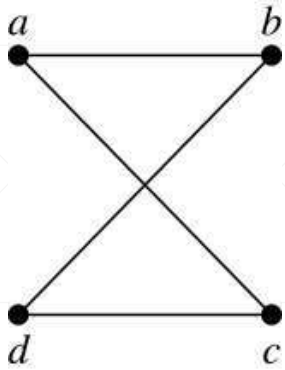
$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

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

- **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  $a_{ii}$  for  $i = 1, 2, 3, \dots, n$ , is 0

## 2. Adjacency Matrices

- Exercise



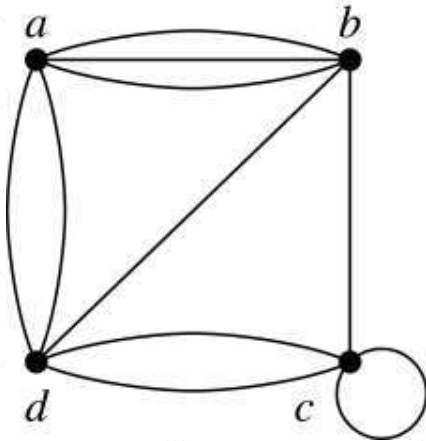
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

## 2. Adjacency Matrices

- Adjacency matrices can also be used to represent graphs with loops and multiple edges.
- A loop at the vertex  $v_i$  is represented by a **1** at the  $(i, j)$ th position of the matrix.
- When multiple edges connect the same pair of vertices  $v_i$  and  $v_j$ , (or if multiple loops are present at the same vertex), the  $(i, j)$ th entry equals the number of edges connecting the pair of vertices.

## 2. Adjacency Matrices

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

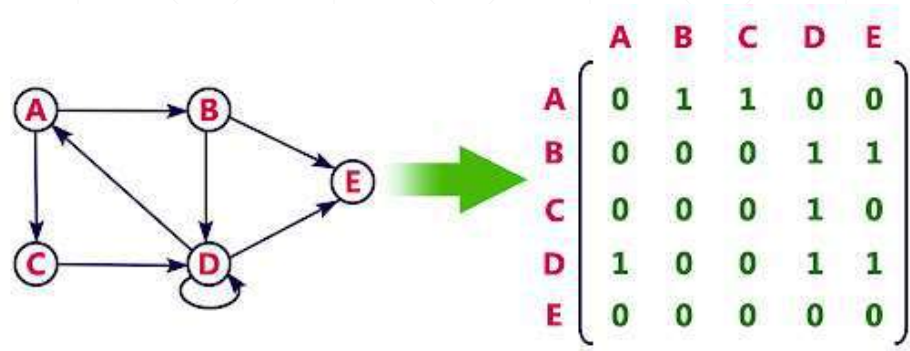


$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

## 2. Adjacency Matrices

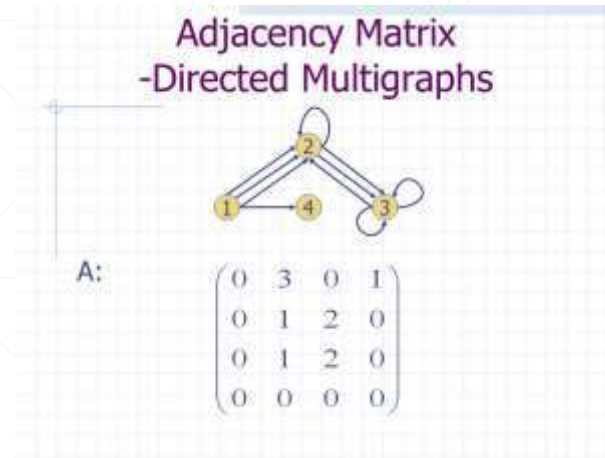
- Adjacency matrices can also be used to represent directed graphs. The matrix for a directed graph  $G = (V, E)$  has a 1 in its  $(i, j)$ th position if there is an edge from  $v_i$  to  $v_j$ , where  $v_1, v_2, \dots, v_n$  is a list of the vertices.
- In other words, if the graph's adjacency matrix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

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



## 2. Adjacency Matrices

- 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 multi-graphs, the value of  $a_{ij}$  is the number of edges connecting  $v_i$  to  $v_j$ .



## 2. Adjacency Matrices

- 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.



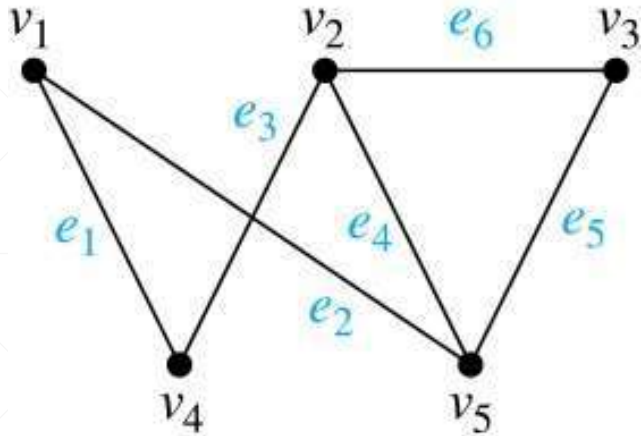
### 3. Incidence Matrices

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

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

### 3. Incidence Matrices

- **Example:** Simple Graph and Incidence Matrix

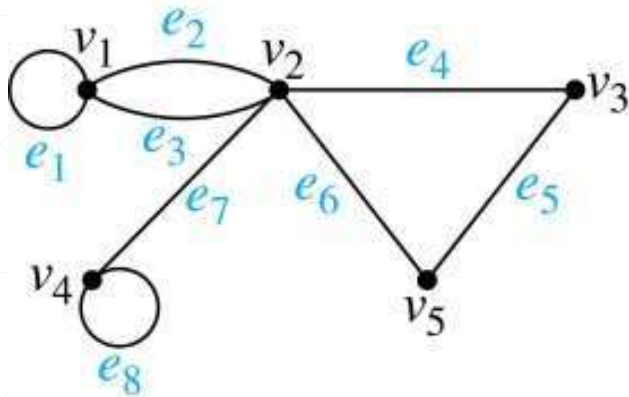


$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

*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  $e_6$ .*

### 3. Incidence Matrices

- **Exercise:** Pseudograph and Incidence Matrix



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

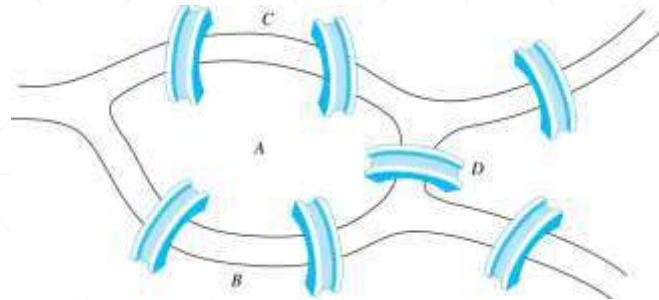
# Euler Path and Circuit

## Lecture 3

---

# Euler Paths and Circuits

- The town of Königsberg, Prussia (now Kalingrad, Russia) was divided into four sections by the branches of the Pregel river. In the 18th century seven bridges connected these regions.

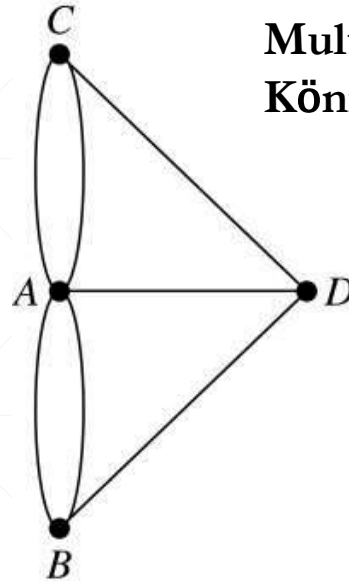
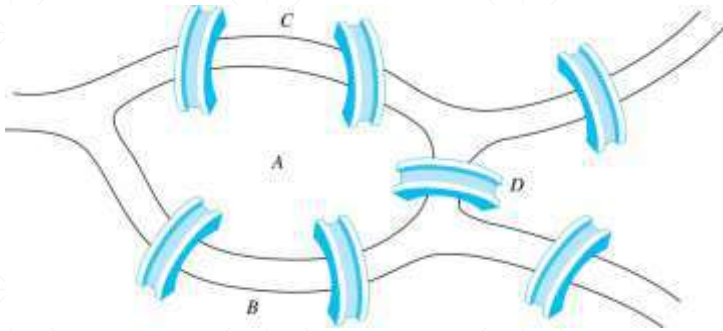


**The 7 Bridges of Königsberg**

- People wondered whether it was possible to follow a path that crosses each bridge exactly once and returns to the starting point.

# Euler Paths and Circuits

- 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.



**Multigraph Model of the Bridges of Königsberg**

# Path and Circuit

- A **path** is a sequence of vertices with the property that each vertex in the sequence is adjacent to the vertex next to it.
- A path that does not repeat vertices is called a simple path.
- A **circuit** is path that begins and ends at the same vertex.

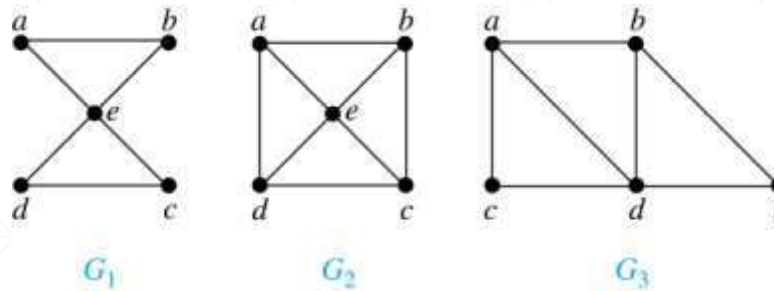
# Euler Paths and Circuits

- An *Euler path* in  $G$  is a simple path containing every edge of  $G$ . **Or** *Euler path* in  $G$  is a path that includes every edge of a  $G$  exactly once.
- An *Euler circuit* in a graph  $G$  is a simple circuit containing every edge of  $G$ . **Or** *Euler circuit* is a circuit that includes each edge exactly once
- An Euler circuit is always an Euler path, but an Euler path may not be an Euler circuit.



# Euler Paths and Circuits

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



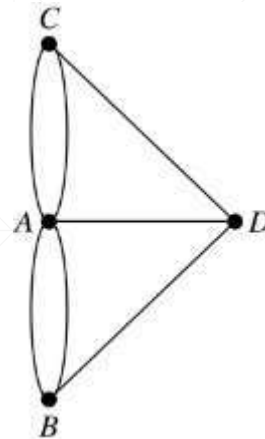
**Solution:** The graph  $G_1$  has an Euler circuit (e.g.,  $a, e, c, d, e, b, a$ ). But, as can easily be verified by inspection, neither  $G_2$  nor  $G_3$  has an Euler circuit. Note that  $G_3$  has an Euler path (e.g.,  $a, c, d, e, b, d, a, b$ ), but there is no Euler path in  $G_2$ , which can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

# Euler Theorem

1. A connected multi-graph with at least two vertices has an Euler circuit if and only if each of its vertices has an even degree
2. The graph has an Euler path if and only if it has exactly two vertices of odd degree.

