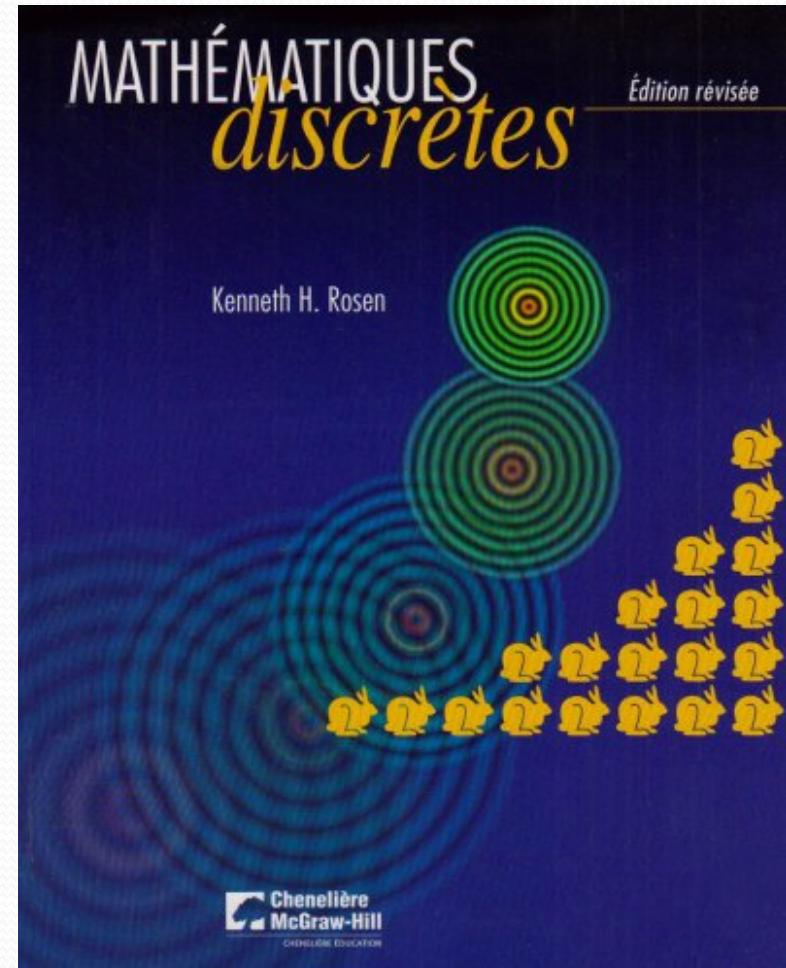
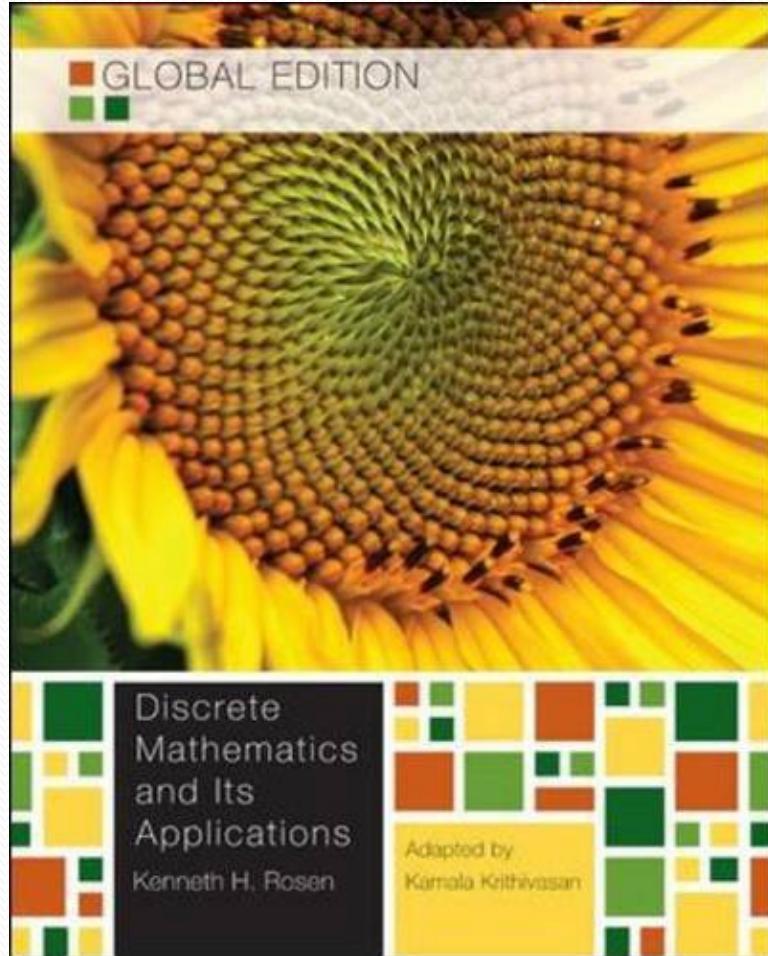


Books



Books

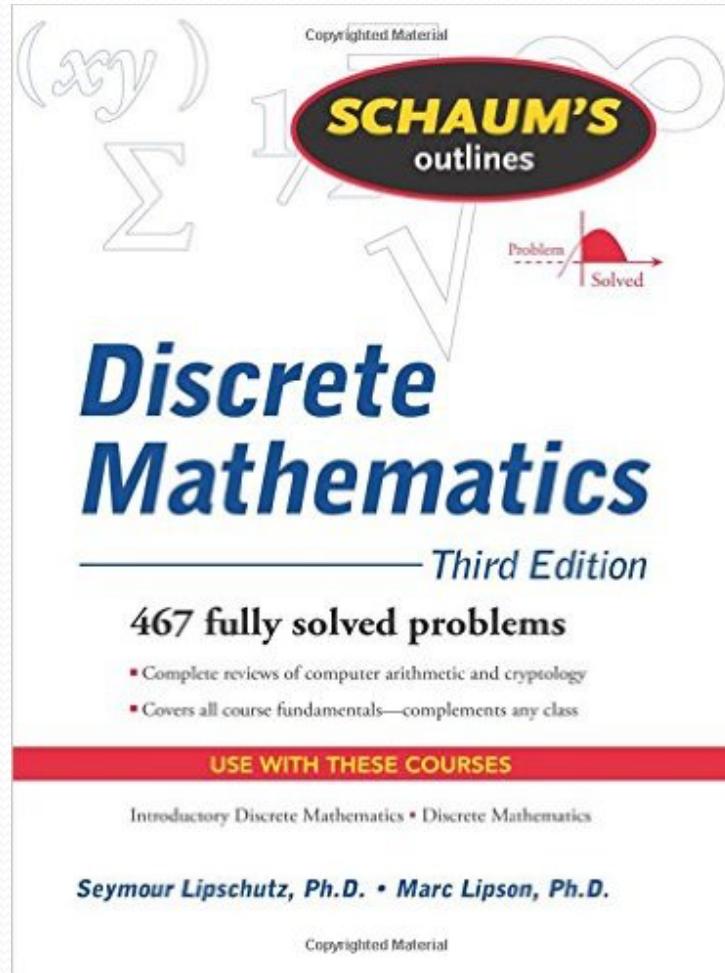


Table of contents

- *Logic
- Sets and functions
- *Counting
- *Advanced counting, recurrence relations
- Relations
- *Graphs
- Trees
- ?Dynamic programming
- ?Markov chains

The Foundations: Logic and Proofs

Chapter 1, Part I: Propositional Logic
Rosen, “Discrete mathematics and its applications”

With Question/Answer Animations

Chapter Summary

- Propositional Logic
 - The Language of Propositions
 - Applications
 - Logical Equivalences
- Predicate Logic
 - The Language of Quantifiers
 - Logical Equivalences
 - Nested Quantifiers
- Proofs
 - Rules of Inference
 - Proof Methods
 - Proof Strategy

Propositional Logic Summary

- The Language of Propositions
 - Connectives
 - Truth Values
 - Truth Tables
- Applications
 - Translating English Sentences
 - System Specifications
 - Logic Puzzles
 - Logic Circuits
- Logical Equivalences
 - Important Equivalences
 - Showing Equivalence
 - Satisfiability

Propositional Logic

Section 1.1

Section Summary

- Propositions
- Connectives
 - Negation
 - Conjunction
 - Disjunction
 - Implication; contrapositive, inverse, converse
 - Biconditional
- 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) $0 + 0 = 2$
- 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/equivalence operator \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.”

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.”

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:

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

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

- 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, conclusion)*.

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)

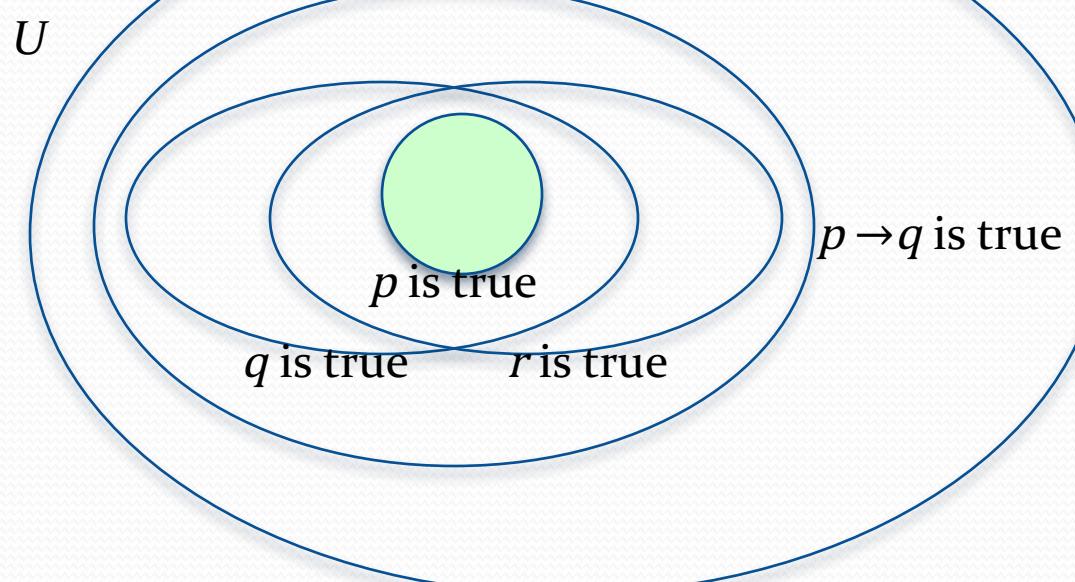
- One way to view the logical conditional is to think of an **obligation** or **contract**.
 - “If I am elected, then I will lower taxes.”
 - “If you get 100% on the final, then you will get an A.”
- If the politician is elected and does not lower taxes, then the voters can say that he or she has broken 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 when p
-
- a **necessary** condition when p is true is q
 - a **sufficient** condition for q is p

Necessary and sufficient

- $p \rightarrow q$ is true can be interpreted from set theory:
 $\{\text{universes: } p \text{ is true}\} \subseteq \{\text{universes: } q \text{ is true}\}$
 - q is necessary for p to be true (but not sufficient)
 - p is sufficient for q to be true



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

Necessary and sufficient

- For instance,
 - If the gift is a car (p) then the gift is a vehicle (q)
 - To be a vehicle (q) is **necessary** in order to be a car (p)
 - To be a car (p) is **sufficient** to be a vehicle (q)
- The gift is a vehicle (q) **if** it is a car (p)
- The gift is a car (p) **only if** it is a vehicle (q)
- \rightarrow if and only if

Converse, Contrapositive, 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 $p \rightarrow q$

Example: Find the converse, inverse, and contrapositive of “Raining is a sufficient condition for me 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.

Biconditional or equivalence

- If p and q are propositions, then we can form the *biconditional* (or *equivalence*) 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

- 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 Biconditional

- 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**

Compound Propositions

- We can build more complicated **compound propositions, logical formula or logical expressions:**
- Atomic propositions are formula
- If R is a formula then $(\neg R)$ is a formula
- If R and S are formula then $(R \vee S)$, $(R \wedge S)$, $(R \rightarrow S)$, $(R \leftrightarrow S)$ all are formula

Truth Tables For Compound Propositions

- Construction of a truth table:
 - Rows
 - Need a row for every possible combination of values for the **atomic** propositions. It enumerates all the possible **universes**.
- 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 (compound) propositions are **logically equivalent** if they always (in all universes) have the same truth value.
- **Example:** Show using a truth table that the conditional is equivalent to the contrapositive.

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 the inverse of an implication are logically 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 the same as $(p \vee q) \rightarrow \neg r$
If the intended meaning is $p \vee (q \rightarrow \neg r)$
then parentheses must be used.

Applications of Propositional Logic

Section 1.2

Applications of Propositional Logic: Summary

- Translating English to Propositional Logic
- System Specifications
- Boolean Searching
- Logic Puzzles
- Logic Circuits
- AI Diagnosis Method (Optional)

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
- “If I go to Harry’s or to the country, then I will not go for 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 .

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

Example

Problem: 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.”

One Solution: 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 set of propositions is **consistent** (cohérent) if it is possible to assign truth values to the proposition variables so that each proposition is true.

In other words, there exists at least one universe where all the propositions are true.

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.”

Solution: Let p denote “The diagnostic message is not stored in the buffer.” Let q denote “The diagnostic message is retransmitted.” The specification can be written as: $p \vee q$, $p \rightarrow q$, $\neg p$. When p is false and q is true all three statements are true. So the specification is consistent.

- What if “The diagnostic message is not retransmitted” is added.

Solution: Now we are adding $\neg q$ and there is no satisfying assignment. So the specification is not consistent.

Logic Puzzles



Raymond
Smullyan
(Born 1919)

- An island has two kinds of inhabitants, *knaves*, who always tell the truth, and *knights*, who always lie.
- You go to the island and meet A and B.
 - A says “B is a knight.”
 - B says “The two of us are of opposite types.”

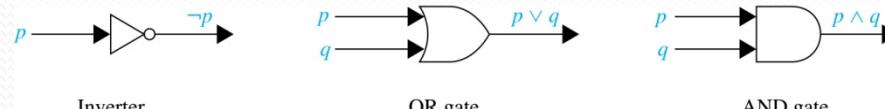
Example: What are the types of A and B?

Solution: Let p and q be the statements that “A is a knight” and “B is a knight”, respectively. So, then $\neg p$ represents the proposition that “A is a knave” and $\neg q$ that “B is a knave”.

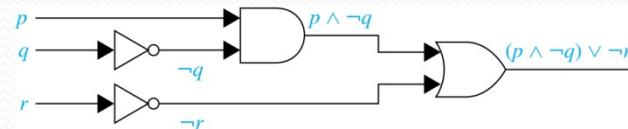
- If A is a knight, then p is true. Since knights tell the truth, q must also be true. Then $(p \wedge \neg q) \vee (\neg p \wedge q)$ would have to be true, but it is not. So, A is not a knight and therefore $\neg p$ must be true.
- If A is a knave, then B must not be a knight since knaves always lie. So, then both $\neg p$ and $\neg q$ hold since both are knaves.

Logic Circuits (Studied in depth in Chapter 12)

- Electronic circuits. Each input/output signal can be viewed as a 0 or 1.
 - 0 represents **False**
 - 1 represents **True**
- Complicated circuits are constructed from three basic circuits called gates.



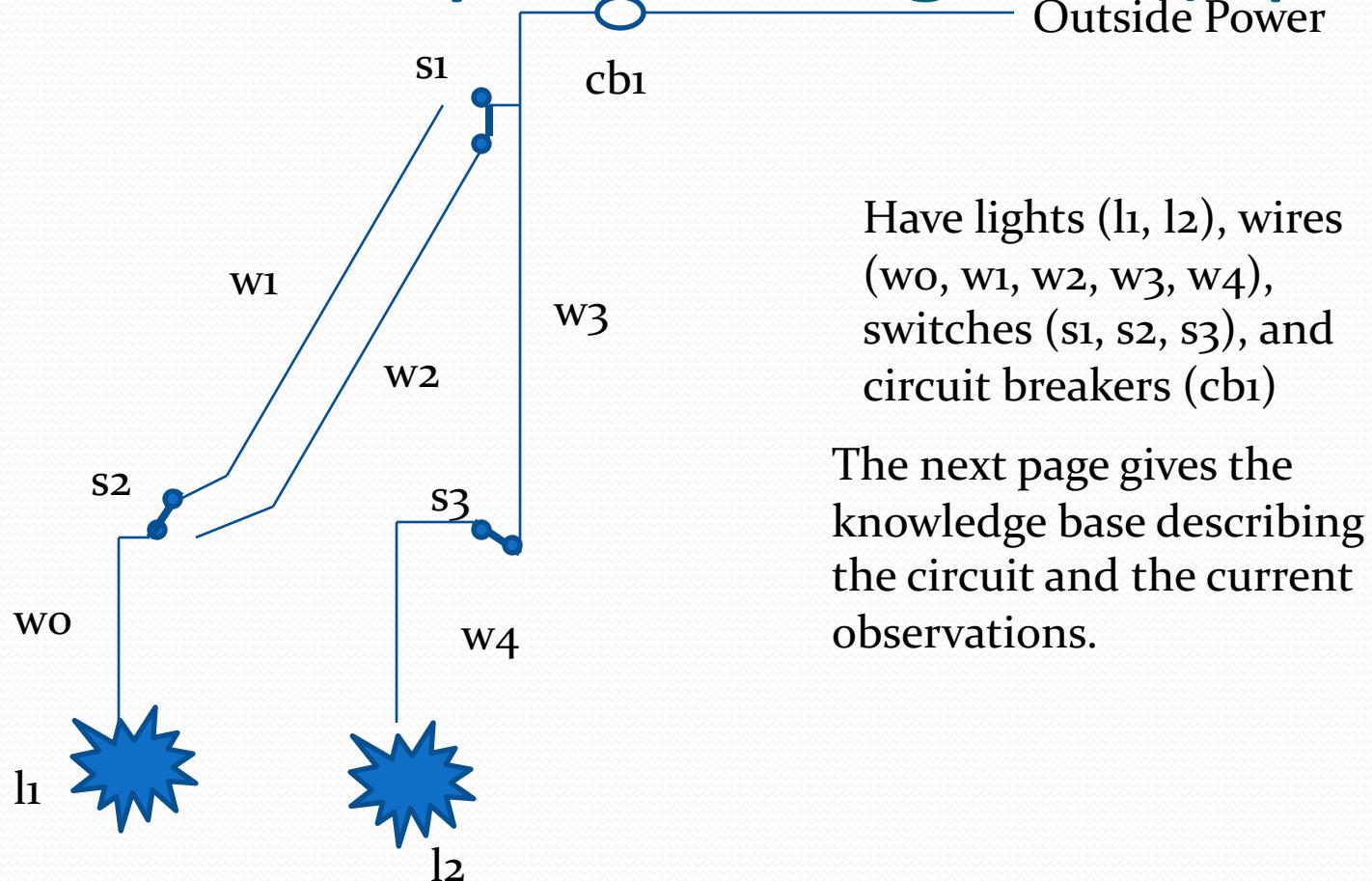
- The inverter (**NOT gate**) takes an input bit and produces the negation of that bit.
- The **OR gate** takes two input bits and produces the value equivalent to the disjunction of the two bits.
- The **AND gate** takes two input bits and produces the value equivalent to the conjunction of the two bits.
- 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:



Diagnosis of Faults in an Electrical System (*Optional*)

- AI Example (from *Artificial Intelligence: Foundations of Computational Agents* by David Poole and Alan Mackworth, 2010)
- Need to represent in propositional logic the features of a piece of machinery or circuitry that are required for the operation to produce observable features. This is called the **Knowledge Base (KB)**.
- We also have observations representing the features that the system is exhibiting now.

Electrical System Diagram (optional)



Have lights (l_1, l_2), wires (w_0, w_1, w_2, w_3, w_4), switches (s_1, s_2, s_3), and circuit breakers (cb_1)

The next page gives the knowledge base describing the circuit and the current observations.

Representing the Electrical System in Propositional Logic

- We need to represent our common-sense understanding of how the electrical system works in propositional logic.
- For example: “ l_1 is lit only when l_1 is a light and if l_1 is receiving current.”
 - $\text{lit_}l_1 \rightarrow \text{light_}l_1 \wedge \text{live_}l_1 \wedge \text{ok_}l_1$
- Also: “ w_0 has current only when w_1 has current, and switch s_2 is in the up position, and s_2 is not broken.”
 - $\text{live_}w_0 \rightarrow \text{live_}w_1 \wedge \text{up_}s_2 \wedge \text{ok_}s_2$
- This task of representing a piece of our common-sense world in logic is a common one in logic-based AI.

Knowledge Base (*opt*)

- live_outside We have outside power.
 - light_l1 Both l1 and l2 are lights.
 - light_l2
 - live_l1 → live_wo
 - live_wo → live_w1 ∧ up_s2 ∧ ok_s2
 - live_wo → live_w2 ∧ down_s2 ∧ ok_s2 ← If s2 is ok and s2 is in a down position and w2 has current, then wo has current.
 - live_w1 → live_w3 ∧ up_s1 ∧ ok_s1
 - live_w2 → live_w3 ∧ down_s1 ∧ ok_s1
 - live_l2 → live_w4
 - live_w4 → live_w3 ∧ up_s3 ∧ ok_s3
 - live_w3 → live_outside ∧ ok_cb1
 - lit_l1 → light_l1 ∧ live_l1 ∧ ok_l1
 - lit_l2 → light_l2 ∧ live_l2 ∧ ok_l2

Observations (*opt*)

- Observations or facts need to be added to the KB
 - Both Switches up
 - up_s1
 - up_s2
 - Both lights are dark
 - \neg lit_l1
 - \neg lit_l2

Diagnosis (*opt*)

- We assume that the components are working ok, unless we are forced to assume otherwise. These atoms are called *assumables*.
- The assumables (ok_cb1, ok_s1, ok_s2, ok_s3, ok_l1, ok_l2) represent the assumption that we assume that the switches, lights, and circuit breakers are ok.
- If the system is working correctly (all assumables are true), the observations and the knowledge base are consistent (i.e., satisfiable).
- The augmented knowledge base is clearly not consistent if the assumables are all true. The switches are both up, but the lights are not lit. Some of the assumables must then be false. This is the basis for the method to diagnose possible faults in the system.
- A diagnosis is a minimal set of assumables which must be false to explain the observations of the system.

Diagnostic Results (*opt*)

- See *Artificial Intelligence: Foundations of Computational Agents* (by David Poole and Alan Mackworth, 2010) for details on this problem and how the method of consistency based diagnosis can determine possible diagnoses for the electrical system.
- The approach yields 7 possible faults in the system. At least one of these must hold:
 - Circuit Breaker 1 is not ok.
 - Both Switch 1 and Switch 2 are not ok.
 - Both Switch 1 and Light 2 are not ok.
 - Both Switch 2 and Switch 3 are not ok.
 - Both Switch 2 and Light 2 are not ok.
 - Both Light 1 and Switch 3 are not ok.
 - Both Light 1 and Light 2 are not ok.

Propositional Equivalences

Section 1.3

Section Summary

- Tautologies, Contradictions, and Contingencies.
- Logical Equivalence
 - Important Logical Equivalences
 - Showing Logical Equivalence
- Normal Forms (*optional, covered in exercises in text*)
 - Disjunctive Normal Form
 - Conjunctive Normal Form
- Propositional Satisfiability
 - Sudoku Example

Tautologies, Contradictions, and Contingencies

- A **tautology** is a proposition which is always true.
 - Example: $p \vee \neg p$
- A **contradiction** is a proposition which is always false.
 - Example: $p \wedge \neg p$
- A **contingency** is a proposition which is neither a tautology nor a contradiction, such as p

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

Logically Equivalent

- Two compound propositions p and q are **logically equivalent** if and only if $p \leftrightarrow q$ is a tautology.
- We can write this as $p \Leftrightarrow q$ or as $p \equiv q$ where p and q are compound propositions.
- Two compound propositions p and q are equivalent if and only if the columns in a truth table giving their truth values agree.
- This truth table shows that $\neg p \vee q$ is logically equivalent to $p \rightarrow q$.

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

De Morgan's Laws



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

Augustus De Morgan

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

1806-1871

Distribution of the negation operator

This truth table shows that De Morgan's Second Law holds.

p	q	$\neg p$	$\neg q$	$(p \vee q)$	$\neg(p \vee q)$	$\neg p \wedge \neg q$
T	T	F	F	T	F	F
T	F	F	T	T	F	F
F	T	T	F	T	F	F
F	F	T	T	F	T	T

Key Logical Equivalences

- Identity laws: $p \wedge T \equiv p, \quad p \vee F \equiv p$
- Domination laws: $p \vee T \equiv T, \quad p \wedge F \equiv F$
- Idempotent laws: $p \vee p \equiv p, \quad p \wedge p \equiv p$
- Double Negation law: $\neg(\neg p) \equiv p$
- Negation laws: $p \vee \neg p \equiv T, \quad p \wedge \neg p \equiv F$

Key Logical Equivalences (*cont*)

- Commutative laws: $p \vee q \equiv q \vee p, \quad p \wedge q \equiv q \wedge p$
- Associative laws: $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
 $(p \vee q) \vee r \equiv p \vee (q \vee r)$
- Distributive laws: $(p \vee (q \wedge r)) \equiv (p \vee q) \wedge (p \vee r)$
 $(p \wedge (q \vee r)) \equiv (p \wedge q) \vee (p \wedge r)$
- Absorption laws: $p \vee (p \wedge q) \equiv p \quad p \wedge (p \vee q) \equiv p$

More Logical Equivalences

TABLE 7 Logical Equivalences Involving Conditional Statements.

$$p \rightarrow q \equiv \neg p \vee q$$

$$p \rightarrow q \equiv \neg q \rightarrow \neg p$$

$$p \vee q \equiv \neg p \rightarrow q$$

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

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

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

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

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

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

TABLE 8 Logical Equivalences Involving Biconditional Statements.

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

$$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$$

$$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$$

$$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$$

Constructing New Logical Equivalences

- We can show that two expressions are logically equivalent by developing a series of logically equivalent statements.
- To prove that $A \equiv B$ we produce a series of equivalences beginning with A and ending with B .

$$\begin{array}{c} A \equiv A_1 \\ \vdots \\ A_n \equiv B \end{array}$$

- Keep in mind that whenever a proposition (represented by a propositional variable) occurs in the equivalences listed earlier, it may be replaced by an arbitrarily complex compound proposition.

Equivalence Proofs

Example: Show that $\neg(p \vee (\neg p \wedge q))$
is logically equivalent to $\neg p \wedge \neg q$

Solution:

$$\begin{aligned}\neg(p \vee (\neg p \wedge q)) &\equiv \neg p \wedge \neg(\neg p \wedge q) && \text{by the second De Morgan law} \\ &\equiv \neg p \wedge [\neg(\neg p) \vee \neg q] && \text{by the first De Morgan law} \\ &\equiv \neg p \wedge (p \vee \neg q) && \text{by the double negation law} \\ &\equiv (\neg p \wedge p) \vee (\neg p \wedge \neg q) && \text{by the second distributive law} \\ &\equiv F \vee (\neg p \wedge \neg q) && \text{because } \neg p \wedge p \equiv F \\ &\equiv (\neg p \wedge \neg q) \vee F && \text{by the commutative law} \\ &\equiv (\neg p \wedge \neg q) && \text{for disjunction} \\ &\equiv \neg p \wedge \neg q && \text{by the identity law for } F\end{aligned}$$

Equivalence Proofs

Example: Show that $(p \wedge q) \rightarrow (p \vee q)$
is a tautology.

Solution:

$$\begin{aligned}(p \wedge q) \rightarrow (p \vee q) &\equiv \neg(p \wedge q) \vee (p \vee q) && \text{by } p \rightarrow q \equiv \neg p \vee q \\ &\equiv (\neg p \vee \neg q) \vee (p \vee q) && \text{by De Morgan law} \\ &\equiv (\neg p \vee p) \vee (\neg q \vee q) && \text{by associativity and} \\ &\quad && \text{commutativity of } \vee \\ &\equiv T \vee T && \neg p \vee p \text{ is a tautology} \\ &\equiv T && T \vee T \text{ is a tautology}\end{aligned}$$

Disjunctive Normal Form

- A propositional formula is in **disjunctive normal form** if it consists of a disjunction of $(1, \dots, n)$ disjuncts where each disjunct consists of a conjunction of $(1, \dots, m)$ **atomic formula** or the negation of an atomic formula.
 - Yes $(\neg p \wedge q) \vee (p \wedge \neg q)$
 - No $p \wedge (p \vee q)$
- Disjunctive Normal Form is important in many applications, such as for the circuit design methods discussed in Chapter 12.

Disjunctive Normal Form

Show that every compound proposition can be put in disjunctive normal form.

Solution: Construct the truth table for the proposition.

- Then a logically equivalent proposition is the disjunction with n disjuncts, where n is the number of rows for which the formula evaluates to T (number of universes in which the proposition is T).
- Each disjunct is composed of m conjuncts where m is the number of distinct atomic propositional variables. Each conjunct includes the positive form of the propositional variable if the variable is assigned T in that row and the negated form if the variable is assigned F in that row.

Disjunctive Normal Form

Show that every compound proposition can be put in disjunctive normal form (follow-up).

- This proposition is in disjunctive normal form because
 - By construction, each disjunct corresponds exactly to one and only one universe which has a true value (bijection) and is T in this universe. For all other universes, it is false
- Thus, the disjunctive normal form (the disjunction of all the disjuncts – a standard OR)
 - is automatically true in each universe for which the true table is true and
 - is automatically false in each universe for which the true table is false: all disjuncts are false.
- Therefore, it is **logically equivalent** to the original proposition.

Disjunctive Normal Form

Example: Find the Disjunctive Normal Form (DNF) of

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

Solution: This proposition is true when r is false or when both p and q are false.

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

Disjunctive Normal Form

- Here is an illustration

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

Conjunctive Normal Form

- A compound proposition is in **Conjunctive Normal Form** (CNF) if it is a conjunction of disjunctions.
- Every proposition can be put in an equivalent CNF.
- Conjunctive Normal Form (CNF) can be obtained by eliminating implications, moving negation inwards and using the distributive and associative laws.
- Important in resolution theorem proving used in artificial Intelligence (AI).
- A compound proposition can be put in conjunctive normal form through repeated application of the logical equivalences covered earlier.

Conjunctive Normal Form

Example: Put the following into CNF:

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

Solution:

1. Eliminate implication signs:

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

2. Move negation inwards; eliminate double negation:

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

3. Convert to CNF using associative/distributive laws

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

Propositional Satisfiability

- A compound proposition is **satisfiable** if there is an assignment of truth values to its variables that make it true.
- When no such assignments exist, the compound proposition is **unsatisfiable**.
- A compound proposition is unsatisfiable if and only if its negation is a tautology and the proposition is a contradiction.

Questions on Propositional Satisfiability

Example: Determine the satisfiability of the following compound propositions:

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

Solution: Satisfiable. Assign T to p , q , and r .

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

Solution: Satisfiable. Assign T to p and F to q .

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

Solution: Not satisfiable. Check each possible assignment of truth values to the propositional variables and none will make the proposition true.

Notation

$\bigvee_{j=1}^n p_j$ is used for $p_1 \vee p_2 \vee \dots \vee p_n$

$\bigwedge_{j=1}^n p_j$ is used for $p_1 \wedge p_2 \wedge \dots \wedge p_n$

Sudoku

- A **Sudoku puzzle** is represented by a 9×9 grid made up of nine 3×3 subgrids, known as **blocks**. Some of the 81 cells of the puzzle are assigned one of the numbers 1, 2, ..., 9.
- The puzzle is solved by assigning numbers to each blank cell so that every row, column and block contains each of the nine possible numbers.
- Example

2	9				4			
		5			1			
4								
			4	2				
6							7	
5								
7		3						5
	1		9					6

Encoding as a Satisfiability Problem

- Let $p(i,j,n)$ denote the proposition that is true when the number n is in the cell in the i th row and the j th column.
- There are $9 \times 9 \times 9 = 729$ such propositions.
- In the sample puzzle $p(5,1,6)$ is true, but $p(5,j,6)$ is false for $j = 2,3,\dots,9$

Encoding (cont)

- For each cell with a given value, assert $p(i,j,n)$, when the cell in row i and column j has the given value.
- Assert that every row contains every number.

$$\bigwedge_{i=1}^9 \bigwedge_{n=1}^9 \bigvee_{j=1}^9 p(i, j, n)$$

- Assert that every column contains every number.

$$\bigwedge_{j=1}^9 \bigwedge_{n=1}^9 \bigvee_{i=1}^9 p(i, j, n)$$

Solving Satisfiability Problems

- To solve a Sudoku puzzle, we need to find an assignment of truth values to the 729 variables of the form $p(i,j,n)$ that makes the conjunction of the assertions true. Those variables that are assigned T yield a solution to the puzzle.
- A truth table can always be used to determine the satisfiability of a compound proposition. But this is too complex even for modern computers for large problems.
- There has been much work on developing efficient methods for solving satisfiability problems as many practical problems can be translated into satisfiability problems.

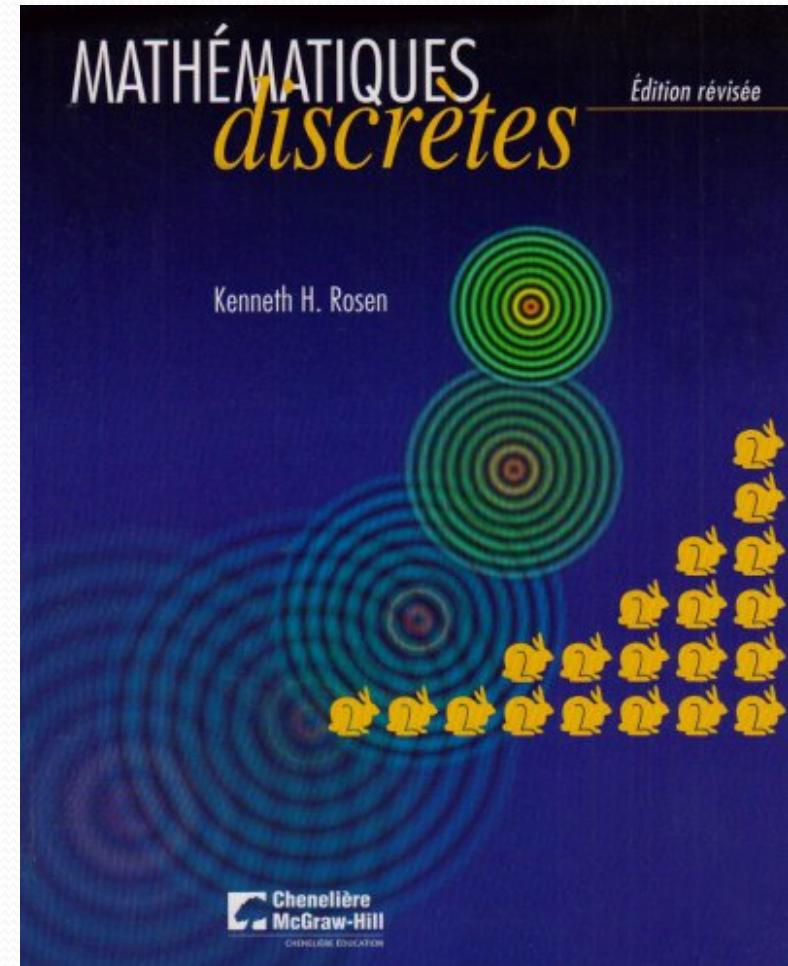
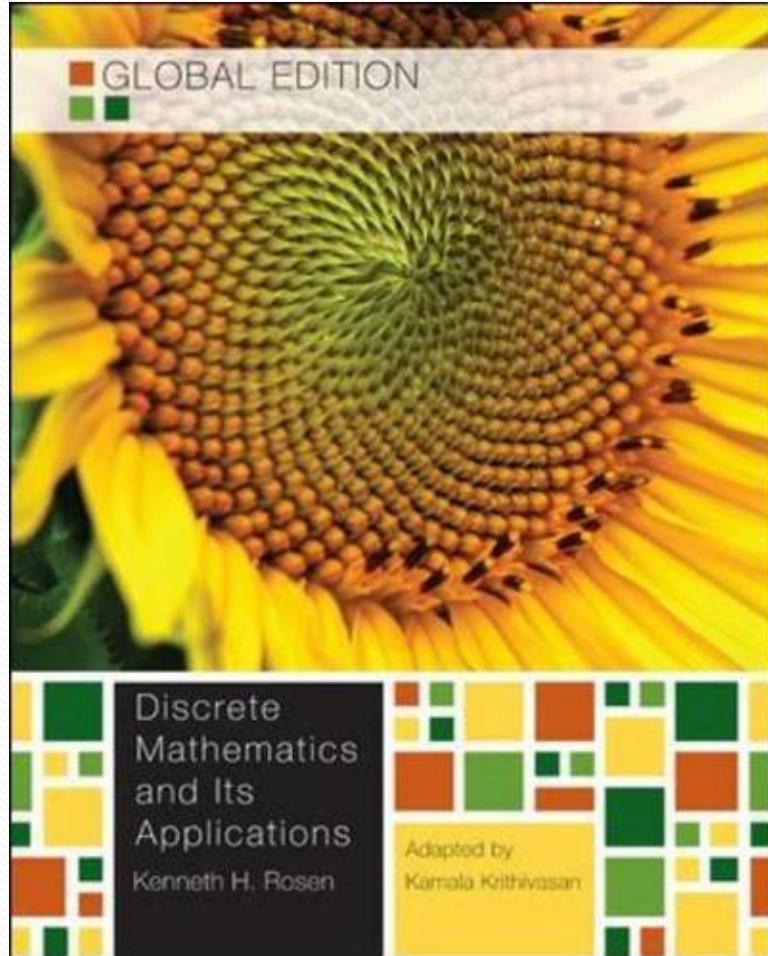
The Foundations: Logic and Proofs

Chapter 1, Part II: Predicate Logic

Rosen, “Discrete mathematics and its applications”

With Question/Answer Animations

Books



Summary

- Predicate Logic (First-Order Logic (FOL), Predicate Calculus)
 - The Language of Quantifiers
 - Logical Equivalences
 - Nested Quantifiers
 - Translation from Predicate Logic to English
 - Translation from English to Predicate Logic

Predicates and Quantifiers

Section 1.4

Section Summary

- Predicates
- Variables
- Quantifiers
 - Universal Quantifier
 - Existential Quantifier
- Negating Quantifiers
 - De Morgan's Laws for Quantifiers
- Translating English to Logic
- Logic Programming (*optional*)

Propositional Logic Not Enough

- If we have:
 - “All men are mortal.”
 - “Socrates is a man.”
- Does it follow that “Socrates is mortal?”
- Can’t be represented in propositional logic. Need a language that talks about objects, their properties, and their relations.
- Later we’ll see how to draw inferences.

Introducing Predicate Logic

- Predicate logic uses the following new features:
 - Variables: x, y, z
 - Predicates: $P(x), M(x)$
 - Quantifiers (*to be covered in a few slides*):
- *Propositional functions* are a generalization of propositions.
 - They contain variables and a predicate, e.g., $P(x)$
 - Variables can be replaced by elements from their *domain*.

Propositional Functions

- 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, as we will see later).
- The statement $P(x)$ is said to be the value of the propositional function P at x .
- For example, let $P(x)$ denote “ $x > 0$ ” and the domain be the integers. Then:
 - $P(-3)$ is false.
 - $P(0)$ is false.
 - $P(3)$ is true.
- Often the domain is denoted by U . So in this example U is the set of integers.

Examples of Propositional Functions

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

$$R(2, -1, 5)$$

Solution: F

$$R(3, 4, 7)$$

Solution: T

$$R(x, 3, z)$$

Solution: Not a Proposition

- Now let “ $x - y = z$ ” be denoted by $Q(x, y, z)$, with U as the integers. Find these truth values:

$$Q(2, -1, 3)$$

Solution: T

$$Q(3, 4, 7)$$

Solution: F

$$Q(x, 3, z)$$

Solution: Not a Proposition

Compound Expressions

- Connectives from propositional logic carry over to predicate logic.
- If $P(x)$ denotes “ $x > 0$,” find these truth values:
 - $P(3) \vee P(-1)$ **Solution:** T
 - $P(3) \wedge P(-1)$ **Solution:** F
 - $P(3) \rightarrow P(-1)$ **Solution:** F
 - $P(-1) \rightarrow P(-1)$ **Solution:** T
- Expressions with variables are not propositions and therefore do not have truth values. For example,
 - $P(3) \wedge P(y)$
 - $P(x) \rightarrow P(y)$
- When used with quantifiers (to be introduced next), these expressions (propositional functions) become propositions.