# Euler Theorem

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

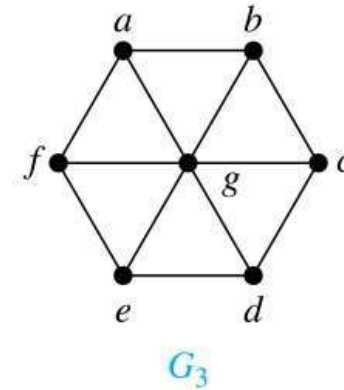
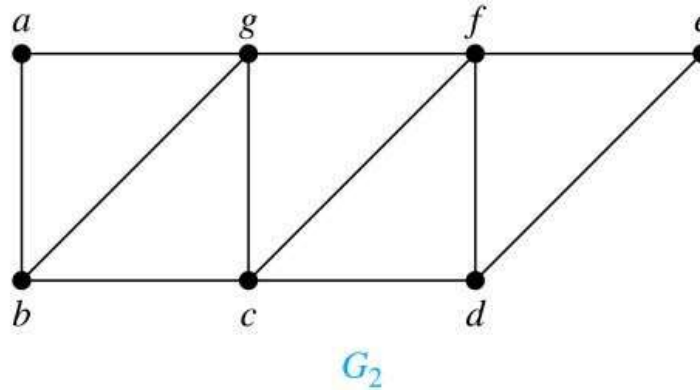
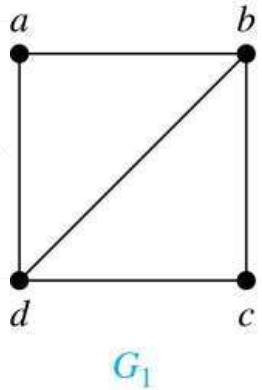


# Necessary Conditions for Euler Circuits and Paths

- 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.

# Euler Paths and Circuits

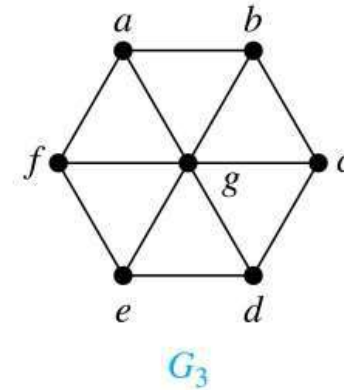
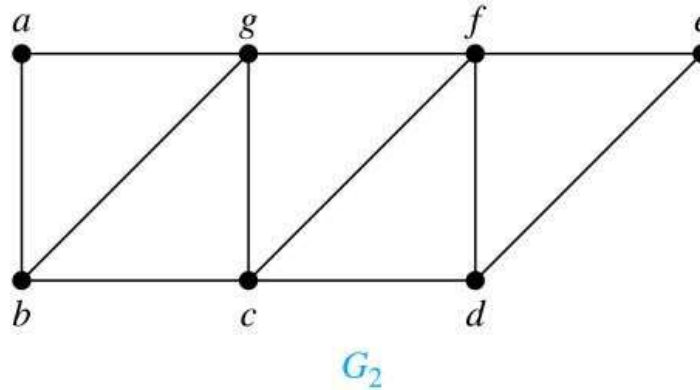
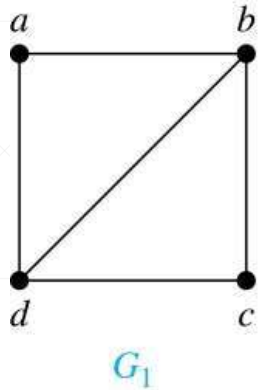
- **Example:**



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

# Euler Paths and Circuits

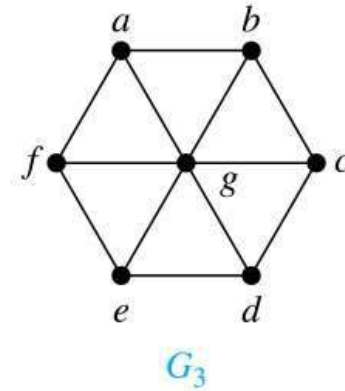
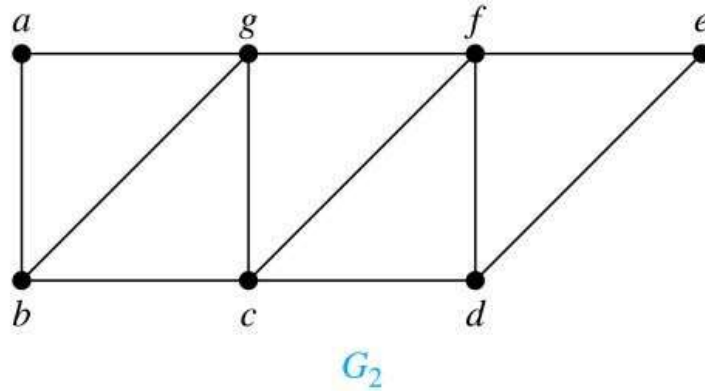
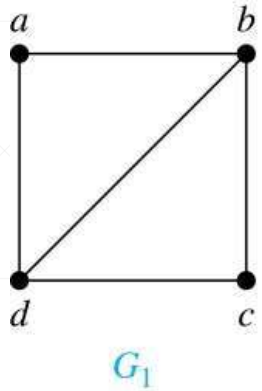
- **Example:**



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

# Euler Paths and Circuits

- **Example:**



- $G_3$  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 Path and Circuit

## Lecture 4

---

# Hamilton Paths and Circuits

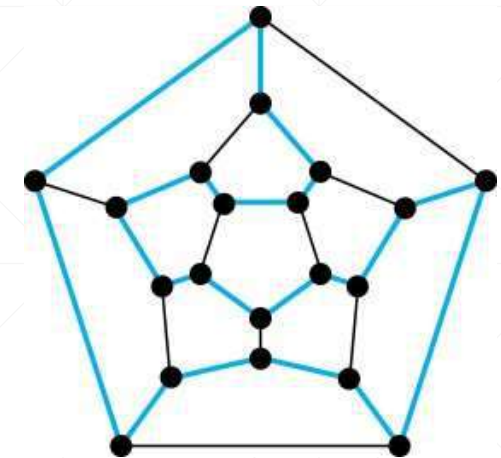
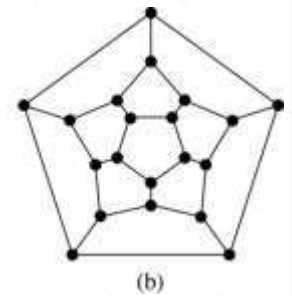
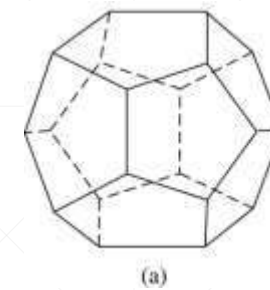
- Euler paths and circuits contained every edge only once. Now we look at paths and circuits that contain every vertex exactly once.
- **Definition:** A simple path in a graph  $G$  that passes through every vertex exactly once is called a *Hamilton path*, and a simple circuit in a graph  $G$  that passes through every vertex exactly once is called a *Hamilton circuit*.
- That is, a simple path  $x_0, x_1, \dots, x_{n-1}, x_n$  in the graph  $G = (V, E)$  is called a Hamilton path if  $V = \{x_0, x_1, \dots, x_{n-1}, x_n\}$  and  $x_i \neq x_j$  for  $0 \leq i < j \leq n$ , and the simple circuit  $x_0, x_1, \dots, x_{n-1}, x_n, x_0$  (with  $n > 0$ ) is a Hamilton circuit if  $x_0, x_1, \dots, x_{n-1}, x_n$  is a Hamilton path.

# Hamilton Paths and Circuits

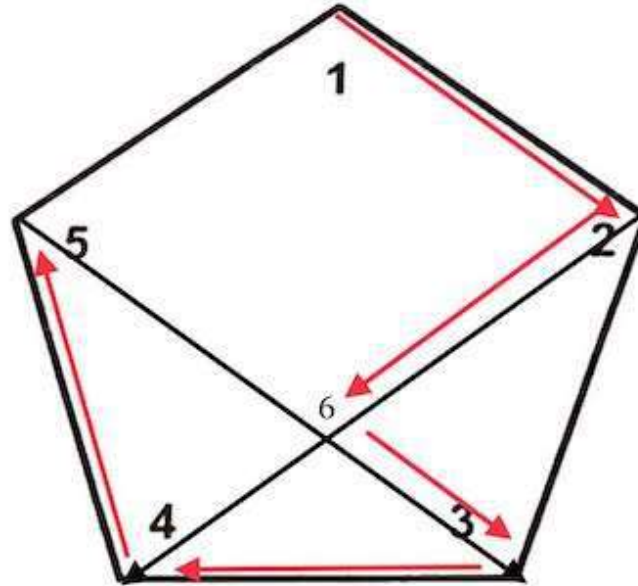


William Rowan  
Hamilton (1805-  
1865)

- 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 plot a circuit visiting 20 cities exactly once.
- The graph form of the puzzle is given in (b).
- The solution (a Hamilton circuit) is given here.



# Hamilton Path



# 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 Hamiltonian 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.

# Discrete Structures

Spring 2024 – Week 11



# Discrete Probability

## Lecture 1

---



# Probability of an Event



Pierre-Simon Laplace  
(1749-1827)

- Pierre-Simon Laplace introduced classical theory of probability in the 18<sup>th</sup> century when he analyzed games of chance
- An *experiment* is a procedure that yields one of a given set of possible outcomes.
- The *sample space* of the experiment is the set of possible outcomes.
- An *event* is a subset of the sample space.

# Probability of an Event

- Laplace's definition of the probability of an event:
- **Definition:** If  $S$  is a finite sample space of equally likely outcomes, and  $E$  is an event, that is, a subset of  $S$ , then the *probability* of  $E$  is

$$p(E) = |E| / |S|$$

- For every event  $E$ , we have  $0 \leq p(E) \leq 1$ .

# Applying Laplace's Definition

- **Example:** An urn contains four blue balls and five red balls. What is the probability that a ball chosen from the urn is blue?
- **Solution:** The probability that the ball is chosen is  $4/9$  since there are nine possible outcomes, and four of these produce a blue ball.

# Applying Laplace's Definition

- **Example:** What is the probability that when two dice are rolled, the sum of the numbers on the two dice is 7?
- **Solution:** By the product rule there are  $6^2 = 36$  possible outcomes. Six of these sum to 7. Hence, the probability of obtaining a 7 is  $6/36 = 1/6$ .

# Applying Laplace's Definition

- **Example:** In a lottery, a player wins a large prize when they pick four digits that match, in correct order, four digits selected by a random mechanical process. What is the probability that a player wins the prize?
- **Solution:** By the product rule there are  $10^4 = 10,000$  ways to pick four digits. Since there is only 1 way to pick the correct digits, the probability of winning the large prize is  $1/10,000 = 0.0001$ .

# Applying Laplace's Definition

- A smaller prize is won if only three digits are matched. What is the probability that a player wins the small prize?
- **Solution:** If exactly three digits are matched, one of the four digits must be incorrect and the other three digits must be correct. For the digit that is incorrect, there are 9 possible choices. Hence, by the sum rule, there a total of 36 possible ways to choose four digits that match exactly three of the winning four digits. The probability of winning the small price is  $36/10,000 = 9/2500 = 0.0036$ .

3 matched digits minus and also minus the probability of if all 4 matched.

# Applying Laplace's Definition

- **Example:** There are many lotteries that award prizes to people who correctly choose a set of six numbers out of the first  $n$  positive integers, where  $n$  is usually between 30 and 60. What is the probability that a person picks the correct six numbers out of 40?
- **Solution:** The number of ways to choose six numbers out of 40 is

$$C(40,6) = 40!/(34!6!) = 3,838,380.$$

Hence, the probability of picking a winning combination is  $1/3,838,380 \approx 0.00000026$

# Applying Laplace's Definition

- **Example:** What is the probability that the numbers 11, 4, 17, 39, and 23 are drawn in that order from a bin with 50 balls labeled with the numbers 1,2, ..., 50 if
  - a) The ball selected is not returned to the bin.
  - b) The ball selected is returned to the bin before the next ball is selected.