Charles Peirce (1839-1914)

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
 - We write $\forall x P(x)$ 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:

- 1) If $P(x)$ denotes “ $x > 0$ ” and U is the integers, then $\forall x P(x)$ is false.
- 2) If $P(x)$ denotes “ $x > 0$ ” and U is the positive integers, then $\forall x P(x)$ is true.
- 3) If $P(x)$ denotes “ x is even” and U is the 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:

1. 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.
2. If $P(x)$ denotes “ $x < 0$ ” and U is the positive integers, then $\exists x P(x)$ is false.
3. If $P(x)$ denotes “ x is even” and U is the integers, then $\exists x P(x)$ is true.

Uniqueness Quantifier (*optional*)

- $\exists!x P(x)$ means that $P(x)$ is true for one and only one x in the universe of discourse.
- This is commonly expressed in English in the following equivalent ways:
 - “There is a unique x such that $P(x)$.”
 - “There is one and only one x such that $P(x)$ ”
- Examples:
 1. If $P(x)$ denotes “ $x + 1 = 0$ ” and U is the integers, then $\exists!x P(x)$ is true.
 2. But if $P(x)$ denotes “ $x > 0$,” then $\exists!x P(x)$ is false.
- The uniqueness quantifier is not really needed as the restriction that there is a unique x such that $P(x)$ can be expressed as:

$$\exists x (P(x) \wedge \forall y (P(y) \rightarrow y=x))$$

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.
- Even if the domains are infinite, we can still think of the quantifiers this fashion, but the loops will not terminate in some cases.

Properties of Quantifiers

- The truth value of $\exists x P(x)$ and $\forall x P(x)$ depends on both the propositional function $P(x)$ and on the domain U .
- **Examples:**
 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 $\forall x (P(x) \vee Q(x))$.

Translating from English to Logic

Example 1: Translate the following sentence into predicate logic: “Every student in this class has taken a course in Java.”

Solution:

First decide on the domain U .

Solution 1: If U is all students in this class, define a propositional function $J(x)$ denoting “ x has taken a course in Java” and translate as $\forall x J(x)$.

Solution 2: But if U is all people, also define a propositional function $S(x)$ denoting “ x is a student in this class” and translate as $\forall x (S(x) \rightarrow J(x))$.

Translating from English to Logic

Example 2: Translate the following sentence into predicate logic: “Some student in this class has taken a course in Java.”

Solution:

First decide on the domain U .

Solution 1: If U is all students in this class, translate as

$$\exists x J(x)$$

Solution 2: But if U is all people, then translate as

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

Returning to the Socrates Example

- Introduce the propositional functions $Man(x)$ denoting “ x is a man” and $Mortal(x)$ denoting “ x is mortal.” Specify the domain as all people.
- The two premises are: $\forall x Man(x) \rightarrow Mortal(x)$
 $Man(Socrates)$
- The conclusion is: $Mortal(Socrates)$
- Later we will show how to prove that the conclusion follows from the premises.

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.
- **Example:** $\forall x \neg\neg S(x) \equiv \forall x S(x)$

Thinking about Quantifiers as Conjunctions and Disjunctions

- If the domain is finite, a universally quantified proposition is logically equivalent to a conjunction of propositions without quantifiers and an existentially quantified proposition is equivalent to a disjunction of propositions without quantifiers.
- If U consists of the integers 1, 2, and 3:

$$\forall x P(x) \equiv P(1) \wedge P(2) \wedge P(3) \equiv \bigwedge_{x \in U} P(x)$$

$$\exists x P(x) \equiv P(1) \vee P(2) \vee P(3) \equiv \bigvee_{x \in U} P(x)$$

- Even if the domains are infinite, you can still think of the quantifiers in this fashion, but the equivalent expressions without quantifiers will be infinitely long.

Negating Quantified Expressions

- 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” and
the domain is students in your class.
- Negating the original statement gives “It is not the case that every student in your class has taken Java.”
This means 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 logically equivalent

Negating Quantified Expressions

(continued)

- Now Consider $\exists x J(x)$

“There is at least 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 logically equivalent

De Morgan's Laws 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 .

- It means that:

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

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

Translation from English to Logic

Examples:

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

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

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

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

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

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

Some Fun with Translating from English into Logical Expressions

- $U = \{\text{fleegles, snurds, thingamabobs}\}$

$F(x)$: x is a fleagle

$S(x)$: x is a snurd

$T(x)$: x is a thingamabob

Translate “Everything is a fleagle”

Solution: $\forall x F(x)$

Translation (cont)

- $U = \{\text{fleegles, snurds, thingamabobs}\}$

$F(x)$: x is a fleagle

$S(x)$: x is a snurd

$T(x)$: x is a thingamabob

“Nothing is a snurd.”

Solution: $\neg \exists x S(x)$ What is this equivalent to?

Solution: $\forall x \neg S(x)$

Translation (cont)

- $U = \{\text{fleegles, snurds, thingamabobs}\}$

$F(x)$: x is a fleagle

$S(x)$: x is a snurd

$T(x)$: x is a thingamabob

“All fleegles are snurds.”

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

Translation (cont)

- $U = \{\text{fleegles, snurds, thingamabobs}\}$

$F(x)$: x is a fleagle

$S(x)$: x is a snurd

$T(x)$: x is a thingamabob

“Some fleegles are thingamabobs.”

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

Translation (cont)

- $U = \{\text{fleegles, snurds, thingamabobs}\}$

$F(x)$: x is a fleagle

$S(x)$: x is a snurd

$T(x)$: x is a thingamabob

“No snurd is a thingamabob.”

Solution: $\neg \exists x (S(x) \wedge T(x))$ What is this equivalent to?

Solution: $\forall x (\neg S(x) \vee \neg T(x))$

Translation (cont)

- $U = \{\text{fleegles, snurds, thingamabobs}\}$

$F(x)$: x is a fleagle

$S(x)$: x is a snurd

$T(x)$: x is a thingamabob

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

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

System Specification Example

- Predicate logic is used for specifying properties that systems must satisfy.
- For example, translate into predicate logic:
 - “Every mail message larger than one megabyte will be compressed.”
 - “If a user is active, then at least one network link will be available.”
- Decide on predicates and domains (left implicit here) for the variables:
 - Let $L(m, y)$ be “Mail message m is larger than y megabytes.”
 - Let $C(m)$ denote “Mail message m will be compressed.”
 - Let $A(u)$ represent “User u is active.”
 - Let $S(n, x)$ represent “Network link n is state x . ”
- Now we have:

$$\forall m(L(m, 1) \rightarrow C(m))$$

$$\exists u A(u) \rightarrow \exists n S(n, \text{available})$$

Some Predicate Calculus Definitions

- An assertion involving predicates and quantifiers is *valid* if it is true
 - for all domains
 - every propositional function substituted for the predicates in the assertion.

Example: $\forall x \neg S(x) \leftrightarrow \neg \exists x S(x)$

- An assertion involving predicates is *satisfiable* if it is true
 - for some domains
 - some propositional functions that can be substituted for the predicates in the assertion.
- Otherwise it is *unsatisfiable*.

Example: $\forall x(F(x) \leftrightarrow T(x))$ not valid but satisfiable

Example: $\forall x(F(x) \wedge \neg F(x))$ unsatisfiable

Logic Programming

- Prolog (from *Programming in Logic*) is a programming language developed in the 1970s by researchers in artificial intelligence (AI).
- Prolog programs include *Prolog facts* and *Prolog rules*.
- As an example of a set of Prolog facts consider the following:

```
instructor(chan, math273).  
instructor(patel, ee222).  
instructor(grossman, cs301).  
enrolled(kevin, math273).  
enrolled(juna, ee222).  
enrolled(juana, cs301).  
enrolled(kiko, math273).  
enrolled(kiko, cs301).
```

- Here the predicates *instructor(p,c)* and *enrolled(s,c)* represent that professor *p* is the instructor of course *c* and that student *s* is enrolled in course *c*.

Logic Programming (cont)

- In Prolog, names beginning with an uppercase letter are variables.
- If we have a predicate *teaches*(*p,s*) representing “professor *p* teaches student *s*,” we can write the rule:
teaches(*P,S*) :- *instructor*(*P,C*) , *enrolled*(*S,C*) .
- This Prolog rule can be viewed as equivalent to the following statement in logic (using our conventions for logical statements).

$$\forall p \forall c \forall s (I(p,c) \wedge E(s,c)) \rightarrow T(p,s))$$

Logic Programming (cont)

- Prolog programs are loaded into a *Prolog interpreter*. The interpreter receives *queries* and returns answers using the Prolog program.
- For example, using our program, the following query may be given:
`?enrolled(kevin,math273) .`
- Prolog produces the response:
`yes`
- Note that the `?` is the prompt given by the Prolog interpreter indicating that it is ready to receive a query.

Logic Programming (cont)

- The query:

```
?enrolled(X,math273).
```

produces the response:

```
X = kevin;  
X = kiko;  
no
```

- The query:

```
?teaches(X,juana).
```

produces the response:

```
X = patel;  
X = grossman;  
no
```

The Prolog interpreter tries to find an instantiation for X. It does so and returns `X = kevin.` Then the user types the ; indicating a request for another answer. When Prolog is unable to find another answer it returns no.

Logic Programming (cont)

- The query:

```
?teaches(chan,X).
```

produces the response:

```
X = kevin;  
X = kiko;  
no
```

- A number of very good Prolog texts are available. *Learn Prolog Now!* is one such text with a free online version at <http://www.learnprolognow.org/>
- There is much more to Prolog and to the entire field of logic programming.

L'histoire du bar

Six collègues de travail entrent dans un bar pour boire un verre. Toutes les tables sont prises et il n'y a plus que six tabourets au comptoir. Ils vont pouvoir s'asseoir mais souhaiteraient optimiser leurs discussions pour que chacun ait des choses à dire à son voisin.

Ils ne veulent en aucun cas parler “boulot” et choisissent de partir sur leurs passions respectives pour trouver la meilleure disposition.

Prénom	Passion(s)
Mathieu	Foot et Jeux vidéos
Etienne	Escalade, Jeux vidéos et Matrices
Alex	Oiseaux et Livres
Nico	Foot
Bertrand	Matrices
Marco	Livres et Matrices

Concepts PROLOG

Les faits

Une situation qui est vrai pour l'univers en question. Peut être apparenté à une proposition.

Ex: Mathieu a comme passion le foot

sera exprimé en PROLOG

passion(mathieu, foot).

Les règles (relations)

Exprime une question à laquelle l'interpréteur peut répondre à l'aide de faits.

Ex: 2 pers. peuvent discuter si elles ont une passion en commun

sera exprimé en PROLOG

discussion(A, B) :- passion(A, P), passion(B, P).

Les faits

```
passion(mathieu, foot). passion(mathieu, jeux).
passion(nico, foot).
passion(alex, oiseaux). passion(alex, livres).
passion(marco, matrices). passion(marco, livres).
passion(bertrand, matrices).
passion(etienne, jeux). passion(etienne, escalade). passion(etienne, matrices).
```

Les règles

```
discussion(A, B) :- passion(A, P), passion(B, P), not(A=B).
```

```
arrangement(A,B,C,D,E,F) :-
```

```
    discussion(A,B), discussion(B,C), discussion(C,D), discussion(D,E), discussion(E,F),
    not(A=B), not(A=C), not(A=D), not(A=E), not(A=F),
    not(B=C), not(B=D), not(B=E), not(B=F),
    not(C=D), not(C=E), not(C=F),
    not(D=E), not(D=F),
    not(E=F).
```

Nested Quantifiers

Section 1.5

Section Summary

- Nested Quantifiers
- Order of Quantifiers
- Translating from Nested Quantifiers into English
- Translating Mathematical Statements into Statements involving Nested Quantifiers.
- Translated English Sentences into Logical Expressions.
- Negating Nested Quantifiers.

Nested Quantifiers

- Nested quantifiers are often necessary to express the meaning of sentences in English as well as important concepts in computer science and mathematics.

Example: “Every real number has an inverse” is

$$\forall x \exists y (x + y = 0)$$

where the domains of x and y are the real numbers.

- We can also think of nested propositional functions:
 $\forall x \exists y (x + y = 0)$ can be viewed as $\forall x Q(x)$ where $Q(x)$ is
 $\exists y P(x, y)$ where $P(x, y)$ is $(x + y = 0)$

Thinking of Nested Quantification

- Nested Loops
 - To see if $\forall x \forall y P(x,y)$ is true, loop through the values of x :
 - At each step, loop through the values for y .
 - If for some pair of x and y , $P(x,y)$ is false, then $\forall x \forall y P(x,y)$ is false and both the outer and inner loop terminate.
 - $\forall x \forall y P(x,y)$ is true if the outer loop ends after stepping through each x .
- To see if $\forall x \exists y P(x,y)$ is true, loop through the values of x :
 - At each step, loop through the values for y .
 - The inner loop ends when a pair x and y is found such that $P(x,y)$ is true.
 - If no y is found such that $P(x,y)$ is true the outer loop terminates as $\forall x \exists y P(x,y)$ has been shown to be false.
- $\forall x \exists y P(x,y)$ is true if the outer loop ends after stepping through each x .
- If the domains of the variables are infinite, then this process can not actually be carried out.

Order of Quantifiers

Examples:

1. Let $P(x,y)$ be the statement “ $x + y = y + x$.” Assume that U is the real numbers. Then $\forall x \forall y P(x,y)$ and $\forall y \forall x P(x,y)$ have the same truth value.
1. Let $Q(x,y)$ be the statement “ $x + y = 0$.” Assume that U is the real numbers. Then $\forall x \exists y P(x,y)$ is true, but $\exists y \forall x P(x,y)$ is false.

Questions on Order of Quantifiers

Example 1: Let U be the real numbers,

Define $P(x,y) : x \cdot y = 0$

What is the truth value of the following:

1. $\forall x \forall y P(x,y)$

Answer: False

2. $\forall x \exists y P(x,y)$

Answer: True

3. $\exists x \forall y P(x,y)$

Answer: True

4. $\exists x \exists y P(x,y)$

Answer: True

Questions on Order of Quantifiers

Example 2: Let U be the real numbers,

Define $P(x,y) : x / y = 1$

What is the truth value of the following:

1. $\forall x \forall y P(x,y)$

Answer: False

2. $\forall x \exists y P(x,y)$

Answer: True

3. $\exists x \forall y P(x,y)$

Answer: False

4. $\exists x \exists y P(x,y)$

Answer: True

Quantifications of Two Variables

Statement	When True?	When False
$\forall x \forall y P(x, y)$	$P(x, y)$ is true for every pair x, y .	There is a pair x, y for which $P(x, y)$ is false.
$\forall y \forall x P(x, y)$		
$\forall x \exists y P(x, y)$	For every x there is a y for which $P(x, y)$ is true.	There is an x such that $P(x, y)$ is false for every y .
$\exists x \forall y P(x, y)$	There is an x for which $P(x, y)$ is true for every y .	For every x there is a y for which $P(x, y)$ is false.
$\exists x \exists y P(x, y)$	There is a pair x, y for which $P(x, y)$ is true.	$P(x, y)$ is false for every pair x, y
$\exists y \exists x P(x, y)$		

Translating Nested Quantifiers into English

Example 1: Translate the statement

$$\forall x (C(x) \vee \exists y (C(y) \wedge F(x, y)))$$

where $C(x)$ is “ x has a computer,” and $F(x, y)$ is “ x and y are friends,” and the domain for both x and y consists of all students in your school.

Solution: Every student in your school has a computer or has a friend who has a computer.

Example 2: Translate the statement

$$\exists x \forall y \forall z ((F(x, y) \wedge F(x, z)) \wedge (y \neq z)) \rightarrow \neg F(y, z))$$

Solution: There is a student for which none of whose friends are also friends with each other.

Translating Mathematical Statements into Predicate Logic

Example : Translate “The sum of two positive integers is always positive” into a logical expression.

Solution:

1. Rewrite the statement to make the implied quantifiers and domains explicit:

“For every two integers, if these integers are both positive, then the sum of these integers is positive.”

2. Introduce the variables x and y , and specify the domain, to obtain:

“For all positive integers x and y , $x + y$ is positive.”

3. The result is:

$$\forall x \forall y ((x > 0) \wedge (y > 0) \rightarrow (x + y > 0))$$

where the domain of both variables consists of all integers

Questions on Translation from English

Choose the obvious predicates and express in predicate logic.

Example 1: “Brothers are siblings.”

Solution: $\forall x \forall y (B(x,y) \rightarrow S(x,y))$

Example 2: “Siblinghood is symmetric.”

Solution: $\forall x \forall y (S(x,y) \rightarrow S(y,x))$

Example 3: “Everybody loves somebody.”

Solution: $\forall x \exists y L(x,y)$

Example 4: “There is someone who is loved by everyone.”

Solution: $\exists y \forall x L(x,y)$

Example 5: “There is someone who loves someone.”

Solution: $\exists x \exists y L(x,y)$

Example 6: “Everyone loves himself”

Solution: $\forall x L(x,x)$

The Foundations: Logic and Proofs

Chapter 1, Part III: Proofs

With Question/Answer Animations

Summary

- Valid Arguments and Rules of Inference
- Proof Methods
- Proof Strategies

Rules of Inference

Section 1.6

Section Summary

- Valid Arguments
- Inference Rules for Propositional Logic
- Using Rules of Inference to Build Arguments
- Rules of Inference for Quantified Statements
- Building Arguments for Quantified Statements

Revisiting the Socrates Example

- We have the two premises:
 - “All men are mortal.”
 - “Socrates is a man.”
- And the conclusion:
 - “Socrates is mortal.”
- How do we get the conclusion from the premises?

The Argument

- We can express the **premises** or **hypotheses** (above the line) and the **conclusion** (below the line) in predicate logic as an argument:

$$\forall x(Man(x) \rightarrow Mortal(x))$$

$$Man(Socrates)$$

$$\therefore Mortal(Socrates)$$

- We will see shortly that this is a valid argument.

Valid Arguments

- We will show how to construct valid arguments in two stages; first for propositional logic and then for predicate logic. The rules of inference are the essential building block in the construction of valid arguments.
 1. Propositional Logic
Inference Rules
 2. Predicate Logic
Inference rules for propositional logic plus additional inference rules to handle variables and quantifiers.

Rules of Inference for Propositional Logic

- An argument is **valid** if for all universes where its premises (or hypotheses) are true, its conclusion is also true.
 - The hypotheses therefore are **filtering** the set of universes: only universes where all the hypotheses are true are considered.
- Definition of a **counter-example**. A counter-example of an argument is a universe where the premises of the argument are true and the conclusion is false.
- An argument is valid if and only if it does not have any counter-example.

Rules of Inference for Propositional Logic

- From the definition of the implication, a valid argument can be expressed by the fact that the following logical expression is a **tautology**.
 - let $p_1, p_2, p_3, \dots, p_m$, be the premises
 - let q be the conclusion
 - then
$$(p_1 \wedge p_2 \wedge p_3 \wedge \dots \wedge p_m) \rightarrow q$$
 - should be a tautology for a valid argument
- In fact, it is a tautology if and only if it is a valid argument.
- We say that $\{p_1, p_2, p_3, \dots, p_m\}$ **logically implies** q .
- This corresponds to proving a theorem.

Rules of Inference for Propositional Logic: Modus Ponens

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

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.”

Modus Tollens

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

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.”

Hypothetical Syllogism

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

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.”

Disjunctive Syllogism

$$\begin{array}{c} p \vee q \\ \neg p \\ \hline \therefore q \end{array}$$

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.”

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.”

Simplification

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

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

Example:

Let p be “I will study discrete math.”

Let q be “I will study English literature.”

“I will study discrete math and English literature”

“Therefore, I will study discrete math.”

Conjunction

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

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.”

Resolution

Resolution plays an important role in AI and is used in Prolog.

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

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.”

Proof by contradiction

- It can be shown that $(\neg p \rightarrow F) \rightarrow p$, where F is the contradiction, is a tautology

p	$\neg p$	F	$\neg p \rightarrow F$	$(\neg p \rightarrow F) \rightarrow p$
T	F	F	T	T
F	T	F	F	T

- So that, instead of proving p , one can prove that $(\neg p \rightarrow F)$ is true instead
 - This could be useful when $(\neg p \rightarrow F)$ is easier to prove than p

Proof by contradiction

- More generally, if we have premises , $p_1, p_2, p_3, \dots, p_m$
- And we want to prove $(p_1 \wedge p_2 \wedge p_3 \wedge \dots \wedge p_m) \rightarrow q$, where q is the conclusion,
- We can prove instead $(p_1 \wedge p_2 \wedge p_3 \wedge \dots \wedge p_m \wedge \neg q) \rightarrow F$
- Indeed, it can easily be shown that
 - $(p \rightarrow q) \leftrightarrow [(p \wedge \neg q) \rightarrow F]$
 - where p could be set to $p = (p_1 \wedge p_2 \wedge p_3 \wedge \dots \wedge p_m)$
- This is called **proof by contradiction** or **reductio ad absurdum**

Proof by contradiction

- Indeed,

p	q	F	$p \wedge \neg q$	$(p \wedge \neg q) \rightarrow F$	$p \rightarrow q$	$(p \rightarrow q) \leftrightarrow [(p \wedge \neg q) \rightarrow F]$
F	F	F	F	T	T	T
F	T	F	F	T	T	T
T	F	F	T	F	F	T
T	T	F	F	T	T	T

Using the Rules of Inference to Build Valid Arguments

- A *valid argument* is thus 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:

$$S_1$$
$$S_2$$
$$\cdot$$
$$\cdot$$
$$\cdot$$
$$S_n$$
$$\therefore C$$

Valid Arguments

Example 1: From the single premise

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

Show that q is a conclusion.

Solution:

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

Valid Arguments

Example 2:

- With these 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.”

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

Continued on next slide →

Valid Arguments

3. 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)

Handling Quantified Statements

- Valid arguments for quantified statements are a sequence of statements.
- Each statement is either a premise or follows from previous statements by rules of inference which include:
 - Rules of Inference for Propositional Logic
 - Rules of Inference for Quantified Statements
- The rules of inference for quantified statements are introduced in the next several slides.

Universal Instantiation (UI)

$$\frac{\forall x P(x)}{\therefore P(c)}$$

Example:

Our domain consists of all dogs and Fido is a dog.

“All dogs are cuddly.”

“Therefore, Fido is cuddly.”

Universal Generalization (UG)

$$\frac{P(c) \text{ for an arbitrary } c}{\therefore \forall x P(x)}$$

Used often implicitly in Mathematical Proofs.

Existential Instantiation (EI)

$$\frac{\exists x P(x)}{\therefore P(c) \text{ for some element } c}$$

Example:

“There is someone who got an A in the course.”

“Let’s call her a and say that a got an A”

Existential Generalization (EG)

$$\frac{P(c) \text{ for some element } c}{\therefore \exists x P(x)}$$

Example:

“Michelle got an A in the class.”

“Therefore, someone got an A in the class.”

Using Rules of Inference

Example 1: Using the rules of inference, construct a valid argument to show that

“John Smith has two legs”

is a consequence of the premises:

“Every man has two legs.” “John Smith is a man.”

Solution: Let $M(x)$ denote “ x is a man” and $L(x)$ “ x has two legs” and let John Smith be a member of the domain.

Valid Argument:

Step	Reason
1. $\forall x(M(x) \rightarrow L(x))$	Premise
2. $M(J) \rightarrow L(J)$	UI from (1)
3. $M(J)$	Premise
4. $L(J)$	Modus Ponens using (2) and (3)

Using Rules of Inference

Example 2: Use the rules of inference to construct a valid argument showing that the conclusion

“Someone who passed the first exam has not read the book.”
follows from the premises

“A student in this class has not read the book.”

“Everyone in this class passed the first exam.”

Solution: Let $C(x)$ denote “ x is in this class,” $B(x)$ denote “ x has read the book,” and $P(x)$ denote “ x passed the first exam.”

First we translate the premises and conclusion into symbolic form.

$$\frac{\exists x(C(x) \wedge \neg B(x))}{\forall x(C(x) \rightarrow P(x))} \\ \therefore \exists x(P(x) \wedge \neg B(x))$$

Continued on next slide →

Using Rules of Inference

Valid Argument:

Step

1. $\exists x(C(x) \wedge \neg B(x))$
2. $C(a) \wedge \neg B(a)$
3. $C(a)$
4. $\forall x(C(x) \rightarrow P(x))$
5. $C(a) \rightarrow P(a)$
6. $P(a)$
7. $\neg B(a)$
8. $P(a) \wedge \neg B(a)$
9. $\exists x(P(x) \wedge \neg B(x))$

Reason

- Premise
EI from (1)
Simplification from (2)
Premise
UI from (4)
MP from (3) and (5)
Simplification from (2)
Conj from (6) and (7)
EG from (8)

Returning to the Socrates Example

$$\forall x(Man(x) \rightarrow Mortal(x))$$

Man(Socrates)

∴ Mortal(Socrates)

Solution for Socrates Example

Valid Argument

Step

1. $\forall x(Man(x) \rightarrow Mortal(x))$

2. $Man(Socrates) \rightarrow Mortal(Socrates)$

3. $Man(Socrates)$

4. $Mortal(Socrates)$

Reason

Premise

UI from (4)

Premise

MP from (2)
and (3)

Universal Modus Ponens

Universal Modus Ponens combines universal instantiation and modus ponens into one rule.

$$\forall x(P(x) \rightarrow Q(x))$$

$P(a)$, where a is a particular
element in the domain

$$\therefore Q(a)$$

This rule could be used in the Socrates example.

Normal Forms

Section 1.7

With Question/Answer Animations

Section Summary

- Disjunctive Normal Form
- Conjunctive Normal Form
- Principal Disjunctive Normal Form
- Principal Conjunctive Normal Form
- Prenex Normal Form
(Normal form for first order logic)

Disjunctive Normal Form (cont)

- A propositional formula is in *disjunctive normal form* if it consists of a disjunction of ($1, \dots, n$) disjuncts where each disjunct consists of a conjunction of ($1, \dots, m$) atomic formulas or the negation of an atomic formula. Are the following expression in DNF?
 - $(p \wedge \neg q) \vee (\neg p \vee q)$ No
 - $p \wedge (p \vee q)$ No
 - $(p \wedge \neg q) \vee (\neg p \wedge q)$ Yes
- Disjunctive Normal Form is important for circuit design methods.

Disjunctive Normal Form (cont)

Example: Show that every compound proposition can be put in disjunctive normal form.

Solution: Construct the truth table for the proposition. Then an equivalent proposition is the disjunction with n disjuncts (where n is the number of rows for which the formula evaluates to T).

Each disjunct has m conjuncts where m is the number of distinct propositional variables. Each conjunct includes the positive form of the propositional variable if the variable is assigned T in that row and the negated form if the variable is assigned F in that row.

This proposition is in disjunctive normal form.

Disjunctive Normal Form (cont)

Example: Find the Disjunctive Normal Form (DNF) of

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

Solution: This proposition is true when r is false or when both p and q are false.

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

Conjunctive Normal Form (cont)

- A compound proposition is in *Conjunctive Normal Form* (CNF) if it is a conjunction of disjunctions.
- Every proposition can be put in an equivalent CNF.
- Conjunctive Normal Form (CNF) can be obtained by eliminating implications, moving negation inwards and using the distributive and associative laws.
- Important in resolution theorem proving used in artificial Intelligence (AI).
- A compound proposition can be put in conjunctive normal form through repeated application of the logical equivalences covered earlier.

Conjunctive Normal Form (cont)

Example: Put the following into CNF:

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

Solution:

1. Eliminate implication signs:

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

2. Move negation inwards; eliminate double negation:

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

3. Convert to CNF using associative/distributive laws

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

Principal Disjunctive Normal Form

Let p and q be propositional variables. Consider the four conjunctions given below.

$$p \wedge q, p \wedge \neg q, \neg p \wedge q, \neg p \wedge \neg q$$

represent conjunctions in which p or $\neg p$ appears as also q or $\neg q$. Each variable occurs either negated or non negated, but both negated and non negated forms of a variable do not occur together in the conjunction. Also $p \wedge q$ and $q \wedge p$ are treated as the same. These four conjunctions are called **minterms** of p and q .

In general if there are n variables, there will be 2^n minterms. **Each minterm is a conjunction in which each variable occurs once either in the negated form or in the non negated form.**

Definition

For a given formula, an equivalent formula consisting of disjunctions of minterms only is known as its **principal disjunctive normal form**. Such a normal form is also called sum-of-products canonical form.

Consider the following truth table

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

Consider the truth table for $p \rightarrow q$ and $p \leftrightarrow q$

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

The principal disjunctive normal form for $p \rightarrow q$ will be $(p \wedge q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)$. This disjunction corresponds to the disjunction of these minterms having T in the respective rows. Similarly $p \leftrightarrow q$ has the following principal disjunctive normal form $(p \wedge q) \vee (\neg p \wedge \neg q)$.

In order to obtain the principal disjunctive normal form of a given formula without constructing its truth table, one may first replace implication (\rightarrow) and equivalence (\leftrightarrow) by using \wedge , \vee , \neg .

Then De Morgan's laws are used wherever necessary and distributive laws are also used in bringing to disjunctive normal form. An elementary product which is a contradiction is dropped.

Minterms are obtained in the disjunctions by introducing the missing factors. Duplications are avoided.

Example:

Obtain principal disjunctive normal form for $p \vee \neg q$

Solution:

$$\begin{aligned} p \vee \neg q &= [p \wedge (q \vee \neg q)] \vee [\neg q \wedge (p \vee \neg p)] \\ &= (p \wedge q) \vee (p \wedge \neg q) \vee (\neg q \wedge p) \vee (\neg q \wedge \neg p) \\ &= (p \wedge q) \vee (p \wedge \neg q) \vee (\neg p \wedge \neg q) \end{aligned}$$

Example: Obtain principal disjunctive normal form for
 $(p \wedge q) \vee (\neg p \wedge r) \vee (q \wedge r)$

Solution:

$$\begin{aligned}(p \wedge q) &= (p \wedge q) \wedge (r \vee \neg r) \\&= (p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r)\end{aligned}$$

$$\begin{aligned}(\neg p \wedge r) &= (\neg p \wedge r) \wedge (q \vee \neg q) \\&= (\neg p \wedge r \wedge q) \vee (\neg p \wedge r \wedge \neg q) \\&= (\neg p \wedge q \wedge r) \vee (\neg p \wedge \neg q \wedge r)\end{aligned}$$

$$\begin{aligned}(q \wedge r) &= (q \wedge r) \wedge (p \vee \neg p) \\&= (q \wedge r \wedge p) \vee (q \wedge r \wedge \neg p) \\&= (p \wedge q \wedge r) \vee (\neg p \wedge q \wedge r)\end{aligned}$$

Avoiding duplication, the given expression is equivalent to
 $(p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (\neg p \wedge q \wedge r) \vee (\neg p \wedge \neg q \wedge r)$

Principal Conjunctive Normal Form

We define maxterm as a dual to minterm. For a given number of variables, the maxterm consists of disjunctions in which each variable or its negation, but not both, appears only once. It can be seen that each of the maxterms has the truth value F for exactly one combination of the truth values of the variables. This is illustrated for two variables below:

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

Definition

For a given formula, an equivalent formula consisting of conjunction of maxterms only is known as its principal conjunctive normal form. This normal form is also called the product of sums canonical form.

Example:

Find principal conjunctive normal form for $(p \leftrightarrow q)$

Solution:

$$\begin{aligned} p \leftrightarrow q &= (p \rightarrow q) \wedge (q \rightarrow p) \\ &= (\neg p \vee q) \wedge (\neg q \vee p) \end{aligned}$$

Example: Find principal conjunctive normal form for
 $[(p \vee q) \wedge \neg p \rightarrow \neg q]$

Solution:

$$\begin{aligned} [(p \vee q) \wedge \neg p \rightarrow \neg q] &= [(p \wedge \neg p) \vee (q \wedge \neg p)] \rightarrow \neg q \\ &= (q \wedge \neg p) \rightarrow \neg q \\ &= \neg(q \wedge \neg p) \vee \neg q \\ &= \neg q \vee \neg \neg p \vee \neg q \\ &= \neg q \vee p \\ &= p \vee \neg q \end{aligned}$$

Definition

A formula F in the first order logic is said to be in a prenex normal form if and only if the formula F is in the form of $(Q_1x_1)\dots(Q_nx_n)(M)$ where every $(Q_i x_i)$, $i = 1, \dots, n$ is either $(\forall x_i)$ or $(\exists x_i)$, and M is a formula containing no quantifiers.

$(Q_1x_1)\dots(Q_nx_n)$ is called the **prefix** and M is called the **matrix** of the formula F .

Example

$$(\forall x)(\forall y)(P(x, y) \wedge Q(y))$$

$\forall x \exists y \forall z(Q(x, y) \rightarrow R(z))$ are in prenex normal form.

Let us now see how to convert a given formula in first order logic to prenex normal form.

We denote two formulas F_1, F_2 as equivalent by $F_1 \leftrightarrow F_2$ if and only if the truth values of F and G are the same under every interpretations.

We know that

$$\neg \forall x P(x) \leftrightarrow \exists x \neg P(x) \quad (1)$$

$$\neg \exists x P(x) \leftrightarrow \forall x \neg P(x) \quad (2)$$

Also \forall distributes over \wedge and \exists over \vee .

\forall does not distribute over \vee and \exists over \wedge .

Also if F has a variable x and G does not contain x , then

$$(Qx)F(x) \vee G \leftrightarrow Q(x)(F(x) \vee G) \quad (3)$$

$$(Qx)F(x) \wedge G \leftrightarrow Q(x)(F(x) \wedge G) \quad (4)$$

We see that if F_1 and F_2 have variable x ,

$$(\forall x F_1(x)) \vee (\forall x F_2(x)) \neq \forall x (F_1(x) \vee F_2(x))$$

But in $F_2(x)$ we can rename the variable x as z and get $\forall x F_1(x) \vee \forall z F_2(z)$ which can be brought to the form $\forall x \forall z (F_1(x) \vee F_2(z))$. Note that F_2 does not contain x and F_1 does not contain z .

Similarly, $\exists x F_1(x) \wedge \exists x F_2(x)$ can be brought to the following form by renaming of variable x as z in F_2 .

$$\begin{aligned} & \exists x F_1(x) \wedge \exists x F_2(x) \\ &= \exists x F_1(x) \wedge \exists z F_2(z) \\ &= \exists x \exists z (F_1(x) \wedge F_2(z)) \end{aligned}$$

Hence it is possible to bring the quantifiers to the left of the formula. The following steps are carried out to bring a formula of first order logic to prenex normal form:

Step 1: Replace \leftrightarrow and \rightarrow using \wedge , \vee , \neg

Step 2: Use double negation and De Morgan's laws repeatedly and the laws (1) and (2).

Step 3: Rename variables if necessary

Step 4: Use rules (3) and (4) to bring the quantifiers to the left.

Example:

Transform the formula into prenex normal form:

$$\forall x P(x) \rightarrow \exists x Q(x)$$

Solution:

$$\forall x P(x) \rightarrow \exists x Q(x)$$

$$\neg \forall x P(x) \vee \exists x Q(x)$$

$$\exists x (\neg P(x)) \vee \exists x Q(x)$$

$$\exists x (\neg P(x)) \vee \exists x Q(x)$$

$$\exists x (\neg P(x) \vee Q(x))$$

Example: Obtain prenex normal form for the formula

$$(\forall x)(\forall y)((\exists z)(P(x, z) \wedge P(y, z)) \rightarrow (\exists u)Q(x, y, u))$$

Solution:

$$(\forall x)(\forall y)(\neg(\exists z)(P(x, z) \wedge P(y, z)) \vee (\exists u)Q(x, y, u))$$

$$(\forall x)(\forall y)((\forall z)\neg(P(x, z) \wedge P(y, z)) \vee (\exists u)Q(x, y, u))$$

$$(\forall x)(\forall y)((\forall z)(\neg P(x, z) \vee \neg P(y, z)) \vee (\exists u)Q(x, y, u))$$

$$(\forall x)(\forall y)(\forall z)(\exists u)((\neg P(x, z) \vee \neg P(y, z)) \vee Q(x, y, u))$$

Proofs: more advanced material

Section 1.8

Section Summary

- Mathematical Proofs
- Forms of Theorems
- Direct Proofs
- Indirect Proofs
 - Proof of the Contrapositive
 - Proof by Contradiction

Proofs of Mathematical Statements

- A *proof* is a valid argument that establishes the truth of a statement.
- In math, CS, and other disciplines, informal proofs which are generally shorter, are generally used.
 - More than one rule of inference are often used in a step.
 - Steps may be skipped.
 - The rules of inference used are not explicitly stated.
 - Easier for to understand and to explain to people.
 - But it is also easier to introduce errors.
- Proofs have many practical applications:
 - verification that computer programs are correct
 - establishing that operating systems are secure
 - enabling programs to make inferences in artificial intelligence
 - showing that system specifications are consistent

Definitions

- A *theorem* is a statement that can be shown to be true using:
 - definitions
 - other theorems
 - *axioms* (statements which are given as true)
 - rules of inference
- A *lemma* is a ‘helping theorem’ or a result which is needed to prove a theorem.
- A *corollary* is a result which follows directly from a theorem.
- Less important theorems are sometimes called *propositions*.
- A *conjecture* is a statement that is being proposed to be true. Once a proof of a conjecture is found, it becomes a theorem. It may turn out to be false.

Forms of Theorems

- Many theorems assert that a property holds for all elements in a domain, such as the integers, the real numbers, or some of the discrete structures that we will study in this class.
- Often the universal quantifier (needed for a precise statement of a theorem) is omitted by standard mathematical convention.

For example, the statement:

“If $x > y$, where x and y are positive real numbers, then $x^2 > y^2$ ”
really means

“For all positive real numbers x and y , if $x > y$, then $x^2 > y^2$.”

Proving Theorems

- Many theorems have the form:

$$\forall x(P(x) \rightarrow Q(x))$$

- To prove them, we show that where c is an arbitrary element of the domain, $P(c) \rightarrow Q(c)$
- By universal generalization the truth of the original formula follows.
- So, we must prove something of the form: $p \rightarrow q$

Proving Conditional Statements: $p \rightarrow q$

- *Trivial Proof:* If we know q is true, then $p \rightarrow q$ is true as well.

“If it is raining then $1=1$.”

- *Vacuous Proof:* If we know p is false then $p \rightarrow q$ is true as well.

“If I am both rich and poor then $2 + 2 = 5$.”

[Even though these examples seem silly, both trivial and vacuous proofs are often used in mathematical induction, as we will see in Chapter 5)]

Even and Odd Integers

Definition: The integer n is even if there exists an integer k such that $n = 2k$, and n is odd if there exists an integer k , such that $n = 2k + 1$. Note that every integer is either even or odd and no integer is both even and odd.

We will need this basic fact about the integers in some of the example proofs to follow. We will learn more about the integers in Chapter 4.

Proving Conditional Statements: $p \rightarrow q$

- *Direct Proof:* Assume that p is true. Use rules of inference, axioms, and logical equivalences to show that q must also be true.

Example: Give a direct proof of the theorem “If n is an odd integer, then n^2 is odd.”

Solution: Assume that n is odd. Then $n = 2k + 1$ for an integer k . Squaring both sides of the equation, we get:

$$n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1 = 2r + 1,$$

where $r = 2k^2 + 2k$, an integer.

We have proved that if n is an odd integer, then n^2 is an odd integer. ◀

(◀ marks the end of the proof. Sometimes QED is used instead.)

Proving Conditional Statements: $p \rightarrow q$

Definition: The real number r is *rational* if there exist integers p and q where $q \neq 0$ such that $r = p/q$

Example: Prove that the sum of two rational numbers is rational.

Solution: Assume r and s are two rational numbers. Then there must be integers p, q and also t, u such that

$$r = p/q, \quad s = t/u, \quad u \neq 0, \quad q \neq 0$$
$$r + s = \frac{p}{q} + \frac{t}{u} = \frac{pu + qt}{qu} = \frac{v}{w} \quad \text{where } v = pu + qt \\ w = qu \neq 0$$

Thus the sum is rational. 

Proving Conditional Statements: $p \rightarrow q$

- *Proof by Contraposition:* Assume $\neg q$ and show $\neg p$ is true also. This is sometimes called an *indirect proof* method. If we give a direct proof of $\neg q \rightarrow \neg p$ then we have a proof of $p \rightarrow q$.
Why does this work?

Example: Prove that if n is an integer and $3n + 2$ is odd, then n is odd.

Solution: Assume n is even. So, $n = 2k$ for some integer k . Thus

$$3n + 2 = 3(2k) + 2 = 6k + 2 = 2(3k + 1) = 2j \text{ for } j = 3k + 1$$

Therefore $3n + 2$ is even. Since we have shown $\neg q \rightarrow \neg p$, $p \rightarrow q$ must hold as well. If n is an integer and $3n + 2$ is odd (not even), then n is odd (not even). ◀

Proving Conditional Statements: $p \rightarrow q$

Example: Prove that for an integer n , if n^2 is odd, then n is odd.

Solution: Use proof by contraposition. Assume n is even (i.e., not odd). Therefore, there exists an integer k such that $n = 2k$. Hence,

$$n^2 = 4k^2 = 2(2k^2)$$

and n^2 is even(i.e., not odd).

We have shown that if n is an even integer, then n^2 is even. Therefore by contraposition, for an integer n , if n^2 is odd, then n is odd. 

Proving Conditional Statements: $p \rightarrow q$

- *Proof by Contradiction:* (AKA *reductio ad absurdum*).

To prove p , assume $\neg p$ and derive a contradiction such as $p \wedge \neg p$. (an indirect form of proof). Since we have shown that $\neg p \rightarrow F$ is true , it follows that the contrapositive $T \rightarrow p$ also holds.

Example: Prove that if you pick 22 days from the calendar, at least 4 must fall on the same day of the week.

Solution: Assume that no more than 3 of the 22 days fall on the same day of the week. Because there are 7 days of the week, we could only have picked 21 days. This contradicts the assumption that we have picked 22 days. ◀

Proof by Contradiction

- A preview of Chapter 4.

Example: Use a proof by contradiction to give a proof that $\sqrt{2}$ is irrational.

Solution: Suppose $\sqrt{2}$ is rational. Then there exists integers a and b with $\sqrt{2} = a/b$, where $b \neq 0$ and a and b have no common factors (see Chapter 4). Then

$$2 = \frac{a^2}{b^2} \quad 2b^2 = a^2$$

Therefore a^2 must be even. If a^2 is even then a must be even (an exercise). Since a is even, $a = 2c$ for some integer c . Thus,

$$2b^2 = 4c^2 \quad b^2 = 2c^2$$

Therefore b^2 is even. Again then b must be even as well.

But then 2 must divide both a and b . This contradicts our assumption that a and b have no common factors. We have proved by contradiction that our initial assumption must be false and therefore $\sqrt{2}$ is irrational .



Proof by Contradiction

- A preview of Chapter 4.

Example: Prove that there is no largest prime number.

Solution: Assume that there is a largest prime number. Call it p_n . Hence, we can list all the primes $2, 3, \dots, p_n$. Form $r = p_1 \times p_2 \times \dots \times p_n + 1$

None of the prime numbers on the list divides r . Therefore, by a theorem in Chapter 4, either r is prime or there is a smaller prime that divides r . This contradicts the assumption that there is a largest prime. Therefore, there is no largest prime. ◀

Theorems that are Biconditional Statements

- To prove a theorem that is a biconditional statement, that is, a statement of the form $p \leftrightarrow q$, we show that $p \rightarrow q$ and $q \rightarrow p$ are both true.

Example: Prove the theorem: “If n is an integer, then n is odd if and only if n^2 is odd.”

Solution: We have already shown (previous slides) that both $p \rightarrow q$ and $q \rightarrow p$. Therefore we can conclude $p \leftrightarrow q$.

Sometimes *iff* is used as an abbreviation for “if and only if,” as in “If n is an integer, then n is odd iff n^2 is odd.”

What is wrong with this?

“Proof” that $1 = 2$

Step

1. $a = b$
2. $a^2 = a \times b$
3. $a^2 - b^2 = a \times b - b^2$
4. $(a - b)(a + b) = b(a - b)$
5. $a + b = b$
6. $2b = b$
7. $2 = 1$

Reason

- Premise
Multiply both sides of (1) by a
Subtract b^2 from both sides of (2)
Algebra on (3)
Divide both sides by $a - b$
Replace a by b in (5) because $a = b$
Divide both sides of (6) by b

Solution: Step 5. $a - b = 0$ by the premise and division by 0 is undefined.

Looking Ahead

- If direct methods of proof do not work:
 - We may need a clever use of a proof by contraposition.
 - Or a proof by contradiction.
 - In the next section, we will see strategies that can be used when straightforward approaches do not work.
 - In Chapter 5, we will see mathematical induction and related techniques.
 - In Chapter 6, we will see combinatorial proofs

Proof Methods and Strategy

Section 1.9

Section Summary

- Proof by Cases
- Existence Proofs
 - Constructive
 - Nonconstructive
- Disproof by Counterexample
- Nonexistence Proofs
- Uniqueness Proofs
- Proof Strategies
- Proving Universally Quantified Assertions
- Open Problems

Proof by Cases

- To prove a conditional statement of the form:

$$(p_1 \vee p_2 \vee \dots \vee p_n) \rightarrow q$$

- Use the tautology

$$\begin{aligned} [(p_1 \vee p_2 \vee \dots \vee p_n) \rightarrow q] &\leftrightarrow \\ [(p_1 \rightarrow q) \wedge (p_2 \rightarrow q) \wedge \dots \wedge (p_n \rightarrow q)] \end{aligned}$$

- Each of the implications $p_i \rightarrow q$ is a *case*.

Proof by Cases

Example: Let $a @ b = \max\{a, b\} = a$ if $a \geq b$, otherwise $a @ b = \max\{a, b\} = b$.

Show that for all real numbers a, b, c

$$(a @ b) @ c = a @ (b @ c)$$

(This means the operation @ is associative.)

Proof: Let a, b , and c be arbitrary real numbers.

Then one of the following 6 cases must hold.

1. $a \geq b \geq c$
2. $a \geq c \geq b$
3. $b \geq a \geq c$
4. $b \geq c \geq a$
5. $c \geq a \geq b$
6. $c \geq b \geq a$

Continued on next slide →

Proof by Cases

Case 1: $a \geq b \geq c$

$(a @ b) = a$, $a @ c = a$, $b @ c = b$

Hence $(a @ b) @ c = a = a @ (b @ c)$

Therefore the equality holds for the first case.

A complete proof requires that the equality be shown to hold for all 6 cases. But the proofs of the remaining cases are similar. Try them.



Without Loss of Generality

Example: Show that if x and y are integers and both $x \cdot y$ and $x+y$ are even, then both x and y are even.

Proof: Use a proof by contraposition. Suppose x and y are not both even. Then, one or both are odd. Without loss of generality, assume that x is odd. Then $x = 2m + 1$ for some integer k .

Case 1: y is even. Then $y = 2n$ for some integer n , so
 $x + y = (2m + 1) + 2n = 2(m + n) + 1$ is odd.

Case 2: y is odd. Then $y = 2n + 1$ for some integer n , so
 $x \cdot y = (2m + 1)(2n + 1) = 2(2m \cdot n + m + n) + 1$ is odd.



We only cover the case where x is odd because the case where y is odd is similar. The use phrase *without loss of generality* (WLOG) indicates this.



Srinivasa Ramanujan
(1887-1920)

Existence Proofs

- Proof of theorems of the form $\exists x P(x)$.
- **Constructive** existence proof:
 - Find an explicit value of c , for which $P(c)$ is true.
 - Then $\exists x P(x)$ is true by Existential Generalization (EG).

Example: Show that there is a positive integer that can be written as the sum of cubes of positive integers in two different ways:

Proof: 1729 is such a number since

$$1729 = 10^3 + 9^3 = 12^3 + 1^3 \quad \blacktriangleleft$$



Godfrey Harold Hardy
(1877-1947)

Nonconstructive Existence Proofs

- In a *nonconstructive* existence proof, we assume no c exists which makes $P(c)$ true and derive a contradiction.

Example: Show that there exist irrational numbers x and y such that x^y is rational.

Proof: We know that $\sqrt{2}$ is irrational. Consider the number $\sqrt{2}^{\sqrt{2}}$. If it is rational, we have two irrational numbers x and y with x^y rational, namely $x = \sqrt{2}$ and $y = \sqrt{2}$. But if $\sqrt{2}^{\sqrt{2}}$ is irrational, then we can let $x = \sqrt{2}^{\sqrt{2}}$ and $y = \sqrt{2}$ so that $x^y = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{(\sqrt{2}\sqrt{2})} = \sqrt{2}^2 = 2$. ◀

Counterexamples

- Recall $\exists x \neg P(x) \equiv \neg \forall x P(x)$.
- To establish that $\neg \forall x P(x)$ is true (or $\forall x P(x)$ is false) find a c such that $\neg P(c)$ is true or $P(c)$ is false.
- In this case c is called a *counterexample* to the assertion $\forall x P(x)$.

Example: “Every positive integer is the sum of the squares of 3 integers.” The integer 7 is a counterexample. So the claim is false.

Uniqueness Proofs

- Some theorems assert the existence of a unique element with a particular property, $\exists!x P(x)$. The two parts of a *uniqueness proof* are
 - *Existence*: We show that an element x with the property exists.
 - *Uniqueness*: We show that if $y \neq x$, then y does not have the property.

Example: Show that if a and b are real numbers and $a \neq 0$, then there is a unique real number r such that $ar + b = 0$.

Solution:

- Existence: The real number $r = -b/a$ is a solution of $ar + b = 0$ because $a(-b/a) + b = -b + b = 0$.
- Uniqueness: Suppose that s is a real number such that $as + b = 0$. Then $ar + b = as + b$, where $r = -b/a$. Subtracting b from both sides and dividing by a shows that $r = s$. 

Proof Strategies for proving $p \rightarrow q$

- Choose a method.
 1. First try a direct method of proof.
 2. If this does not work, try an indirect method (e.g., try to prove the contrapositive).
- For whichever method you are trying, choose a strategy.
 1. First try *forward reasoning*. Start with the axioms and known theorems and construct a sequence of steps that end in the conclusion. Start with p and prove q , or start with $\neg q$ and prove $\neg p$.
 2. If this doesn't work, try *backward reasoning*. When trying to prove q , find a statement p that we can prove with the property $p \rightarrow q$.

Backward Reasoning

Example: Suppose that two people play a game taking turns removing, 1, 2, or 3 stones at a time from a pile that begins with 15 stones. The person who removes the last stone wins the game. Show that the first player can win the game no matter what the second player does.

Proof: Let n be the last step of the game.

Step n: Player₁ can win if the pile contains 1, 2, or 3 stones.

Step n-1: Player₂ will have to leave such a pile if the pile that he/she is faced with has 4 stones.

Step n-2: Player₁ can leave 4 stones when there are 5, 6, or 7 stones left at the beginning of his/her turn.

Step n-3: Player₂ must leave such a pile, if there are 8 stones .

Step n-4: Player₁ has to have a pile with 9, 10, or 11 stones to ensure that there are 8 left.

Step n-5: Player₂ needs to be faced with 12 stones to be forced to leave 9, 10, or 11.

Step n-6: Player₁ can leave 12 stones by removing 3 stones.

Now reasoning forward, the first player can ensure a win by removing 3 stones and leaving 12.

Universally Quantified Assertions

- To prove theorems of the form $\forall x P(x)$, assume x is an arbitrary member of the domain and show that $P(x)$ must be true. Using UG it follows that $\forall x P(x)$.

Example: An integer x is even if and only if x^2 is even.

Solution: The quantified assertion is

$$\forall x [x \text{ is even} \leftrightarrow x^2 \text{ is even}]$$

We assume x is arbitrary.

Recall that $p \leftrightarrow q$ is equivalent to $(p \rightarrow q) \wedge (q \rightarrow p)$

So, we have two cases to consider. These are considered in turn.

Continued on next slide →

Universally Quantified Assertions

Case 1. We show that if x is even then x^2 is even using a direct proof (the *only if* part or *necessity*).

If x is even then $x = 2k$ for some integer k .

Hence $x^2 = 4k^2 = 2(2k^2)$ which is even since it is an integer divisible by 2.

This completes the proof of case 1.

Case 2 on next slide →

Universally Quantified Assertions

Case 2. We show that if x^2 is even then x must be even (the *if* part or *sufficiency*). We use a proof by contraposition.

Assume x is not even and then show that x^2 is not even.

If x is not even then it must be odd. So, $x = 2k + 1$ for some k . Then $x^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$ which is odd and hence not even. This completes the proof of case 2.

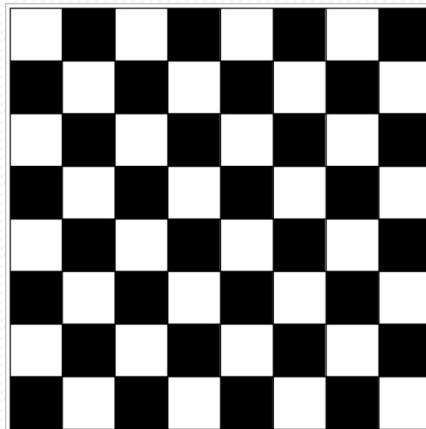
Since x was arbitrary, the result follows by UG.

Therefore we have shown that x is even if and only if x^2 is even. ◀

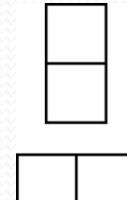
Proof and Disproof: Tilings

Example 1: Can we tile the standard checkerboard using dominos?

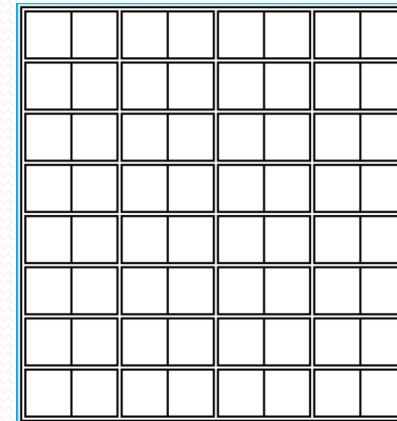
Solution: Yes! One example provides a constructive existence proof.



The Standard Checkerboard



Two Dominoes



One Possible Solution

Tilings

Example 2: Can we tile a checkerboard obtained by removing one of the four corner squares of a standard checkerboard?

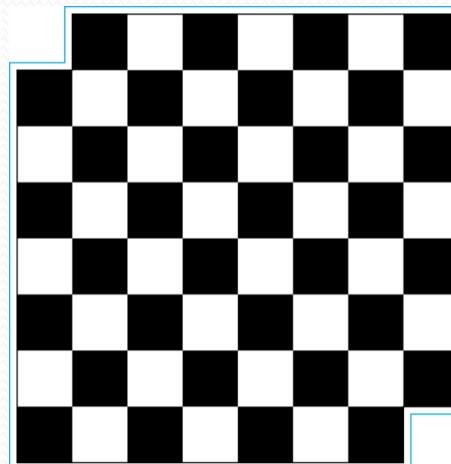
Solution:

- Our checkerboard has $64 - 1 = 63$ squares.
- Since each domino has two squares, a board with a tiling must have an even number of squares.
- The number 63 is not even.
- We have a contradiction.

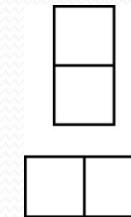


Tilings

Example 3: Can we tile a board obtained by removing both the upper left and the lower right squares of a standard checkerboard?



Nonstandard Checkerboard



Dominoes

Continued on next slide →

Tilings

Solution:

- There are 62 squares in this board.
- To tile it we need 31 dominos.
- *Key fact:* Each domino covers one black and one white square.
- Therefore the tiling covers 31 black squares and 31 white squares.
- Our board has either 30 black squares and 32 white squares or 32 black squares and 30 white squares.
- Contradiction!



The Role of Open Problems

- Unsolved problems have motivated much work in mathematics. Fermat's Last Theorem was conjectured more than 300 years ago. It has only recently been finally solved.

Fermat's Last Theorem: The equation $x^n + y^n = z^n$ has no solutions in integers x , y , and z , with $xyz \neq 0$ whenever n is an integer with $n > 2$.

A proof was found by Andrew Wiles in the 1990s.

An Open Problem

- **The $3x + 1$ Conjecture:** Let T be the transformation that sends an even integer x to $x/2$ and an odd integer x to $3x + 1$. For all positive integers x , when we repeatedly apply the transformation T , we will eventually reach the integer 1.

For example, starting with $x = 13$:

$$T(13) = 3 \cdot 13 + 1 = 40, T(40) = 40/2 = 20, T(20) = 20/2 = 10,$$

$$T(10) = 10/2 = 5, T(5) = 3 \cdot 5 + 1 = 16, T(16) = 16/2 = 8,$$

$$T(8) = 8/2 = 4, T(4) = 4/2 = 2, T(2) = 2/2 = 1$$

The conjecture has been verified using computers up to $5.6 \cdot 10^{13}$.

Additional Proof Methods

- Later we will see many other proof methods:
 - Mathematical induction, which is a useful method for proving statements of the form $\forall n P(n)$, where the domain consists of all positive integers.
 - Structural induction, which can be used to prove such results about recursively defined sets.
 - Cantor diagonalization is used to prove results about the size of infinite sets.
 - Combinatorial proofs use counting arguments.

Basic Structures: Sets, Functions, Sequences, Sums, and Matrices

Chapter 2

With Question/Answer Animations

Chapter Summary

- Sets
 - The Language of Sets
 - Set Operations
 - Set Identities
- Functions
 - Types of Functions
 - Operations on Functions
 - Computability
- Sequences and Summations
 - Types of Sequences
 - Summation Formulae
- Set Cardinality
 - Countable Sets
- Matrices
 - Matrix Arithmetic

Sets

Section 2.1

Section Summary

- Definition of sets
- Describing Sets
 - Roster Method
 - Set-Builder Notation
- Some Important Sets in Mathematics
- Empty Set and Universal Set
- Subsets and Set Equality
- Cardinality of Sets
- Tuples
- Cartesian Product

Introduction

- Sets are one of the basic building blocks for the types of objects considered in discrete mathematics.
 - Important for counting.
 - Programming languages have set operations.
- Set theory is an important branch of mathematics.
 - Many different systems of axioms have been used to develop set theory.
 - Here we are not concerned with a formal set of axioms for set theory. Instead, we will use what is called naïve set theory.

Sets

- 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 $a \notin A$

Describing a Set: 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.

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

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 strictly 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

N = *natural numbers* = {0,1,2,3,...}

Z = *integers* = {...,-3,-2,-1,0,1,2,3,...}

Z⁺ = *positive integers* = {1,2,3,.....}

R = set of *real numbers*

R⁺ = set of *positive real numbers*

C = set of *complex numbers*.

Q = set of rational numbers

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

- Positive rational numbers:

$$\mathbf{Q}^+ = \{x \in \mathbf{R} \mid x = p/q, \text{ for some positive integers } p, q\}$$

Interval Notation

$$[a,b] = \{x \mid a \leq x \leq b\}$$

$$[a,b) = \{x \mid a \leq x < b\}$$

$$(a,b] = \{x \mid a < x \leq b\}$$

$$(a,b) = \{x \mid a < x < b\}$$

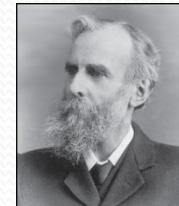
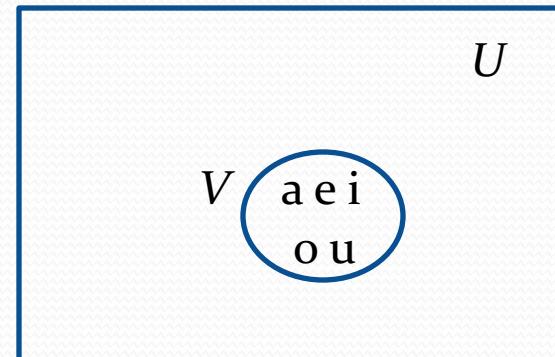
closed interval $[a,b]$

open interval (a,b) or $]a,b[$

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



John Venn (1834-1923)
Cambridge, UK

Russell's Paradox

- Let S be the set of all sets which are not members of themselves. A paradox results from trying to answer the question “Is S a member of itself?”
- Related Paradox:
 - Henry is a barber who shaves all people who do not shave themselves. A paradox results from trying to answer the question “Does Henry shave himself?”



Bertrand Russell (1872-1970)
Cambridge, UK
Nobel Prize Winner

Some things to remember

- Sets can be elements of sets.

$$\{\{1,2,3\}, a, \{b,c\}\}$$

$$\{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$$

- The empty set is different from a set containing the empty set.

$$\emptyset \neq \{ \emptyset \}$$

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.
 1. Because $a \in \emptyset$ is always false, $\emptyset \subseteq S$, for every set S .
 2. Because $a \in S \rightarrow a \in S$, $S \subseteq S$, 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.

Another look at Equality of Sets

- Recall that two sets A and B are *equal*, denoted by

$A = B$, iff

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

- Using logical equivalences we have that $A = B$ iff

$$\forall x[(x \in A \rightarrow x \in B) \wedge (x \in B \rightarrow x \in A)]$$

- This is equivalent to

$$A \subseteq B \quad \text{and} \quad B \subseteq A$$

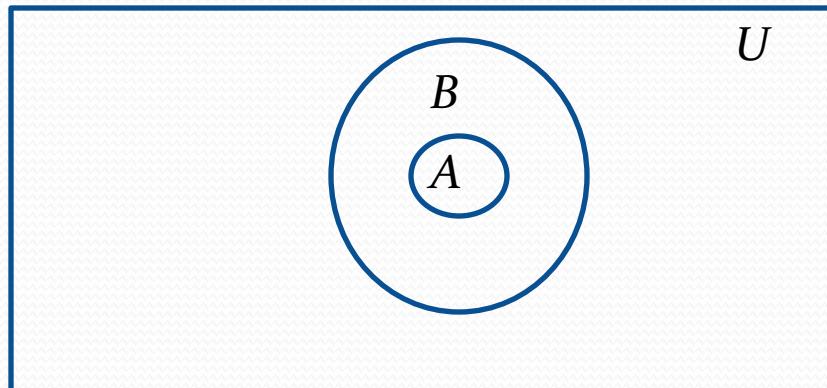
Proper Subsets

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 .

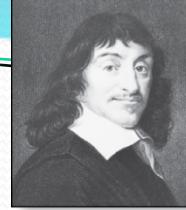
Example: If $A = \{a,b\}$ then

$$\mathcal{P}(A) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$$

- If a set has n elements, then the cardinality of the power set is 2^n . (In Chapters 5 and 6, we will discuss different ways to show this.)

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 $b = d$.



René Descartes
(1596-1650)

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:

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

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

- **Definition:** A subset R of the Cartesian product $A \times B$ is called a *relation* from the set A to the set B . (Relations will be covered in depth in Chapter 9.)

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

Truth Sets of Quantifiers

- Given a predicate P and a domain D , we define the *truth set* of P to be the set of elements in D for which $P(x)$ is true. The truth set of $P(x)$ is denoted by

$$\{x \in D | P(x)\}$$

- Example:** The truth set of $P(x)$ where the domain is the integers and $P(x)$ is “ $|x| = 1$ ” is the set $\{-1, 1\}$

Set Operations

Section 2.2

Section Summary

- Set Operations
 - Union
 - Intersection
 - Complementation
 - Difference
- More on Set Cardinality
- Set Identities
- Proving Identities
- Membership Tables

Boolean Algebra

- Propositional calculus and set theory are both instances of an algebraic system called a *Boolean Algebra*. This is discussed in Chapter 12.
- The operators in set theory are analogous to the corresponding operator in propositional calculus.
- As always there must be a universal set U . All sets are assumed to be subsets of U .

Union

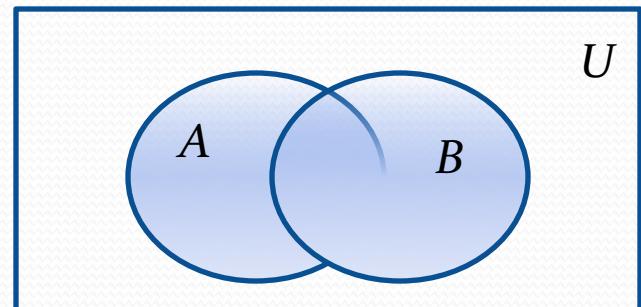
- **Definition:** Let A and B be sets. The *union* of the sets A and B , denoted by $A \cup B$, is the set:

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

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

Solution: $\{1,2,3,4,5\}$

Venn Diagram for $A \cup B$



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? $\{1,2,3\} \cap \{3,4,5\}$?

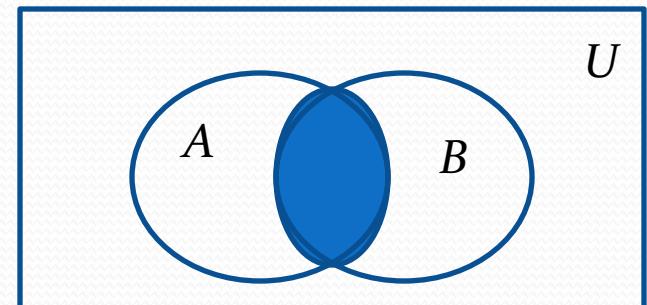
Solution: $\{3\}$

- **Example:** What is?

$$\{1,2,3\} \cap \{4,5,6\} ?$$

Solution: \emptyset

Venn Diagram for $A \cap B$



Complement

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

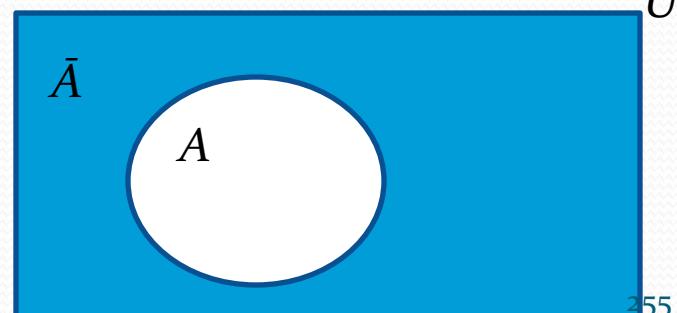
$$\bar{A} = \{x \in U \mid x \notin A\}$$

(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: $\{x \mid x \leq 70\}$

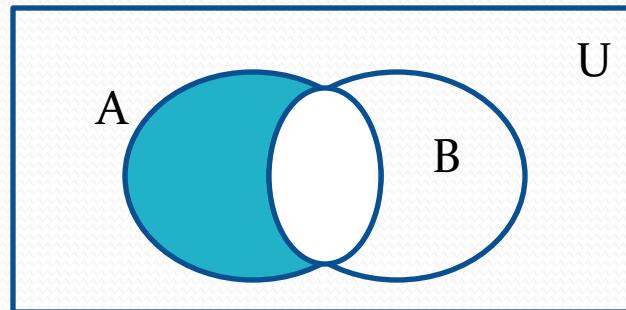
Venn Diagram for Complement



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 .

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

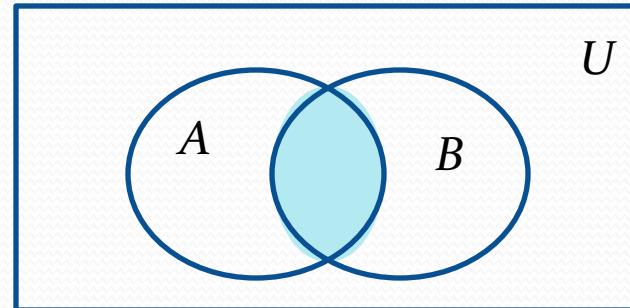


Venn Diagram for $A - B$

The Cardinality of the Union of Two Sets

- Inclusion-Exclusion

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



Venn Diagram for $A, B, A \cap B, A \cup B$

- **Example:** Let A be the math majors in your class and B be the CS majors. To count the number of students who are either math majors or CS majors, add the number of math majors and the number of CS majors, and subtract the number of joint CS/math majors.
- We will return to this principle in Chapter 6 and Chapter 8 where we will derive a formula for the cardinality of the union of n sets, where n is a positive integer.

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

4. \bar{B}

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

5. $A - B$

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

6. $B - A$

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

Symmetric Difference (*optional*)

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

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

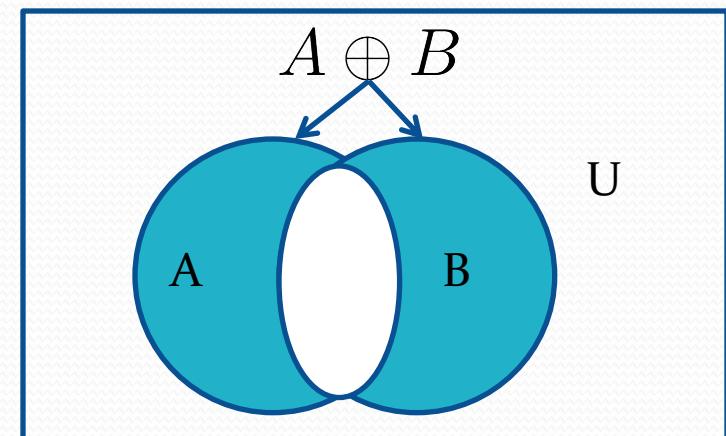
Example:

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

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

What is:

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



Venn Diagram

Set Identities

- Identity laws

$$A \cup \emptyset = A \quad A \cap U = A$$

- Domination laws

$$A \cup U = U \quad A \cap \emptyset = \emptyset$$

- Idempotent laws

$$A \cup A = A \quad A \cap A = A$$

- Complementation law

$$\overline{(\overline{A})} = A$$

Continued on next slide →

Set Identities

- Commutative laws

$$A \cup B = B \cup A \qquad A \cap B = B \cap A$$

- Associative laws

$$A \cup (B \cup C) = (A \cup B) \cup C$$

$$A \cap (B \cap C) = (A \cap B) \cap C$$

- Distributive laws

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

Continued on next slide →

Set Identities

- De Morgan's laws

$$\overline{A \cup B} = \overline{A} \cap \overline{B} \quad \overline{A \cap B} = \overline{A} \cup \overline{B}$$

- Absorption laws

$$A \cup (A \cap B) = A \quad A \cap (A \cup B) = A$$

- Complement laws

$$A \cup \overline{A} = U \quad A \cap \overline{A} = \emptyset$$

Proving Set Identities

- Different ways to prove set identities:
 1. Prove that each set (side of the identity) is a subset of the other.
 2. Use set builder notation and propositional logic.
 3. Membership Tables: Verify that elements in the same combination of sets always either belong or do not belong to the same side of the identity. Use 1 to indicate it is in the set and a 0 to indicate that it is not.

Proof of Second De Morgan Law

Example: Prove that $\overline{A \cap B} = \overline{A} \cup \overline{B}$

Solution: We prove this identity by showing that:

1) $\overline{A \cap B} \subseteq \overline{A} \cup \overline{B}$ and

2) $\overline{A} \cup \overline{B} \subseteq \overline{A \cap B}$

Continued on next slide →

Proof of Second De Morgan Law

These steps show that:

$$\overline{A \cap B} \subseteq \overline{A} \cup \overline{B}$$

$$x \in \overline{A \cap B}$$

by assumption

$$x \notin A \cap B$$

defn. of complement

$$\neg((x \in A) \wedge (x \in B))$$

defn. of intersection

$$\neg(x \in A) \vee \neg(x \in B)$$

1st De Morgan Law for Prop Logic

$$x \notin A \vee x \notin B$$

defn. of negation

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

defn. of complement

$$x \in \overline{A} \cup \overline{B}$$

defn. of union

Continued on next slide →

Proof of Second De Morgan Law

These steps show that:

$$x \in \overline{A} \cup \overline{B}$$

$$(x \in \overline{A}) \vee (x \in \overline{B})$$

$$(x \notin A) \vee (x \notin B)$$

$$\neg(x \in A) \vee \neg(x \in B)$$

$$\neg((x \in A) \wedge (x \in B))$$

$$\neg(x \in A \cap B)$$

$$x \in \overline{A \cap B}$$

$$\overline{A} \cup \overline{B} \subseteq \overline{A \cap B}$$

by assumption

defn. of union

defn. of complement

defn. of negation

by 1st De Morgan Law for Prop Logic

defn. of intersection

defn. of complement



Set-Builder Notation: Second De Morgan Law

$$\begin{aligned}\overline{A \cap B} &= \{x | x \notin A \cap B\} && \text{by defn. of complement} \\ &= \{x | \neg(x \in (A \cap B))\} && \text{by defn. of does not belong symbol} \\ &= \{x | \neg(x \in A \wedge x \in B)\} && \text{by defn. of intersection} \\ &= \{x | \neg(x \in A) \vee \neg(x \in B)\} && \text{by 1st De Morgan law} \\ &&& \text{for Prop Logic} \\ &= \{x | x \notin A \vee x \notin B\} && \text{by defn. of not belong symbol} \\ &= \{x | x \in \overline{A} \vee x \in \overline{B}\} && \text{by defn. of complement} \\ &= \{x | x \in \overline{A} \cup \overline{B}\} && \text{by defn. of union} \\ &= \overline{A} \cup \overline{B} && \text{by meaning of notation}\end{aligned}$$



Membership Table

Example: Construct a membership table to show that the distributive law holds.

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

Solution:

A	B	C	$B \cap C$	$A \cup (B \cap C)$	$A \cup B$	$A \cup C$	$(A \cup B) \cap (A \cup C)$
1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	0	0	0	1	1	1	1
0	1	1	1	1	1	1	1
0	1	0	0	0	1	0	0
0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0

Generalized Unions and Intersections

- Let A_1, A_2, \dots, A_n be an indexed collection of sets.

We define:

$$\bigcup_{i=1}^n A_i = A_1 \cup A_2 \cup \dots \cup A_n$$

$$\bigcap_{i=1}^n A_i = A_1 \cap A_2 \cap \dots \cap A_n$$

These are well defined, since union and intersection are associative.

- For $i = 1, 2, \dots$, let $A_i = \{i, i + 1, i + 2, \dots\}$. Then,

$$\bigcup_{i=1}^n A_i = \bigcup_{i=1}^n \{i, i + 1, i + 2, \dots\} = \{1, 2, 3, \dots\}$$

$$\bigcap_{i=1}^n A_i = \bigcap_{i=1}^n \{i, i + 1, i + 2, \dots\} = \{n, n + 1, n + 2, \dots\} = A_n$$

Functions

Section 2.3

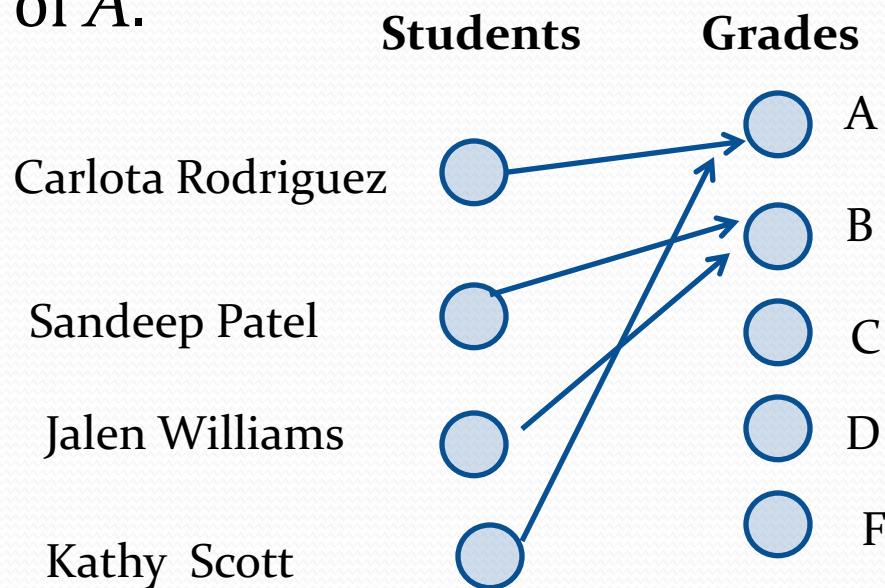
Section Summary

- Definition of a Function.
 - Domain, Cdomain
 - Image, Preimage
- Injection, Surjection, Bijection
- Inverse Function
- Function Composition
- Graphing Functions
- Floor, Ceiling, Factorial
- Partial Functions (optional)

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 are sometimes called *mappings* or *transformations*.



Functions

- A function $f: A \rightarrow B$ can also be defined as a subset of $A \times B$ (a relation). This subset is restricted to be a relation where no two elements of the relation have the same first element.
- Specifically, a function f from A to B contains one, and only one ordered pair (a, b) for every element $a \in A$.

$$\forall x[x \in A \rightarrow \exists y[y \in B \wedge (x, y) \in f]]$$

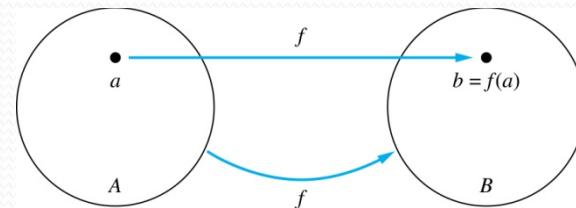
and

$$\forall x, y_1, y_2[((x, y_1) \in f \wedge (x, y_2)) \rightarrow y_1 = y_2]$$

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 .
- B is called the of f .
- If $f(a) = b$,
 - then b is called the *image* of a under f .
 - a is called the *preimage* of b .
- The range of f is the set of all images of points in A under f . We denote it by $f(A)$.
- 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.



Representing Functions

- Functions may be specified in different ways:
 - An explicit statement of the assignment.
Students and grades example.
 - A formula.
$$f(x) = x + 1$$
 - A computer program.
 - A Java program that when given an integer n , produces the n th Fibonacci Number (covered in the next section and also in Chapter 5).

Questions

$f(a) = ? \quad z$

The image of d is ? $\quad z$

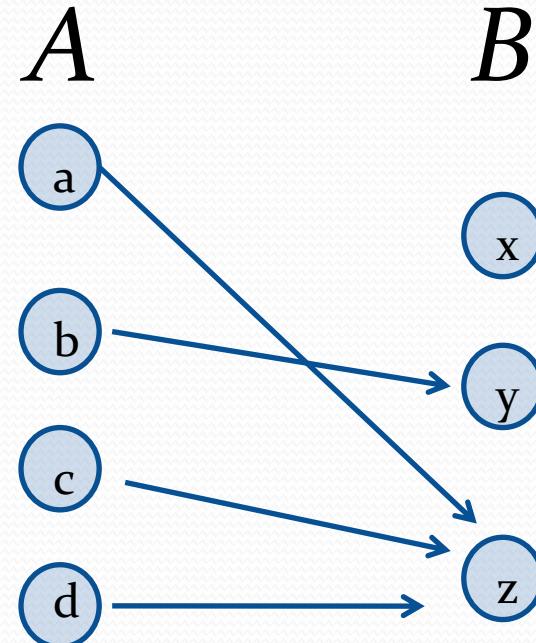
The domain of f is ? A

The codomain of f is ? B

The preimage of y is ? b

$f(A) = ?$

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



Question on Functions and Sets

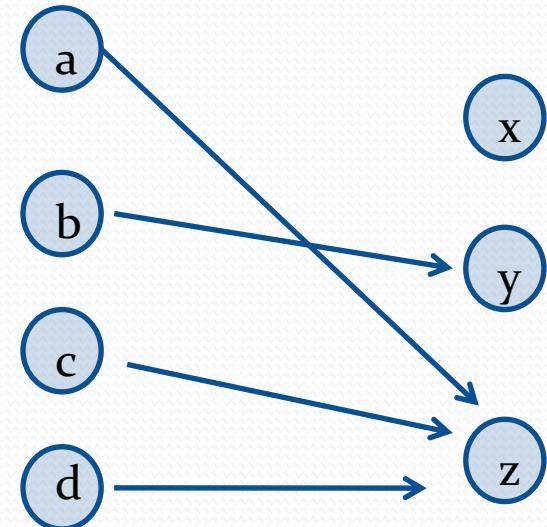
- If $f : A \rightarrow B$ and S is a subset of A , then

$$f(S) = \{f(s) | s \in S\}$$

A B

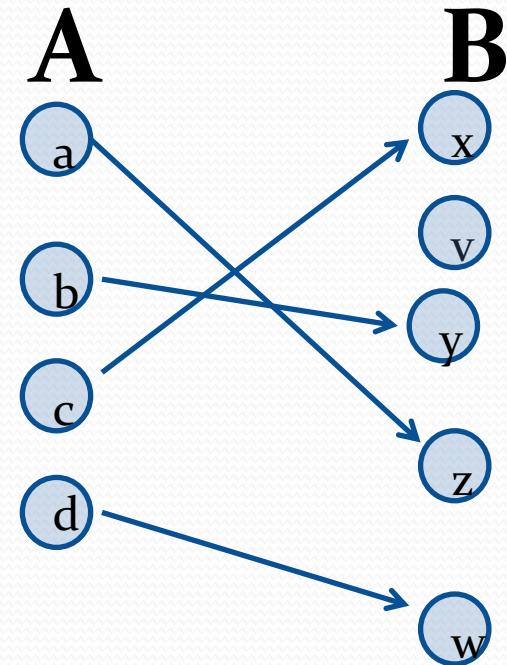
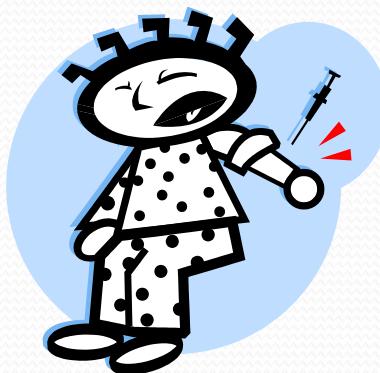
$f\{a,b,c,\}$ is ? $\{y,z\}$

$f\{c,d\}$ is ? $\{z\}$



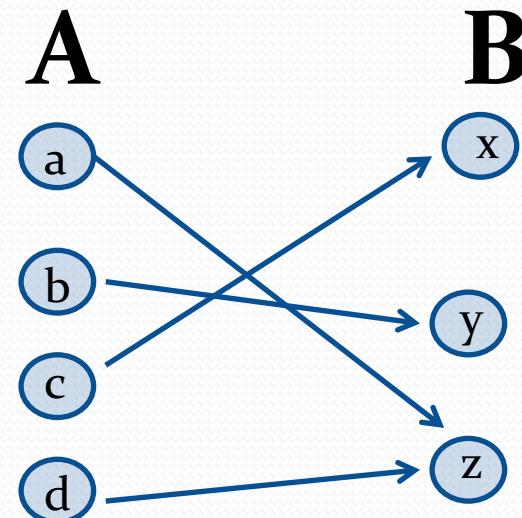
Injections

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.