# Applying Laplace's Definition

- **Solution:** Use the product rule in each case.
  - a) *Sampling without replacement:* The probability is  $1/254,251,200$  since there are  $50 \cdot 49 \cdot 47 \cdot 46 \cdot 45 = 254,251,200$  ways to choose the five balls.
  - b) *Sampling with replacement:* The probability is  $1/50^5 = 1/312,500,000$  since  $50^5 = 312,500,000$ .

# The Probability of Complements

- **Theorem 1:** Let  $E$  be an event in sample space  $S$ . The probability of the event  $\overline{E} = S - E$ , the complementary event of  $E$ , is given by

$$p(\overline{E}) = 1 - p(E).$$

- **Proof:** Using the fact that  $|\overline{E}| = |S| - |E|$ ,

$$p(\overline{E}) = \frac{|S| - |E|}{|S|} = 1 - \frac{|E|}{|S|} = 1 - p(E).$$

# The Probability of Complements

- **Example:** A sequence of 10 bits is chosen randomly. What is the probability that at least one of these bits is 0?
- **Solution:** Let  $E$  be the event that at least one of the 10 bits is 0. Then  $\overline{E}$  is the event that all of the bits are 1s. The size of the sample space  $S$  is  $2^{10}$ . Hence,

$$p(E) = 1 - p(\overline{E}) = 1 - \frac{|\overline{E}|}{|S|} = 1 - \frac{1}{2^{10}} = 1 - \frac{1}{1024} = \frac{1023}{1024}.$$

# The Probability of Unions of Events

- **Theorem 2:** Let  $E_1$  and  $E_2$  be events in the sample space  $S$ . Then

$$p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)$$

**Proof:** Given the inclusion-exclusion formula from Section 2.2,  $|A \cup B| = |A| + |B| - |A \cap B|$ , it follows that

$$\begin{aligned} p(E_1 \cup E_2) &= \frac{|E_1 \cup E_2|}{|S|} = \frac{|E_1| + |E_2| - |E_1 \cap E_2|}{|S|} \\ &= \frac{|E_1|}{|S|} + \frac{|E_2|}{|S|} - \frac{|E_1 \cap E_2|}{|S|} \\ &= p(E_1) + p(E_2) - p(E_1 \cap E_2). \end{aligned}$$

# The Probability of Unions of Events

- **Example:** What is the probability that a positive integer selected at random from the set of positive integers not exceeding 100 is divisible by either 2 or 5?
- **Solution:** Let  $E_1$  be the event that the integer is divisible by 2 and  $E_2$  be the event that it is divisible 5? Then the event that the integer is divisible by 2 or 5 is  $E_1 \cup E_2$  and  $E_1 \cap E_2$  is the event that it is divisible by 2 and 5.

It follows that:

$$\begin{aligned} p(E_1 \cup E_2) &= p(E_1) + p(E_2) - p(E_1 \cap E_2) \\ &= 50/100 + 20/100 - 10/100 = 3/5. \end{aligned}$$

# Probability Theory

Lecture 2 and Lecture 3

---

# Assigning Probabilities

- Laplace's definition from the previous section, assumes that all outcomes are equally likely. Now we introduce a more general definition of probabilities that avoids this restriction.
- Let  $\mathcal{S}$  be a sample space of an experiment with a finite number of outcomes. We assign a probability  $p(s)$  to each outcome  $s$ , so that:
  - i.  $0 \leq p(s) \leq 1$  for each  $s \in \mathcal{S}$
  - ii.  $\sum_{s \in \mathcal{S}} p(s) = 1$

# Assigning Probabilities

- The function  $p$  from the set of all outcomes of the sample space  $\mathcal{S}$  is called a *probability distribution*.



# Assigning Probabilities

- **Example:** What probabilities should we assign to the outcomes  $H$ (heads) and  $T$  (tails) when a fair coin is flipped? What probabilities should be assigned to these outcomes when the coin is biased so that heads comes up twice as often as tails?
- **Solution:** For a fair coin, we have  $p(H) = p(T) = 1/2$ .
- For a biased coin, we have  $p(H) = 2p(T)$ .
- Because  $p(H) + p(T) = 1$ , it follows that

$$2p(T) + p(T) = 3p(T) = 1.$$

Hence,  $p(T) = 1/3$  and  $p(H) = 2/3$ .

# Uniform Distribution

- **Definition:** Suppose that  $S$  is a set with  $n$  elements. The *uniform distribution* assigns the probability  $1/n$  to each element of  $S$ . (Note that we could have used Laplace's definition here.)
- **Example:** Consider again the coin flipping example, but with a fair coin. Now  $p(H) = p(T) = 1/2$ .

# Probability of an Event

- **Definition:** The probability of the event  $E$  is the sum of the probabilities of the outcomes in  $E$ .

$$p(E) = \sum_{s \in E} p(s)$$

- Note that now no assumption is being made about the distribution.

# Example

- **Example:** Suppose that a die is biased so that 3 appears twice as often as each other number, but that the other five outcomes are equally likely. What is the probability that an odd number appears when we roll this die?
- **Solution:** We want the probability of the event  $E = \{1,3,5\}$
- We have  $p(3) = 2/7$  and
- $p(1) = p(2) = p(4) = p(5) = p(6) = 1/7$ .
- Hence,  $p(E) = p(1) + p(3) + p(5) = 1/7 + 2/7 + 1/7 = 4/7$ .

# Probabilities of Complements and Unions of Events

- Complements:  $p(\overline{E}) = 1 - p(E)$  still holds.
- Since each outcome is in either  $E$  or  $\overline{E}$ , but not both,

$$\sum_{s \in S} p(s) = 1 = p(E) + p(\overline{E}).$$

- Unions:  $p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)$  also still holds under the new definition.

# Conditional Probability

- Definition: Let  $E$  and  $F$  be events with  $p(F) > 0$ .
- The conditional probability of  $E$  given  $F$ , denoted by  $P(E | F)$ , is defined as:

$$p(E|F) = \frac{p(E \cap F)}{p(F)}$$

# Conditional Probability

- **Example:** A bit string of length four is generated at random so that each of the 16 bit strings of length 4 is equally likely. What is the probability that it contains at least two consecutive 0s, given that its first bit is a 0?
- **Solution:** Let  $E$  be the event that the bit string contains at least two consecutive 0s, and  $F$  be the event that the first bit is a 0.
- Since  $E \cap F = \{0000, 0001, 0010, 0011, 0100\}$
- $p(E \cap F) = 5/16$ .
- Because 8 bit strings of length 4 start with a 0,  $p(F) = 8/16 = 1/2$ .  
Hence,

$$p(E|F) = \frac{p(E \cap F)}{p(F)} = \frac{5/16}{1/2} = \frac{5}{8}.$$

# Conditional Probability

- **Example:** What is the conditional probability that a family with two children has two boys, given that they have at least one boy. Assume that each of the possibilities  $BB$ ,  $BG$ ,  $GB$ , and  $GG$  is equally likely where  $B$  represents a boy and  $G$  represents a girl.
- **Solution:** Let  $E$  be the event that the family has two boys and let  $F$  be the event that the family has at least one boy. Then  $E = \{BB\}$ ,  $F = \{BB, BG, GB\}$ , and  $E \cap F = \{BB\}$ .
- It follows that  $p(F) = 3/4$  and  $p(E \cap F) = 1/4$ .
- Hence, 
$$p(E|F) = \frac{p(E \cap F)}{p(F)} = \frac{1/4}{3/4} = \frac{1}{3}.$$



# Independence

- **Definition:** The events  $E$  and  $F$  are independent if and only if

$$p(E \cap F) = p(E)p(F).$$

# Independence

- **Example:** Suppose  $E$  is the event that a randomly generated bit string of length four begins with a 1 and  $F$  is the event that this bit string contains an even number of 1s. Are  $E$  and  $F$  independent if the 16 bit strings of length four are equally likely?
- **Solution:** There are eight bit strings of length four that begin with a 1, and eight bit strings of length four that contain an even number of 1s. Since the number of bit strings of length 4 is 16,

$$p(E) = p(F) = 8/16 = 1/2.$$

- Since  $E \cap F = \{1111, 1100, 1010, 1001\}$ ,  $p(E \cap F) = 4/16 = 1/4$ .
- We conclude that  $E$  and  $F$  are independent, because  $p(E \cap F) = 1/4 = (1/2)(1/2) = p(E)p(F)$

# Independence

- **Example:** Assume that each of the four ways a family can have two children ( $BB, GG, BG, GB$ ) is equally likely. Are the events  $E$ , that a family with two children has two boys, and  $F$ , that a family with two children has at least one boy, independent?
- **Solution:** Because  $E = \{BB\}$ ,  $p(E) = 1/4$ .
- We saw previously that that  $p(F) = 3/4$  and  $p(E \cap F) = 1/4$ .
- The events  $E$  and  $F$  are not independent since
$$p(E) p(F) = 3/16 \neq 1/4 = p(E \cap F) .$$

# Pair-wise and Mutual Independence

- **Definition:** The events  $E_1, E_2, \dots, E_n$  are *pairwise independent* if and only if  $p(E_i \cap E_j) = p(E_i) p(E_j)$  for all pairs  $i$  and  $j$  with  $i \leq j \leq n$ .
- The events are *mutually independent* if

$$p(E_{i_1} \cap E_{i_2} \cap \dots \cap E_{i_m}) = p(E_{i_1})p(E_{i_2}) \dots p(E_{i_m})$$

whenever  $i_j, j = 1, 2, \dots, m$ , are integers with

$$1 \leq i_1 < i_2 < \dots < i_m \leq n \quad \text{and} \quad m \geq 2.$$

James Bernoulli  
(1654 – 1705)



# Bernoulli Trials

- **Definition:** Suppose an experiment can have only two possible outcomes, *e.g.*, the flipping of a coin or the random generation of a bit.
  - Each performance of the experiment is called a *Bernoulli trial*.
  - One outcome is called a *success* and the other a *failure*.
  - If  $p$  is the probability of success and  $q$  the probability of failure, then  $p + q = 1$ .
- Many problems involve determining the probability of  $k$  successes when an experiment consists of  $n$  mutually independent Bernoulli trials.

# Bernoulli Trials

- **Example:** A coin is biased so that the probability of heads is  $2/3$ . What is the probability that exactly four heads occur when the coin is flipped seven times?
- **Solution:** There are  $2^7 = 128$  possible outcomes. The number of ways four of the seven flips can be heads is  $C(7,4)$ . The probability of each of the outcomes is  $(2/3)^4(1/3)^3$  since the seven flips are independent. Hence, the probability that exactly four heads occur is
- $$C(7,4) (2/3)^4(1/3)^3 = (35 \cdot 16) / 2^7 = 560 / 2187.$$

# Practice Problems

---

# Practice Problem

- What is the probability that a randomly selected integer chosen from the first 100 positive integers is odd?
- **Solution:** Determine the number of odd integers in the range and divide it by the total number of integers.



# Practice Problem

- What is the probability that the sum of the numbers on two dice is even when they are rolled?
- **Solution:** Determine the number of favorable outcomes (even sums) and divide it by the total number of possible outcomes.

# Practice Problem

- What is the probability that a fair die never comes up an even number when it is rolled six times?
- Solution: Since we want to find the probability of never rolling an even number in six rolls, we need to consider the favorable outcomes where only odd numbers appear on each roll.

# Practice Problem

- What is the probability that a positive integer not exceeding 100 selected at random is divisible by 5 or 7?

# Practice Problem

- What is the probability that Abby, Barry, and Sylvia win the first, second, and third prizes, respectively, in a drawing if 200 people enter a contest and
  - a) no one can win more than one prize.
  - b) winning more than one prize is allowed.

# Practice Problem

- Find the probability of each outcome when a biased die is rolled, if rolling a 2 or rolling a 4 is three times as likely as rolling each of the other four numbers on the die and it is equally likely to roll a 2 or a 4.