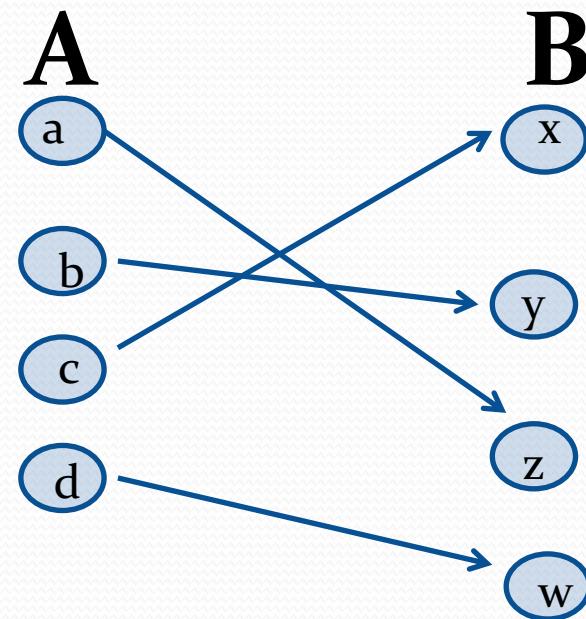
Surjections

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.



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).



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.

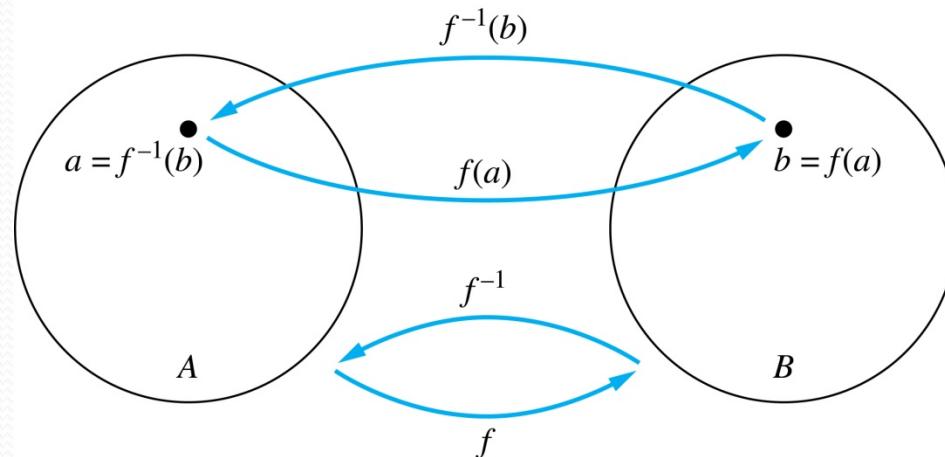
Example 2: Is the function $f(x) = x^2$ from the set of integers onto?

Solution: No, f is not onto because there is no integer x with $x^2 = -1$, for example.

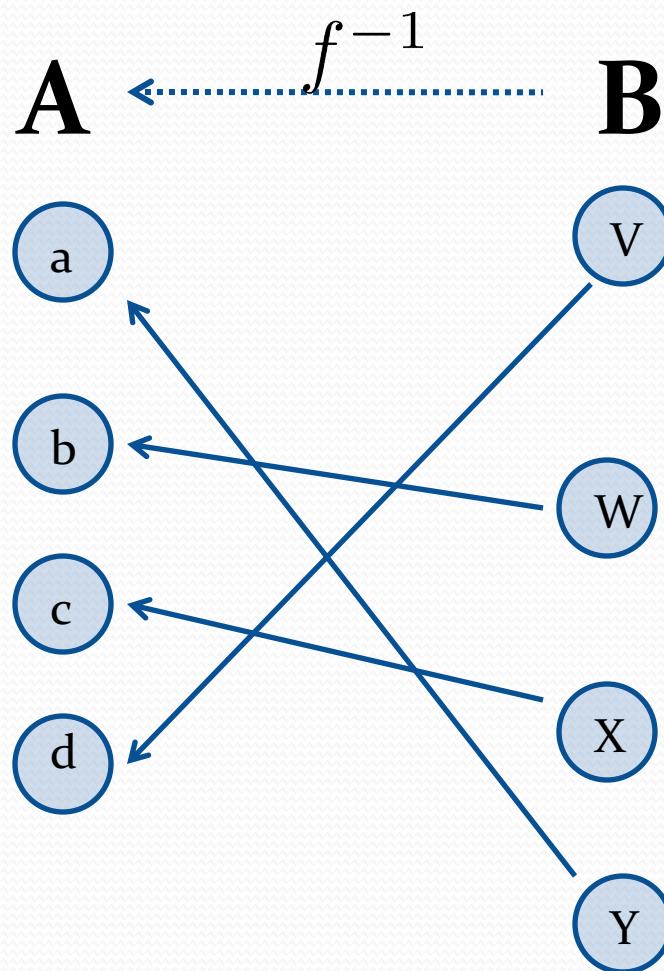
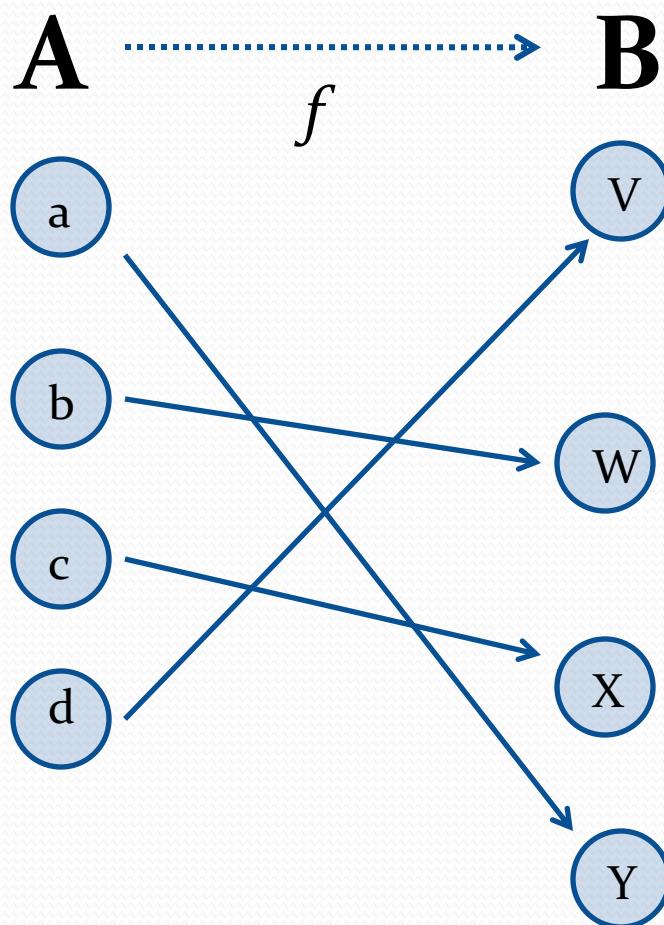
Inverse Functions

Definition: Let f be a bijection from A to B . Then the *inverse* of f , denoted f^{-1} , is the function from B to A defined as $f^{-1}(y) = x$ iff $f(x) = y$

No inverse exists unless f is a bijection. Why?



Inverse Functions



Questions

Example 1: Let f be the function from $\{a,b,c\}$ to $\{1,2,3\}$ such that $f(a) = 2$, $f(b) = 3$, and $f(c) = 1$. Is f invertible and if so what is its inverse?

Solution: The function f is invertible because it is a one-to-one correspondence. The inverse function f^{-1} reverses the correspondence given by f , so $f^{-1}(1) = c$, $f^{-1}(2) = a$, and $f^{-1}(3) = b$.

Questions

Example 2: Let $f: \mathbf{Z} \rightarrow \mathbf{Z}$ be such that $f(x) = x + 1$. Is f invertible, and if so, what is its inverse?

Solution: The function f is invertible because it is a one-to-one correspondence. The inverse function f^{-1} reverses the correspondence so $f^{-1}(y) = y - 1$.

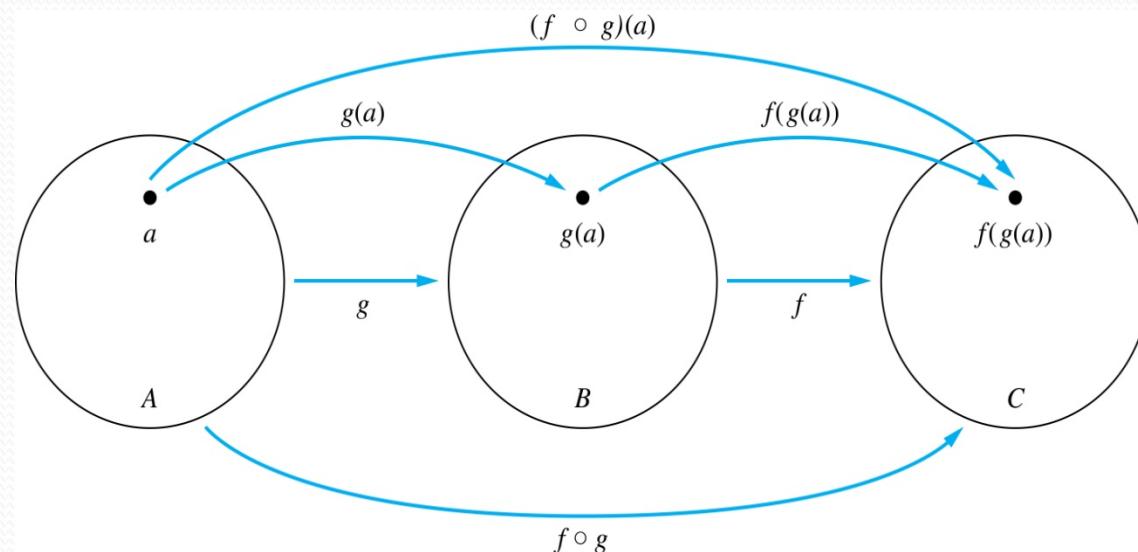
Questions

Example 3: Let $f: \mathbf{R} \rightarrow \mathbf{R}$ be such that $f(x) = x^2$. Is f invertible, and if so, what is its inverse?

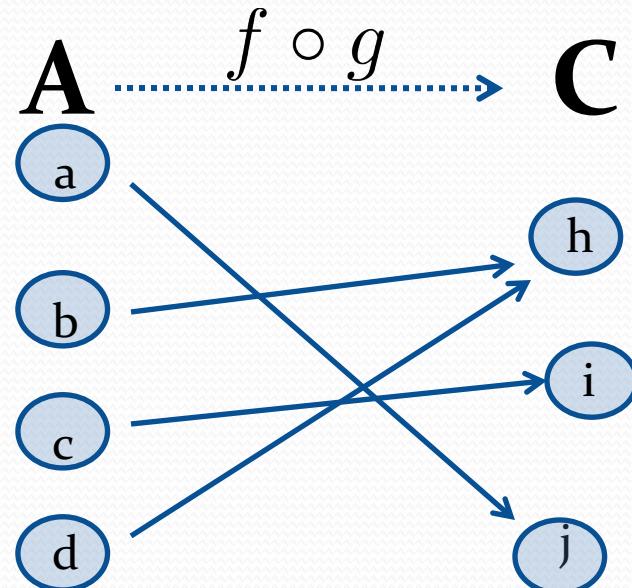
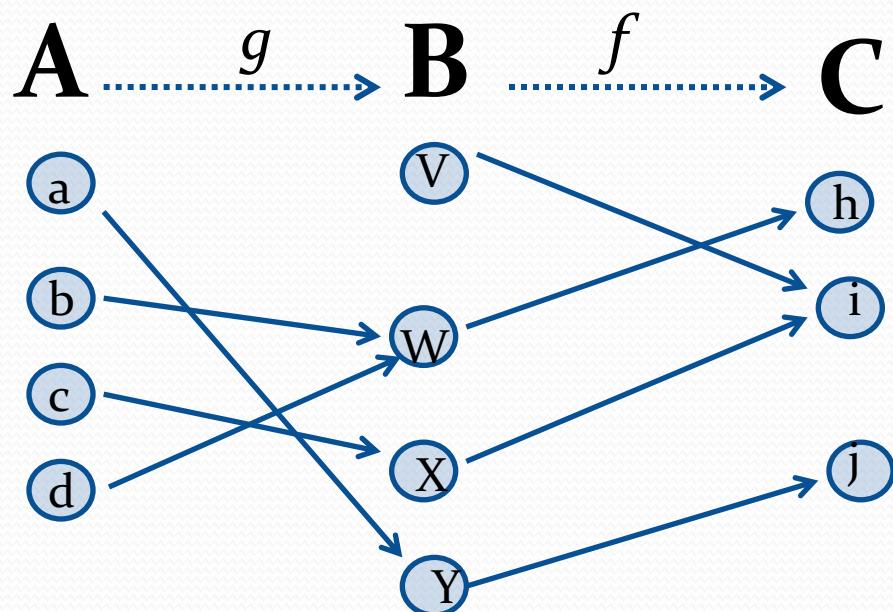
Solution: The function f is not invertible because it is not one-to-one .

Composition

- **Definition:** Let $f: B \rightarrow C$, $g: A \rightarrow B$. The *composition of f with g* , denoted $f \circ g$ is the function from A to C defined by $f \circ g(x) = f(g(x))$



Composition



Composition

Example 1: If $f(x) = x^2$ and $g(x) = 2x + 1$,
then

$$f(g(x)) = (2x + 1)^2$$

and

$$g(f(x)) = 2x^2 + 1$$

Composition Questions

Example 2: Let g be the function from the set $\{a, b, c\}$ to itself such that $g(a) = b$, $g(b) = c$, and $g(c) = a$. Let f be the function from the set $\{a, b, c\}$ to the set $\{1, 2, 3\}$ such that $f(a) = 3$, $f(b) = 2$, and $f(c) = 1$.

What is the composition of f and g , and what is the composition of g and f .

Solution: The composition $f \circ g$ is defined by

$$f \circ g (a) = f(g(a)) = f(b) = 2.$$

$$f \circ g (b) = f(g(b)) = f(c) = 1.$$

$$f \circ g (c) = f(g(c)) = f(a) = 3.$$

Note that $g \circ f$ is not defined, because the range of f is not a subset of the domain of g .

Composition Questions

Example 2: Let f and g be functions from the set of integers to the set of integers defined by $f(x) = 2x + 3$ and $g(x) = 3x + 2$.

What is the composition of f and g , and also the composition of g and f ?

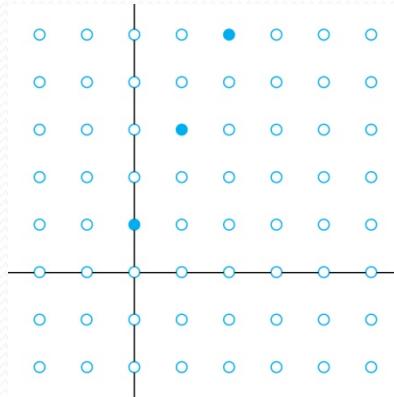
Solution:

$$f \circ g (x) = f(g(x)) = f(3x + 2) = 2(3x + 2) + 3 = 6x + 7$$

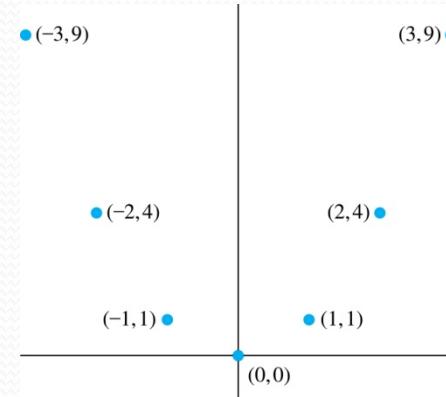
$$g \circ f (x) = g(f(x)) = g(2x + 3) = 3(2x + 3) + 2 = 6x + 11$$

Graphs of Functions

- Let f be a function from the set A to the set B . The *graph* of the function f is the set of ordered pairs $\{(a,b) \mid a \in A \text{ and } f(a) = b\}$.



Graph of $f(n) = 2n + 1$
from Z to Z



Graph of $f(x) = x^2$
from Z to Z

Some Important Functions

- The *floor* function, denoted

$$f(x) = \lfloor x \rfloor$$

is the largest integer less than or equal to x .

- The *ceiling* function, denoted

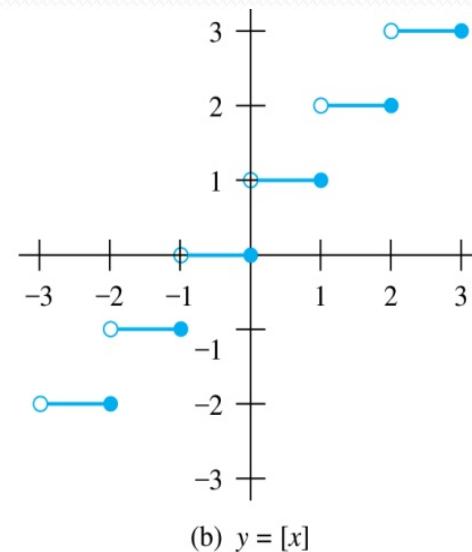
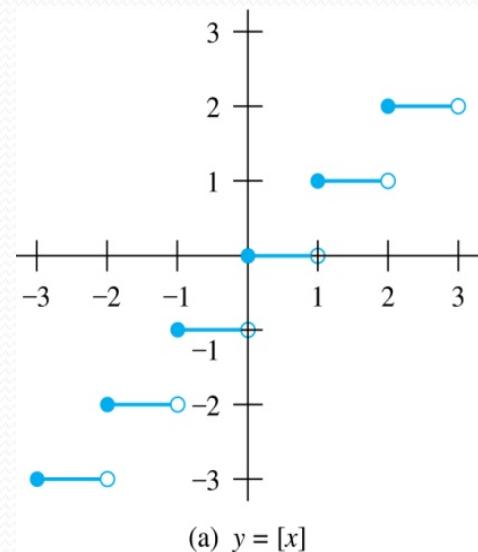
$$f(x) = \lceil x \rceil$$

is the smallest integer greater than or equal to x

Example: $\lceil 3.5 \rceil = 4$ $\lfloor 3.5 \rfloor = 3$

$$\lceil -1.5 \rceil = -1 \quad \lfloor -1.5 \rfloor = -2$$

Floor and Ceiling Functions



Graph of (a) Floor and (b) Ceiling Functions

Floor and Ceiling Functions

TABLE 1 Useful Properties of the Floor and Ceiling Functions.

(n is an integer, x is a real number)

(1a) $\lfloor x \rfloor = n$ if and only if $n \leq x < n + 1$

(1b) $\lceil x \rceil = n$ if and only if $n - 1 < x \leq n$

(1c) $\lfloor x \rfloor = n$ if and only if $x - 1 < n \leq x$

(1d) $\lceil x \rceil = n$ if and only if $x \leq n < x + 1$

(2) $x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$

(3a) $\lfloor -x \rfloor = -\lceil x \rceil$

(3b) $\lceil -x \rceil = -\lfloor x \rfloor$

(4a) $\lfloor x + n \rfloor = \lfloor x \rfloor + n$

(4b) $\lceil x + n \rceil = \lceil x \rceil + n$

Proving Properties of Functions

Example: Prove that x is a real number, then

$$\lfloor 2x \rfloor = \lfloor x \rfloor + \lfloor x + 1/2 \rfloor$$

Solution: Let $x = n + \varepsilon$, where n is an integer and $0 \leq \varepsilon < 1$.

Case 1: $\varepsilon < 1/2$

- $2x = 2n + 2\varepsilon$ and $\lfloor 2x \rfloor = 2n$, since $0 \leq 2\varepsilon < 1$.
- $\lfloor x + 1/2 \rfloor = n$, since $x + 1/2 = n + (1/2 + \varepsilon)$ and $0 \leq 1/2 + \varepsilon < 1$.
- Hence, $\lfloor 2x \rfloor = 2n$ and $\lfloor x \rfloor + \lfloor x + 1/2 \rfloor = n + n = 2n$.

Case 2: $\varepsilon \geq 1/2$

- $2x = 2n + 2\varepsilon = (2n + 1) + (2\varepsilon - 1)$ and $\lfloor 2x \rfloor = 2n + 1$, since $0 \leq 2\varepsilon - 1 < 1$.
- $\lfloor x + 1/2 \rfloor = \lfloor n + (1/2 + \varepsilon) \rfloor = \lfloor n + 1 + (\varepsilon - 1/2) \rfloor = n + 1$ since $0 \leq \varepsilon - 1/2 < 1$.
- Hence, $\lfloor 2x \rfloor = 2n + 1$ and $\lfloor x \rfloor + \lfloor x + 1/2 \rfloor = n + (n + 1) = 2n + 1$. ◀

Factorial Function

Definition: $f: \mathbf{N} \rightarrow \mathbf{Z}^+$, denoted by $f(n) = n!$ is the product of the first n positive integers when n is a nonnegative integer.

$$f(n) = 1 \cdot 2 \cdots (n - 1) \cdot n, \quad f(0) = 0! = 1$$

Examples:

$$f(1) = 1! = 1$$

$$f(2) = 2! = 1 \cdot 2 = 2$$

$$f(6) = 6! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720$$

$$f(20) = 2,432,902,008,176,640,000.$$

Stirling's Formula:

$$n! \sim \sqrt{2\pi n}(n/e)^n$$

$$f(n) \sim g(n) \doteq \lim_{n \rightarrow \infty} f(n)/g(n) = 1$$

Partial Functions (*optional*)

Definition: A *partial function* f from a set A to a set B is an assignment to each element a in a subset of A , called the *domain of definition* of f , of a unique element b in B .

- The sets A and B are called the *domain* and *codomain* of f , respectively.
- We say that f is *undefined* for elements in A that are not in the domain of definition of f .
- When the domain of definition of f equals A , we say that f is a *total function*.

Example: $f: \mathbf{N} \rightarrow \mathbf{R}$ where $f(n) = \sqrt{n}$ is a partial function from \mathbf{Z} to \mathbf{R} where the domain of definition is the set of nonnegative integers. Note that f is undefined for negative integers.

Sequences and Summations

Section 2.4

Section Summary

- Sequences.
 - Examples: Geometric Progression, Arithmetic Progression
- Recurrence Relations
 - Example: Fibonacci Sequence
- Summations
- Special Integer Sequences (*optional*)

Introduction

- Sequences are 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

$$a_n = \frac{1}{n} \quad \{a_n\} = \{a_1, a_2, a_3, \dots\}$$

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

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:

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

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 $d = 2$:

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

Strings

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

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

Recurrence Relations

Definition: A *recurrence relation* for the sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more of the previous terms of the sequence, namely, a_0, a_1, \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

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

$$a_2 = 5 + 3 = 8$$

$$a_3 = 8 + 3 = 11$$

Questions about Recurrence Relations

Example 2: Let $\{a_n\}$ be a sequence that satisfies the recurrence relation $a_n = a_{n-1} - a_{n-2}$ for $n = 2, 3, 4, \dots$ 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*.
- Various methods for solving recurrence relations will be covered in Chapter 8 where recurrence relations will be studied in greater depth.
- Here we illustrate by example the method of iteration in which we need to guess the formula. The guess can be proved correct by the method of induction (Chapter 5).

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 $a_1 = 2$.

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

.

.

.

$$= a_2 + 3(n-2) = (a_1 + 3) + 3(n-2) = 2 + 3(n-1)$$

Financial Application

Example: Suppose that a person deposits \$10,000.00 in a savings account at a bank yielding 11% per year with interest compounded annually. How much will be in the account after 30 years?

Let P_n denote the amount in the account after 30 years. P_n satisfies the following recurrence relation:

$$P_n = P_{n-1} + 0.11P_{n-1} = (1.11) P_{n-1}$$

with the initial condition $P_0 = 10,000$

Continued on next slide →

Financial Application

$$P_n = P_{n-1} + 0.11P_{n-1} = (1.11) P_{n-1}$$

with the initial condition $P_0 = 10,000$

Solution: Forward Substitution

$$P_1 = (1.11)P_0$$

$$P_2 = (1.11)P_1 = (1.11)^2 P_0$$

$$P_3 = (1.11)P_2 = (1.11)^3 P_0$$

:

$$P_n = (1.11)P_{n-1} = (1.11)^n P_0 = (1.11)^n 10,000$$

$P_n = (1.11)^n 10,000$ (Can prove by induction, covered in Chapter 5)

$$P_{30} = (1.11)^{30} 10,000 = \$228,992.97$$

Special Integer Sequences (*opt*)

- Given a few terms of a sequence, try to identify the sequence. Conjecture a formula, recurrence relation, or some other rule.
- Some questions to ask?
 - Are there repeated terms of the same value?
 - Can you obtain a term from the previous term by adding an amount or multiplying by an amount?
 - Can you obtain a term by combining the previous terms in some way?
 - Are there cycles among the terms?
 - Do the terms match those of a well known sequence?

Questions on Special Integer Sequences (opt)

Example 1: Find formulae for the sequences with the following first five terms: $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$

Solution: Note that the denominators are powers of 2. The sequence with $a_n = 1/2^n$ is a possible match. This is a geometric progression with $a = 1$ and $r = \frac{1}{2}$.

Example 2: Consider 1,3,5,7,9

Solution: Note that each term is obtained by adding 2 to the previous term. A possible formula is $a_n = 2n + 1$. This is an arithmetic progression with $a = 1$ and $d = 2$.

Example 3: 1, -1, 1, -1, 1

Solution: The terms alternate between 1 and -1. A possible sequence is $a_n = (-1)^n$. This is a geometric progression with $a = 1$ and $r = -1$.

Useful Sequences

TABLE 1 Some Useful Sequences.

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

Guessing Sequences (*optional*)

Example: Conjecture a simple formula for a_n if the first 10 terms of the sequence $\{a_n\}$ are 1, 7, 25, 79, 241, 727, 2185, 6559, 19681, 59047.

Solution: Note the ratio of each term to the previous approximates 3. So now compare with the sequence 3^n . We notice that the n th term is 2 less than the corresponding power of 3. So a good conjecture is that $a_n = 3^n - 2$.

Integer Sequences (*optional*)

- Integer sequences appear in a wide range of contexts. Later we will see the sequence of prime numbers (Chapter 4), the number of ways to order n discrete objects (Chapter 6), the number of moves needed to solve the Tower of Hanoi puzzle with n disks (Chapter 8), and the number of rabbits on an island after n months (Chapter 8).
- Integer sequences are useful in many fields such as biology, engineering, chemistry and physics.
- On-Line Encyclopedia of Integer Sequences (OESIS) contains over 200,000 sequences. Began by Neil Sloane in the 1960s (printed form). Now found at
<http://oeis.org/Spuzzle.html>

Integer Sequences (*optional*)

- Here are three interesting sequences to try from the OESIS site. To solve each puzzle, find a rule that determines the terms of the sequence.
- Guess the rules for forming for the following sequences:
 - 2, 3, 3, 5, 10, 13, 39, 43, 172, 177, ...
 - Hint: Think of adding and multiplying by numbers to generate this sequence.
 - 0, 0, 0, 0, 4, 9, 5, 1, 1, 0, 55, ...
 - Hint: Think of the English names for the numbers representing the position in the sequence and the Roman Numerals for the same number.
 - 2, 4, 6, 30, 32, 34, 36, 40, 42, 44, 46, ...
 - Hint: Think of the English names for numbers, and whether or not they have the letter 'e.'
- The answers and many more can be found at
<http://oeis.org/Spuzzle.html>

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

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

- 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:** $r^0 + r^1 + r^2 + r^3 + \cdots + r^n = \sum_0^n r^j$

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

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

Product Notation (*optional*)

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

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

represents

$$a_m \times a_{m+1} \times \cdots \times a_n$$

Geometric Series

Sums of terms of geometric progressions

$$\sum_{j=0}^n ar^j = \begin{cases} \frac{ar^{n+1}-a}{r-1} & r \neq 1 \\ (n+1)a & r = 1 \end{cases}$$

Proof: Let $S_n = \sum_{j=0}^n ar^j$

To compute S_n , first multiply both sides of the equality by r and then manipulate the resulting sum as follows:

$$rS_n = r \sum_{j=0}^n ar^j$$

$$= \sum_{j=0}^n ar^{j+1}$$

Continued on next slide →

Geometric Series

$$\begin{aligned} &= \sum_{j=0}^n ar^{j+1} && \text{From previous slide.} \\ &= \sum_{k=1}^{n+1} ar^k && \text{Shifting the index of summation with } k = j + 1. \\ &= \left(\sum_{k=0}^n ar^k \right) + (ar^{n+1} - a) && \text{Removing } k = n + 1 \text{ term and} \\ &&& \text{adding } k = 0 \text{ term.} \\ &= S_n + (ar^{n+1} - a) && \text{Substituting } S \text{ for summation formula} \end{aligned}$$

∴

$$rS_n = S_n + (ar^{n+1} - a)$$

$$S_n = \frac{ar^{n+1} - a}{r - 1} \quad \text{if } r \neq 1$$

$$S_n = \sum_{j=0}^n ar^j = \sum_{j=0}^n a = (n + 1)a \quad \text{if } r = 1$$

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)

Cardinality of Sets

Section 2.5

Section Summary

- Cardinality
- Countable Sets
- Computability

Cardinality

Definition: The *cardinality* of a set A is equal to the cardinality of a set B , denoted

$$|A| = |B|,$$

if and only if there is a one-to-one correspondence (*i.e.*, a bijection) from A to B .

- If there is a one-to-one function (*i.e.*, an injection) from A to B , the cardinality of A is less than or the same as the cardinality of B and we write $|A| \leq |B|$.
- When $|A| \leq |B|$ and A and B have different cardinality, we say that the cardinality of A is less than the cardinality of B and write $|A| < |B|$.

Cardinality

Definition

A set S is finite with cardinality $n \in \mathbb{N}$ if there is a bijection from the set $\{0, 1, \dots, n-1\}$ to S . A set is infinite if it is not finite.

Some facts which could easily be seen are:

1. If S' is infinite and is a subset of S , S is infinite.
2. Every subset of a finite set is finite.
3. If $f : S \rightarrow T$ be an injection and S is infinite, then T is infinite.
4. If S is an infinite set $P(S)$ is infinite.
5. If S and T are infinite sets. $S \cup T$ is infinite.
6. If S is infinite and $T \neq \emptyset$, then $S \times T$ is infinite.
7. If S is infinite and $T \neq \emptyset$, the set of functions from T to S is infinite.

Definition

The sets A and B have the same cardinality if and only if there is a one-to-one correspondence from A to B. When A and B have the same cardinality, we write $|A| = |B|$.

For infinite sets the definition of cardinality provides a relative measure of the sizes of two sets, rather than a measure of the size of one particular set. We can also define what it means for one set to have a smaller cardinality than another set.

Cardinality

- **Definition:** A set that is either finite or has the same cardinality as the set of positive integers (\mathbb{Z}^+) is called *countable*. A set that is not countable is *uncountable*.
- The set of real numbers \mathbf{R} is an uncountable set (see later).
- When an infinite set is countable (*countably infinite*) its cardinality is \aleph_0 (where \aleph is aleph, the 1st letter of the Hebrew alphabet). We write $|S| = \aleph_0$ and say that S has cardinality “aleph null.”

Showing that a Set is Countable

- An infinite set is countable if and only if it is possible to list the elements of the set in a sequence (indexed by the positive integers).
- The reason for this is that a one-to-one correspondence f from the set of positive integers to a set S can be expressed in terms of a sequence $a_1, a_2, \dots, a_n, \dots$ where $a_1 = f(1)$, $a_2 = f(2), \dots, a_n = f(n), \dots$

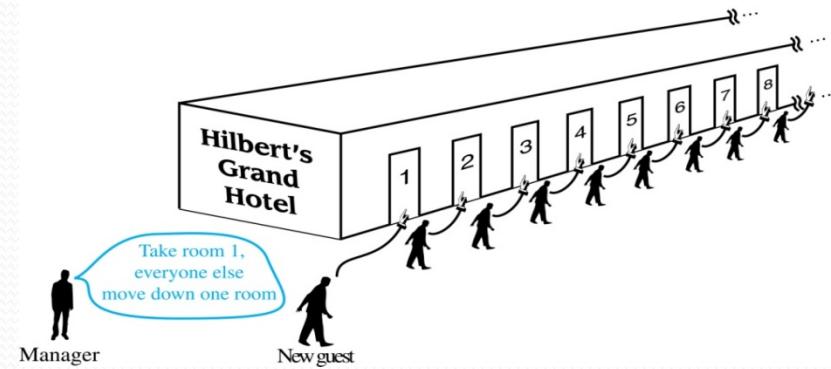


Hilbert's Grand Hotel

David Hilbert

The Grand Hotel (example due to David Hilbert) has countably infinite number of rooms, each occupied by a guest. We can always accommodate a new guest at this hotel. How is this possible?

Explanation: Because the rooms of Grand Hotel are countable, we can list them as Room 1, Room 2, Room 3, and so on. When a new guest arrives, we move the guest in Room 1 to Room 2, the guest in Room 2 to Room 3, and in general the guest in Room n to Room $n + 1$, for all positive integers n . This frees up Room 1, which we assign to the new guest, and all the current guests still have rooms.

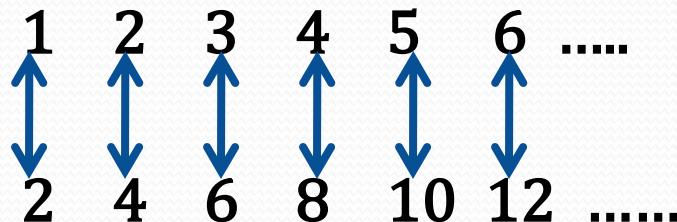


The hotel can also accommodate a countable number of new guests, all the guests on a countable number of buses where each bus contains a countable number of guests (see exercises).

Showing that a Set is Countable

Example 1: Show that the set of positive even integers E is countable set.

Solution: Let $f(x) = 2x$.



Then f is a bijection from \mathbb{N} to E since f is both one-to-one and onto. To show that it is one-to-one, suppose that $f(n) = f(m)$. Then $2n = 2m$, and so $n = m$. To see that it is onto, suppose that t is an even positive integer. Then $t = 2k$ for some positive integer k and $f(k) = t$.



Showing that a Set is Countable

Example 2: Show that the set of integers \mathbf{Z} is countable.

Solution: Can list in a sequence:

0, 1, -1, 2, -2, 3, -3 ,.....

Or can define a bijection from \mathbf{N} to \mathbf{Z} :

- When n is even: $f(n) = n/2$
- When n is odd: $f(n) = -(n-1)/2$



The Positive Rational Numbers are Countable

- **Definition:** A *rational number* can be expressed as the ratio of two integers p and q such that $q \neq 0$.
 - $\frac{3}{4}$ is a rational number
 - $\sqrt{2}$ is not a rational number.

Example 3: Show that the positive rational numbers are countable.

Solution: The positive rational numbers are countable since they can be arranged in a sequence:

$$r_1, r_2, r_3, \dots$$

The next slide shows how this is done.



The Positive Rational Numbers are Countable

First row $q = 1$.

Second row $q = 2$.

etc.

Constructing the List

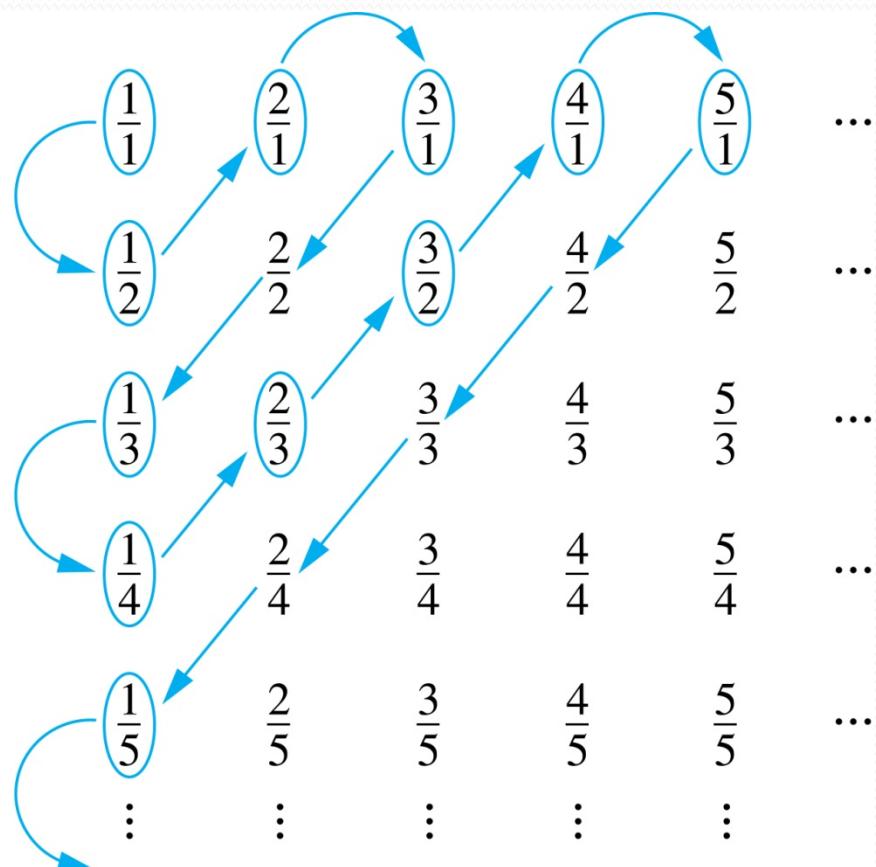
First list p/q with $p + q = 2$.

Next list p/q with $p + q = 3$

And so on.

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

Terms not circled
are not listed
because they
repeat previously
listed terms



Strings

Example 4: Show that the set of finite strings S over a finite alphabet A is countably infinite.

Assume an alphabetical ordering of symbols in A

Solution: Show that the strings can be listed in a sequence. First list

1. All the strings of length 0 in alphabetical order.
2. Then all the strings of length 1 in lexicographic (as in a dictionary) order.
3. Then all the strings of length 2 in lexicographic order.
4. And so on.

This implies a bijection from \mathbb{N} to S and hence it is a countably infinite set.



Example

Let $\Sigma = \{a, b\}$. The set of strings over Σ^* is a countably infinite set. An enumeration τ of Σ^* is possible. $\tau(0) = \lambda$, the empty string. Assuming a comes before b in Σ . $\tau(1) = a$ and $\tau(2) = b$. $\tau(3)$ to $\tau(6)$ are strings of length 2 and so on. What is the 49th string in the enumeration?