# Practice Problem

- What is the conditional probability that exactly four heads appear when a fair coin is flipped five times, given that the first flip came up heads?

# Discrete Structures

Spring 2024 – Week 12



# Trees

---

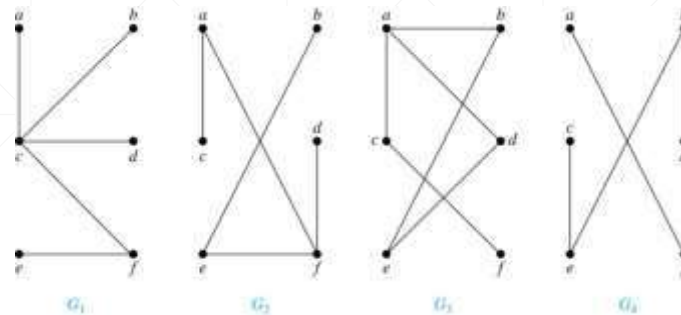
---



# Trees

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

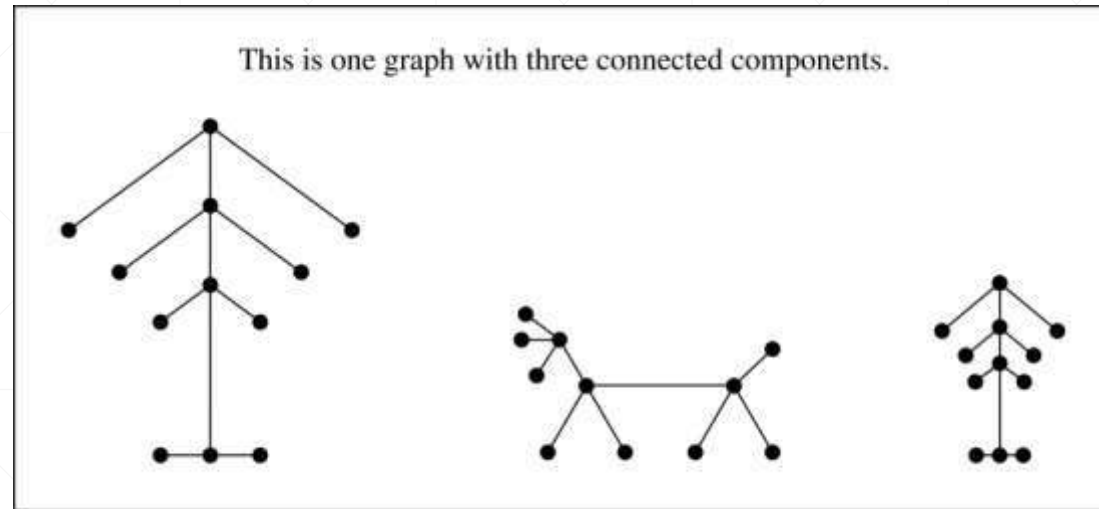
**Example:** Which of these graphs are trees?



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

# Trees

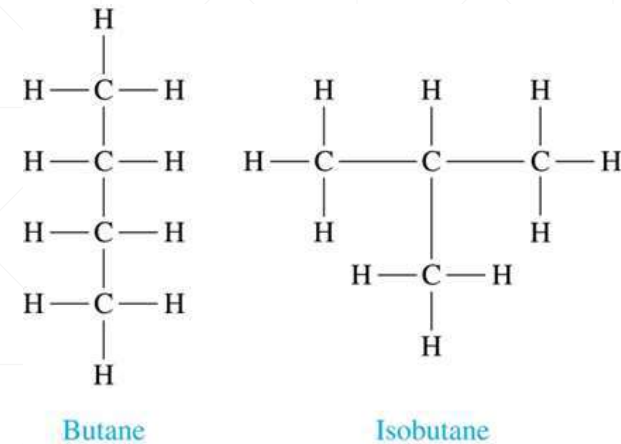
**Definition:** A *forest* is a graph that has no simple circuit, but is not connected. Each of the connected components in a forest is a tree.



# Trees as Models

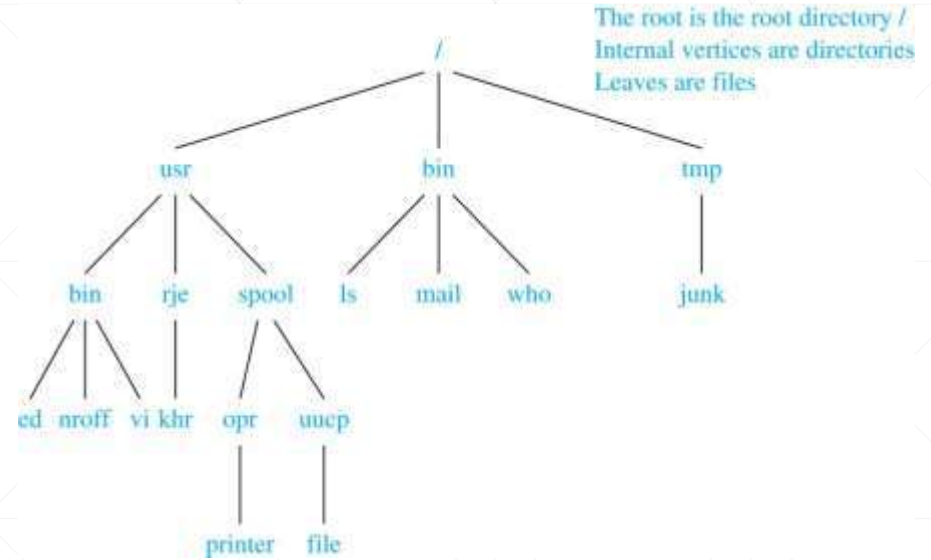
- Trees are used as models in computer science, chemistry, geology, botany, psychology, and many other areas.
- Trees were introduced by the mathematician Cayley in 1857 in his work counting the number of isomers of saturated hydrocarbons. The two isomers of butane are shown at the right.

Arthur Cayley  
(1821-1895)



# Trees as Models

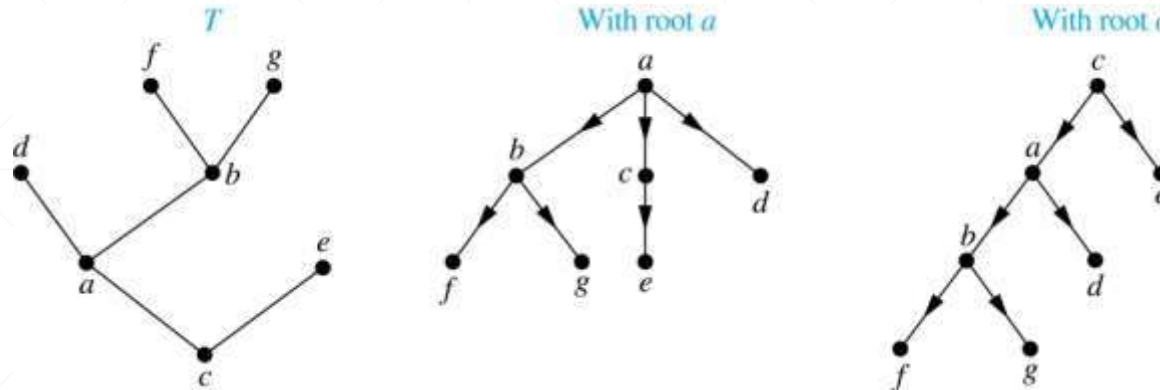
- The organization of a computer file system into directories, subdirectories, and files is naturally represented as a tree.
- Trees are used to represent the structure of organizations.



# Rooted Trees

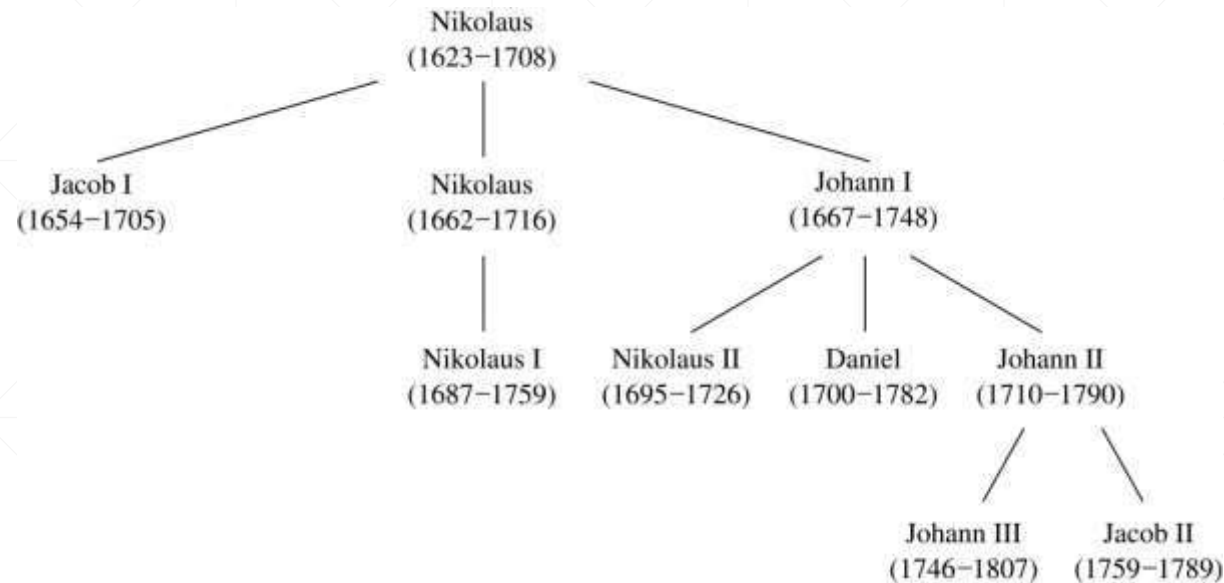
**Definition:** A *rooted tree* is a tree in which one vertex has been designated as the *root* and every edge is directed away from the root.

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*.
- 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.
- A vertex of a rooted tree with no children is called a *leaf*. Vertices that have children are called *internal vertices*.
- If  $a$  is a vertex in a tree, the *subtree* with  $a$  as its root is the subgraph of the tree consisting of  $a$  and its descendants and all edges incident to these descendants.

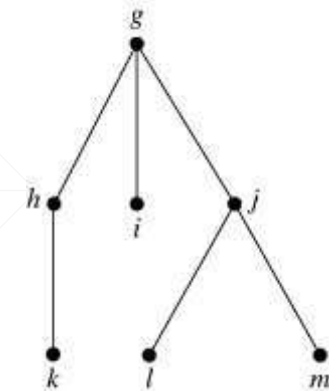
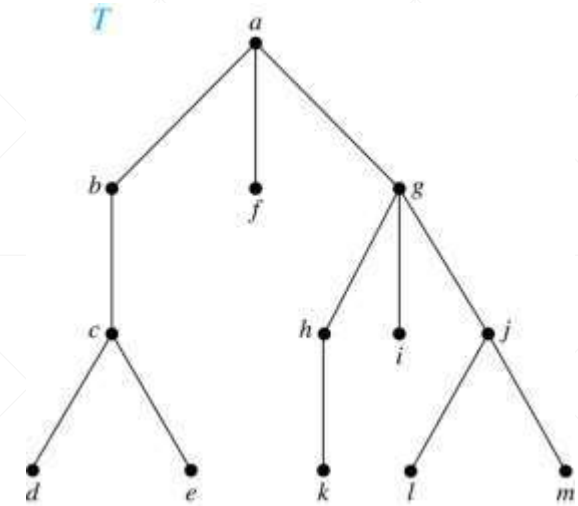
# Terminology for Rooted Trees

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

- (i) Find the parent of  $c$ , the children of  $g$ , the siblings of  $b$ , the ancestors of  $e$ , and the descendants of  $b$ .
- (ii) Find all internal vertices and all leaves.
- (iii) What is the subtree rooted at  $G$ ?