$\tau(0)$ – string of length 0

$\tau(1)$ – $\tau(2)$ – string of length 1

$\tau(3)$ – $\tau(6)$ – string of length 2

$\tau(7)$ – $\tau(14)$ – string of length 3

$\tau(15)$ – $\tau(30)$ – string of length 4

$\tau(31)$ – $\tau(62)$ – string of length 5

Hence the 49th string is of length 5. Among these $\tau(31)$ to $\tau(46)$ begin with a and $\tau(47)$ to $\tau(62)$ begin with b.

$\tau(47) = b a a a a$

$\tau(48) = b a a a b$

$\tau(49) = b a a b a$

Hence the 49th string in the enumeration is b a a b a.

The set of all Java programs is countable.

Example 5: Show that the set of all Java programs is countable.

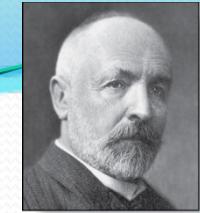
Solution: Let S be the set of strings constructed from the characters which can appear in a Java program. Use the ordering from the previous example. Take each string in turn:

- Feed the string into a Java compiler. (A Java compiler will determine if the input program is a syntactically correct Java program.)
- If the compiler says YES, this is a syntactically correct Java program, we add the program to the list.
- We move on to the next string.

In this way we construct an implied bijection from \mathbf{N} to the set of Java programs. Hence, the set of Java programs is countable.



Georg Cantor
(1845-1918)



The Real Numbers are Uncountable

Example: Show that the set of real numbers is uncountable.

Solution: The method is called the Cantor diagonalization argument, and is a proof by contradiction.

1. Suppose \mathbf{R} is countable. Then the real numbers between 0 and 1 are also countable (any subset of a countable set is countable - an exercise in the text).
2. The real numbers between 0 and 1 can be listed in order r_1, r_2, r_3, \dots .
3. Let the decimal representation of this listing be $r_1 = 0.d_{11}d_{12}d_{13}d_{14}d_{15}d_{16} \dots$
 $r_2 = 0.d_{21}d_{22}d_{23}d_{24}d_{25}d_{26} \dots$
 $r_3 = 0.d_{31}d_{32}d_{33}d_{34}d_{35}d_{36} \dots$
 \vdots
4. Form a new real number r with the decimal expansion $0.d_1d_2d_3d_4 \dots$ where $d_i = 3$ if $d_{ii} \neq 3$ and $d_i = 4$ if $d_{ii} = 3$.
5. r is not equal to any of the r_1, r_2, r_3, \dots . Because it differs from r_i in its i th position after the decimal point. Therefore there exists a real number between 0 and 1 that is not on the list since every real number has a unique decimal expansion. Hence, all the real numbers between 0 and 1 cannot be listed, so the set of real numbers between 0 and 1 is uncountable.
6. Since a set with an uncountable subset is uncountable (an exercise), the set of real numbers is uncountable.



Theorem

Every infinite set contains a countably infinite subset.

Proof

Let S be an infinite set. Select a sequence of elements $\langle a_0, a_1, \dots \rangle$ from S as follows:

Select a_0 from S

Select a_1 from $S - \{a_0\}$

Select a_2 from $S - \{a_0, a_1\}$

and so on. a_i will be selected from $S - \{a_0, a_1, \dots, a_{i-1}\}$.

Each $S - \{a_0, \dots, a_i\}$ is infinite.

Otherwise $(S - \{a_0, \dots, a_i\}) \cup \{a_0, \dots, a_i\} = S$ will be finite.

This set $\{a_0, \dots, a_i, \dots\}$ is countably infinite subset of S .

Theorem

The union of a countable collection of countable sets is countable.

Proof

Let S_0, S_1, S_2, \dots be a countable collection of countable sets and $S_i = \langle a_{i0}, a_{i1}, a_{i2}, \dots \rangle$

$$S_0 : (a_{00} \ a_{01} \ a_{02} \ a_{03} \dots)$$



$$S_1 : (a_{10} \ a_{11} \ a_{12} \ a_{13} \dots)$$



$$S_2 : (a_{20} \ a_{21} \ a_{22} \ a_{23} \dots)$$



$$S_3 : (a_{30} \ a_{31} \ a_{32} \ a_{33} \dots)$$

Enumerate the elements as shown in the diagram.

Let b_0, b_1, \dots be the sequence of elements.

$$b_0 = a_{00}$$

$$b_1 = a_{01}$$

$$b_2 = a_{10}$$

$$b_3 = a_{02}$$

$$b_4 = a_{11}$$

$b_5 = a_{20}$ and so on.

This is an enumeration of the elements of $\bigcup_{i=0}^{\infty} A_i$ and hence $\bigcup_{i=0}^{\infty} A_i$ is countable.

Theorem

Let Σ be a finite alphabet and Σ^* the set of all strings over Σ . Then $P(\Sigma^*)$ is uncountable.

Proof

Let $\langle x_0, x_1, x_2, \dots \rangle$ be an enumeration of strings in Σ^* and if possible let $\langle A_0, A_1, \dots \rangle$ be an enumeration of the sets in $P(\Sigma^*)$. Construct a binary matrix M as follows.

	x_0	x_1	x_2	\dots
A_0	a_{00}	a_{01}	a_{02}	\dots
A_1	a_{10}	a_{11}	a_{12}	\dots
A_2	a_{20}	a_{21}	a_{22}	\dots
\vdots	\vdots	\vdots	\vdots	

$a_{ij} = 1$ if $x_j \in A_i$ and $a_{ij} = 0$ if $x_j \notin A_i$. Now we construct a set A as follows $x_i \in A$ if $a_{ii} = 0$. i.e., $x_i \notin A_i$, $A = \{x_i \in A \mid x_i \notin A_i, i \in N\}$

Now we can see that A cannot be any A_j and hence cannot appear in the enumeration $\langle A_0, A_1, \dots \rangle$ even though $A \in P(\Sigma^*)$. This is seen by the following contradiction. Suppose if possible that $A = A_j$.

A consists of strings x_j such that $a_{jj} = 0$. So if $x_j \in A$, then $a_{jj} = 0$ and by the way we constructed the matrix M , if $a_{jj} = 0$, $x_j \notin A_j$.

Hence $A \neq A_j$. Hence we cannot have an enumeration of the sets in $P(\Sigma^*)$ and so $P(\Sigma^*)$ is an uncountable set.

Definition

A set S is of cardinality c if there is a bijection from $[0, 1]$ to S . The label c is given to this because of the fact that the set $[0, 1]$ is often called a continuum.

Definition

Let S and T be sets. Then S and T are equipotent or have the same cardinality, denoted by $|S| = |T|$, if there is a bijection from S to T .

The following result follows immediately.

Theorem

Equipotence is an equivalence relation over any collection of sets.

Definition

The cardinality of S is no greater than the cardinality of T , denoted as $|S| \leq |T|$, if there is an injection from S to T . The cardinality of S is less than the cardinality of T , written $|S| < |T|$, if there exists an injection but no bijection from S to T .

It is easy to see that if S and T are sets, then exactly one of the following three conditions will hold:

1. $|S| < |T|$
2. $|S| = |T|$
3. $|T| < |S|$

Also we can see that if $|S| \leq |T|$ and $|T| \leq |S|$, then $|T| = |S|$.

Computability (Optional)

- **Definition:** We say that a function is **computable** if there is a computer program in some programming language that finds the values of this function. If a function is not computable we say it is **uncomputable**.
- There are uncomputable functions. We have shown that the set of Java programs is countable. Exercise 38 in the text shows that there are uncountably many different functions from a particular countably infinite set (i.e., the positive integers) to itself. Therefore (Exercise 39) there must be uncomputable functions.

Matrices

Section 2.6

Section Summary

- Definition of a Matrix
- Matrix Arithmetic
- Transposes and Powers of Arithmetic
- Zero-One matrices

Matrices

- Matrices are useful discrete structures that can be used in many ways. For example, they are used to:
 - describe certain types of functions known as linear transformations.
 - Express which vertices of a graph are connected by edges (see Chapter 10).
- In later chapters, we will see matrices used to build models of:
 - Transportation systems.
 - Communication networks.
- Algorithms based on matrix models will be presented in later chapters.
- Here we cover the aspect of matrix arithmetic that will be needed later.

Matrix

Definition: A *matrix* is a rectangular array of numbers. A matrix with m rows and n columns is called an $m \times n$ matrix.

- The plural of matrix is *matrices*.
- A matrix with the same number of rows as columns is called *square*.
- Two matrices are *equal* if they have the same number of rows and the same number of columns and the corresponding entries in every position are equal.

3×2 matrix

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

Notation

- Let m and n be positive integers and let

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

- The i th row of \mathbf{A} is the $1 \times n$ matrix $[a_{i1}, a_{i2}, \dots, a_{in}]$. The j th column of \mathbf{A} is the $m \times 1$ matrix:
- The (i,j) th *element* or *entry* of \mathbf{A} is the element a_{ij} . We can use $\mathbf{A} = [a_{ij}]$ to denote the matrix with its (i,j) th element equal to a_{ij} .

$$\begin{bmatrix} a_{1j} \\ a_{2j} \\ \cdot \\ \cdot \\ a_{mj} \end{bmatrix}$$

Matrix Arithmetic: Addition

Defintion: Let $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$ be $m \times n$ matrices.

The sum of \mathbf{A} and \mathbf{B} , denoted by $\mathbf{A} + \mathbf{B}$, is the $m \times n$ matrix that has $a_{ij} + b_{ij}$ as its (i,j) th element. In other words, $\mathbf{A} + \mathbf{B} = [a_{ij} + b_{ij}]$.

Example:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 2 & -3 \\ 3 & 4 & 0 \end{bmatrix} + \begin{bmatrix} 3 & 4 & -1 \\ 1 & -3 & 0 \\ -1 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 & -2 \\ 3 & -1 & -3 \\ 2 & 5 & 2 \end{bmatrix}$$

Note that matrices of different sizes can not be added.

Matrix Multiplication

Definition: Let \mathbf{A} be an $n \times k$ matrix and \mathbf{B} be a $k \times n$ matrix. The *product* of \mathbf{A} and \mathbf{B} , denoted by \mathbf{AB} , is the $m \times n$ matrix that has its (i,j) th element equal to the sum of the products of the corresponding elements from the i th row of \mathbf{A} and the j th column of \mathbf{B} . In other words, if $\mathbf{AB} = [c_{ij}]$ then $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{kj}b_{2j}$.

Example:

$$\begin{bmatrix} 1 & 0 & 4 \\ 2 & 1 & 1 \\ 3 & 1 & 0 \\ 0 & 2 & 2 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ 1 & 1 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 14 & 4 \\ 8 & 9 \\ 7 & 13 \\ 8 & 2 \end{bmatrix}$$

The product of two matrices is undefined when the number of columns in the first matrix is not the same as the number of rows in the second.

Illustration of Matrix Multiplication

- The Product of $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \textcolor{red}{a_{i1}} & \textcolor{red}{a_{i2}} & \dots & \textcolor{red}{a_{ik}} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{m1} & a_{m2} & \dots & a_{mk} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} b_{11} & a_{12} & \dots & \textcolor{red}{b_{1j}} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & \textcolor{red}{b_{2j}} & \dots & b_{2n} \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ b_{k1} & b_{k2} & \dots & \textcolor{red}{b_{kj}} & \dots & b_{kn} \end{bmatrix}$$

$$\mathbf{AB} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \textcolor{red}{c_{ij}} \\ \cdot & \cdot & & \cdot \\ c_{m1} & c_{m2} & \dots & c_{mn} \end{bmatrix}$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{ik}b_{kj}$$

Matrix Multiplication is not Commutative

Example: Let

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

Does $AB = BA$?

Solution:

$$AB = \begin{bmatrix} 2 & 2 \\ 5 & 3 \end{bmatrix}$$

$$BA = \begin{bmatrix} 4 & 3 \\ 3 & 2 \end{bmatrix}$$

$AB \neq BA$

Identity Matrix and Powers of Matrices

Definition: The *identity matrix of order n* is the $m \times n$ matrix $\mathbf{I}_n = [\delta_{ij}]$, where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$.

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \ddots & & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \quad \mathbf{AI}_n = \mathbf{I}_m \mathbf{A} = \mathbf{A}$$

when \mathbf{A} is an $m \times n$ matrix

Powers of square matrices can be defined. When \mathbf{A} is an $n \times n$ matrix, we have:

$$\mathbf{A}^0 = \mathbf{I}_n \quad \mathbf{A}^r = \underbrace{\mathbf{AAA}\cdots\mathbf{A}}_{r \text{ times}}$$

Transposes of Matrices

Definition: Let $\mathbf{A} = [a_{ij}]$ be an $m \times n$ matrix. The *transpose* of \mathbf{A} , denoted by \mathbf{A}^t , is the $n \times m$ matrix obtained by interchanging the rows and columns of \mathbf{A} .

If $\mathbf{A}^t = [b_{ij}]$, then $b_{ij} = a_{ji}$ for $i = 1, 2, \dots, n$
and $j = 1, 2, \dots, m$.

The transpose of the matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ is the matrix $\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$.

Transposes of Matrices

Definition: A square matrix A is called symmetric if $A = A^t$. Thus $A = [a_{ij}]$ is symmetric if $a_{ij} = a_{ji}$ for i and j with $1 \leq i \leq n$ and $1 \leq j \leq n$.

The matrix $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ is square.

Square matrices do not change when their rows and columns are interchanged.

Zero-One Matrices

Definition: A matrix all of whose entries are either 0 or 1 is called a *zero-one matrix*. (These will be used in Chapters 9 and 10.)

Algorithms operating on discrete structures represented by zero-one matrices are based on Boolean arithmetic defined by the following Boolean operations:

$$b_1 \wedge b_2 = \begin{cases} 1 & \text{if } b_1 = b_2 = 1 \\ 0 & \text{otherwise} \end{cases} \quad b_1 \vee b_2 = \begin{cases} 1 & \text{if } b_1 = 1 \text{ or } b_2 = 1 \\ 0 & \text{otherwise} \end{cases}$$

Zero-One Matrices

Definition: Let $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$ be an $m \times n$ zero-one matrices.

- The *join* of \mathbf{A} and \mathbf{B} is the zero-one matrix with (i,j) th entry $a_{ij} \vee b_{ij}$. The *join* of \mathbf{A} and \mathbf{B} is denoted by $\mathbf{A} \vee \mathbf{B}$.
- The *meet* of \mathbf{A} and \mathbf{B} is the zero-one matrix with (i,j) th entry $a_{ij} \wedge b_{ij}$. The *meet* of \mathbf{A} and \mathbf{B} is denoted by $\mathbf{A} \wedge \mathbf{B}$.

Joins and Meets of Zero-One Matrices

Example: Find the join and meet of the zero-one matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}.$$

Solution: The join of \mathbf{A} and \mathbf{B} is

$$\mathbf{A} \vee \mathbf{B} = \begin{bmatrix} 1 \vee 0 & 0 \vee 1 & 1 \vee 0 \\ 0 \vee 1 & 1 \vee 1 & 0 \vee 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

The meet of \mathbf{A} and \mathbf{B} is

$$\mathbf{A} \wedge \mathbf{B} = \begin{bmatrix} 1 \wedge 0 & 0 \wedge 1 & 1 \wedge 0 \\ 0 \wedge 1 & 1 \wedge 1 & 0 \wedge 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Boolean Product of Zero-One Matrices

Definition: Let $\mathbf{A} = [a_{ij}]$ be an $m \times k$ zero-one matrix and $\mathbf{B} = [b_{ij}]$ be a $k \times n$ zero-one matrix. The *Boolean product* of \mathbf{A} and \mathbf{B} , denoted by $\mathbf{A} \odot \mathbf{B}$, is the $m \times n$ zero-one matrix with (i,j) th entry

$$c_{ij} = (a_{i1} \wedge b_{1j}) \vee (a_{i2} \wedge b_{2j}) \vee \dots \vee (a_{ik} \wedge b_{kj}).$$

Example: Find the Boolean product of \mathbf{A} and \mathbf{B} , where

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

Continued on next slide →

Boolean Product of Zero-One Matrices

Solution: The Boolean product $\mathbf{A} \odot \mathbf{B}$ is given by

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} (1 \wedge 1) \vee (0 \wedge 0) & (1 \wedge 1) \vee (0 \wedge 1) & (1 \wedge 0) \vee (0 \wedge 1) \\ (0 \wedge 1) \vee (1 \wedge 0) & (0 \wedge 1) \vee (1 \wedge 1) & (0 \wedge 0) \vee (1 \wedge 1) \\ (1 \wedge 1) \vee (0 \wedge 0) & (1 \wedge 1) \vee (0 \wedge 1) & (1 \wedge 0) \vee (0 \wedge 1) \end{bmatrix}$$

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

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

Boolean Powers of Zero-One Matrices

Definition: Let A be a square zero-one matrix and let r be a positive integer. The r th Boolean power of A is the Boolean product of r factors of A , denoted by $A^{[r]}$. Hence,

$$A^{[r]} = \underbrace{A \odot A \odot \dots \odot A}_{r \text{ times}}.$$

We define $A^{[r]}$ to be I_n .

(The Boolean product is well defined because the Boolean product of matrices is associative.)

Boolean Powers of Zero-One Matrices

Example: Let $\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$.

Find \mathbf{A}^n for all positive integers n .

Solution:

$$\mathbf{A}^{[2]} = \mathbf{A} \odot \mathbf{A} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}^{[3]} = \mathbf{A}^{[2]} \odot \mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{A}^{[4]} = \mathbf{A}^{[3]} \odot \mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{A}^{[5]} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \mathbf{A}^{[n]} = \mathbf{A}^5 \quad \text{for all positive integers } n \text{ with } n \geq 5.$$

Induction and recursion

Chapter 5

With Question/Answer Animations

Chapter Summary

- Mathematical Induction
- Strong Induction
- Well-Ordering
- Recursive Definitions
- Structural Induction
- Recursive Algorithms
- Program Correctness (*not yet included in overheads*)

Mathematical Induction

Section 5.1

Section Summary

- Mathematical Induction
- Examples of Proof by Mathematical Induction
- Mistaken Proofs by Mathematical Induction
- Guidelines for Proofs by Mathematical Induction

Climbing an Infinite Ladder

Suppose we have an infinite ladder:

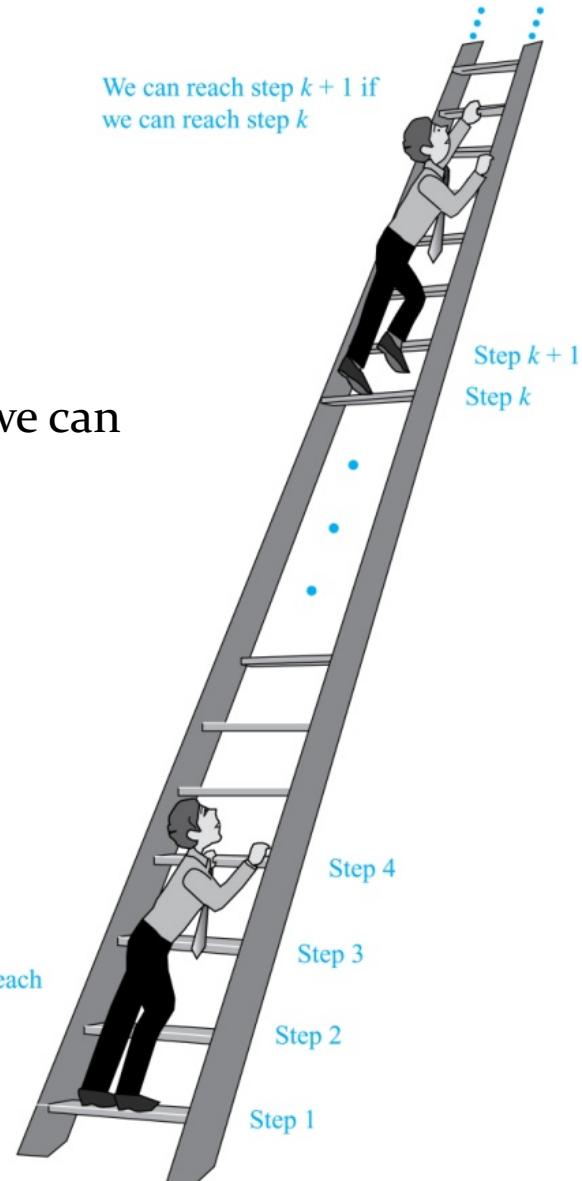
1. We can reach the first rung of the ladder.
2. If we can reach a particular rung of the ladder, then we can reach the next rung.

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.

This example motivates proof by mathematical induction.

We can reach step $k + 1$ if we can reach step k



Principle of 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.

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. We will see examples of this soon.

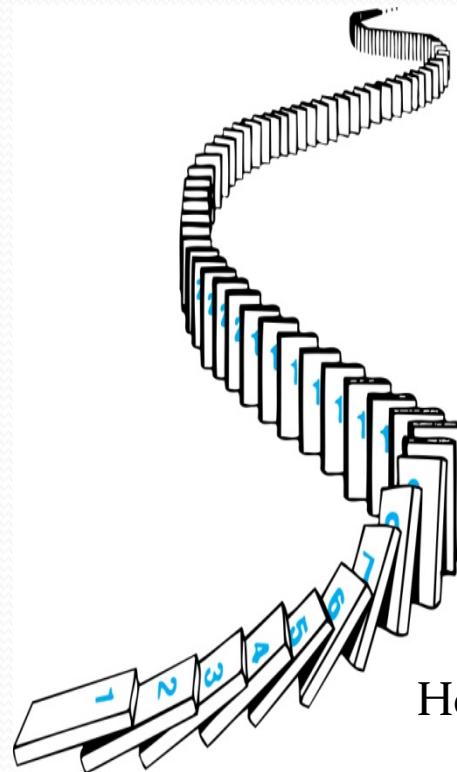
Validity of Mathematical Induction

- Mathematical induction is valid because of the well ordering property, which states that every nonempty subset of the set of positive integers has a least element (*see Section 5.2 and Appendix 1*). Here is the proof:
 - Suppose that $P(1)$ holds and $P(k) \rightarrow P(k + 1)$ is true for all positive integers k .
 - Assume there is at least one positive integer n for which $P(n)$ is false. Then the set S of positive integers for which $P(n)$ is false is nonempty.
 - By the well-ordering property, S has a least element, say m .
 - We know that m can not be 1 since $P(1)$ holds.
 - Since m is positive and greater than 1, $m - 1$ must be a positive integer. Since $m - 1 < m$, it is not in S , so $P(m - 1)$ must be true.
 - But then, since the conditional $P(k) \rightarrow P(k + 1)$ for every positive integer k holds, $P(m)$ must also be true. This contradicts $P(m)$ being false.
 - Hence, $P(n)$ must be true for every positive integer n .

Remembering How Mathematical Induction Works

Consider an infinite sequence of dominoes, labeled 1,2,3, ..., 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 .

We also know that if whenever the k th domino is knocked over, it knocks over the $(k + 1)$ 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 .

Proving a Summation Formula by Mathematical Induction

Example: Show that: $\sum_{i=1}^n = \frac{n(n + 1)}{2}$

Solution:

- 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,

$$\begin{aligned}1 + 2 + \dots + k + (k + 1) &= \frac{k(k + 1)}{2} + (k + 1) \\&= \frac{k(k + 1) + 2(k + 1)}{2} \\&= \frac{(k + 1)(k + 2)}{2}\end{aligned}$$

Note: Once we have this conjecture, mathematical induction can be used to prove it correct.



Conjecturing and Proving Correct a Summation Formula

Example: Conjecture and prove correct a formula for the sum of the first n positive odd integers. Then prove your conjecture.

Solution: We have: $1 = 1$, $1 + 3 = 4$, $1 + 3 + 5 = 9$, $1 + 3 + 5 + 7 = 16$, $1 + 3 + 5 + 7 + 9 = 25$.

- We can conjecture that the sum of the first n positive odd integers is n^2 ,

$$1 + 3 + 5 + \dots + (2n - 1) + (2n + 1) = n^2.$$

- We prove the conjecture is proved correct with mathematical induction.
- BASIS STEP: $P(1)$ is true since $1^2 = 1$.
- INDUCTIVE STEP: $P(k) \rightarrow P(k + 1)$ for every positive integer k .

Assume the inductive hypothesis holds and then show that $P(k)$ holds has well.

Inductive Hypothesis: $1 + 3 + 5 + \dots + (2k - 1) = k^2$

- So, assuming $P(k)$, it follows that:

$$\begin{aligned}1 + 3 + 5 + \dots + (2k - 1) + (2k + 1) &= [1 + 3 + 5 + \dots + (2k - 1)] + (2k + 1) \\&= k^2 + (2k + 1) \quad (\text{by the inductive hypothesis}) \\&= k^2 + 2k + 1 \\&= (k + 1)^2\end{aligned}$$

- Hence, we have shown that $P(k + 1)$ follows from $P(k)$. Therefore the sum of the first n positive odd integers is n^2 .



Proving Inequalities

Example: Use mathematical induction to prove that $n < 2^n$ for all positive integers n .

Solution: Let $P(n)$ be the proposition that $n < 2^n$.

- BASIS STEP: $P(1)$ is true since $1 < 2^1 = 2$.
- INDUCTIVE STEP: Assume $P(k)$ holds, i.e., $k < 2^k$, for an arbitrary positive integer k .
- Must show that $P(k + 1)$ holds. Since by the inductive hypothesis, $k < 2^k$, it follows that:

$$k + 1 < 2^k + 1 \leq 2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$$

Therefore $n < 2^n$ holds for all positive integers n . ◀

Proving Inequalities

Example: Use mathematical induction to prove that $2^n < n!$, for every integer $n \geq 4$.

Solution: Let $P(n)$ be the proposition that $2^n < n!$.

- BASIS STEP: $P(4)$ is true since $2^4 = 16 < 4! = 24$.
- INDUCTIVE STEP: Assume $P(k)$ holds, i.e., $2^k < k!$ for an arbitrary integer $k \geq 4$. To show that $P(k + 1)$ holds:

$$2^{k+1} = 2 \cdot 2^k$$

$$< 2 \cdot k! \quad (\text{by the inductive hypothesis})$$

$$< (k + 1)k!$$

$$= (k + 1)!$$

Therefore, $2^n < n!$ holds, for every integer $n \geq 4$. 

Note that here the basis step is $P(4)$, since $P(0), P(1), P(2)$, and $P(3)$ are all false.

Proving Divisibility Results

Example: Use mathematical induction to prove that $n^3 - n$ is divisible by 3, for every positive integer n .

Solution: Let $P(n)$ be the proposition that $n^3 - n$ is divisible by 3.

- BASIS STEP: $P(1)$ is true since $1^3 - 1 = 0$, which is divisible by 3.
- INDUCTIVE STEP: Assume $P(k)$ holds, i.e., $k^3 - k$ is divisible by 3, for an arbitrary positive integer k . To show that $P(k + 1)$ follows:

$$\begin{aligned}(k + 1)^3 - (k + 1) &= (k^3 + 3k^2 + 3k + 1) - (k + 1) \\ &= (k^3 - k) + 3(k^2 + k)\end{aligned}$$

By the inductive hypothesis, the first term $(k^3 - k)$ is divisible by 3 and the second term is divisible by 3 since it is an integer multiplied by 3. So by part (i) of Theorem 1 in Section 4.1 , $(k + 1)^3 - (k + 1)$ is divisible by 3.

Therefore, $n^3 - n$ is divisible by 3, for every integer positive integer n . ◀

Number of Subsets of a Finite Set

Example: Use mathematical induction to show that if S is a finite set with n elements, where n is a nonnegative integer, then S has 2^n subsets.

(Chapter 6 uses combinatorial methods to prove this result.)

Solution: Let $P(n)$ be the proposition that a set with n elements has 2^n subsets.

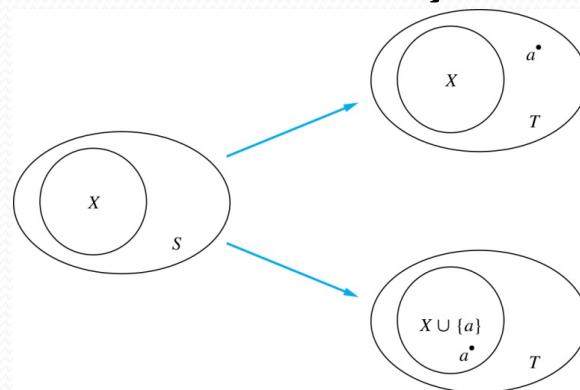
- Basis Step: $P(0)$ is true, because the empty set has only itself as a subset and $2^0 = 1$.
- Inductive Step: Assume $P(k)$ is true for an arbitrary nonnegative integer k .

continued →

Number of Subsets of a Finite Set

Inductive Hypothesis: For an arbitrary nonnegative integer k , every set with k elements has 2^k subsets.

- Let T be a set with $k + 1$ elements. Then $T = S \cup \{a\}$, where $a \in T$ and $S = T - \{a\}$. Hence $|T| = k$.
- For each subset X of S , there are exactly two subsets of T , i.e., X and $X \cup \{a\}$.



- By the inductive hypothesis S has 2^k subsets. Since there are two subsets of T for each subset of S , the number of subsets of T is $2 \cdot 2^k = 2^{k+1}$.



Tiling Checkerboards

Example: Show that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes.

A right triomino is an L-shaped tile which covers three squares at a time.



Solution: Let $P(n)$ be the proposition that every $2^n \times 2^n$ checkerboard with one square removed can be tiled using right triominoes. Use mathematical induction to prove that $P(n)$ is true for all positive integers n .

- BASIS STEP: $P(1)$ is true, because each of the four 2×2 checkerboards with one square removed can be tiled using one right triomino.



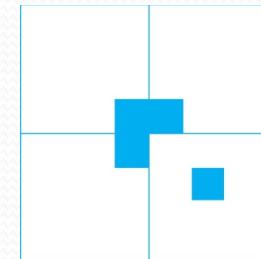
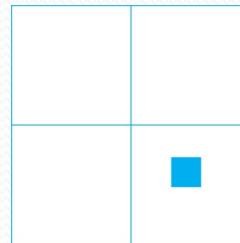
- INDUCTIVE STEP: Assume that $P(k)$ is true for every $2^k \times 2^k$ checkerboard, for some positive integer k .

continued →

Tiling Checkerboards

Inductive Hypothesis: Every $2^k \times 2^k$ checkerboard, for some positive integer k , with one square removed can be tiled using right triominoes.

- Consider a $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed. Split this checkerboard into four checkerboards of size $2^k \times 2^k$, by dividing it in half in both directions.



- Remove a square from one of the four $2^k \times 2^k$ checkerboards. By the inductive hypothesis, this board can be tiled. Also by the inductive hypothesis, the other three boards can be tiled with the square from the corner of the center of the original board removed. We can then cover the three adjacent squares with a triominoe.
- Hence, the entire $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed can be tiled using right triominoes.



An Incorrect “Proof” by Mathematical Induction

Example: Let $P(n)$ be the statement that every set of n lines in the plane, no two of which are parallel, meet in a common point. Here is a “proof” that $P(n)$ is true for all positive integers $n \geq 2$.

- BASIS STEP: The statement $P(2)$ is true because any two lines in the plane that are not parallel meet in a common point.
- INDUCTIVE STEP: The inductive hypothesis is the statement that $P(k)$ is true for the positive integer $k \geq 2$, i.e., every set of k lines in the plane, no two of which are parallel, meet in a common point.
- We must show that if $P(k)$ holds, then $P(k + 1)$ holds, i.e., if every set of k lines in the plane, no two of which are parallel, $k \geq 2$, meet in a common point, then every set of $k + 1$ lines in the plane, no two of which are parallel, meet in a common point.

continued →

An Incorrect “Proof” by Mathematical Induction

Inductive Hypothesis: Every set of k lines in the plane, where $k \geq 2$, no two of which are parallel, meet in a common point.

- Consider a set of $k + 1$ distinct lines in the plane, no two parallel. By the inductive hypothesis, the first k of these lines must meet in a common point p_1 . By the inductive hypothesis, the last k of these lines meet in a common point p_2 .
- If p_1 and p_2 are different points, all lines containing both of them must be the same line since two points determine a line. This contradicts the assumption that the lines are distinct. Hence, $p_1 = p_2$ lies on all $k + 1$ distinct lines, and therefore $P(k + 1)$ holds. Assuming that $k \geq 2$, distinct lines meet in a common point, then every $k + 1$ lines meet in a common point.
- There must be an error in this proof since the conclusion is absurd. But where is the error?
 - **Answer:** $P(k) \rightarrow P(k + 1)$ only holds for $k \geq 3$. It is not the case that $P(2)$ implies $P(3)$. The first two lines must meet in a common point p_1 and the second two must meet in a common point p_2 . They do not have to be the same point since only the second line is common to both sets of lines.

Guidelines: Mathematical Induction Proofs

Template for Proofs by Mathematical Induction

1. Express the statement that is to be proved in the form “for all $n \geq b$, $P(n)$ ” for a fixed integer b .
2. Write out the words “Basis Step.” Then show that $P(b)$ is true, taking care that the correct value of b is used. This completes the first part of the proof.
3. Write out the words “Inductive Step.”
4. State, and clearly identify, the inductive hypothesis, in the form “assume that $P(k)$ is true for an arbitrary fixed integer $k \geq b$.”
5. State what needs to be proved under the assumption that the inductive hypothesis is true. That is, write out what $P(k + 1)$ says.
6. Prove the statement $P(k + 1)$ making use of the assumption $P(k)$. Be sure that your proof is valid for all integers k with $k \geq b$, taking care that the proof works for small values of k , including $k = b$.
7. Clearly identify the conclusion of the inductive step, such as by saying “this completes the inductive step.”
8. After completing the basis step and the inductive step, state the conclusion, namely that by mathematical induction, $P(n)$ is true for all integers n with $n \geq b$.

Strong Induction and Well-Ordering

Section 5.2

Section Summary

- Strong Induction
- Example Proofs using Strong Induction
- Using Strong Induction in Computational Geometry
(not yet included in overheads)
- Well-Ordering Property

Strong Induction

- *Strong Induction:* To prove that $P(n)$ is true for all positive integers n , where $P(n)$ is a propositional function, complete two steps:
 - *Basis Step:* Verify that the proposition $P(1)$ is true.
 - *Inductive Step:* Show the conditional statement $[P(1) \wedge P(2) \wedge \dots \wedge P(k)] \rightarrow P(k + 1)$ holds for all positive integers k .

Strong Induction is sometimes called the *second principle of mathematical induction* or *complete induction*.

Strong Induction and the Infinite Ladder

Strong induction tells us that we can reach all rungs if:

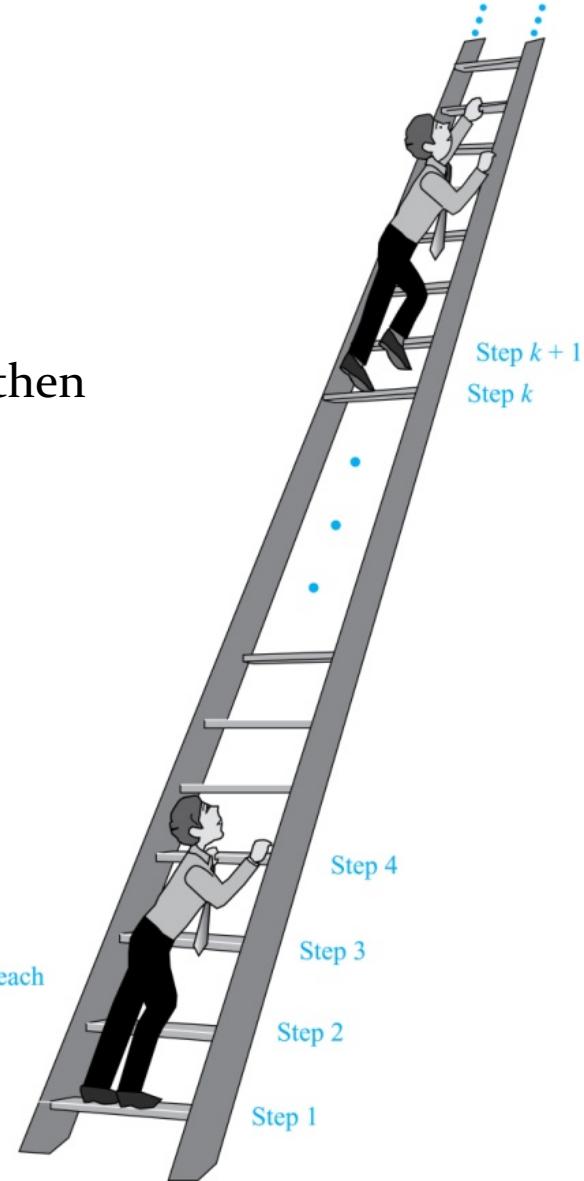
1. We can reach the first rung of the ladder.
2. For every integer k , if we can reach the first k rungs, then we can reach the $(k + 1)$ st rung.

To conclude that we can reach every rung by strong induction:

- BASIS STEP: $P(1)$ holds
- INDUCTIVE STEP: Assume $P(1) \wedge P(2) \wedge \dots \wedge P(k)$ holds for an arbitrary integer k , and show that $P(k + 1)$ must also hold.

We will have then shown by strong induction that for every positive integer n , $P(n)$ holds, i.e., we can reach the n th rung of the ladder.

We can reach
step 1



Proof using Strong Induction

Example: Suppose we can reach the first and second rungs of an infinite ladder, and we know that if we can reach a rung, then we can reach two rungs higher. Prove that we can reach every rung.

(Try this with mathematical induction.)

Solution: Prove the result using strong induction.

- **BASIS STEP:** We can reach the first step.
- **INDUCTIVE STEP:** The inductive hypothesis is that we can reach the first k rungs, for any $k \geq 2$. We can reach the $(k + 1)$ st rung since we can reach the $(k - 1)$ st rung by the inductive hypothesis.
- Hence, we can reach all rungs of the ladder. 

Which Form of Induction Should Be Used?

- We can always use strong induction instead of mathematical induction. But there is no reason to use it if it is simpler to use mathematical induction. (*See page 335 of text.*)
- In fact, the principles of mathematical induction, strong induction, and the well-ordering property are all equivalent. (*Exercises 41-43*)
- Sometimes it is clear how to proceed using one of the three methods, but not the other two.

Completion of the proof of the Fundamental Theorem of Arithmetic

Example: Show that if n is an integer greater than 1, then n can be written as the product of primes.

Solution: Let $P(n)$ be the proposition that n can be written as a product of primes.

- BASIS STEP: $P(2)$ is true since 2 itself is prime.
- INDUCTIVE STEP: The inductive hypothesis is $P(j)$ is true for all integers j with $2 \leq j \leq k$. To show that $P(k + 1)$ must be true under this assumption, two cases need to be considered:
 - If $k + 1$ is prime, then $P(k + 1)$ is true.
 - Otherwise, $k + 1$ is composite and can be written as the product of two positive integers a and b with $2 \leq a \leq b < k + 1$. By the inductive hypothesis a and b can be written as the product of primes and therefore $k + 1$ can also be written as the product of those primes.

Hence, it has been shown that every integer greater than 1 can be written as the product of primes.

(uniqueness proved in Section 4.3)



Proof using Strong Induction

Example: Prove that every amount of postage of 12 cents or more can be formed using just 4-cent and 5-cent stamps.

Solution: Let $P(n)$ be the proposition that postage of n cents can be formed using 4-cent and 5-cent stamps.

- BASIS STEP: $P(12)$, $P(13)$, $P(14)$, and $P(15)$ hold.
 - $P(12)$ uses three 4-cent stamps.
 - $P(13)$ uses two 4-cent stamps and one 5-cent stamp.
 - $P(14)$ uses one 4-cent stamp and two 5-cent stamps.
 - $P(15)$ uses three 5-cent stamps.
- INDUCTIVE STEP: The inductive hypothesis states that $P(j)$ holds for $12 \leq j \leq k$, where $k \geq 15$. Assuming the inductive hypothesis, it can be shown that $P(k + 1)$ holds.
- Using the inductive hypothesis, $P(k - 3)$ holds since $k - 3 \geq 12$. To form postage of $k + 1$ cents, add a 4-cent stamp to the postage for $k - 3$ cents.

Hence, $P(n)$ holds for all $n \geq 12$.



Proof of Same Example using Mathematical Induction

Example: Prove that every amount of postage of 12 cents or more can be formed using just 4-cent and 5-cent stamps.

Solution: Let $P(n)$ be the proposition that postage of n cents can be formed using 4-cent and 5-cent stamps.

- BASIS STEP: Postage of 12 cents can be formed using three 4-cent stamps.
- INDUCTIVE STEP: The inductive hypothesis $P(k)$ for any positive integer k is that postage of k cents can be formed using 4-cent and 5-cent stamps. To show $P(k + 1)$ where $k \geq 12$, we consider two cases:
 - If at least one 4-cent stamp has been used, then a 4-cent stamp can be replaced with a 5-cent stamp to yield a total of $k + 1$ cents.
 - Otherwise, no 4-cent stamp have been used and at least three 5-cent stamps were used. Three 5-cent stamps can be replaced by four 4-cent stamps to yield a total of $k + 1$ cents.

Hence, $P(n)$ holds for all $n \geq 12$.



Well-Ordering Property

- *Well-ordering property:* Every nonempty set of nonnegative integers has a least element.
- The well-ordering property is one of the axioms of the positive integers listed in Appendix 1.
- The well-ordering property can be used directly in proofs, as the next example illustrates.
- The well-ordering property can be generalized.
 - **Definition:** A set is *well ordered* if every subset has a least element.
 - \mathbb{N} is well ordered under \leq .
 - The set of finite strings over an alphabet using lexicographic ordering is well ordered.
 - We will see a generalization of induction to sets other than the integers in the next section.

Well-Ordering Property

Example: Use the well-ordering property to prove the division algorithm, which states that if a is an integer and d is a positive integer, then there are unique integers q and r with $0 \leq r < d$, such that $a = dq + r$.

Solution: Let S be the set of nonnegative integers of the form $a - dq$, where q is an integer. The set is nonempty since $-dq$ can be made as large as needed.

- By the well-ordering property, S has a least element $r = a - dq_0$. The integer r is nonnegative. It also must be the case that $r < d$. If it were not, then there would be a smaller nonnegative element in S , namely,
$$a - d(q_0 + 1) = a - dq_0 - d = r - d > 0.$$
- Therefore, there are integers q and r with $0 \leq r < d$.
(uniqueness of q and r is Exercise 37)



Recursive Definitions and Structural Induction

Section 5.3

Section Summary

- Recursively Defined Functions
- Recursively Defined Sets and Structures
- Structural Induction
- Generalized Induction

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 $f(i) = a_i$. This was done using recurrence relations in Section 2.4.

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:

- $f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9$
- $f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21$
- $f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45$
- $f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93$

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
(1170- 1250)



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 .

- $f_2 = f_1 + f_0 = 1 + 0 = 1$
- $f_3 = f_2 + f_1 = 1 + 1 = 2$
- $f_4 = f_3 + f_2 = 2 + 1 = 3$
- $f_5 = f_4 + f_3 = 3 + 2 = 5$

In Chapter 8, we will use the Fibonacci numbers to model population growth of rabbits. This was an application described by Fibonacci himself.

Next, we use strong induction to prove a result about the Fibonacci numbers.

Fibonacci Numbers

Example 4: Show that whenever $n \geq 3$, $f_n > \alpha^{n-2}$, where $\alpha = (1 + \sqrt{5})/2$.

Solution: Let $P(n)$ be the statement $f_n > \alpha^{n-2}$. Use strong induction to show that $P(n)$ is true whenever $n \geq 3$.

- BASIS STEP: $P(3)$ holds since $\alpha < 2 = f_3$
 $P(4)$ holds since $\alpha^2 = (3 + \sqrt{5})/2 < 3 = f_4$.
- INDUCTIVE STEP: Assume that $P(j)$ holds, i.e., $f_j > \alpha^{j-2}$ for all integers j with $3 \leq j \leq k$, where $k \geq 4$. Show that $P(k+1)$ holds, i.e., $f_{k+1} > \alpha^{k-1}$.
 - Since $\alpha^2 = \alpha + 1$ (because α is a solution of $x^2 - x - 1 = 0$),

$$\alpha^{k-1} = \alpha^2 \cdot \alpha^{k-3} = (\alpha + 1) \cdot \alpha^{k-3} = \alpha \cdot \alpha^{k-3} + 1 \cdot \alpha^{k-3} = \alpha^{k-2} + \alpha^{k-3}$$

- By the inductive hypothesis, because $k \geq 4$ we have

$$f_{k-1} > \alpha^{k-3}, \quad f_{k-2} > \alpha^{k-2}.$$

- Therefore, it follows that

$$f_{k+1} = f_{k+1} + f_{k-1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}.$$

- Hence, $P(k+1)$ is true.

Why does
this equality
hold?



Gabriel Lamé
(1795-1870)



Lamé's Theorem

Lamé's Theorem: Let a and b be positive integers with $a \geq b$. Then the number of divisions used by the Euclidian algorithm to find $\gcd(a,b)$ is less than or equal to five times the number of decimal digits in b .

Proof: When we use the Euclidian algorithm to find $\gcd(a,b)$ with $a \geq b$,

- n divisions are used to obtain
(with $a = r_0, b = r_1$):

$$\begin{aligned} r_0 &= r_1 q_1 + r_2 & 0 \leq r_2 < r_1, \\ r_1 &= r_2 q_2 + r_3 & 0 \leq r_3 < r_2, \\ &\vdots & \\ r_{n-2} &= r_{n-1} q_{n-1} + r_n & 0 \leq r_n < r_{n-1}, \\ r_{n-1} &= r_n q_n. & \end{aligned}$$

- Since each quotient q_1, q_2, \dots, q_{n-1} is at least 1 and $q_n \geq 2$:

$$\begin{aligned} r_n &\geq 1 = f_2, \\ r_{n-1} &\geq 2 \quad r_n \geq 2 f_2 = f_3, \\ r_{n-2} &\geq r_{n-1} + r_n \geq f_3 + f_2 = f_4, \\ &\vdots \\ r_2 &\geq r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n, \\ b = r_1 &\geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1}. \end{aligned}$$

continued →

Lamé's Theorem

- It follows that if n divisions are used by the Euclidian algorithm to find $\gcd(a,b)$ with $a \geq b$, then $b \geq f_{n+1}$. By Example 4, $f_{n+1} > \alpha^{n-1}$, for $n > 2$, where $\alpha = (1 + \sqrt{5})/2$. Therefore, $b > \alpha^{n-1}$.
- Because $\log_{10} \alpha \approx 0.208 > 1/5$, $\log_{10} b > (n-1) \log_{10} \alpha > (n-1)/5$. Hence,

$$n-1 < 5 \cdot \log_{10} b.$$

- Suppose that b has k decimal digits. Then $b < 10^k$ and $\log_{10} b < k$. It follows that $n - 1 < 5k$ and since k is an integer, $n \leq 5k$. 
- As a consequence of Lamé's Theorem, $O(\log b)$ divisions are used by the Euclidian algorithm to find $\gcd(a,b)$ whenever $a > b$.
 - By Lamé's Theorem, the number of divisions needed to find $\gcd(a,b)$ with $a > b$ is less than or equal to $5 (\log_{10} b + 1)$ since the number of decimal digits in b (which equals $\lfloor \log_{10} b \rfloor + 1$) is less than or equal to $\log_{10} b + 1$.

Lamé's Theorem was the first result in computational complexity

Recursively Defined Sets and Structures

Recursive definitions of sets have two parts:

- The *basis step* specifies an initial collection of elements.
- The *recursive step* gives the rules for forming new elements in the set from those already known to be in the set.
- Sometimes the recursive definition has an *exclusion rule*, which specifies that the set contains nothing other than those elements specified in the basis step and generated by applications of the rules in the recursive step.
- We will always assume that the exclusion rule holds, even if it is not explicitly mentioned.
- We will later develop a form of induction, called *structural induction*, to prove results about recursively defined sets.

Recursively Defined Sets and Structures

Example : Subset of Integers S :

BASIS STEP: $3 \in S$.

RECURSIVE STEP: If $x \in S$ and $y \in S$, then $x + y$ is in S .

- Initially 3 is in S , then $3 + 3 = 6$, then $3 + 6 = 9$, etc.

Example: The natural numbers \mathbf{N} .

BASIS STEP: $0 \in \mathbf{N}$.

RECURSIVE STEP: If n is in \mathbf{N} , then $n + 1$ is in \mathbf{N} .

- Initially 0 is in S , then $0 + 1 = 1$, then $1 + 1 = 2$, etc.

Strings

Definition: The set Σ^* of *strings* over the alphabet Σ :

BASIS STEP: $\lambda \in \Sigma^*$ (λ is the empty string)

RECURSIVE STEP: If w is in Σ^* and x is in Σ ,
then $wx \in \Sigma^*$.

Example: If $\Sigma = \{0,1\}$, the strings in Σ^* are the set of
all bit strings, $\lambda, 0, 1, 00, 01, 10, 11$, etc.

Example: If $\Sigma = \{a,b\}$, show that aab is in Σ^* .

- Since $\lambda \in \Sigma^*$ and $a \in \Sigma$, $a \in \Sigma^*$.
- Since $a \in \Sigma^*$ and $a \in \Sigma$, $aa \in \Sigma^*$.
- Since $aa \in \Sigma^*$ and $b \in \Sigma$, $aab \in \Sigma^*$.

String Concatenation

Definition: Two strings can be combined via the operation of *concatenation*. Let Σ be a set of symbols and Σ^* be the set of strings formed from the symbols in Σ . We can define the concatenation of two strings, denoted by \cdot , recursively as follows.

BASIS STEP: If $w \in \Sigma^*$, then $w \cdot \lambda = w$.

RECURSIVE STEP: If $w_1 \in \Sigma^*$ and $w_2 \in \Sigma^*$ and $x \in \Sigma$, then $w \cdot (w_2 x) = (w_1 \cdot w_2)x$.

- Often $w_1 \cdot w_2$ is written as $w_1 w_2$.
- If $w_1 = abra$ and $w_2 = cadabra$, the concatenation $w_1 w_2 = abracadabra$.

Length of a String

Example: Give a recursive definition of $l(w)$, the length of the string w .

Solution: The length of a string can be recursively defined by:

$$l(w) = 0;$$

$$l(wx) = l(w) + 1 \text{ if } w \in \Sigma^* \text{ and } x \in \Sigma.$$

Balanced Parentheses

Example: Give a recursive definition of the set of balanced parentheses P .

Solution:

BASIS STEP: $() \in P$

RECURSIVE STEP: If $w \in P$, then $(w) \in P$, $(w) \in P$ and $w() \in P$.

- Show that $((())()$ is in P .
- Why is $))((()$ not in P ?

Well-Formed Formulae in Propositional Logic

Definition: The set of *well-formed formulae* in propositional logic involving T, F, propositional variables, and operators from the set $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

BASIS STEP: T, F, and s , where s is a propositional variable, are well-formed formulae.

RECURSIVE STEP: If E and F are well formed formulae, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$, $(E \leftrightarrow F)$, are well-formed formulae.

Examples: $((p \vee q) \rightarrow (q \wedge F))$ is a well-formed formula.
 $p q \wedge$ is not a well formed formula.

Rooted Trees

Definition: The set of *rooted trees*, where a rooted tree consists of a set of vertices containing a distinguished vertex called the *root*, and edges connecting these vertices, can be defined recursively by these steps:

BASIS STEP: A single vertex r is a rooted tree.

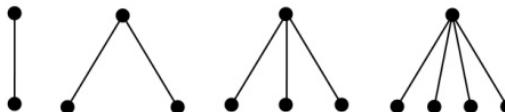
RECURSIVE STEP: Suppose that T_1, T_2, \dots, T_n are disjoint rooted trees with roots r_1, r_2, \dots, r_n , respectively. Then the graph formed by starting with a root r , which is not in any of the rooted trees T_1, T_2, \dots, T_n , and adding an edge from r to each of the vertices r_1, r_2, \dots, r_n , is also a rooted tree.

Building Up Rooted Trees

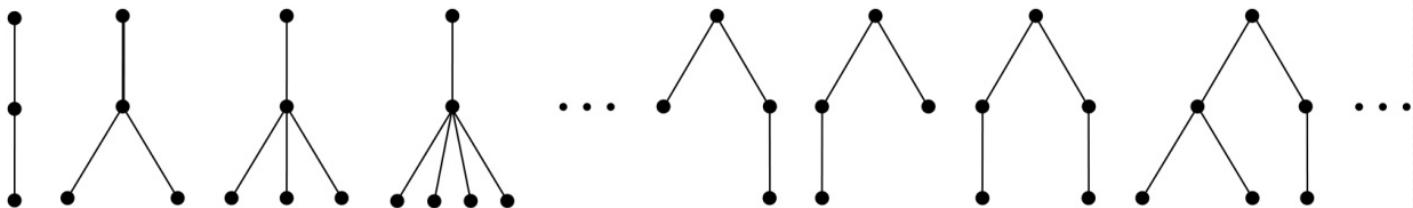
Basis step



Step 1



Step 2



- Trees are studied extensively in Chapter 11.
- Next we look at a special type of tree, the full binary tree.

Full Binary Trees

Definition: The set of *full binary trees* can be defined recursively by these steps.

BASIS STEP: There is a full binary tree consisting of only a single vertex r .

RECURSIVE STEP: If T_1 and T_2 are disjoint full binary trees, there is a full binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2 .

Building Up Full Binary Trees

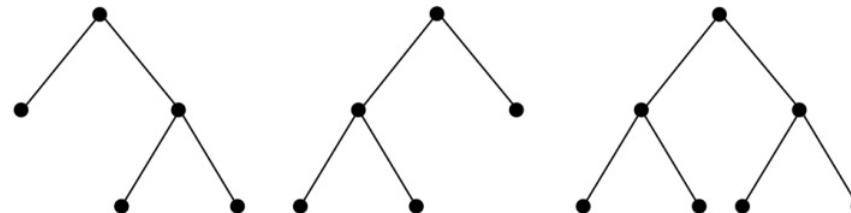
Basis step



Step 1



Step 2



Induction and Recursively Defined Sets

Example: Show that the set S defined by specifying that $3 \in S$ and that if $x \in S$ and $y \in S$, then $x + y$ is in S , is the set of all positive integers that are multiples of 3.

Solution: Let A be the set of all positive integers divisible by 3. To prove that $A = S$, show that A is a subset of S and S is a subset of A .

- $A \subset S$: Let $P(n)$ be the statement that $3n$ belongs to S .

BASIS STEP: $3 \cdot 1 = 3 \in S$, by the first part of recursive definition.

INDUCTIVE STEP: Assume $P(k)$ is true. By the second part of the recursive definition, if $3k \in S$, then since $3 \in S$, $3k + 3 = 3(k + 1) \in S$. Hence, $P(k + 1)$ is true.

- $S \subset A$:

BASIS STEP: $3 \in S$ by the first part of recursive definition, and $3 = 3 \cdot 1$.

INDUCTIVE STEP: The second part of the recursive definition adds $x + y$ to S , if both x and y are in S . If x and y are both in A , then both x and y are divisible by 3. By part (i) of Theorem 1 of Section 4.1, it follows that $x + y$ is divisible by 3.

- We used mathematical induction to prove a result about a recursively defined set. Next we study a more direct form induction for proving results about recursively defined sets.

Structural Induction

Definition: To prove a property of the elements of a recursively defined set, we use *structural induction*.

BASIS STEP: Show that the result holds for all elements specified in the basis step of the recursive definition.

RECURSIVE STEP: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

- The validity of structural induction can be shown to follow from the principle of mathematical induction.

Full Binary Trees

Definition: The *height* $h(T)$ of a full binary tree T is defined recursively as follows:

- **BASIS STEP:** The height of a full binary tree T consisting of only a root r is $h(T) = 0$.
- **RECURSIVE STEP:** If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has height $h(T) = 1 + \max(h(T_1), h(T_2))$.
- The number of vertices $n(T)$ of a full binary tree T satisfies the following recursive formula:
 - **BASIS STEP:** The number of vertices of a full binary tree T consisting of only a root r is $n(T) = 1$.
 - **RECURSIVE STEP:** If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has the number of vertices $n(T) = 1 + n(T_1) + n(T_2)$.

Structural Induction and Binary Trees

Theorem: If T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.

Proof: Use structural induction.

- **BASIS STEP:** The result holds for a full binary tree consisting only of a root, $n(T) = 1$ and $h(T) = 0$. Hence, $n(T) = 1 \leq 2^{0+1} - 1 = 1$.
- **RECURSIVE STEP:** Assume $n(T_1) \leq 2^{h(T_1)+1} - 1$ and also $n(T_2) \leq 2^{h(T_2)+1} - 1$ whenever T_1 and T_2 are full binary trees.

$$\begin{aligned} n(T) &= 1 + n(T_1) + n(T_2) && (\text{by recursive formula of } n(T)) \\ &\leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) && (\text{by inductive hypothesis}) \\ &\leq 2 \cdot \max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 \\ &= 2 \cdot 2^{\max(h(T_1), h(T_2))+1} - 1 && (\max(2^x, 2^y) = 2^{\max(x, y)}) \\ &= 2 \cdot 2^{h(t)} - 1 && (\text{by recursive definition of } h(T)) \\ &= 2^{h(t)+1} - 1 \end{aligned}$$

Generalized Induction

- *Generalized induction* is used to prove results about sets other than the integers that have the well-ordering property. (*explored in more detail in Chapter 9*)
- For example, consider an ordering on $\mathbf{N} \times \mathbf{N}$, ordered pairs of nonnegative integers. Specify that (x_1, y_1) is less than or equal to (x_2, y_2) if either $x_1 < x_2$, or $x_1 = x_2$ and $y_1 < y_2$. This is called the *lexicographic ordering*.
- Strings are also commonly ordered by a *lexicographic ordering*.
- The next example uses generalized induction to prove a result about ordered pairs from $\mathbf{N} \times \mathbf{N}$.

Generalized Induction

Example: Suppose that $a_{m,n}$ is defined for $(m,n) \in \mathbb{N} \times \mathbb{N}$ by $a_{0,0} = 0$ and

$$a_{m,n} = \begin{cases} a_{m-1,n} + 1 & \text{if } n = 0 \text{ and } m > 0 \\ a_{m,n-1} + n & \text{if } n > 0 \end{cases} .$$

Show that $a_{m,n} = m + n(n + 1)/2$ is defined for all $(m,n) \in \mathbb{N} \times \mathbb{N}$.

Solution: Use generalized induction.

BASIS STEP: $a_{0,0} = 0 = 0 + (0 \cdot 1)/2$

INDUCTIVE STEP: Assume that $a_{m',n'} = m' + n'(n' + 1)/2$

whenever (m',n') is less than (m,n) in the lexicographic ordering of $\mathbb{N} \times \mathbb{N}$.

- If $n = 0$, by the inductive hypothesis we can conclude

$$a_{m,n} = a_{m-1,n} + 1 = m - 1 + n(n + 1)/2 + 1 = m + n(n + 1)/2 .$$

- If $n > 0$, by the inductive hypothesis we can conclude

$$a_{m,n} = a_{m-1,n} + 1 = m + n(n - 1)/2 + n = m + n(n + 1)/2 .$$



Recursive Algorithms

Section 5.4

Section Summary

- Recursive Algorithms
- Proving Recursive Algorithms Correct
- Recursion and Iteration (*not yet included in overheads*)
- Merge Sort

Recursive Algorithms

Definition: An algorithm is called *recursive* if it solves a problem by reducing it to an instance of the same problem with smaller input.

- For the algorithm to terminate, the instance of the problem must eventually be reduced to some initial case for which the solution is known.

Recursive Factorial Algorithm

Example: Give a recursive algorithm for computing $n!$, where n is a nonnegative integer.

- **Solution:** Use the recursive definition of the factorial function.

```
procedure factorial(n: nonnegative integer)
  if n = 0 then return 1
  else return n·(n – 1)
  {output is n!}
```

Recursive Exponentiation Algorithm

Example: Give a recursive algorithm for computing a^n , where a is a nonzero real number and n is a nonnegative integer.

Solution: Use the recursive definition of a^n .

```
procedure power(a: nonzero real number, n: nonnegative integer)
if n = 0 then return 1
else return a·power (a, n – 1)
{output is  $a^n$ }
```

Recursive GCD Algorithm

Example: Give a recursive algorithm for computing the greatest common divisor of two nonnegative integers a and b with $a < b$.

Solution: Use the reduction

$$\gcd(a,b) = \gcd(b \bmod a, a)$$

and the condition $\gcd(0,b) = b$ when $b > 0$.

```
procedure gcd(a,b: nonnegative integers  
           with  $a < b$ )
```

```
if  $a = 0$  then return  $b$ 
```

```
else return  $\gcd(b \bmod a, a)$ 
```

```
{output is  $\gcd(a, b)$ }
```

Recursive Modular Exponentiation Algorithm

Example: Devise a recursive algorithm for computing $b^n \bmod m$, where b , n , and m are integers with $m \geq 2$, $n \geq 0$, and $1 \leq b \leq m$.

- **Solution:** *(see text for full explanation)*

```
procedure mpower(b,m,n: integers with  $b > 0$  and  $m \geq 2$ ,  $n \geq 0$ )
if  $n = 0$  then
    return 1
else if  $n$  is even then
    return mpower( $b, n/2, m$ )2 mod  $m$ 
else
    return ( $mpower(b, \lfloor n/2 \rfloor, m)$ 2 mod  $m \cdot b \bmod m$ ) mod  $m$ 
{output is  $b^n \bmod m$ }
```

Recursive Binary Search Algorithm

Example: Construct a recursive version of a binary search algorithm.

Solution: Assume we have a_1, a_2, \dots, a_n , an increasing sequence of integers. Initially i is 1 and j is n . We are searching for x .

```
procedure binary search(i, j, x : integers, 1 ≤ i ≤ j ≤ n)
  m := ⌊(i + j)/2⌋
  if x = am then
    return m
  else if (x < am and i < m) then
    return binary search(i, m - 1, x)
  else if (x > am and j > m) then
    return binary search(m + 1, j, x)
  else return 0
{output is location of x in a1, a2, ..., an if it appears, otherwise 0}
```

Proving Recursive Algorithms Correct

- Both mathematical and strong induction are useful techniques to show that recursive algorithms always produce the correct output.

Example: Prove that the algorithm for computing the powers of real numbers is correct.

```
procedure power(a: nonzero real number, n: nonnegative integer)
if n = 0 then return 1
else return a·power (a, n – 1)
{output is  $a^n$ }
```

Solution: Use mathematical induction on the exponent *n*.

BASIS STEP: $a^0 = 1$ for every nonzero real number *a*, and $\text{power}(a, 0) = 1$.

INDUCTIVE STEP: The inductive hypothesis is that $\text{power}(a, k) = a^k$, for all $a \neq 0$.
Assuming the inductive hypothesis, the algorithm correctly computes a^{k+1} , since

$$\text{power}(a, k + 1) = a \cdot \text{power} (a, k) = a \cdot a^k = a^{k+1}.$$



Merge Sort

- *Merge Sort* works by iteratively splitting a list (with an even number of elements) into two sublists of equal length until each sublist has one element.
- Each sublist is represented by a balanced binary tree.
- At each step a pair of sublists is successively merged into a list with the elements in increasing order. The process ends when all the sublists have been merged.
- The succession of merged lists is represented by a binary tree.

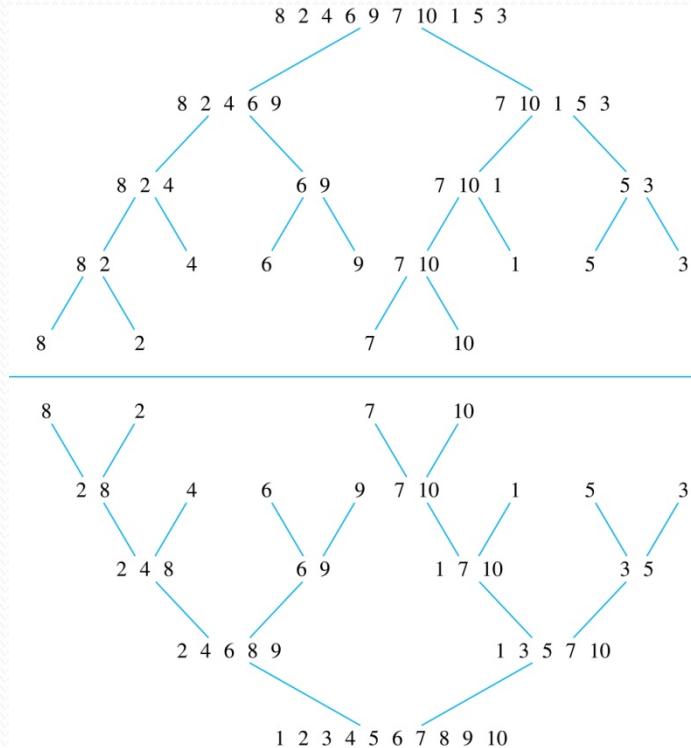
Merge Sort

Example: Use merge sort to put the list

8,2,4,6,9,7,10, 1, 5, 3

into increasing order.

Solution:



Recursive Merge Sort

Example: Construct a recursive merge sort algorithm.

Solution: Begin with the list of n elements L .

```
procedure mergesort( $L = a_1, a_2, \dots, a_n$ )
if  $n > 1$  then
     $m := \lfloor n/2 \rfloor$ 
     $L_1 := a_1, a_2, \dots, a_m$ 
     $L_2 := a_{m+1}, a_{m+2}, \dots, a_n$ 
     $L := \text{merge}(\text{mergesort}(L_1), \text{mergesort}(L_2))$ 
{ $L$  is now sorted into elements in increasing order}
```

continued →

Recursive Merge Sort

- Subroutine *merge*, which merges two sorted lists.

```
procedure merge( $L_1, L_2$  :sorted lists)
 $L :=$  empty list
while  $L_1$  and  $L_2$  are both nonempty
    remove smaller of first elements of  $L_1$  and  $L_2$  from its list;
        put at the right end of  $L$ 
    if this removal makes one list empty
        then remove all elements from the other list and append them to  $L$ 
return  $L$  { $L$  is the merged list with the elements in increasing order}
```

Complexity of Merge: Two sorted lists with m elements and n elements can be merged into a sorted list using no more than $m + n - 1$ comparisons.

Merging Two Lists

Example: Merge the two lists 2,3,5,6 and 1,4.

Solution:

TABLE 1 Merging the Two Sorted Lists 2, 3, 5, 6 and 1, 4.

<i>First List</i>	<i>Second List</i>	<i>Merged List</i>	<i>Comparison</i>
2 3 5 6	1 4		1 < 2
2 3 5 6	4	1	2 < 4
3 5 6	4	1 2	3 < 4
5 6	4	1 2 3	4 < 5
5 6		1 2 3 4	
		1 2 3 4 5 6	

Complexity of Merge Sort

Complexity of Merge Sort: The number of comparisons needed to merge a list with n elements is $O(n \log n)$.

- For simplicity, assume that n is a power of 2, say 2^m .
- At the end of the splitting process, we have a binary tree with m levels, and 2^m lists with one element at level m .
- The merging process begins at level m with the pairs of 2^m lists with one element combined into 2^{m-1} lists of two elements. Each merger takes two one comparison.
- The procedure continues , at each level ($k = m, m-1, m-1, \dots, 3, 2, 1$) 2^k lists with 2^{m-k} elements are merged into 2^{k-1} lists, with 2^{m-k+1} elements at level $k-1$.
 - We know (by the complexity of the merge subroutine) that each merger takes at most $2^{m-k} + 2^{m-k} - 1 = 2^{m-k+1} - 1$ comparisons.

continued →

Complexity of Merge Sort

- Summing over the number of comparisons at each level, shows that

$$\sum_{k=1}^m 2^{k-1}(2^{m-k+1} - 1) = \sum_{k=1}^m 2^m - \sum_{k=1}^m 2^{k-1} = m2^m - (2^m - 1) = n \log n - n + 1,$$

because $m = \log n$ and $n = 2^m$.

(The expression $\sum_{k=1}^m 2^{k-1}$ in the formula above is evaluated as $2^m - 1$ using the formula for the sum of the terms of a geometric progression, from Section 2.4.)

- In Chapter 11, we'll see that the fastest comparison-based sorting algorithms have $O(n \log n)$ time complexity. So, merge sort achieves the best possible big-O estimate of time complexity.

Counting

Chapter 6

With Question/Answer Animations

Chapter Summary

- The Basics of Counting
- The Pigeonhole Principle
- Permutations and Combinations
- Binomial Coefficients and Identities
- Generalized Permutations and Combinations

The Basics of Counting

Section 6.1

Section Summary

- The Product Rule
- The Sum Rule
- The Subtraction Rule
- The Division Rule
- Examples, Examples, and Examples
- Tree Diagrams

Basic Counting Principles: The Product Rule

The Product Rule: A procedure can be broken down into a sequence of two tasks. There are n_1 ways to do the first task and n_2 ways to do the second task. Then there are $n_1 \cdot n_2$ ways to do the procedure.

Example: How many bit strings of length seven are there?

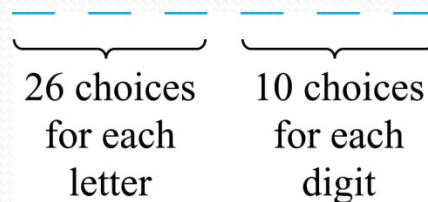
Solution: Since each of the seven bits is either a 0 or a 1, the answer is $2^7 = 128$.

The Product Rule

Example: How many different license plates can be made if each plate contains a sequence of three uppercase English letters followed by three digits?

Solution: By the product rule,

there are $26 \cdot 26 \cdot 26 \cdot 10 \cdot 10 \cdot 10 = 17,576,000$ different possible license plates.



Counting Functions

Counting Functions: How many functions are there from a set with m elements to a set with n elements?

Solution: Since a function represents a choice of one of the n elements of the codomain for each of the m elements in the domain, the product rule tells us that there are $n \cdot n \cdots n = n^m$ such functions.

Counting One-to-One Functions: How many one-to-one functions are there from a set with m elements to one with n elements?

Solution: Suppose the elements in the domain are a_1, a_2, \dots, a_m . There are n ways to choose the value of a_1 and $n-1$ ways to choose a_2 , etc. The product rule tells us that there are $n(n-1)(n-2)\cdots(n-m+1)$ such functions.

Telephone Numbering Plan

Example: The *North American numbering plan (NANP)* specifies that a telephone number consists of 10 digits, consisting of a three-digit area code, a three-digit office code, and a four-digit station code. There are some restrictions on the digits.

- Let X denote a digit from 0 through 9.
- Let N denote a digit from 2 through 9.
- Let Y denote a digit that is 0 or 1.
- In the old plan (in use in the 1960s) the format was $NYX-NNX-XXX$.
- In the new plan, the format is $XXX-XXX-XXX$.

How many different telephone numbers are possible under the old plan and the new plan?

Solution: Use the Product Rule.

- There are $8 \cdot 2 \cdot 10 = 160$ area codes with the format NYX .
- There are $8 \cdot 10 \cdot 10 = 800$ area codes with the format NNX .
- There are $8 \cdot 8 \cdot 10 = 640$ office codes with the format NNX .
- There are $10 \cdot 10 \cdot 10 \cdot 10 = 10,000$ station codes with the format $XXXX$.

Number of old plan telephone numbers: $160 \cdot 640 \cdot 10,000 = 1,024,000,000$.

Number of new plan telephone numbers: $800 \cdot 800 \cdot 10,000 = 6,400,000,000$.

Counting Subsets of a Finite Set

Counting Subsets of a Finite Set: Use the product rule to show that the number of different subsets of a finite set S is $2^{|S|}$.

Solution: When the elements of S are listed in an arbitrary order, there is a one-to-one correspondence between subsets of S and bit strings of length $|S|$. When the i th element is in the subset, the bit string has a 1 in the i th position and a 0 otherwise.

By the product rule, there are $2^{|S|}$ such bit strings, and therefore $2^{|S|}$ subsets.

Product Rule in Terms of Sets

- 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 of each set.
- The task of choosing an element in the Cartesian product $A_1 \times A_2 \times \dots \times A_m$ is done by choosing an element in A_1 , an element in A_2 , ..., and an element in A_m .
- By the product rule, it follows that:
$$|A_1 \times A_2 \times \dots \times A_m| = |A_1| \cdot |A_2| \cdot \dots \cdot |A_m|.$$

Basic Counting Principles: The Sum Rule

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

Example: The mathematics department must choose either a student or a faculty member as a representative for a university committee. How many 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: By the sum rule it follows that there are $37 + 83 = 120$ possible ways to pick a representative.

The Sum Rule in terms of sets.

- The sum rule can be phrased in terms of sets.

$|A \cup B| = |A| + |B|$ as long as A and B are disjoint sets.

- Or more generally,

$$|A_1 \cup A_2 \cup \dots \cup A_m| = |A_1| + |A_2| + \dots + |A_m|$$

when $A_i \cap A_j = \emptyset$ for all i, j .

- The case where the sets have elements in common will be discussed when we consider the subtraction rule and taken up fully in Chapter 8.

Combining the Sum and Product Rule

Example: 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: Use the product rule.

$$26 + 26 \cdot 10 = 286$$

Counting Passwords

- Combining the sum and product rule allows us to solve more complex problems.

Example: 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. We find that:

$$P_6 = 36^6 - 26^6 = 2,176,782,336 - 308,915,776 = 1,867,866,560.$$

$$\begin{aligned}P_7 &= 36^7 - 26^7 = \\&78,364,164,096 - 8,031,810,176 = 70,332,353,920.\end{aligned}$$

$$\begin{aligned}P_8 &= 36^8 - 26^8 = \\&2,821,109,907,456 - 208,827,064,576 = 2,612,282,842,880.\end{aligned}$$

Consequently, $P = P_6 + P_7 + P_8 = 2,684,483,063,360$.

Internet Addresses

- Version 4 of the Internet Protocol (IPv4) uses 32 bits.

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			

- 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.
 - 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.

Counting Internet Addresses

Example: How many different IPv4 addresses are available for computers on the internet?

Solution: Use both the sum and the product rule. 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.
$$x_A = 127 \cdot 16,777,214 = 2,130,706,178.$$
- To find, x_B : $2^{14} = 16,384$ netids. $2^{16} - 2 = 16,534$ hostids.
$$x_B = 16,384 \cdot 16,534 = 1,073,709,056.$$
- To find, x_C : $2^{21} = 2,097,152$ netids. $2^8 - 2 = 254$ hostids.
$$x_C = 2,097,152 \cdot 254 = 532,676,608.$$
- Hence, the total number of available IPv4 addresses is

$$\begin{aligned}x &= x_A + x_B + x_C \\&= 2,130,706,178 + 1,073,709,056 + 532,676,608 \\&= 3,737,091,842.\end{aligned}$$

Not Enough Today !!

The newer IPv6 protocol solves the problem of too few addresses.

Basic Counting Principles: Subtraction Rule

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*:

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

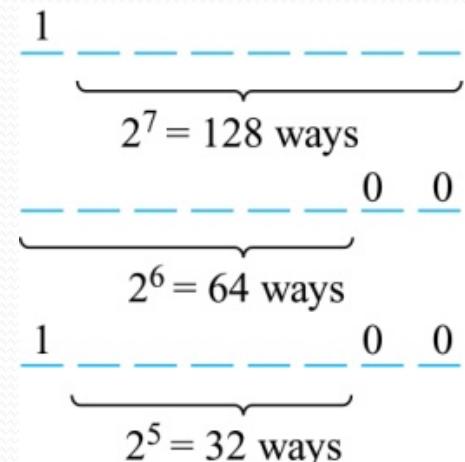
Counting Bit Strings

Example: How many bit strings of length eight either start with a 1 bit or end with the two bits 00?

Solution: Use the subtraction rule.

- Number of bit strings of length eight that start with a 1 bit: $2^7 = 128$
- Number of bit strings of length eight that start with bits 00: $2^6 = 64$
- Number of bit strings of length eight that start with a 1 bit and end with bits 00 : $2^5 = 32$

Hence, the number is $128 + 64 - 32 = 160$.



Basic Counting Principles: Division Rule

Division Rule: There are n/d ways to do a task if it can be done using a procedure that can be carried out in n ways, and for every way w , exactly d of the n ways correspond to way w .

- Restated in terms of sets: If the finite set A is the union of n pairwise disjoint subsets each with d elements, then $n = |A|/d$.
- In terms of functions: If f is a function from A to B , where both are finite sets, and for every value $y \in B$ there are exactly d values $x \in A$ such that $f(x) = y$, then $|B| = |A|/d$.

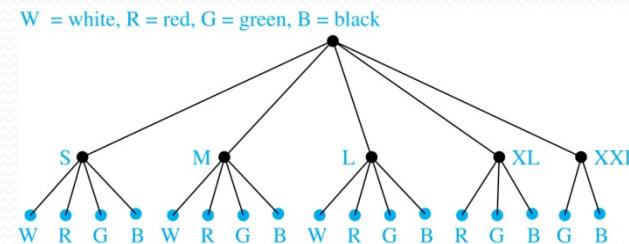
Example: How many ways are there to seat four people around a circular table, where two seatings are considered the same when each person has the same left and right neighbor?

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 seatings 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.

Tree Diagrams

- **Tree Diagrams:** We can solve many counting problems through the use of *tree diagrams*, where a branch represents a possible choice and the leaves represent possible outcomes.
- **Example:** Suppose that “I Love Discrete Math” T-shirts come in five different sizes: S,M,L,XL, and XXL. Each size comes in four colors (white, red, green, and black), except XL, which comes only in red, green, and black, and XXL, which comes only in green and black. What is the minimum number of stores that the campus book store needs to stock to have one of each size and color available?
- **Solution:** Draw the tree diagram.



- The store must stock 17 T-shirts.

The Pigeonhole Principle

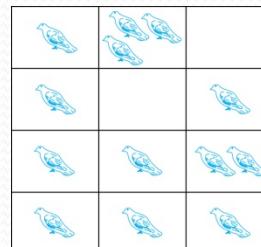
Section 6.2

Section Summary

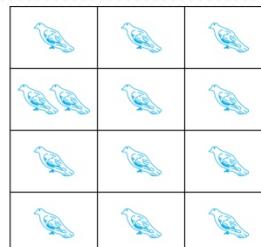
- The Pigeonhole Principle
- The Generalized Pigeonhole Principle

The Pigeonhole Principle

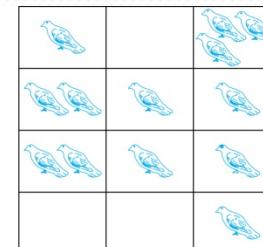
- If a flock of 20 pigeons roosts in a set of 19 pigeonholes, one of the pigeonholes must have more than 1 pigeon.



(a)



(b)



(c)

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 contradiction. 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

Corollary 1: A function f from a set with $k + 1$ elements to a set with k elements is not one-to-one.

Proof: Use the pigeonhole principle.

- Create a box for each element y in the codomain of f .
- Put in the box for y all of the elements x from the domain such that $f(x) = y$.
- Because there are $k + 1$ elements and only k boxes, at least one box has two or more elements.

Hence, f can't be one-to-one.



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.

Example (optional): Show that for every integer n there is a multiple of n that has only 0s and 1s in its decimal expansion.

Solution: Let n be a positive integer. Consider the $n + 1$ integers 1, 11, 111,, 11...1 (where the last has $n + 1$ 1s). There are n possible remainders when an integer is divided by n . By the pigeonhole principle, when each of the $n + 1$ integers is divided by n , at least two must have the same remainder. Subtract the smaller from the larger and the result is a multiple of n that has only 0s and 1s in its decimal expansion.

The Generalized Pigeonhole Principle

The Generalized Pigeonhole Principle: If N objects are placed into k boxes, then there is at least one box containing at least $[N/k]$ objects.

Here, $[N/k]$ is the smallest integer n for which $n \geq N/k$.

Proof: We use a proof by contraposition. Suppose that none of the boxes contains more than $[N/k] - 1$ objects. Then the total number of objects is at most

$$k \left(\left\lceil \frac{N}{k} \right\rceil - 1 \right) < k \left(\left(\frac{N}{k} + 1 \right) - 1 \right) = N,$$

where the inequality $\left[\frac{N}{k} \right] < \frac{N}{k} + 1$ has been used. This is a contradiction because there is a total of N objects. ◀

Example: Among 100 people there are at least $\lceil 100/12 \rceil = 9$ who were born in the same month. Note that $100/12 = 8.333$

The Generalized Pigeonhole Principle

Example: a) 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?

b) How many must be selected to guarantee that at least three hearts are selected?