**Solution:**

- (i) The parent of  $c$  is  $b$ . The children of  $g$  are  $h$ ,  $i$ , and  $j$ . The siblings of  $b$  are  $f$  and  $g$ . The ancestors of  $e$  are  $c$ ,  $b$ , and  $a$ . The descendants of  $b$  are  $c$ ,  $d$ , and  $e$ .
- (ii) The internal vertices are  $a$ ,  $b$ ,  $c$ ,  $g$ ,  $h$ , and  $j$ . The leaves are  $d$ ,  $e$ ,  $f$ ,  $i$ ,  $k$ ,  $l$ , and  $m$ .
- (iii) We display the subtree rooted at  $g$ .

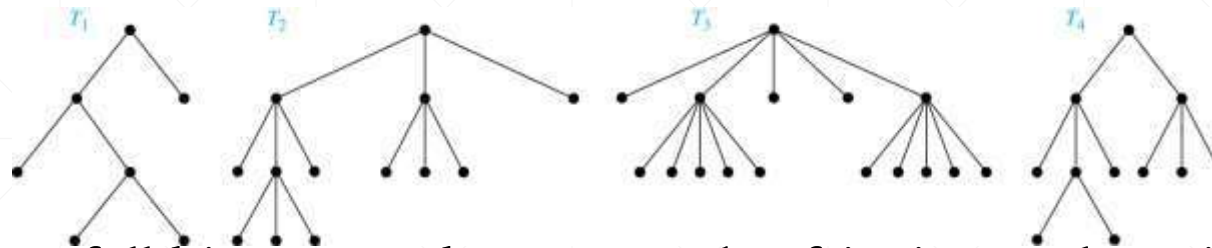




# $m$ -ary Rooted Trees

**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.

# 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.

**Definition:** A *binary tree* is an ordered rooted tree 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.

# Ordered Rooted Trees

**Example:** Consider the binary tree  $T$ .

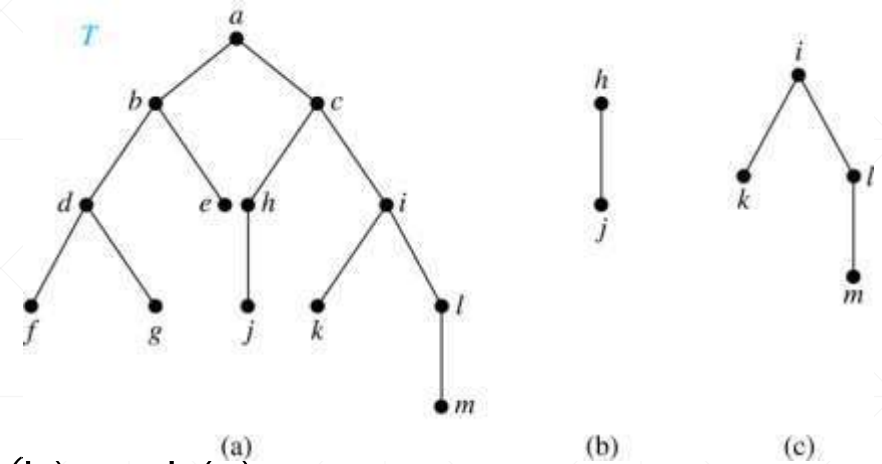
(i) What are the left and right children of  $d$ ?

(ii) What are the left and right subtrees 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).



# Level of vertices and height of trees

- When working with trees, we often want to have rooted trees where the subtrees 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.

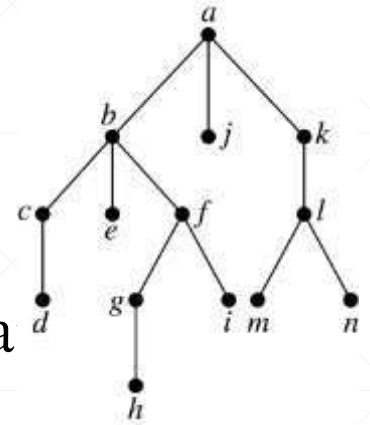
# Level of vertices and height of trees

## Example:

- (i) Find the level of each vertex in the tree to the right.
- (ii) What is the height of the tree?

## Solution:

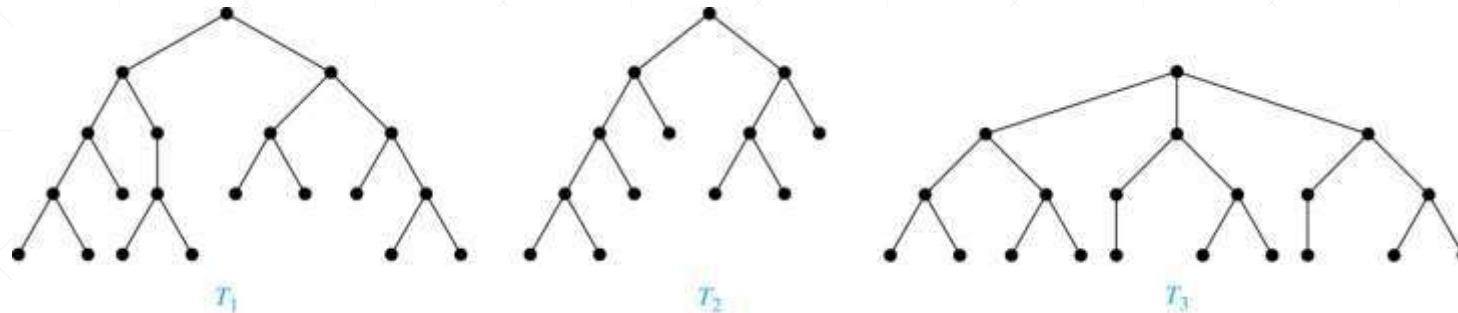
- (i) The root  $a$  is at level 0. Vertices  $b, j$ , and  $k$  are at level 1. Vertices  $c, e, f$ , and  $l$  are at level 2. Vertices  $d, g, i, m$ , and  $n$  are at level 3. Vertex  $h$  is at level 4.
- (ii) The height is 4, since 4 is the largest level of any vertex.



# 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.

# Tree Traversal

---

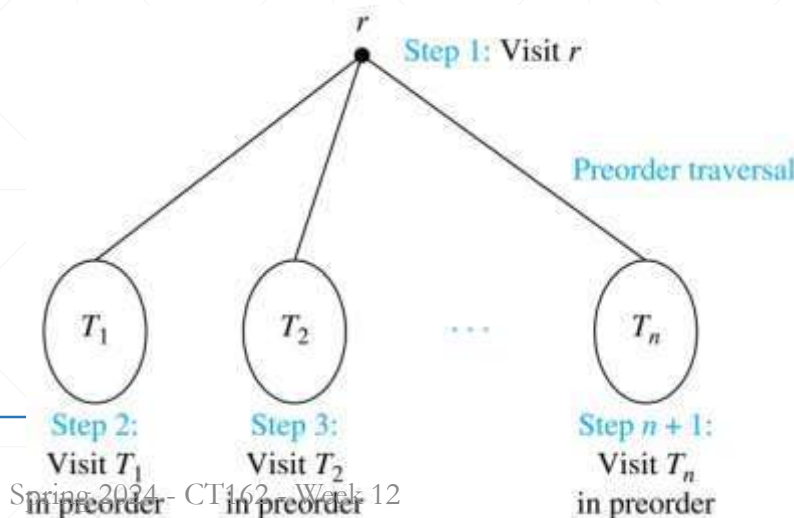
# Tree Traversal

- 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

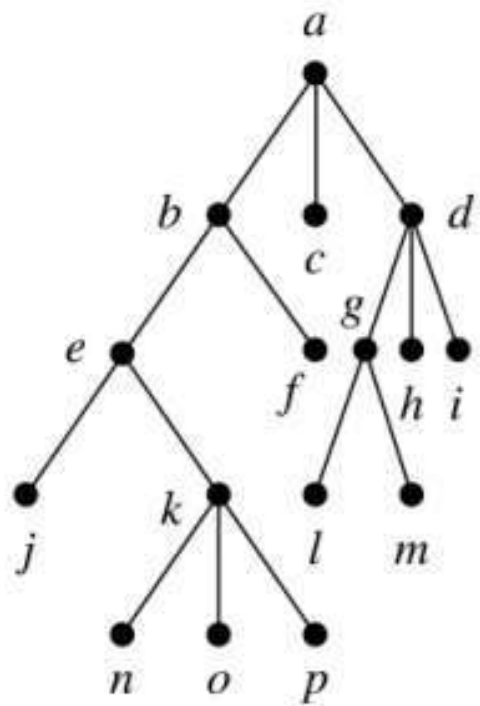
**Definition:** Let  $T$  be an ordered rooted tree with root  $r$ . If  $T$  consists only of  $r$ , then  $r$  is the *preorder traversal* of  $T$ . Otherwise, suppose that  $T_1, T_2, \dots, T_n$  are the subtrees of  $r$  from left to right in  $T$ . The preorder traversal begins by visiting  $r$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



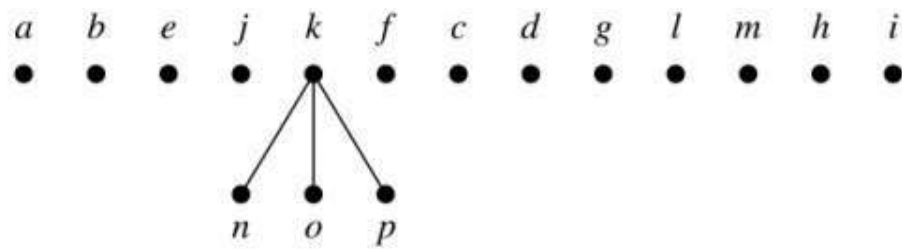
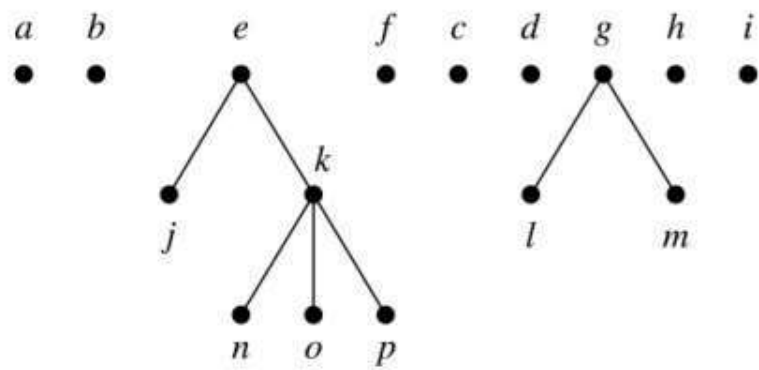
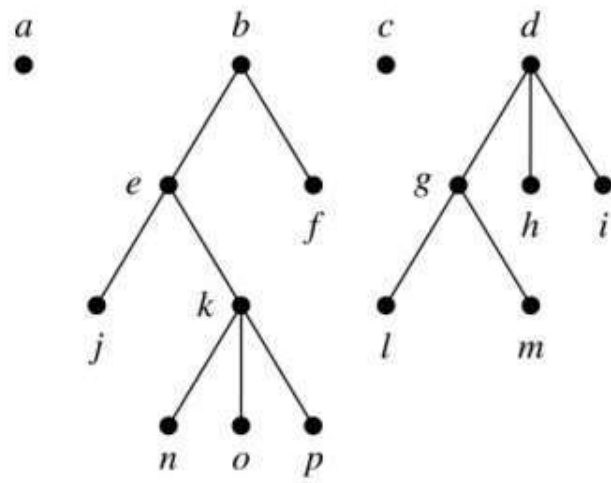
# Preorder Traversal (*continued*)

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

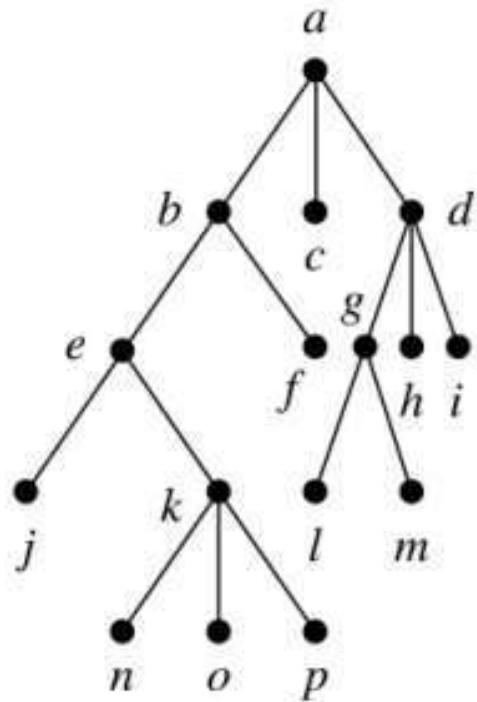
# Preorder Traversal (*continued*)



**Preorder traversal:** Visit root,  
visit subtrees left to right



# Preorder Traversal (*continued*)

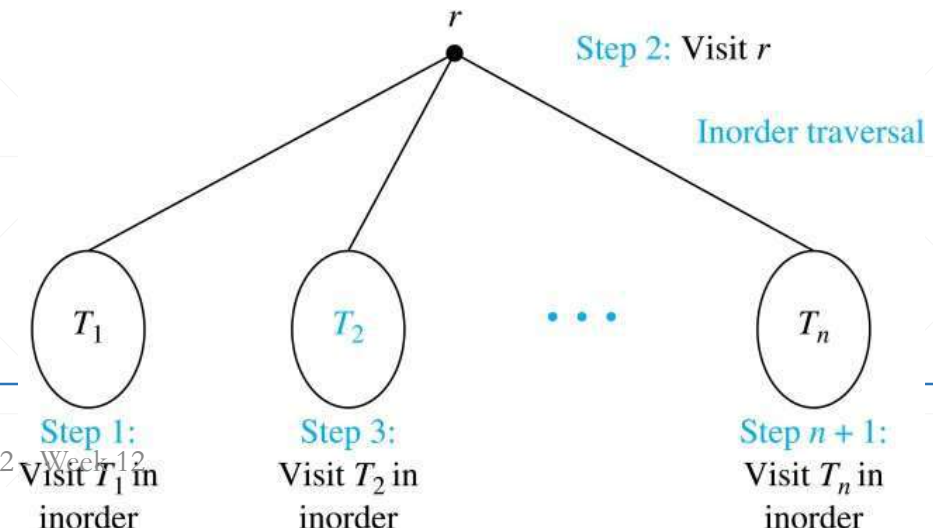


**Preorder traversal:** Visit root,  
visit subtrees left to right

*a b e j k n o p f c d g l m h i*

# Inorder Traversal

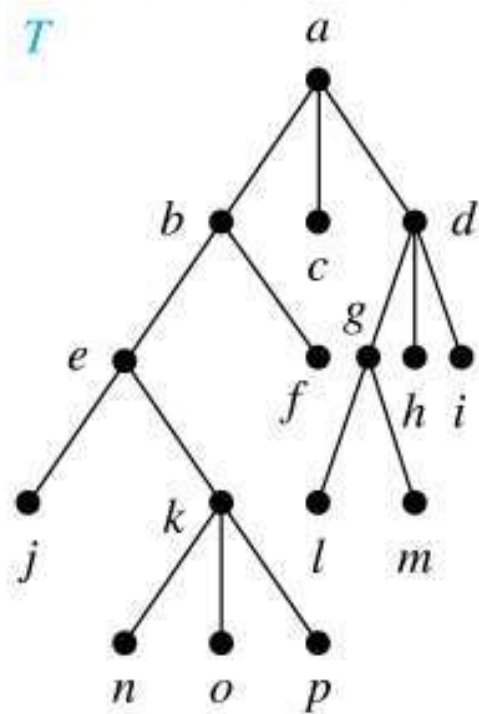
**Definition:** Let  $T$  be an ordered rooted tree with root  $r$ . If  $T$  consists only of  $r$ , then  $r$  is the *inorder traversal* of  $T$ . Otherwise, suppose that  $T_1, T_2, \dots, T_n$  are the subtrees of  $r$  from left to right in  $T$ . The inorder traversal begins by traversing  $T_1$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

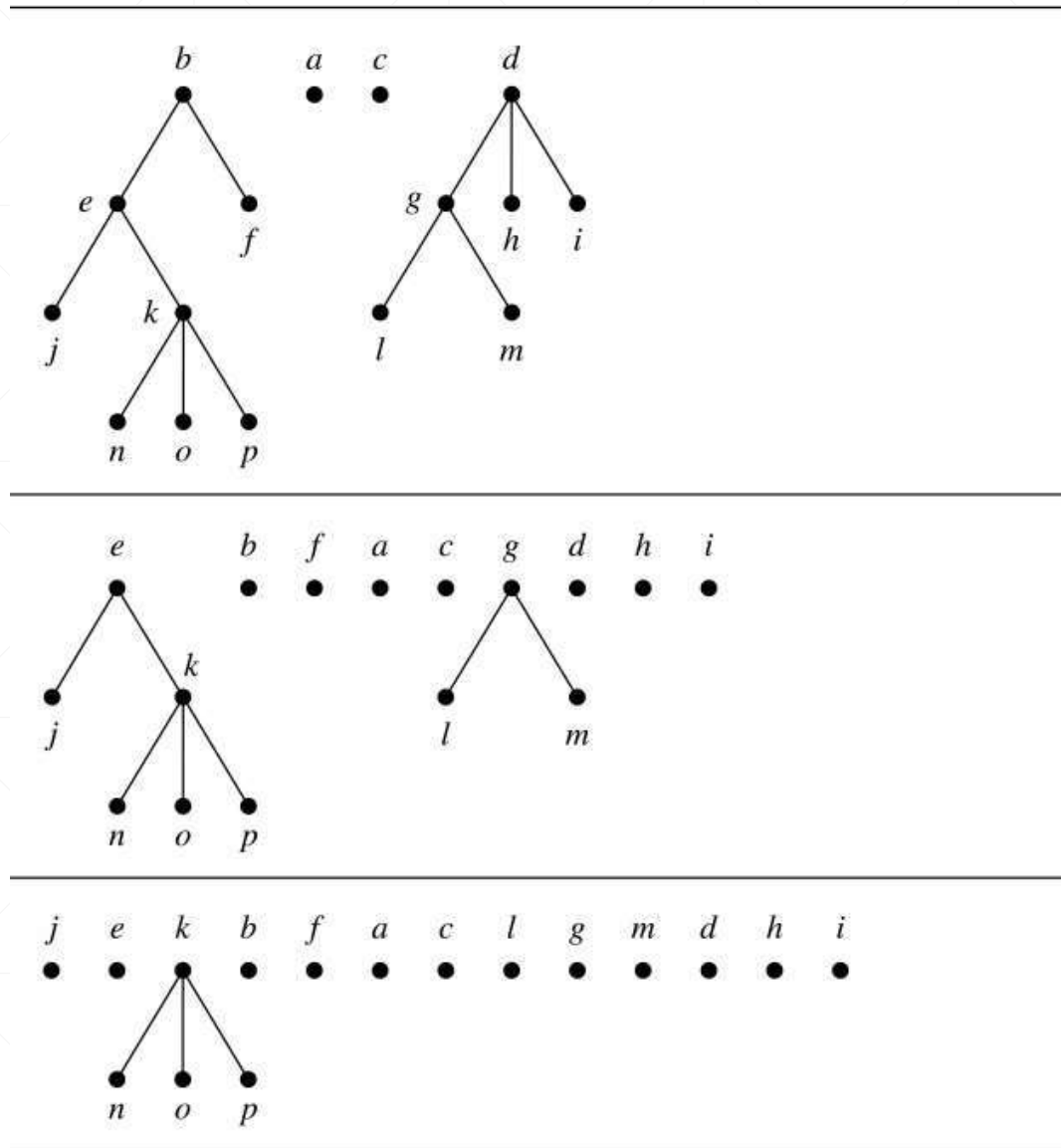
- **procedure** *inorder* ( $T$ : ordered rooted tree)
- $r := \text{root of } T$
- **if**  $r$  is a leaf **then** list  $r$
- **else**
  - $l := \text{first child of } r \text{ from left to right}$
  - $T(l) := \text{subtree with } l \text{ as its root}$
  - *inorder*( $T(l)$ )
  - list( $r$ )
  -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

# Inorder Traversal (*continued*)

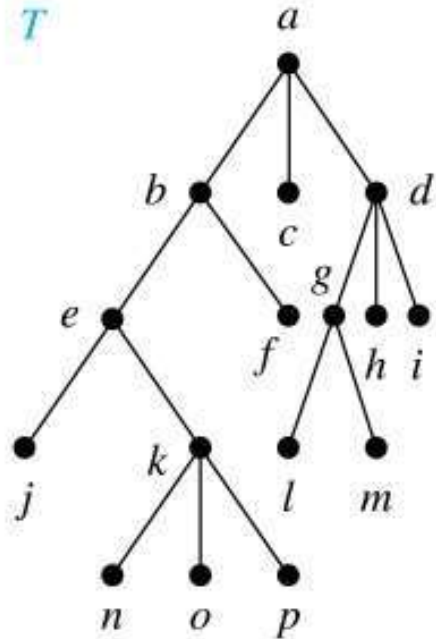


**Inorder traversal:** Visit leftmost subtree, visit root, visit other subtrees left to right





# Inorder Traversal (*continued*)



**Inorder traversal:** Visit leftmost subtree, visit root, visit other subtrees left to right

---

---

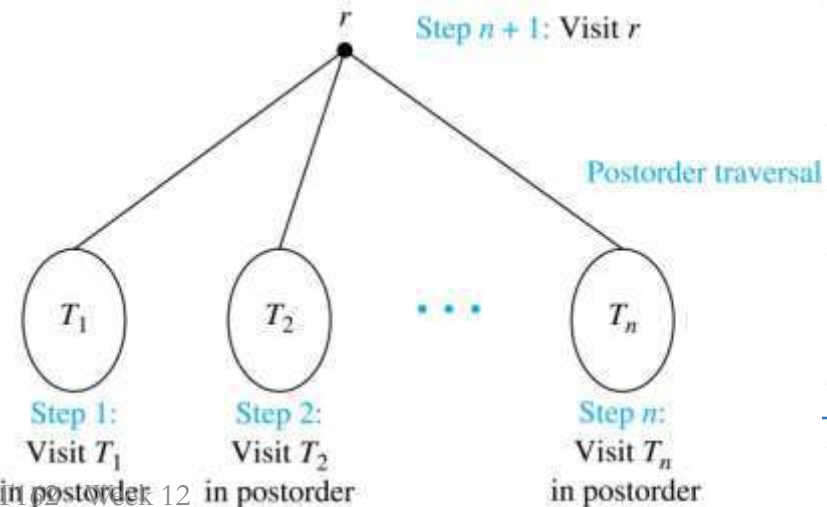
<i>j</i>	<i>e</i>	<i>n</i>	<i>k</i>	<i>o</i>	<i>p</i>	<i>b</i>	<i>f</i>	<i>a</i>	<i>c</i>	<i>l</i>	<i>g</i>	<i>m</i>	<i>d</i>	<i>h</i>	<i>i</i>
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