Solution: a) 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 $N = 2 \cdot 4 + 1 = 9$.

b) A deck contains 13 hearts and 39 cards which are not hearts. So, if we select 41 cards, we may have 39 cards which are not hearts along with 2 hearts. However, when we select 42 cards, we must have at least three hearts. (Note that the generalized pigeonhole principle is not used here.)

Permutations and Combinations

Section 6.3

Section Summary

- Permutations
- Combinations
- Combinatorial Proofs

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* or *arrangement*.

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 .
- The number of r -permutations or arrangement 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, $P(3,2) = 6$.

A Formula for the 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.

- 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: 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 (*continued*)

Example: 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 (*continued*)

Example: 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

Definition: A 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*. (*We will see the notation again in the binomial theorem in Section 6.4.*)

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

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

This is a special case of a general result. →

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

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

Hence, $C(n, r) = C(n, n - r)$. ◀

This result can be proved without using algebraic manipulation. →

Combinatorial Proofs

- **Definition 1:** A *combinatorial proof* of an identity is a proof that uses one of the following methods.
 - A *double counting proof* uses counting arguments to prove that both sides of an identity count the same objects, but in different ways.
 - A *bijective proof* shows that there is a bijection between the sets of objects counted by the two sides of the identity.

Combinatorial Proofs

- Here are two combinatorial proofs that

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

when r and n are nonnegative integers with $r < n$:

- *Bijective Proof:* Suppose that S is a set with n elements. The function that maps a subset A of S to \bar{A} is a bijection between the subsets of S with r elements and the subsets with $n - r$ elements. Since there is a bijection between the two sets, they must have the same number of elements. ◀
- *Double Counting Proof:* By definition the number of subsets of S with r elements is $C(n, r)$. Each subset A of S can also be described by specifying which elements are not in A , i.e., those which are in \bar{A} . Since the complement of a subset of S with r elements has $n - r$ elements, there are also $C(n, n - r)$ subsets of S with r elements. ◀

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

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

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

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

Binomial Coefficients and Identities

Section 6.4

Section Summary

- The Binomial Theorem
- Pascal's Identity and Triangle
- Other Identities Involving Binomial Coefficients

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?
 - 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 one 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:

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

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 $n-j$ xs 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 $j = 13$.

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

A Useful Identity

Corollary 1: With $n \geq 0$, $\sum_{k=0}^n \binom{n}{k} = 2^n$.

Proof (using binomial theorem): With $x = 1$ and $y = 1$, from the binomial theorem we see that:

$$2^n = (1 + 1)^n = \sum_{k=0}^n \binom{n}{k} 1^k 1^{(n-k)} = \sum_{k=0}^n \binom{n}{k}.$$



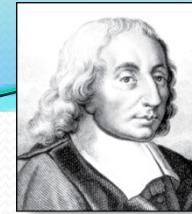
Proof (combinatorial): Consider the subsets of a set with n elements. There are $\binom{n}{0}$ subsets with zero elements, $\binom{n}{1}$ with one element, $\binom{n}{2}$ with two elements, ..., and $\binom{n}{n}$ with n elements. Therefore the total is

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

Since, we know that a set with n elements has 2^n subsets, we conclude:

$$\sum_{k=0}^n \binom{n}{k} = 2^n.$$





Pascal's Identity

Pascal's Identity: If n and k are integers with $n \geq k \geq 0$, then

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}.$$

Proof (combinatorial): Let T be a set where $|T| = n + 1$, $a \in T$, and $S = T - \{a\}$. There are $\binom{n+1}{k}$ subsets of T containing k elements. Each of these subsets either:

- contains a with $k - 1$ other elements, or
- contains k elements of S and not a .

There are

- $\binom{n}{k-1}$ subsets of k elements that contain a , since there are $\binom{n}{k-1}$ subsets of $k - 1$ elements of S ,
- $\binom{n}{k}$ subsets of k elements of T that do not contain a , because there are $\binom{n}{k}$ subsets of k elements of S .

Hence,

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}.$$



*See Exercise 19
for an algebraic
proof.*

Pascal's Triangle

The n th row in the triangle consists of the binomial coefficients $\binom{n}{k}$, $k = 0, 1, \dots, n$.

$\binom{0}{0}$		1
$\binom{1}{0} \binom{1}{1}$		1 1
$\binom{2}{0} \binom{2}{1} \binom{2}{2}$		1 2 1
$\binom{3}{0} \binom{3}{1} \binom{3}{2} \binom{3}{3}$	By Pascal's identity: $\binom{6}{4} + \binom{6}{5} = \binom{7}{5}$	1 3 3 1
$\binom{4}{0} \binom{4}{1} \binom{4}{2} \binom{4}{3} \binom{4}{4}$		1 4 6 4 1
$\binom{5}{0} \binom{5}{1} \binom{5}{2} \binom{5}{3} \binom{5}{4} \binom{5}{5}$		1 5 10 10 5 1
$\binom{6}{0} \binom{6}{1} \binom{6}{2} \binom{6}{3} \binom{6}{4} \binom{6}{5} \binom{6}{6}$		1 6 15 20 15 6 1
$\binom{7}{0} \binom{7}{1} \binom{7}{2} \binom{7}{3} \binom{7}{4} \binom{7}{5} \binom{7}{6} \binom{7}{7}$		1 7 21 35 35 21 7 1
$\binom{8}{0} \binom{8}{1} \binom{8}{2} \binom{8}{3} \binom{8}{4} \binom{8}{5} \binom{8}{6} \binom{8}{7} \binom{8}{8}$		1 8 28 56 70 56 28 8 1
...		...
(a)		(b)

By Pascal's identity, adding two adjacent binomial coefficients results in the binomial coefficient in the next row between these two coefficients.

Generalized Permutations and Combinations

Section 6.5

Section Summary

- Permutations with Repetition
- Combinations with Repetition
- Permutations with Indistinguishable Objects
- Distributing Objects into Boxes

Permutations with Repetition

Theorem 1: The number of r -permutations of a set of n objects with repetition allowed is $PR(n, r) = n^r$.

Proof: There are n ways to select an element of the set for each of the r positions in the r -permutation when repetition is allowed. Hence, by the product rule there are n^r r -permutations with repetition. ◀

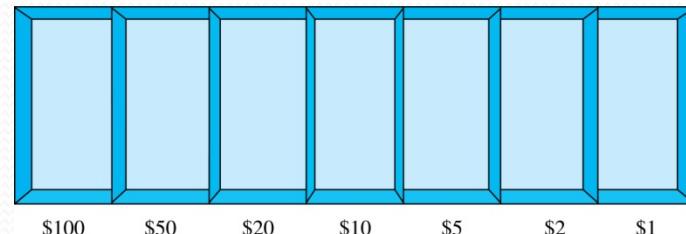
Example: How many strings of length r can be formed from the uppercase letters of the English alphabet?

Solution: The number of such strings is 26^r , which is the number of r -permutations of a set with 26 elements.

Combinations with Repetition

Example: How many ways are there to select five bills from a box containing at least five of each of the following denominations: \$1, \$2, \$5, \$10, \$20, \$50, and \$100? $CR(n,r) = CR(7,5)$

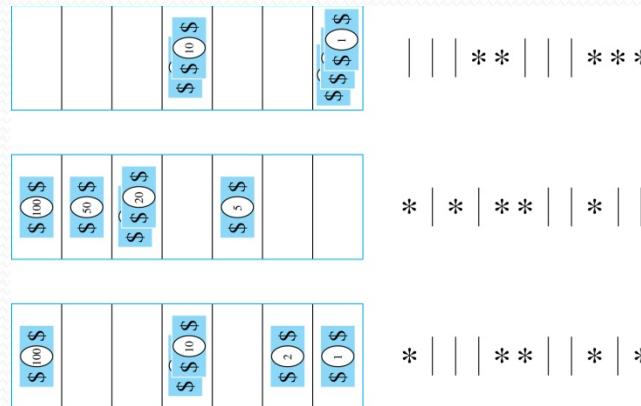
Solution: Place the selected bills in the appropriate position of a cash box illustrated below:



continued →

Combinations with Repetition

- Some possible ways of placing the five bills:



- The number of ways to select five bills of seven types corresponds to the number of ways to arrange six ($7 - 1$) bars and five stars in a row.
- This is exactly equivalent to specify the position (11) of the stars (5).
- Which is the number of unordered selections of 5 objects (stars) from a set of 11 (positions). Hence, there are

$$C(11, 5) = \frac{11!}{5!6!} = 462$$

ways to choose five bills among seven types of bills.

Combinations with Repetition

Theorem 2: The number of r -combinations from a set with n elements when repetition of elements is allowed is

$$CR(n, r) = C(n + r - 1, r) = C(n + r - 1, n - 1).$$

Proof: Each r -combination of a set with n elements with repetition allowed can be represented by a list of $n - 1$ bars and r stars. The bars mark the n cells containing a star for each time the i th element of the set occurs in the combination.

The number of such lists is $C(n + r - 1, r)$, because each list is a choice of the r positions to place the stars, from the total of $n + r - 1$ positions to place the stars (and the bars).

This is also equal to $C(n + r - 1, n - 1)$, which is the number of ways to place the $n - 1$ bars.



Combinations with Repetition

Example: How many solutions does the equation

$$x_1 + x_2 + x_3 = 11$$

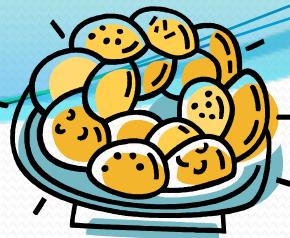
have, where x_1 , x_2 and x_3 are nonnegative integers?

Solution: Each solution corresponds to a way to select 11 items from a set with three types of element: x_1 elements of type one, x_2 of type two, and x_3 of type three.

By Theorem 2 it follows that there are $CR(3,11) =$

$$C(3 + 11 - 1, 11) = C(13, 11) = C(13, 2) = \frac{13 \cdot 12}{1 \cdot 2} = 78$$

solutions.



Combinations with Repetition

Example: Suppose that a cookie shop has four different kinds of cookies. How many different ways can six cookies be chosen?

Solution: The number of ways to choose six cookies is the number of 6-combinations of a set with four elements. By Theorem 2

$$C(9, 6) = C(9, 3) = \frac{9 \cdot 8 \cdot 7}{1 \cdot 2 \cdot 3} = 84$$

is the number of ways to choose six cookies from the four kinds.

Summarizing the Formulas for Counting Permutations and Combinations with and without Repetition

- Is the order important ?
 - Yes: permutations or arrangements
 - No: combinations
- Are repetitions allowed ?
 - Yes: with repetitions
 - No: without repetition

TABLE 1 Combinations and Permutations With and Without Repetition.

Type	Repetition Allowed?	Formula
r -permutations	No	$\frac{n!}{(n - r)!}$
r -combinations	No	$\frac{n!}{r! (n - r)!}$
r -permutations	Yes	n^r
r -combinations	Yes	$\frac{(n + r - 1)!}{r! (n - 1)!}$

Permutations with Indistinguishable Objects

Example: How many different strings can be made by reordering the letters of the word *SUCCESS*.

Solution: There are seven possible positions for the three Ss, two Cs, one U, and one E.

- The three Ss can be placed in $C(7,3)$ different ways, leaving four positions free.
- The two Cs can be placed in $C(4,2)$ different ways, leaving two positions free.
- The U can be placed in $C(2,1)$ different ways, leaving one position free.
- The E can be placed in $C(1,1)$ way.

By the product rule, the number of different strings is:

$$C(7,3)C(4,2)C(2,1)C(1,1) = \frac{7!}{3!4!} \cdot \frac{4!}{2!2!} \cdot \frac{2!}{1!1!} \cdot \frac{1!}{1!0!} = \frac{7!}{3!2!1!1!} = 420.$$

The reasoning can be generalized to the following theorem. →

Permutations with Indistinguishable Objects

Theorem 3: The number of different permutations of n objects, where there are n_1 indistinguishable objects of type 1, n_2 indistinguishable objects of type 2, ..., and n_k indistinguishable objects of type k , is:

$$\frac{n!}{n_1!n_2!\cdots n_k!} .$$

Proof: By the product rule the total number of permutations is:

$$C(n, n_1) C(n - n_1, n_2) \cdots C(n - n_1 - n_2 - \cdots - n_{k-1}, n_k) \text{ since:}$$

- The n_1 objects of type one can be placed in the n positions in $C(n, n_1)$ ways, leaving $n - n_1$ positions.
- Then the n_2 objects of type two can be placed in the $n - n_1$ positions in $C(n - n_1, n_2)$ ways, leaving $n - n_1 - n_2$ positions.
- Continue in this fashion, until n_k objects of type k are placed in $C(n - n_1 - n_2 - \cdots - n_{k-1}, n_k)$ ways.

The product can be manipulated into the desired result as follows:

$$\frac{n!}{n_1!(n-n_1)!} \frac{(n-n_1)!}{n_2!(n-n_1-n_2)!} \cdots \frac{(n-n_1-\cdots-n_{k-1})!}{n_k!0!} = \frac{n!}{n_1!n_2!\cdots n_k!} .$$



Distributing Objects into Boxes

- Many counting problems can be solved by counting the ways objects can be placed in boxes.
 - The objects may be either different from each other (*distinguishable*) or identical (*indistinguishable*).
 - The boxes may be labeled (*distinguishable*) or unlabeled (*indistinguishable*).

Distributing Objects into Boxes

- *Distinguishable objects and distinguishable boxes.*
 - There are $n!/(n_1!n_2!\cdots n_k!)$ ways to distribute n distinguishable objects into k distinguishable boxes.
 - (*See Exercises 47 and 48 for two different proofs.*)
 - Example: There are $52!/(5!5!5!5!32!)$ ways to distribute hands of 5 cards each to four players.
- *Indistinguishable objects and distinguishable boxes.*
 - There are $C(n + r - 1, n - 1)$ ways to place r indistinguishable objects into n distinguishable boxes.
 - Proof based on one-to-one correspondence between n -combinations from a set with k -elements when repetition is allowed and the ways to place n indistinguishable objects into k distinguishable boxes.
 - Example: There are $C(8 + 10 - 1, 10) = C(17, 10) = 19,448$ ways to place 10 indistinguishable objects into 8 distinguishable boxes.

Distributing Objects into Boxes

- *Distinguishable objects and indistinguishable boxes.*
 - Example: There are 14 ways to put four employees into three indistinguishable offices (see *Example 10*).
 - There is no simple closed formula for the number of ways to distribute n distinguishable objects into j indistinguishable boxes.
 - See the text for a formula involving *Stirling numbers of the second kind*.
- *Indistinguishable objects and indistinguishable boxes.*
 - Example: There are 9 ways to pack six copies of the same book into four identical boxes (see *Example 11*).
 - The number of ways of distributing n indistinguishable objects into k indistinguishable boxes equals $p_k(n)$, the number of ways to write n as the sum of at most k positive integers in increasing order.
 - No simple closed formula exists for this number.

Advanced Counting Techniques

Chapter 8

Chapter Summary

- Applications of Recurrence Relations
- Solving Linear Recurrence Relations
 - Homogeneous Recurrence Relations
 - Nonhomogeneous Recurrence Relations
- Generating Functions
- Inclusion-Exclusion
- Applications of Inclusion-Exclusion

Applications of Recurrence Relations

Section 8.1

Recurrence Relations

(recalling definitions from Chapter 2)

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.

Rabbits and the Fibonacci Numbers

Example: A young pair of rabbits (one of each gender) is placed on an island.

A pair of rabbits does not breed until they are 2 months old. After they are 2 months old, each pair of rabbits produces another pair each month.

Find a recurrence relation for the number of pairs of rabbits on the island after n months, assuming that rabbits never die.

This is the original problem considered by Leonardo Pisano (Fibonacci) in the thirteenth century.

Rabbits and the Fiobonacci Numbers (cont.)

Reproducing pairs (at least two months old)	Young pairs (less than two months old)	Month	Reproducing pairs	Young pairs	Total pairs
		1	0	1	1
		2	0	1	1
		3	1	1	2
		4	1	2	3
		5	2	3	5
		6	3	5	8
					

Modeling the Population Growth of Rabbits on an Island

Rabbits and the Fibonacci Numbers (cont.)

Solution: Let f_n be the number of pairs of rabbits after n months.

- There are $f_1 = 1$ pairs of rabbits on the island at the end of the first month.
- We also have $f_2 = 1$ because the pair does not breed during the first month.
- To find the number of pairs on the island after n months,
 - add the number on the island after the previous month, f_{n-1} ,
 - and the number of newborn pairs, which equals f_{n-2} , because each newborn pair comes from a pair at least two months old.

Consequently the sequence $\{f_n\}$ satisfies the recurrence relation $f_n = f_{n-1} + f_{n-2}$ for $n \geq 3$ with the initial conditions $f_1 = 1$ and $f_2 = 1$. The number of pairs of rabbits on the island after n months is given by the n th Fibonacci number.

Counting Bit Strings

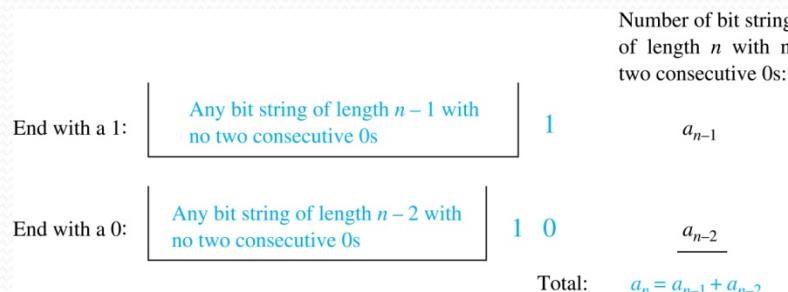
Example: Find a recurrence relation and give initial conditions for the number of bit strings of length n without two consecutive 0s. How many such bit strings are there of length five?

Solution: Let a_n denote the number of bit strings of length n without two consecutive 0s. To obtain a recurrence relation for $\{a_n\}$ note that the number of bit strings of length n that do not have two consecutive 0s is the number of bit strings ending with a 0 plus the number of such bit strings ending with a 1.

Now assume that $n \geq 3$.

- The bit strings of length n ending with 1 without two consecutive 0s are the bit strings of length $n - 1$ with no two consecutive 0s with a 1 added at the end. Hence, there are a_{n-1} such bit strings.
- The bit strings of length n ending with 0 without two consecutive 0s are the bit strings of length $n - 2$ with no two consecutive 0s with 10 at the end. Hence, there are a_{n-2} such bit strings.

We conclude that $a_n = a_{n-1} + a_{n-2}$ for $n \geq 3$.



Bit Strings (*continued*)

The initial conditions are:

- $a_1 = 2$, since both the bit strings 0 and 1 do not have consecutive 0s.
- $a_2 = 3$, since the bit strings 01, 10, and 11 do not have consecutive 0s, while 00 does.

To obtain a_5 , we use the recurrence relation three times to find that:

- $a_3 = a_2 + a_1 = 3 + 2 = 5$
- $a_4 = a_3 + a_2 = 5 + 3 = 8$
- $a_5 = a_4 + a_3 = 8 + 5 = 13$

Note that $\{a_n\}$ satisfies the same recurrence relation as the Fibonacci sequence. Since $a_1 = f_3$ and $a_2 = f_4$, we conclude that $a_n = f_{n+2}$.

Solving Linear Recurrence Relations

Section 8.2

Section Summary

- Linear Homogeneous Recurrence Relations
- Solving Linear Homogeneous Recurrence Relations with Constant Coefficients.
- Solving Linear Nonhomogeneous Recurrence Relations with Constant Coefficients.

Linear Homogeneous Recurrence Relations

Definition: A *linear homogeneous recurrence relation of degree k with constant coefficients* is a recurrence relation of the form $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$, where c_1, c_2, \dots, c_k are real numbers, and $c_k \neq 0$

- it is *linear* because the right-hand side is a sum of the previous terms of the sequence each multiplied by a function of n .
- it is *homogeneous* because no terms occur that are not multiples of the a_j s. Each coefficient is a constant.
- the *degree* is k because a_n is expressed in terms of the previous k terms of the sequence.

By induction, a sequence satisfying such a recurrence relation is uniquely determined by the recurrence relation and the k initial conditions $a_0 = C_1, a_1 = C_2, \dots, a_{k-1} = C_{k-1}$.

Linear Homogeneous Recurrence Relations

Assume $\{b_n\}$ and $\{d_n\}$ are solutions of the linear homogeneous recurrence relation of degree k with constant coefficients

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

Then, $\alpha_1 b_n + \alpha_2 d_n$ is also a solution of this recurrence relation.

Examples of Linear Homogeneous Recurrence Relations

- $P_n = (1.11)P_{n-1}$ linear homogeneous recurrence relation of degree one
- $f_n = f_{n-1} + f_{n-2}$ linear homogeneous recurrence relation of degree two
- $a_n = a_{n-1} + a_{n-2}^2$ not linear
- $H_n = 2H_{n-1} + 1$ not homogeneous
- $B_n = nB_{n-1}$ coefficients are not constants

Solving Linear Homogeneous Recurrence Relations

- The basic approach is to look for solutions of the form $a_n = r^n$, where r is a constant.
- Note that $a_n = r^n$ is a solution to the recurrence equation $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$ if and only if $r^n = c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k}$.
- Algebraic manipulation yields the *characteristic equation*:
$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_{k-1} r - c_k = 0$$
- The sequence $\{a_n\}$ with $a_n = r^n$ is a solution if and only if r is a solution to the characteristic equation.
- The solutions to the characteristic equation are called the *characteristic roots* of the recurrence relation. The roots are used to give an explicit formula for all the solutions of the recurrence relation.

Solving Linear Homogeneous Recurrence Relations of Degree Two

Theorem 1: Let c_1 and c_2 be real numbers. Suppose that $r^2 - c_1r - c_2 = 0$ has two distinct roots r_1 and r_2 . Then the sequence $\{a_n\}$ is a solution to the recurrence relation $a_n = c_1a_{n-1} + c_2a_{n-2}$ with initial conditions if and only if

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$$

for $n = 0, 1, 2, \dots$, where α_1 and α_2 are constants.

This direct solution can represent all the possible sequences verifying $a_n = c_1a_{n-1} + c_2a_{n-2}$.

Solving Linear Homogeneous Recurrence Relations of Degree Two

Proof: The roots $(r_1)^n$ and $(r_2)^n$ are solutions of the recurrence equation. Thus, $\alpha_1(r_1)^n + \alpha_2(r_2)^n$ with arbitrary α_1, α_2 is also a solution of the recurrence equation.

- Moreover, if $r_1 \neq r_2$, we can find uniquely α_1 and α_2 in order to verify the initial conditions (IC), $\alpha_1 + \alpha_2 = a_0$ and $\alpha_1 r_1 + \alpha_2 r_2 = a_1$. Thus, $\alpha_1(r_1)^n + \alpha_2(r_2)^n$ verifies the recurrence relation for $n = 0, 1$.
- Then, since, for a given set of IC, the sequence $\{a_n\}$ for $n = 2, \dots$ is determined uniquely by $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ and $\alpha_1(r_1)^n + \alpha_2(r_2)^n$ verifies exactly the recurrence equation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ for the same initial conditions, we must have

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$$

for $n = 0, 1, 2, \dots$, where α_1 and α_2 are set by the IC.

Using Theorem 1

Example: What is the solution to the recurrence relation

$$a_n = a_{n-1} + 2a_{n-2} \text{ with } a_0 = 2 \text{ and } a_1 = 7?$$

Solution: The characteristic equation is $r^2 - r - 2 = 0$.

Its roots are $r = 2$ and $r = -1$. Therefore, $\{a_n\}$ is a solution to the recurrence relation if and only if $a_n = \alpha_1 2^n + \alpha_2 (-1)^n$, for some constants α_1 and α_2 .

To find the constants α_1 and α_2 , note that

$$a_0 = 2 = \alpha_1 + \alpha_2 \text{ and } a_1 = 7 = \alpha_1 2 + \alpha_2 (-1).$$

Solving these equations, we find that $\alpha_1 = 3$ and $\alpha_2 = -1$.

Hence, the solution is the sequence $\{a_n\}$ with $a_n = 3 \cdot 2^n - (-1)^n$.

An Explicit Formula for the Fibonacci Numbers

We can use Theorem 1 to find an explicit formula for the Fibonacci numbers. The sequence of Fibonacci numbers satisfies the recurrence relation $f_n = f_{n-1} + f_{n-2}$ with the initial conditions: $f_0 = 0$ and $f_1 = 1$.

Solution: The roots of the characteristic equation $r^2 - r - 1 = 0$ are

$$r_1 = \frac{1+\sqrt{5}}{2}$$

$$r_2 = \frac{1-\sqrt{5}}{2}$$

Fibonacci Numbers (*continued*)

Therefore by Theorem 1

$$f_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2} \right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2} \right)^n$$

for some constants α_1 and α_2 .

Using the initial conditions $f_0 = 0$ and $f_1 = 1$, we have

$$f_0 = \alpha_1 + \alpha_2 = 0$$

$$f_1 = \alpha_1 \left(\frac{1+\sqrt{5}}{2} \right) + \alpha_2 \left(\frac{1-\sqrt{5}}{2} \right) = 1.$$

Solving, we obtain $\alpha_1 = \frac{1}{\sqrt{5}}$, $\alpha_2 = -\frac{1}{\sqrt{5}}$.

Hence,

$$f_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^n$$

The Solution when there is a Repeated Root

Theorem 2: Let c_1 and c_2 be real numbers with $c_2 \neq 0$. Suppose that $r^2 - c_1r - c_2 = 0$ has one repeated root r_0 . Then the sequence $\{a_n\}$ is a solution to the recurrence relation $a_n = c_1a_{n-1} + c_2a_{n-2}$ if and only if

$$a_n = \alpha r_0^n + \alpha_2 n r_0^n$$

for $n = 0, 1, 2, \dots$, where α_1 and α_2 are constants.

Using Theorem 2

Example: What is the solution to the recurrence relation
 $a_n = 6a_{n-1} - 9a_{n-2}$ with $a_0 = 1$ and $a_1 = 6$?

Solution: The characteristic equation is $r^2 - 6r + 9 = 0$.

The only root is $r = 3$. Therefore, $\{a_n\}$ is a solution to the recurrence relation if and only if
$$a_n = \alpha_1 3^n + \alpha_2 n(3)^n$$

where α_1 and α_2 are constants.

To find the constants α_1 and α_2 , note that

$$a_0 = 1 = \alpha_1 \quad \text{and} \quad a_1 = 6 = \alpha_1 \cdot 3 + \alpha_2 \cdot 3.$$

Solving, we find that $\alpha_1 = 1$ and $\alpha_2 = 1$.

Hence,

$$a_n = 3^n + n3^n.$$

Solving Linear Homogeneous Recurrence Relations of Arbitrary Degree

This theorem can be used to solve linear homogeneous recurrence relations with constant coefficients of any degree when the characteristic equation has distinct roots.

Theorem 3: Let c_1, c_2, \dots, c_k be real numbers. Suppose that the characteristic equation

$$r^k - c_1 r^{k-1} - \dots - c_k = 0$$

has k distinct roots r_1, r_2, \dots, r_k . Then a sequence $\{a_n\}$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

if and only if

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^m + \dots + \alpha_k r_k^n$$

for $n = 0, 1, 2, \dots$, where $\alpha_1, \alpha_2, \dots, \alpha_k$ are constants.

The General Case with Repeated Roots Allowed

Theorem 4: Let c_1, c_2, \dots, c_k be real numbers. Suppose that the characteristic equation

$$r^k - c_1 r^{k-1} - \dots - c_k = 0$$

has t distinct roots r_1, r_2, \dots, r_t with multiplicities m_1, m_2, \dots, m_t , respectively so that $m_i \geq 1$ for $i = 0, 1, 2, \dots, t$ and $m_1 + m_2 + \dots + m_t = k$. Then a sequence $\{a_n\}$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

if and only if

$$\begin{aligned} a_n &= (\alpha_{1,0} + \alpha_{1,1}n + \dots + \alpha_{1,m_1-1}n^{m_1-1})r_1^n \\ &\quad + (\alpha_{2,0} + \alpha_{2,1}n + \dots + \alpha_{2,m_2-1}n^{m_2-1})r_2^n \\ &\quad + \dots + (\alpha_{t,0} + \alpha_{t,1}n + \dots + \alpha_{t,m_t-1}n^{m_t-1})r_t^n \end{aligned}$$

for $n = 0, 1, 2, \dots$, where $\alpha_{i,j}$ are constants for $1 \leq i \leq t$ and $0 \leq j \leq m_{i-1}$.

Linear Nonhomogeneous Recurrence Relations with Constant Coefficients

Definition: A *linear nonhomogeneous recurrence relation with constant coefficients* is a recurrence relation of the form:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k} + F(n),$$

where c_1, c_2, \dots, c_k are real numbers, and $F(n)$ is a function not identically zero depending only on n : the source term

The recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k},$$

is called the associated homogeneous recurrence relation.

Linear Nonhomogeneous Recurrence Relations with Constant Coefficients (*cont.*)

The following are linear nonhomogeneous recurrence relations with constant coefficients:

$$a_n = a_{n-1} + 2^n,$$

$$a_n = a_{n-1} + a_{n-2} + n^2 + n + 1,$$

$$a_n = 3a_{n-1} + n3^n,$$

$$a_n = a_{n-1} + a_{n-2} + a_{n-3} + n!$$

where the following are the associated linear homogeneous recurrence relations, respectively:

$$a_n = a_{n-1},$$

$$a_n = a_{n-1} + a_{n-2},$$

$$a_n = 3a_{n-1},$$

$$a_n = a_{n-1} + a_{n-2} + a_{n-3}$$

Solving Linear Nonhomogeneous Recurrence Relations with Constant Coefficients

Theorem 5: If $\{a_n^{(p)}\}$ is a particular solution of the nonhomogeneous linear recurrence relation with constant coefficients

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + F(n),$$

then every solution is of the form $\{a_n^{(p)} + a_n^{(h)}\}$, where $\{a_n^{(h)}\}$ is the general solution of the associated homogeneous recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}.$$

Solving Linear Nonhomogeneous Recurrence Relations with Constant Coefficients

Proof: Let $\{a_n^{(p)}\}$ be a particular solution of $a_n^{(p)} = c_1 a_{n-1}^{(p)} + c_2 a_{n-2}^{(p)} + \cdots + c_k a_{n-k}^{(p)} + F(n)$ with the corresponding IC

- Let $\{b_n\}$ be any other solution of the form $b_n = c_1 b_{n-1} + c_2 b_{n-2} + \cdots + c_k b_{n-k} + F(n)$
- We thus have that $(b_n - a_n^{(p)}) = c_1 (b_{n-1} - a_{n-1}^{(p)}) + c_2 (b_{n-2} - a_{n-2}^{(p)}) + \cdots + c_k (b_{n-k} - a_{n-k}^{(p)})$ which is linear homogeneous with general solution $\{a_n^{(h)}\}$
- Therefore $(b_n - a_n^{(p)}) = a_n^{(h)}$ and $\{b_n\}$ must be of the form $b_n = a_n^{(p)} + a_n^{(h)}$, where $\{a_n^{(h)}\}$ is a solution of the associated homogeneous recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}.$$

Solving Linear Nonhomogeneous Recurrence Relations with Constant Coefficients (*continued*)

Example: Find all solutions of the recurrence relation $a_n = 3a_{n-1} + 2n$.

What is the solution with $a_1 = 3$?

Solution: The associated linear homogeneous equation is $a_n = 3a_{n-1}$.

Its solutions are $a_n^{(h)} = \alpha 3^n$, where α is a constant.

Because $F(n) = 2n$ is a polynomial in n of degree one, to find a particular solution we might try a linear function in n , say $p_n = cn + d$, where c and d are constants. Suppose that $p_n = cn + d$ is such a solution.

Then $a_n = 3a_{n-1} + 2n$ becomes $cn + d = 3(c(n-1) + d) + 2n$.

Simplifying yields $(2 + 2c)n + (2d - 3c) = 0$. It follows that $cn + d$ is a solution if and only if $2 + 2c = 0$ and $2d - 3c = 0$. Therefore, $cn + d$ is a solution if and only if $c = -1$ and $d = -3/2$. Consequently, $a_n^{(p)} = -n - 3/2$ is a particular solution.

By Theorem 5, all solutions are of the form $a_n = a_n^{(p)} + a_n^{(h)} = -n - 3/2 + \alpha 3^n$, where α is a constant.

To find the solution with $a_1 = 3$, let $n = 1$ in the above formula for the general solution.

Then $3 = -1 - 3/2 + 3\alpha$, and $\alpha = 11/6$. Hence, the solution is $a_n = -n - 3/2 + (11/6)3^n$.

Solving Recurrence Relations using substitution

Using the idea of substitution helps in solving recurrence relations. The following two examples illustrate this.

Example

Solve the recurrence relation $a_n^2 - 2a_{n-1}^2 = 1$ for $n \geq 1$ where $a_0 = 1$.

This is a nonlinear equation. This can be converted to a linear equation by using the substitution $b_n = a_n^2$. Now the equation becomes $b_n - 2b_{n-1} = 1$ with $b_0 = a_0^2 = 1^2 = 1$.

Solving $b_n - 2b_{n-1} = 1$ with $b_0 = 1$ the homogeneous solution is $a2^n$ and the particular solution is c . So $b_n = a2^n + c$ where a and c are constants. Using the fact that $b_0 = 1$ and $b_1 = 3$ we get $a + c = 1$ and $2a + c = 3$, which gives $a = 2$ and $b = -1$. Hence the solution is $b_n = 2 \cdot 2^n - 1 = 2^{n+1} - 1$.

Substituting back $a_n^2 = 2^{n+1} - 1$

$$a_n = \sqrt{2^{n+1} - 1}.$$

Even though $a_n = \pm\sqrt{b_n}$, since $a_0 = 1$, a_n cannot be $-\sqrt{b_n}$.

The state-space approach

The state-space approach

- We have the recurrence equation:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + F(n)$$

- Let us put it in vector/matrix form:

$$\mathbf{a}_n = \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_{n-k+1} \end{bmatrix} \quad \mathbf{a}_{n-1} = \begin{bmatrix} a_{n-1} \\ a_{n-2} \\ \vdots \\ a_{n-k} \end{bmatrix} \quad \mathbf{f}_n = \begin{bmatrix} F(n) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

The state-space approach

- We also define the matrix of coefficients:

$$\mathbf{C} = \begin{bmatrix} c_1 & c_2 & c_3 & \cdots & c_{k-1} & c_k \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

- We thus obtain a first-order equation:

$$\mathbf{a}_n = \mathbf{C}\mathbf{a}_{n-1} + \mathbf{f}_n$$

The state-space approach

- with the initial condition at $n = 0$:

\mathbf{a}_0 is given

- Let us try to solve this vector equation

$$\begin{cases} \mathbf{a}_1 = \mathbf{C}\mathbf{a}_0 + \mathbf{f}_1 \\ \mathbf{a}_2 = \mathbf{C}\mathbf{a}_1 + \mathbf{f}_2 = \mathbf{C}^2\mathbf{a}_0 + \mathbf{C}\mathbf{f}_1 + \mathbf{f}_2 \\ \mathbf{a}_3 = \mathbf{C}\mathbf{a}_2 + \mathbf{f}_3 = \mathbf{C}^3\mathbf{a}_0 + \mathbf{C}^2\mathbf{f}_1 + \mathbf{C}\mathbf{f}_2 + \mathbf{f}_3 \\ \vdots \end{cases}$$

The state-space approach

- From which we infer that

$$\mathbf{a}_n = \mathbf{C}^n \mathbf{a}_0 + \sum_{k=1}^n \mathbf{C}^{k-1} \mathbf{f}_{n-k+1}$$

- Which is the solution

Generating Functions

Section 8.4

Generating Functions

Definition: The *generating function for the sequence* $a_0, a_1, \dots, a_k, \dots$ of real numbers is the infinite series

$$G(x) = a_0 + a_1x + \cdots + a_kx^k + \cdots = \sum_{k=0}^{\infty} a_kx^k.$$

Examples:

- The sequence $\{a_k\}$ with $a_k = 3$ has the generating function $\sum_{k=0}^{\infty} 3x^k$.
- The sequence $\{a_k\}$ with $a_k = k + 1$ has the generating function $\sum_{k=0}^{\infty} (k + 1)x^k$.
- The sequence $\{a_k\}$ with $a_k = 2^k$ has the generating function $\sum_{k=0}^{\infty} 2^k x^k$.

Generating Functions for Finite Sequences

- Generating functions for finite sequences of real numbers can be defined by extending a finite sequence a_0, a_1, \dots, a_n into an infinite sequence by setting $a_{n+1} = 0, a_{n+2} = 0, \dots$, and so on.
- The generating function $G(x)$ of this infinite sequence $\{a_n\}$ is a polynomial of degree n because no terms of the form a_jx^j with $j > n$ occur, that is,

$$G(x) = a_0 + a_1x + \cdots + a_n x^n.$$

Generating Functions for Finite Sequences (continued)

Example: What is the generating function for the sequence 1,1,1,1,1,1?

Solution: The generating function of 1,1,1,1,1,1 is $1 + x + x^2 + x^3 + x^4 + x^5$.

By Theorem 1 of Section 2.4, we have

$$(x^6 - 1)/(x - 1) = 1 + x + x^2 + x^3 + x^4 + x^5$$

when $x \neq 1$.

Consequently $G(x) = (x^6 - 1)/(x - 1)$ is the generating function of the sequence.

Useful Generating Functions

TABLE 1 Useful Generating Functions.

$G(x)$	a_k
$(1+x)^n = \sum_{k=0}^n C(n, k)x^k$ $= 1 + C(n, 1)x + C(n, 2)x^2 + \cdots + x^n$	$C(n, k)$
$(1+ax)^n = \sum_{k=0}^n C(n, k)a^k x^k$ $= 1 + C(n, 1)ax + C(n, 2)a^2x^2 + \cdots + a^n x^n$	$C(n, k)a^k$
$(1+x^r)^n = \sum_{k=0}^n C(n, k)x^{rk}$ $= 1 + C(n, 1)x^r + C(n, 2)x^{2r} + \cdots + x^{rn}$	$C(n, k/r)$ if $r \mid k$; 0 otherwise
$\frac{1-x^{n+1}}{1-x} = \sum_{k=0}^n x^k = 1 + x + x^2 + \cdots + x^n$	1 if $k \leq n$; 0 otherwise
$\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k = 1 + x + x^2 + \cdots$	1
$\frac{1}{1-ax} = \sum_{k=0}^{\infty} a^k x^k = 1 + ax + a^2x^2 + \cdots$	a^k
$\frac{1}{1-x^r} = \sum_{k=0}^{\infty} x^{rk} = 1 + x^r + x^{2r} + \cdots$	1 if $r \mid k$; 0 otherwise
$\frac{1}{(1-x)^2} = \sum_{k=0}^{\infty} (k+1)x^k = 1 + 2x + 3x^2 + \cdots$	$k+1$
$\frac{1}{(1-x)^n} = \sum_{k=0}^{\infty} C(n+k-1, k)x^k$ $= 1 + C(n, 1)x + C(n+1, 2)x^2 + \cdots$	$C(n+k-1, k) = C(n+k-1, n-1)$
$\frac{1}{(1+x)^n} = \sum_{k=0}^{\infty} C(n+k-1, k)(-1)^k x^k$ $= 1 - C(n, 1)x + C(n+1, 2)x^2 - \cdots$	$(-1)^k C(n+k-1, k) = (-1)^k C(n+k-1, n-1)$
$\frac{1}{(1-ax)^n} = \sum_{k=0}^{\infty} C(n+k-1, k)a^k x^k$ $= 1 + C(n, 1)ax + C(n+1, 2)a^2x^2 + \cdots$	$C(n+k-1, k)a^k = C(n+k-1, n-1)a^k$
$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$	$1/k!$
$\ln(1+x) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} x^k = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots$	$(-1)^{k+1}/k$

Note: The series for the last two generating functions can be found in most calculus books when power series are discussed.

Counting Problems and Generating Functions

Example: Find the number of solutions of

$$e_1 + e_2 + e_3 = 17,$$

where e_1 , e_2 , and e_3 are nonnegative integers with
 $2 \leq e_1 \leq 5$, $3 \leq e_2 \leq 6$, and $4 \leq e_3 \leq 7$.

Solution: The number of solutions is the coefficient of x^{17} in the expansion of

$$(x^2 + x^3 + x^4 + x^5)(x^3 + x^4 + x^5 + x^6)(x^4 + x^5 + x^6 + x^7).$$

This follows because a term equal to x^{17} is obtained in the product by picking a term in the first sum x^{e_1} , a term in the second sum x^{e_2} , and a term in the third sum x^{e_3} , where $e_1 + e_2 + e_3 = 17$.