---

---

# Postorder Traversal

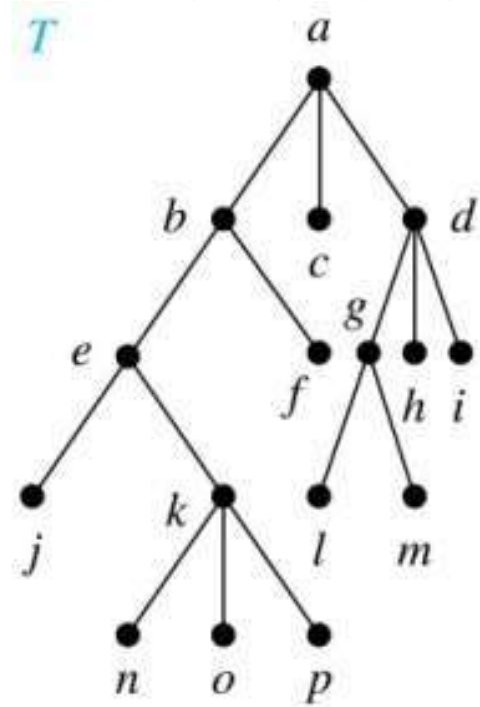
**Definition:** Let  $T$  be an ordered rooted tree with root  $r$ . If  $T$  consists only of  $r$ , then  $r$  is the *postorder traversal* of  $T$ . Otherwise, suppose that  $T_1, T_2, \dots, T_n$  are the subtrees of  $r$  from left to right in  $T$ . The postorder traversal begins by traversing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



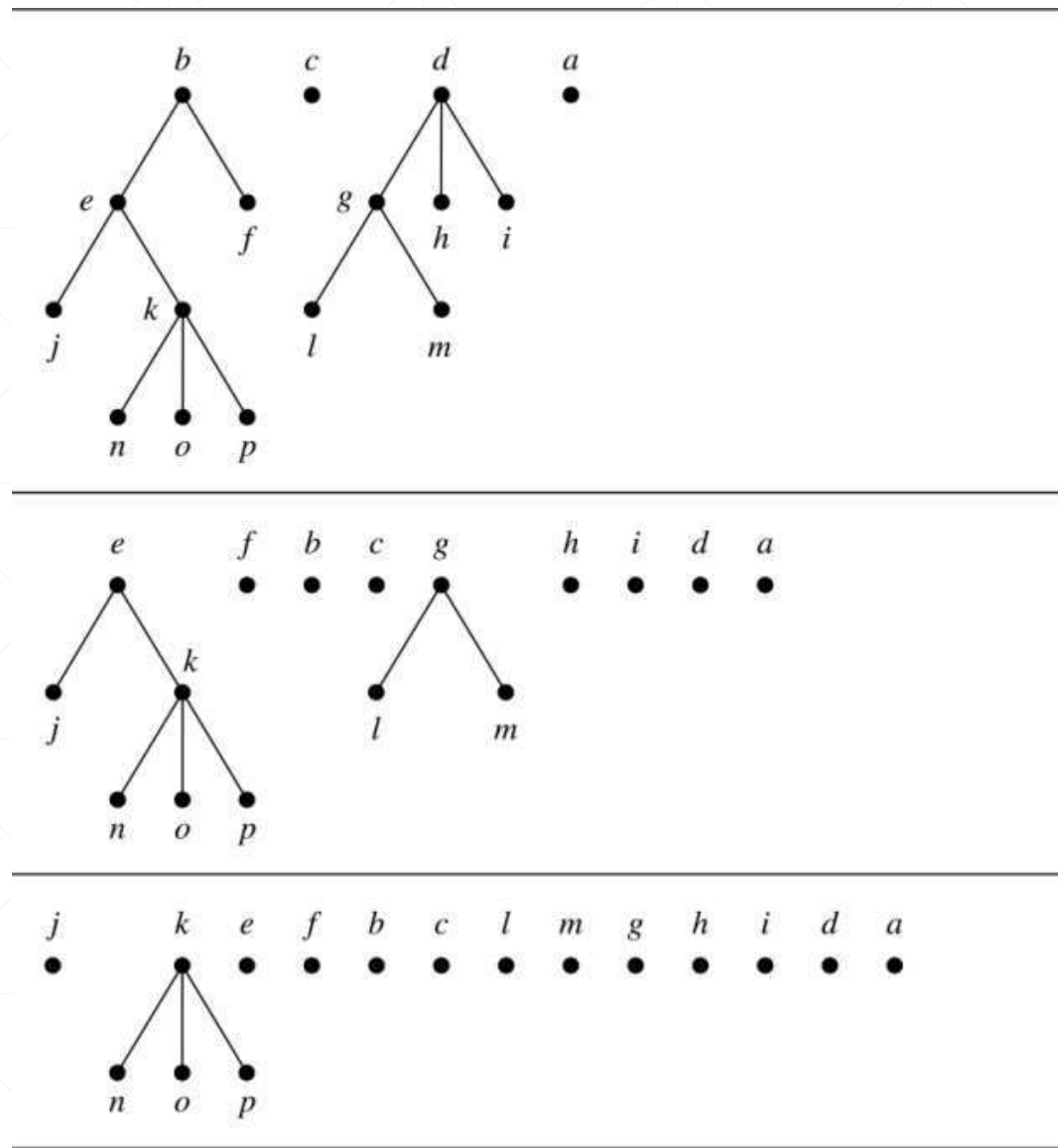
# Postorder Traversal

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

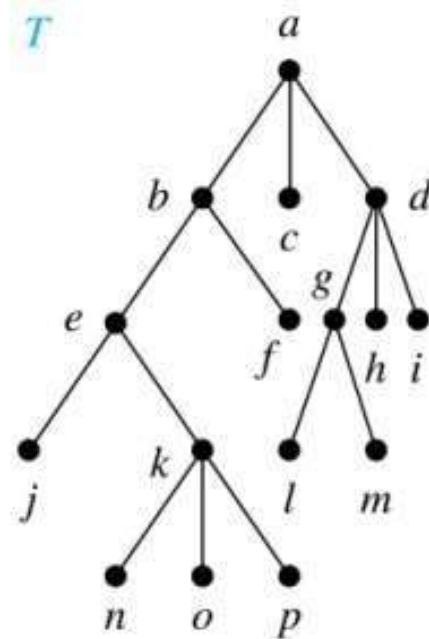
# Postorder Traversal (*continued*)



**Postorder traversal:** Visit subtrees left to right; visit root



# Postorder Traversal (*continued*)



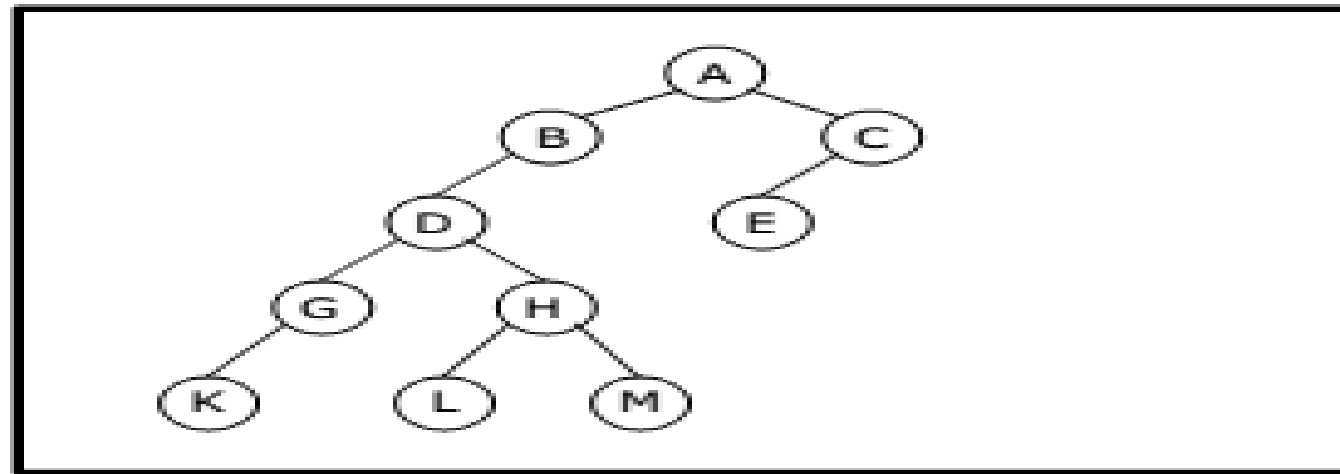
Postorder traversal: Visit  
subtrees left to right; visit root

---

<i>j</i>	<i>n</i>	<i>o</i>	<i>p</i>	<i>k</i>	<i>e</i>	<i>f</i>	<i>b</i>	<i>c</i>	<i>l</i>	<i>m</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>d</i>	<i>a</i>
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

---

# Exercise!



Binary Tree



# Solution!

- Preorder traversal yields:  
A, B, D, G, K, H, L, M, C, E
- Postorder traversal yields:  
K, G, L, M, H, D, B, E, C, A
- Inorder traversal yields:  
K, G, D, L, H, M, B, A, E, C

Pre, Post and Inorder Traversing

# Discrete Structures

Spring 2024 – Week 13



# Boolean Algebra

---

# Introduction to Boolean Algebra

- Boolean algebra has rules for working with elements from the set  $\{0, 1\}$  together with the operators  $+$  (Boolean sum),  $\cdot$  (Boolean product), and  $\bar{\phantom{x}}$  (complement).
- These operators are defined by:
- *Boolean sum:*  $1 + 1 = 1, 1 + 0 = 1, 0 + 1 = 1, 0 + 0 = 0$
- *Boolean product:*  $1 \cdot 1 = 1, 1 \cdot 0 = 0, 0 \cdot 1 = 0, 0 \cdot 0 = 0$
- *Complement:*  $\bar{0} = 1, \bar{1} = 0$

# Introduction to Boolean Algebra

**Example:** Find the value of  $1 \cdot 0 + \overline{(0 + 1)}$

$$\begin{aligned}\text{Solution : } 1 \cdot 0 + \overline{(0 + 1)} &= 0 + \bar{1} \\ &= 0 + 0 \\ &= 0\end{aligned}$$

# Boolean Expressions and Boolean Functions

## Definition:

Let  $B = \{0, 1\}$

Then  $B^n = \{(x_1, x_2, \dots, x_n) \mid x_i \in B \text{ for } 1 \leq i \leq n\}$  is the set of all possible  $n$ -tuples of 0s and 1s.

The variable  $x$  is called a *Boolean variable* if it assumes values only from  $B$ , that is, if its only possible values are 0 and 1.

A function from  $B^n$  to  $B$  is called a *Boolean function of degree  $n$* .

# Boolean Expressions and Boolean Functions

**Example:** The function  $F(x, y) = xȳ$  from the set of ordered pairs of Boolean variables to the set  $\{0, 1\}$  is a Boolean function of degree 2.

TABLE 1		
$x$	$y$	$F(x, y)$
1	1	0
1	0	1
0	1	0
0	0	0

# Boolean Expressions and Boolean Functions (continued)

**Example:** Find the values of the Boolean function represented by  $F(x, y, z) = xy + \bar{z}$ .

**Solution:** We use a table with a row for each combination of values of  $x$ ,  $y$ , and  $z$  to compute the values of  $F(x, y, z)$ .

TABLE 2					
$x$	$y$	$z$	$xy$	$\bar{z}$	$F(x, y, z) = xy + \bar{z}$
1	1	1	1	0	1
1	1	0	1	1	1
1	0	1	0	0	0
1	0	0	0	1	1
0	1	1	0	0	0
0	1	0	0	1	1
0	0	1	0	0	0
0	0	0	0	1	1



# Boolean Expressions and Boolean Functions (*continued*)

- **Definition:** Boolean functions  $F$  and  $G$  of  $n$  variables are equal if and only if  $F(b_1, b_2, \dots, b_n) = G(b_1, b_2, \dots, b_n)$  whenever  $b_1, b_2, \dots, b_n$  belong to  $B$ . Two different Boolean expressions that represent the same function are *equivalent*.
- **Definition:** The complement of the Boolean function  $F$  is the function  $\bar{F}$ , where  $\bar{F}(x_1, x_2, \dots, x_n) = \overline{F(x_1, x_2, \dots, x_n)}$ .

# Boolean Expressions and Boolean Functions (*continued*)

**Definition:** Let  $F$  and  $G$  be Boolean functions of degree  $n$ . The Boolean sum  $F + G$  and the Boolean product  $FG$  are defined by

$$(F + G)(x_1, x_2, \dots, x_n) = F(x_1, x_2, \dots, x_n) + G(x_1, x_2, \dots, x_n)$$

$$(FG)(x_1, x_2, \dots, x_n) = F(x_1, x_2, \dots, x_n)G(x_1, x_2, \dots, x_n)$$

# Identities of Boolean Algebra

Each identity can be proved using a table.

TABLE 5 Boolean Identities.	
Identity	Name
$\overline{\overline{x}} = x$	Law of the double complement
$x + x = x$ $x \cdot x = x$	Idempotent laws
$x + 0 = x$ $x \cdot 1 = x$	Identity laws
$x + 1 = 1$ $x \cdot 0 = 0$	Domination laws
$x + y = y + x$ $xy = yx$	Commutative laws

All identities in Table 5, except for the first and the last two come in pairs. Each element of the pair is the *dual* of the other (obtained by switching Boolean sums and Boolean products and 0's and 1's).

The Boolean identities correspond to the identities of propositional logic (Section 1.3) and the set identities (Section 2.2).

# Identities of Boolean Algebra

$x + (y + z) = (x + y) + z$ $x(yz) = (xy)z$	Associative laws
$x + yz = (x + y)(x + z)$ $x(y + z) = xy + xz$	Distributive laws
$\overline{(xy)} = \bar{x} + \bar{y}$ $\overline{(x + y)} = \bar{x} \bar{y}$	De Morgan's laws
$x + xy = x$ $x(x + y) = x$	Absorption laws
$x + \bar{x} = 1$	Unit property
$x\bar{x} = 0$	Zero property

Each identity can be proved using a table.

All identities in Table 5, except for the first and the last two come in pairs. Each element of the pair is the *dual* of the other (obtained by switching Boolean sums and Boolean products and 0's and 1's).

The Boolean identities correspond to the identities of propositional logic (Section 1.3) and the set identities (Section 2.2).

# Identities of Boolean Algebra

**Example:** Show that the distributive law  $x(y + z) = xy + xz$  is valid.

**Solution:** We show that both sides of this identity always take the same value by constructing this table.

TABLE 6 Verifying One of the Distributive Laws.							
$x$	$y$	$z$	$y + z$	$xy$	$xz$	$x(y + z)$	$xy + xz$
1	1	1	1	1	1	1	1
1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	1
1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0

# Formal Definition of a Boolean Algebra

**Definition:** A *Boolean algebra* is a set  $B$  with two binary operations  $\vee$  and  $\wedge$ , elements  $0$  and  $1$ , and a unary operation  $\neg$  such that for all  $x, y$ , and  $z$  in  $B$ :

$$\begin{aligned}x \vee 0 &= x \\ x \wedge 1 &= x\end{aligned}$$

*identity laws*

$$\begin{aligned}x \vee \bar{x} &= 1 \\ x \wedge \bar{x} &= 0\end{aligned}$$

*complement laws*

$$\begin{aligned}(x \vee y) \vee z &= x \vee (y \vee z) \\ (x \wedge y) \wedge z &= x \wedge (y \wedge z)\end{aligned}$$

*associative laws*

$$\begin{aligned}x \vee y &= y \vee x \\ x \wedge y &= y \wedge x\end{aligned}$$

*commutative laws*

$$\begin{aligned}x \vee (y \wedge z) &= (x \vee y) \wedge (x \vee z) \\ x \wedge (y \vee z) &= (x \wedge y) \vee (x \wedge z)\end{aligned}$$

*distributive laws*

The set of propositional variables with the operators  $\wedge$  and  $\vee$ , elements **T** and **F**, and the negation operator  $\neg$  is a Boolean algebra.

The set of subsets of a universal set with the operators  $\cup$  and  $\cap$ , the empty set ( $\emptyset$ ), universal set ( $U$ ), and the set complementation operator ( $\bar{\phantom{x}}$ ) is a Boolean algebra.

# Representing Boolean Functions

---

# Sum-of-Products Expansion

**Example:** Find Boolean expressions that represent the functions

(i)  $F(x, y, z)$  and (ii)  $G(x, y, z)$  in Table 1.

**Solution:**

(i) To represent  $F$  we need the one term  $x\bar{y}z$  because this expression has the value 1 when  $x = z = 1$  and  $y = 0$ .

(ii) To represent the function  $G$ , we use the sum  $xy\bar{z} + \bar{x}y\bar{z}$  because this expression has the value 1 when  $x = y = 1$  and  $z = 0$ , or  $x = z = 0$  and  $y = 1$ .

TABLE 1				
$x$	$y$	$z$	$F$	$G$
1	1	1	0	0
1	1	0	0	1
1	0	1	1	0
1	0	0	0	0
0	1	1	0	0
0	1	0	0	1
0	0	1	0	0
0	0	0	0	0

The general principle is that each combination of values of the variables for which the function has the value 1 requires a term in the Boolean sum that is the Boolean product of the variables or their complements.



# Sum-of-Products Expansion (*cont*)

**Definition:** A *literal* is a Boolean variable or its complement. A *minterm* of the Boolean variables  $x_1, x_2, \dots, x_n$  is a Boolean product  $y_1 y_2 \cdots y_n$ , where  $y_i = x_i$  or  $y_i = \bar{x}_i$ . Hence, a minterm is a product of  $n$  literals, with one literal for each variable.

The minterm  $y_1, y_2, \dots, y_n$  has value 1 if and only if each  $x_i$  is 1. This occurs if and only if  $x_i = 1$  when  $y_i = x_i$  and  $x_i = 0$  when  $y_i = \bar{x}_i$ .

**Definition:** The sum of minterms that represents the function is called the *sum-of-products expansion* or the *disjunctive normal form* of the Boolean function.

# Sum-of-Products Expansion (*cont*)

**Example:** Find the sum-of-products expansion for the function  $F(x,y,z) = (x + y)\bar{z}$ .

**Solution:** We use two methods, first using a table and second using Boolean identities.

(i) Form the sum of the minterms corresponding to each row of the table that has the value 1.

Including a term for each row of the table for which  $F(x,y,z) = 1$  gives us  $F(x,y,z) = xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z}$ .

TABLE 2					
$x$	$y$	$z$	$x + y$	$\bar{z}$	$(x + y)\bar{z}$
1	1	1	1	0	0
1	1	0	1	1	1
1	0	1	1	0	0
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	1	1
0	0	1	0	0	0
0	0	0	0	1	0

# Sum-of-Products Expansion (*cont*)

(ii) We now use Boolean identities to find the disjunctive normal form of  $F(x,y,z)$ :

$$\begin{aligned} F(x,y,z) &= (x + y) \bar{z} \\ &= x\bar{z} + y\bar{z} \quad \text{distributive law} \\ &= x1\bar{z} + 1y\bar{z} \quad \text{identity law} \\ &= x(y + \bar{y})\bar{z} + (x + \bar{x})y\bar{z} \quad \text{unit property} \\ &= xy\bar{z} + x\bar{y}\bar{z} + xy\bar{z} + \bar{x}y\bar{z} \quad \text{distributive law} \\ &= xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z} \quad \text{idempotent law} \end{aligned}$$

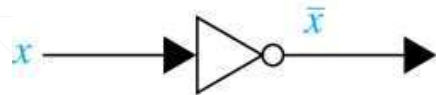
# Logic Gates

---

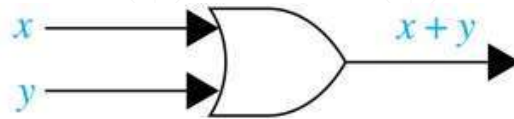
---

# Logic Gates

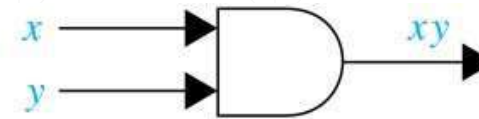
- We construct circuits using *gates*, which take as input the values of two or more Boolean variables and produce one or more bits as output, and *inverters*, which take the value of a Boolean variable as input and produce the complement of this value as output.



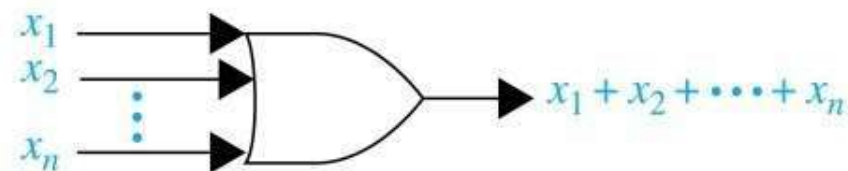
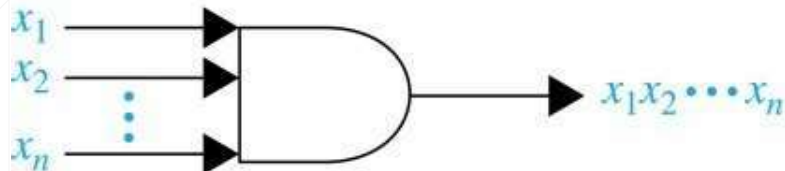
(a) Inverter



(b) OR gate

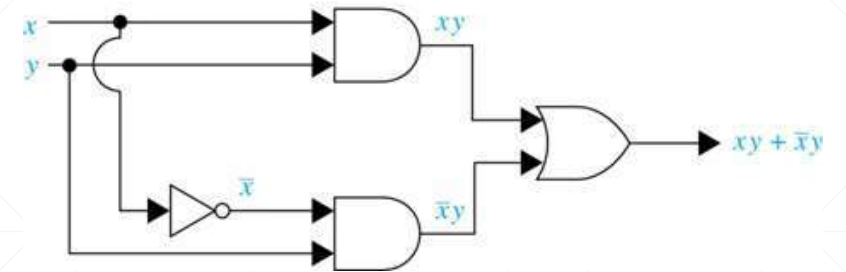
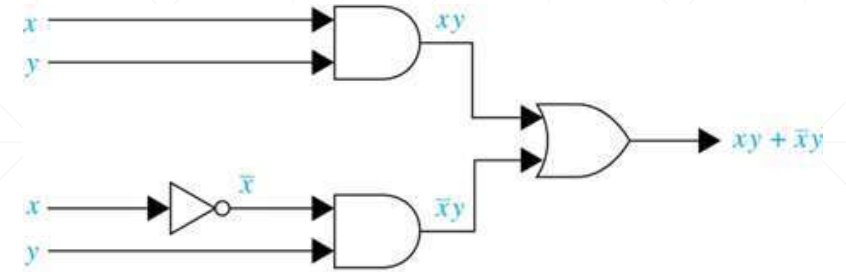


(c) AND gate



# Combinations of Gates

- Combinatorial circuits can be constructed using a combination of inverters, OR gates, and AND gates. Gates may share input and the output of one or more gates may be input to another.
- We show two ways of constructing a circuit that produces the output  $xy + \bar{x}y$ .



# Combinations of Gates

**Example:** Construct circuits that produce these outputs

(a)  $(x + y)\bar{x}$

(b)  $\bar{x} \overline{(y + z)}$

(c)  $(x + y + z)(\bar{x}\bar{y}\bar{z})$