There are three solutions since the coefficient of x^{17} in the product is 3.

Inclusion-Exclusion

Section 8.5

Principle of Inclusion-Exclusion

- In Section 2.2, we developed the following formula for the number of elements in the union of two finite sets:

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

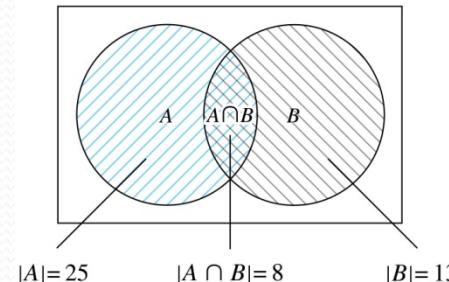
- We will generalize this formula to finite sets of any size.

Two Finite Sets

Example: In a discrete mathematics class every student is a major in computer science or mathematics or both. The number of students having computer science as a major (possibly along with mathematics) is 25; the number of students having mathematics as a major (possibly along with computer science) is 13; and the number of students majoring in both computer science and mathematics is 8. How many students are in the class?

Solution: $|A \cup B| = |A| + |B| - |A \cap B|$
 $= 25 + 13 - 8 = 30$

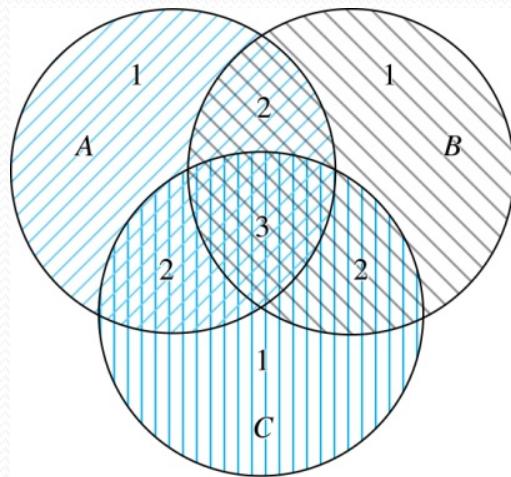
$$|A \cup B| = |A| + |B| - |A \cap B| = 25 + 13 - 8 = 30$$



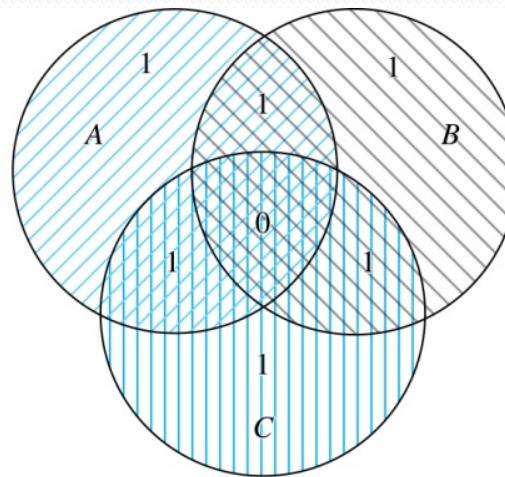
Three Finite Sets

$$|A \cup B \cup C| =$$

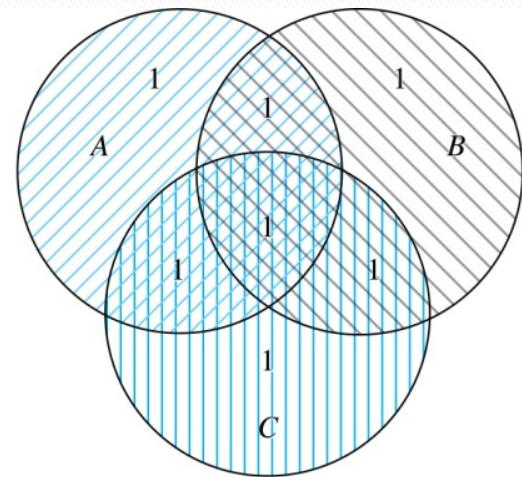
$$|A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$



(a) Count of elements by
 $|A|+|B|+|C|$



(b) Count of elements by
 $|A|+|B|+|C|-|A \cap B|-|A \cap C|-|B \cap C|$



(c) Count of elements by
 $|A|+|B|+|C|-|A \cap B|-|A \cap C|-|B \cap C|+|A \cap B \cap C|$

Three Finite Sets Continued

Example: A total of 1232 students have taken a course in Spanish, 879 have taken a course in French, and 114 have taken a course in Russian. Further, 103 have taken courses in both Spanish and French, 23 have taken courses in both Spanish and Russian, and 14 have taken courses in both French and Russian. If 2092 students have taken a course in at least one of Spanish French and Russian, how many students have taken a course in all 3 languages.

Solution: Let S be the set of students who have taken a course in Spanish, F the set of students who have taken a course in French, and R the set of students who have taken a course in Russian. Then, we have $|S| = 1232$, $|F| = 879$, $|R| = 114$, $|S \cap F| = 103$, $|S \cap R| = 23$, $|F \cap R| = 14$, and $|S \cup F \cup R| = 2092$.

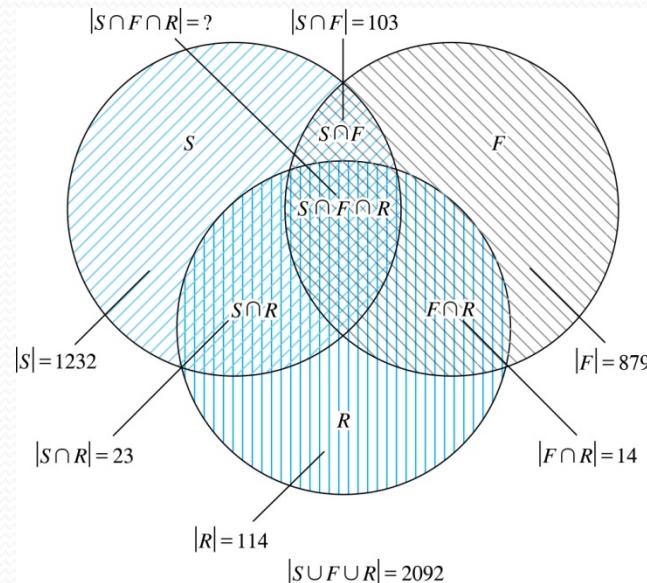
Using the equation

$$|S \cup F \cup R| = |S| + |F| + |R| - |S \cap F| - |S \cap R| - |F \cap R| + |S \cap F \cap R|,$$

we obtain $2092 = 1232 + 879 + 114 - 103 - 23 - 14 + |S \cap F \cap R|$.

Solving for $|S \cap F \cap R|$ yields 7.

Illustration of Three Finite Set Example



The Principle of Inclusion-Exclusion

Theorem 1. The Principle of Inclusion-Exclusion:

Let A_1, A_2, \dots, A_n be finite sets. Then:

$$|A_1 \cup A_2 \cup \dots \cup A_n| =$$

$$\sum_{1 \leq i \leq n} |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| +$$

$$\sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| - \dots + (-1)^{n+1} |A_1 \cap A_2 \cap \dots \cap A_n|$$

The Principle of Inclusion-Exclusion *(continued)*

Proof: An element in the union has to be counted exactly once in the right-hand side of the equation. Consider an element a that is a member of r of the sets A_1, \dots, A_n , where $1 \leq r \leq n$.

What is the number of combinations of these r sets?

- It is counted $C(r,1)$ times by $\sum |A_i|$
- It is counted $C(r,2)$ times by $\sum |A_i \cap A_j|$
- In general, it is counted $C(r,m)$ times by the summation of m of the sets A_i .

The Principle of Inclusion-Exclusion (continued)

- Thus the element is counted exactly

$$C(r,1) - C(r,2) + C(r,3) - \cdots + (-1)^{r+1} C(r,r)$$

times by the right hand side of the equation.

- By Corollary 2 of Section 6.4, we have

$$C(r,0) - C(r,1) + C(r,2) - \cdots + (-1)^r C(r,r) = 0.$$

- Hence,

$$1 = C(r,0) = C(r,1) - C(r,2) + \cdots + (-1)^{r+1} C(r,r).$$

Relations

Chapter 9

Chapter Summary

- Relations and Their Properties
- n -ary Relations and Their Applications (*not currently included in overheads*)
- Representing Relations
- Closures of Relations (*not currently included in overheads*)
- Equivalence Relations
- Partial Orderings

Relations and Their Properties

Section 9.1

Section Summary

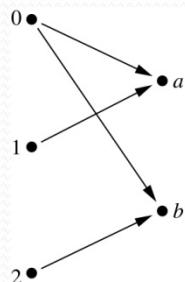
- Relations and Functions
- Properties of Relations
 - Reflexive Relations
 - Symmetric and Antisymmetric Relations
 - Transitive Relations
- Combining Relations

Binary Relations

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

Example:

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



R	a	b
0	×	×
1	×	
2		×

Relations are more general than functions. A function is a relation where exactly one element of B is related to each element of A .

Binary Relation on a Set

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

Example:

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

Binary Relation on a Set (*cont.*)

Question: How many relations are there on a set A ?

Solution: Because a relation on A is the same thing as a subset of $A \times A$, we count the subsets of $A \times A$. Since $A \times A$ has n^2 elements when A has n elements, and a set with m elements has 2^m subsets, there are $2^{|A|^2}$ subsets of $A \times A$. Therefore, there are $2^{|A|^2}$ relations on a set A .

Binary Relations on a Set (cont.)

Example: Consider these relations on the set of integers:

$$R_1 = \{(a,b) \mid a \leq b\},$$

$$R_4 = \{(a,b) \mid a = b\},$$

$$R_2 = \{(a,b) \mid a > b\},$$

$$R_5 = \{(a,b) \mid a = b + 1\},$$

$$R_3 = \{(a,b) \mid a = b \text{ or } a = -b\},$$

$$R_6 = \{(a,b) \mid a + b \leq 3\}.$$

Note that these relations are on an infinite set and each of these relations is an infinite set.

Which of these relations contain each of the pairs

(1,1), (1, 2), (2, 1), (1, -1), and (2, 2)?

Solution: Checking the conditions that define each relation, we see that the pair (1,1) is in R_1 , R_3 , R_4 , and R_6 : (1,2) is in R_1 and R_6 : (2,1) is in R_2 , R_5 , and R_6 : (1, -1) is in R_2 , R_3 , and R_6 : (2,2) is in R_1 , R_3 , and R_4 .

Reflexive Relations

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

$$\forall x[x \in U \rightarrow (x,x) \in R]$$

Example: The following relations on the integers are reflexive:

$$R_1 = \{(a,b) \mid a \leq b\},$$

$$R_3 = \{(a,b) \mid a = b \text{ or } a = -b\},$$

$$R_4 = \{(a,b) \mid a = b\}.$$

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

The following relations are not reflexive:

$$R_2 = \{(a,b) \mid a > b\} \text{ (note that } 3 \not> 3\text{),}$$

$$R_5 = \{(a,b) \mid a = b + 1\} \text{ (note that } 3 \neq 3 + 1\text{),}$$

$$R_6 = \{(a,b) \mid a + b \leq 3\} \text{ (note that } 4 + 4 \not\leq 3\text{).}$$

Symmetric Relations

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

$$\forall x \forall y [(x,y) \in R \rightarrow (y,x) \in R]$$

Example: The following relations on the integers are symmetric:

$$R_3 = \{(a,b) \mid a = b \text{ or } a = -b\},$$

$$R_4 = \{(a,b) \mid a = b\},$$

$$R_6 = \{(a,b) \mid a + b \leq 3\}.$$

The following are not symmetric:

$$R_1 = \{(a,b) \mid a \leq b\} \text{ (note that } 3 \leq 4, \text{ but } 4 \not\leq 3\text{)},$$

$$R_2 = \{(a,b) \mid a > b\} \text{ (note that } 4 > 3, \text{ but } 3 \not> 4\text{)},$$

$$R_5 = \{(a,b) \mid a = b + 1\} \text{ (note that } 4 = 3 + 1, \text{ but } 3 \neq 4 + 1\text{)}.$$

Antisymmetric Relations

Definition: A relation R on a set A such that for all $a, b \in A$ if $(a, b) \in R$ and $(b, a) \in R$, then $a = b$ is called *antisymmetric*.

Written symbolically, R is antisymmetric if and only if

$$\forall x \forall y [(x, y) \in R \wedge (y, x) \in R \rightarrow x = y]$$

- **Example:** The following relations on the integers are antisymmetric:

$$R_1 = \{(a, b) \mid a \leq b\},$$

For any integer, if $a \leq b$ and $a \leq b$, then $a = b$.

$$R_2 = \{(a, b) \mid a > b\},$$

$$R_4 = \{(a, b) \mid a = b\},$$

$$R_5 = \{(a, b) \mid a = b + 1\}.$$

The following relations are not antisymmetric:

$$R_3 = \{(a, b) \mid a = b \text{ or } a = -b\}$$

(note that both $(1, -1)$ and $(-1, 1)$ belong to R_3),

$$R_6 = \{(a, b) \mid a + b \leq 3\}$$
 (note that both $(1, 2)$ and $(2, 1)$ belong to R_6).

Transitive Relations

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

$$\forall x \forall y \forall z [(x,y) \in R \wedge (y,z) \in R \rightarrow (x,z) \in R]$$

- **Example:** The following relations on the integers are transitive:

$$R_1 = \{(a,b) \mid a \leq b\},$$

$$R_2 = \{(a,b) \mid a > b\},$$

$$R_3 = \{(a,b) \mid a = b \text{ or } a = -b\},$$

$$R_4 = \{(a,b) \mid a = b\}.$$

For every integer, $a \leq b$ and $b \leq c$, then $b \leq c$.

The following are not transitive:

$R_5 = \{(a,b) \mid a = b + 1\}$ (note that both $(3,2)$ and $(4,3)$ belong to R_5 , but not $(3,3)$),

$R_6 = \{(a,b) \mid a + b \leq 3\}$ (note that both $(2,1)$ and $(1,2)$ belong to R_6 , but not $(2,2)$).

Combining Relations

- Given two relations R_1 and R_2 , we can combine them using basic set operations to form new relations such as $R_1 \cup R_2$, $R_1 \cap R_2$, $R_1 - R_2$, and $R_2 - R_1$.
- Example:** Let $A = \{1,2,3\}$ and $B = \{1,2,3,4\}$. The relations $R_1 = \{(1,1),(2,2),(3,3)\}$ and $R_2 = \{(1,1),(1,2),(1,3),(1,4)\}$ can be combined using basic set operations to form new relations:

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

$$R_1 \cap R_2 = \{(1,1)\} \quad R_1 - R_2 = \{(2,2), (3,3)\}$$

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

Composition

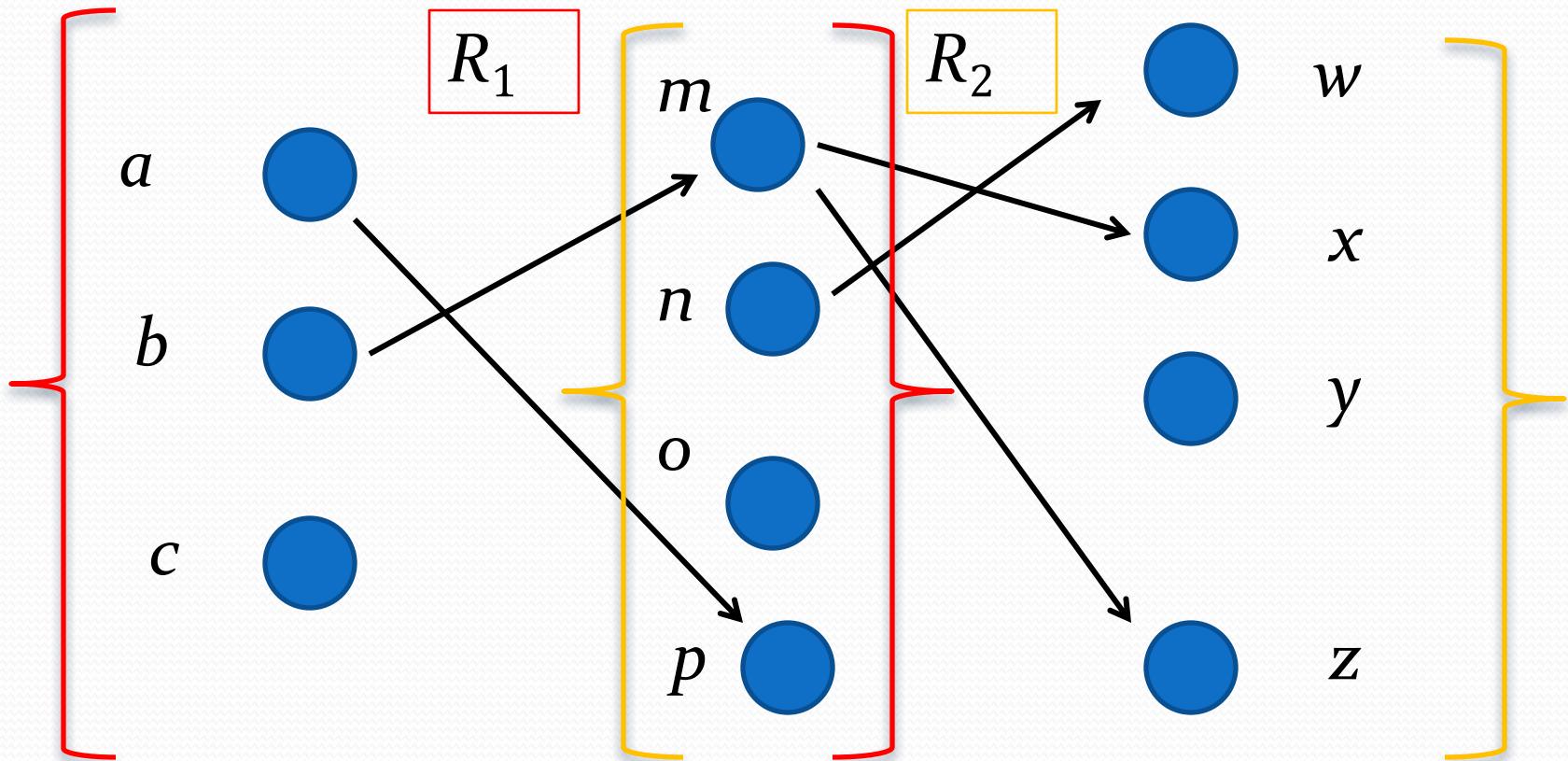
Definition: Suppose

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

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

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

Representing the Composition of a Relation



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

Powers of a Relation

Definition: Let R be a binary relation on A . Then the powers R^n of the relation R can be defined inductively by:

- Basis Step: $R^1 = R$
- Inductive Step: $R^{n+1} = R^n \circ R$

(see the slides for Section 9.3 for further insights)

The powers of a transitive relation are subsets of the relation. This is established by the following theorem:

Theorem 1: The relation R on a set A is transitive iff $R^n \subseteq R$ for $n = 1, 2, 3, \dots$

(the proof is not required)

Representing Relations

Section 9.3

Section Summary

- Representing Relations using Matrices
- Representing Relations using Digraphs

Representing Relations Using Matrices

- A relation between finite sets can be represented using a zero-one matrix.
- Suppose R is a relation from $A = \{a_1, a_2, \dots, a_m\}$ to $B = \{b_1, b_2, \dots, b_n\}$.
 - The elements of the two sets can be listed in any particular arbitrary order. When $A = B$, we use the same ordering.
- The relation R is represented by the matrix $M_R = [m_{ij}]$, where
$$m_{ij} = \begin{cases} 1 & \text{if } (a_i, b_j) \in R, \\ 0 & \text{if } (a_i, b_j) \notin R. \end{cases}$$
- The matrix representing R has a 1 as its (i,j) entry when a_i is related to b_j and a 0 if a_i is not related to b_j .

Examples of Representing Relations Using Matrices

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

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

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

Examples of Representing Relations Using Matrices (cont.)

Example 2: Let $A = \{a_1, a_2, a_3\}$ and $B = \{b_1, b_2, b_3, b_4, b_5\}$. Which ordered pairs are in the relation R represented by the matrix

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

Solution: Because R consists of those ordered pairs (a_i, b_j) with $m_{ij} = 1$, it follows that:

$$R = \{(a_1, b_2), (a_2, b_1), (a_2, b_3), (a_2, b_4), (a_3, b_1), (a_3, b_3), (a_3, b_5)\}.$$

Matrices of Relations on Sets

- If R is a reflexive relation, all the elements on the main diagonal of M_R are equal to 1.

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

- R is a symmetric relation, if and only if $m_{ij} = 1$ whenever $m_{ji} = 1$. R is an antisymmetric relation, if and only if $m_{ij} = 0$ or $m_{ji} = 0$ when $i \neq j$.

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

(a) Symmetric

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

(b) Antisymmetric

Example of a Relation on a Set

Example 3: Suppose that the relation R on a set is represented by the matrix

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

Is R reflexive, symmetric, and/or antisymmetric?

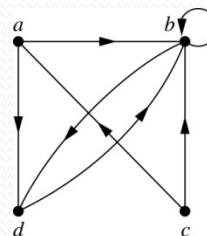
Solution: Because all the diagonal elements are equal to 1, R is reflexive. Because M_R is symmetric, R is symmetric and not antisymmetric because both $m_{1,2}$ and $m_{2,1}$ are 1.

Representing Relations Using Digraphs

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

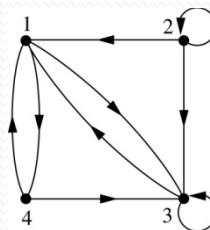
- An edge of the form (a,a) is called a *loop*.

Example 7: A drawing of the directed graph with vertices a , b , c , and d , and edges (a, b) , (a, d) , (b, b) , (b, d) , (c, a) , (c, b) , and (d, b) is shown here.



Examples of Digraphs Representing Relations

Example 8: What are the ordered pairs in the relation represented by this directed graph?

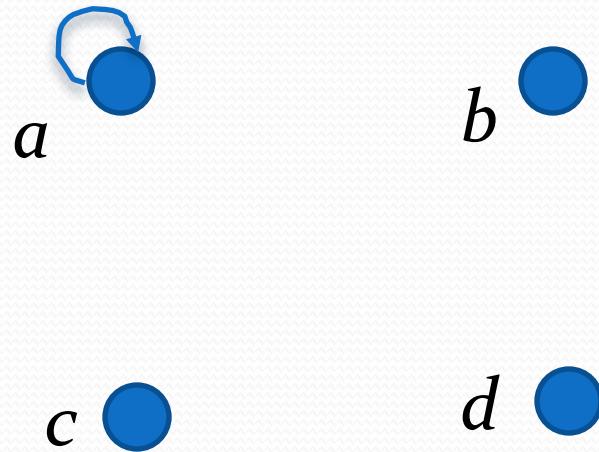


Solution: The ordered pairs in the relation are
 $(1, 3)$, $(1, 4)$, $(2, 1)$, $(2, 2)$, $(2, 3)$, $(3, 1)$, $(3, 3)$,
 $(4, 1)$, and $(4, 3)$

Determining which Properties a Relation has from its Digraph

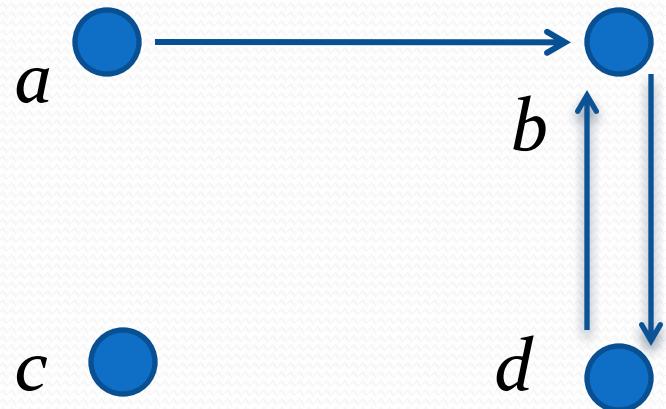
- *Reflexivity*: A loop must be present at all vertices in the graph.
- *Symmetry*: If (x,y) is an edge, then so is (y,x) .
- *Antisymmetry*: If (x,y) with $x \neq y$ is an edge, then (y,x) is not an edge.
- *Transitivity*: If (x,y) and (y,z) are edges, then so is (x,z) .

Determining which Properties a Relation has from its Digraph – Example 1



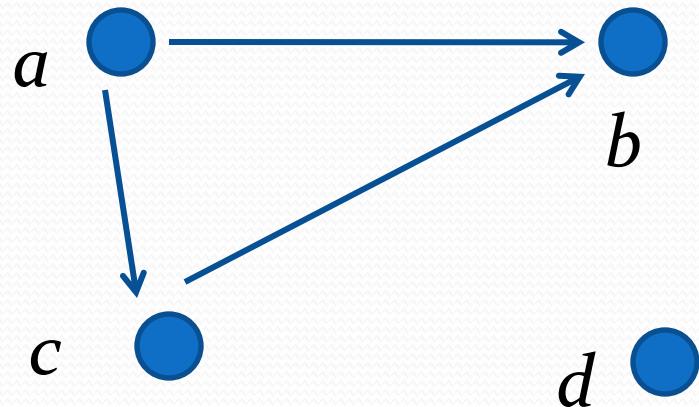
- *Reflexive?* No, not every vertex has a loop
- *Symmetric?* Yes (trivially), there is no edge from one vertex to another
- *Antisymmetric?* Yes (trivially), there is no edge from one vertex to another
- *Transitive?* Yes, (trivially) since there is no edge from one vertex to another

Determining which Properties a Relation has from its Digraph – Example 2



- *Reflexive?* No, there are no loops
- *Symmetric?* No, there is an edge from a to b , but not from b to a
- *Antisymmetric?* No, there is an edge from d to b and b to d
- *Transitive?* No, there are edges from a to c and from c to b ,
but there is no edge from a to d

Determining which Properties a Relation has from its Digraph – Example 3



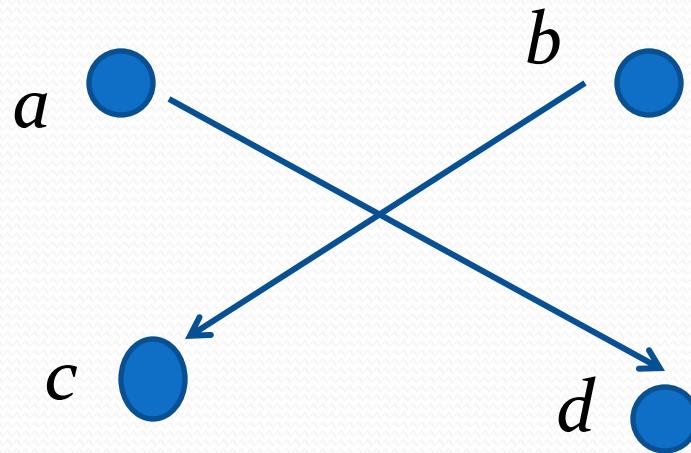
Reflexive? No, there are no loops

Symmetric? No, for example, there is no edge from c to a

Antisymmetric? Yes, whenever there is an edge from one vertex to another, there is not one going back

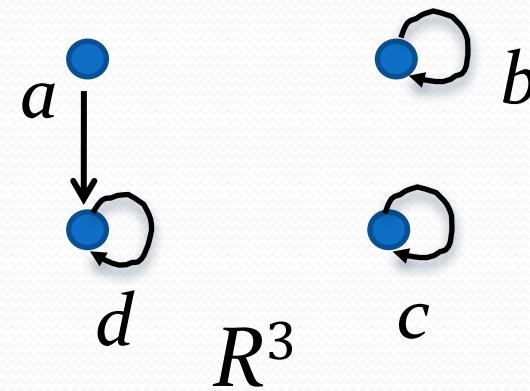
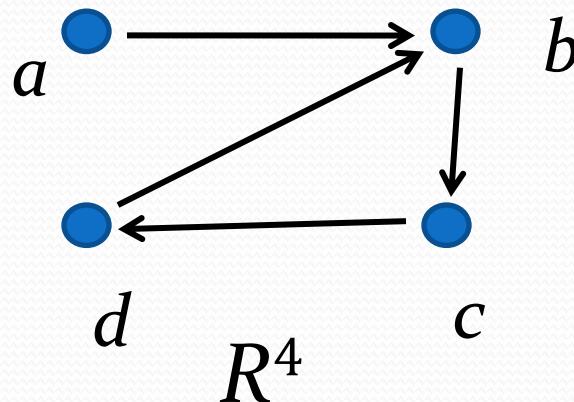
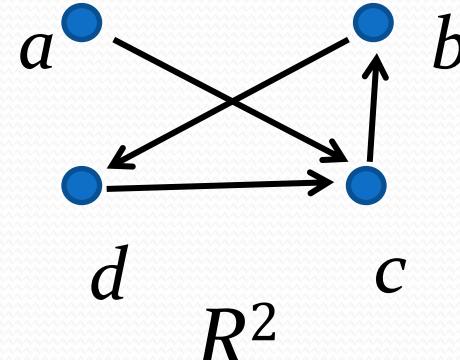
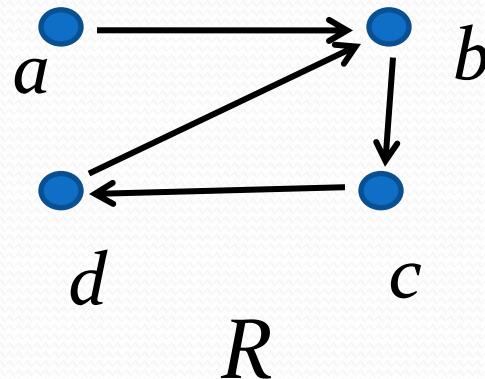
Transitive? No, there is no edge from a to b

Determining which Properties a Relation has from its Digraph – Example 4



- *Reflexive?* No, there are no loops
- *Symmetric?* No, for example, there is no edge from d to a
- *Antisymmetric?* Yes, whenever there is an edge from one vertex to another, there is not one going back
- *Transitive?* Yes (trivially), there are no two edges where the first edge ends at the vertex where the second edge begins

Example of the Powers of a Relation



The pair (x,y) is in R^n if there is a path of length n from x to y in R (following the direction of the arrows).

Equivalence Relations

Section 9.5

Section Summary

- Equivalence Relations
- Equivalence Classes
- Equivalence Classes and Partitions

Equivalence Relations

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

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

Strings

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

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

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

Congruence Modulo m

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

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

is an equivalence relation on the set of integers.

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

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

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

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

Divides

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

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

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

Equivalence Classes

Definition 3: Let R be an equivalence relation on a set A . The set of all elements that are related to an element a of A is called the *equivalence class* of a . The equivalence class of a with respect to R is denoted by $[a]_R$.

When only one relation is under consideration, we can write $[a]$, without the subscript R , for this equivalence class.

Note that $[a]_R = \{s | (a,s) \in R\}$.

- If $b \in [a]_R$, then b is called a representative of this equivalence class. Any element of a class can be used as a representative of the class.
- The equivalence classes of the relation congruence modulo m are called the *congruence classes modulo m* . The congruence class of an integer a modulo m is denoted by $[a]_m$, so $[a]_m = \{..., a-2m, a-m, a+2m, a+2m, ...\}$. For example,

$$[0]_4 = \{..., -8, -4, 0, 4, 8, ...\}$$

$$[1]_4 = \{..., -7, -3, 1, 5, 9, ...\}$$

$$[2]_4 = \{..., -6, -2, 2, 6, 10, ...\}$$

$$[3]_4 = \{..., -5, -1, 3, 7, 11, ...\}$$

Equivalence Classes and Partitions

Theorem 1: let R be an equivalence relation on a set A .
These statements for elements a and b of A are equivalent:

- (i) aRb
- (ii) $[a] = [b]$
- (iii) $[a] \cap [b] = \emptyset$

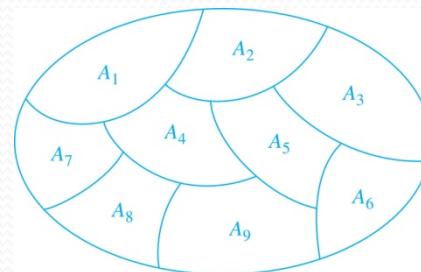
Proof: We show that (i) implies (ii). Assume that aRb . Now suppose that $c \in [a]$. Then aRc . Because aRb and R is symmetric, bRa . Because R is transitive and bRa and aRc , it follows that bRc . Hence, $c \in [b]$. Therefore, $[a] \subseteq [b]$. A similar argument (omitted here) shows that $[b] \subseteq [a]$. Since $[a] \subseteq [b]$ and $[b] \subseteq [a]$, we have shown that $[a] = [b]$.

(other proofs not required, but present in the book)

Partition of a Set

Definition: A *partition* of a set S is a collection of disjoint nonempty subsets of S that have S as their union. In other words, the collection of subsets A_i , where $i \in I$ (where I is an index set), forms a partition of S if and only if

- $A_i \neq \emptyset$ for $i \in I$,
- $A_i \cap A_j = \emptyset$ when $i \neq j$,
- and $\bigcup_{i \in I} A_i = S$.



A Partition of a Set

An Equivalence Relation Partitions a Set

- Let R be an equivalence relation on a set A . The union of all the equivalence classes of R is all of A , since an element a of A is in its own equivalence class $[a]_R$. In other words,

$$\bigcup_{a \in A} [a]_R = A.$$

- From Theorem 1, it follows that these equivalence classes are either equal or disjoint, so $[a]_R \cap [b]_R = \emptyset$ when $[a]_R \neq [b]_R$.
- Therefore, the equivalence classes form a partition of A , because they split A into disjoint subsets.

An Equivalence Relation Partitions a Set (*continued*)

Theorem 2: Let R be an equivalence relation on a set S . Then the equivalence classes of R form a partition of S . Conversely, given a partition $\{A_i \mid i \in I\}$ of the set S , there is an equivalence relation R that has the sets A_i , $i \in I$, as its equivalence classes.

Proof: We have already shown the first part of the theorem.

For the second part, assume that $\{A_i \mid i \in I\}$ is a partition of S . Let R be the relation on S consisting of the pairs (x, y) where x and y belong to the same subset A_i in the partition. We must show that R satisfies the properties of an equivalence relation.

- *Reflexivity:* For every $a \in S$, $(a, a) \in R$, because a is in the same subset as itself.
- *Symmetry:* If $(a, b) \in R$, then b and a are in the same subset of the partition, so $(b, a) \in R$.
- *Transitivity:* If $(a, b) \in R$ and $(b, c) \in R$, then a and b are in the same subset of the partition, as are b and c . Since the subsets are disjoint and b belongs to both, the two subsets of the partition must be identical. Therefore, $(a, c) \in R$ since a and c belong to the same subset of the partition.

Partial Orderings

Section 9.6

Section Summary

- Partial Orderings and Partially-ordered Sets
- Lexicographic Orderings
- Hasse Diagrams
- Lattices (*not currently in overheads*)
- Topological Sorting (*not currently in overheads*)

Partial Orderings

Definition 1: A relation R on a set S is called a *partial ordering*, or *partial order*, if it is reflexive, antisymmetric, and transitive. A set together with a partial ordering R is called a *partially ordered set*, or *poset*, and is denoted by (S, R) . Members of S are called *elements* of the poset.

Partial Orderings (*continued*)

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

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

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

Partial Orderings (*continued*)

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

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

Partial Orderings (*continued*)

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

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

The properties all follow from the definition of set inclusion.

Comparability

Definition 2: The elements a and b of a poset (S, \leq) are *comparable* if either $a \leq b$ or $b \leq a$. When a and b are elements of S so that neither $a \leq b$ nor $b \leq a$, then a and b are called *incomparable*.

The symbol \leq is used to denote the relation in any poset.

Definition 3: If (S, \leq) is a poset and every two elements of S are comparable, S is called a *totally ordered* or *linearly ordered set*, and \leq is called a *total order* or a *linear order*. A totally ordered set is also called a *chain*.

Definition 4: (S, \leq) is a well-ordered set if it is a poset such that \leq is a total ordering and every nonempty subset of S has a least element.

Lexicographic Order

Definition: Given two posets (A_1, \leqslant_1) and (A_2, \leqslant_2) , the *lexicographic ordering* on $A_1 \times A_2$ is defined by specifying that (a_1, a_2) is less than (b_1, b_2) , that is,

$$(a_1, a_2) < (b_1, b_2),$$

either if $a_1 <_1 b_1$ or if $a_1 = b_1$ and $a_2 <_2 b_2$.

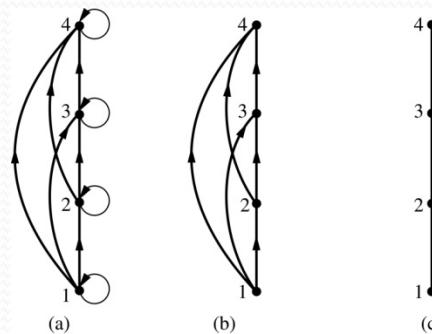
- This definition can be easily extended to a lexicographic ordering on strings (see text).

Example: Consider strings of lowercase English letters. A lexicographic ordering can be defined using the ordering of the letters in the alphabet. This is the same ordering as that used in dictionaries.

- *discreet* < *discrete*, because these strings differ in the seventh position and *e* < *t*.
- *discreet* < *discreteness*, because the first eight letters agree, but the second string is longer.

Hasse Diagrams

Definition: A *Hasse diagram* is a visual representation of a partial ordering that leaves out edges that must be present because of the reflexive and transitive properties.



A partial ordering is shown in (a) of the figure above. The loops due to the reflexive property are deleted in (b). The edges that must be present due to the transitive property are deleted in (c). The Hasse diagram for the partial ordering (a), is depicted in (c).

Procedure for Constructing a Hasse Diagram

- To represent a finite poset (S, \leq) using a Hasse diagram, start with the directed graph of the relation:
 - Remove the loops (a, a) present at every vertex due to the reflexive property.
 - Remove all edges (x, y) for which there is an element $z \in S$ such that $x < z$ and $z < y$. These are the edges that must be present due to the transitive property.
 - Arrange each edge so that its initial vertex is below the terminal vertex. Remove all the arrows, because all edges point upwards toward their terminal vertex.

Graphs

Chapter 10

Chapter Summary

- Graphs and Graph Models
- Graph Terminology and Special Types of Graphs
- Representing Graphs and Graph Isomorphism
- Connectivity
- Euler and Hamiltonian Graphs
- Shortest-Path Problems (*not currently included in overheads*)
- Planar Graphs (*not currently included in overheads*)
- Graph Coloring (*not currently included in overheads*)

Graphs and Graph Models

Section 10.1

Section Summary

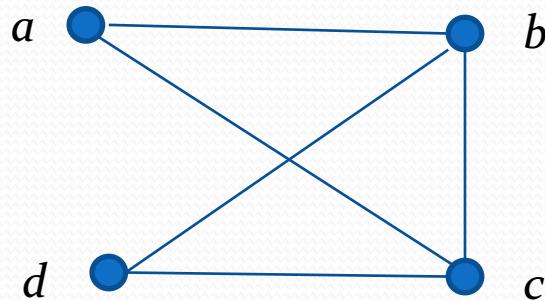
- Introduction to Graphs
- Graph Taxonomy
- Graph Models

Graphs

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

Example:

This is a graph with four vertices and five edges.



Remarks:

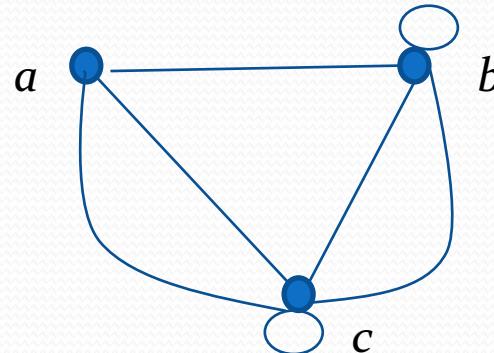
- The graphs we study here are unrelated to graphs of functions studied in Chapter 2.
- We have a lot of freedom when we draw a picture of a graph. All that matters is the connections made by the edges, not the particular geometry depicted. For example, the lengths of edges, whether edges cross, how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

Some Terminology

- In a *simple graph* each edge connects two different vertices and no two edges connect the same pair of vertices.
- *Multigraphs* may have multiple edges connecting the same two vertices. When m different edges connect the vertices u and v , we say that $\{u,v\}$ is an edge of *multiplicity* m .
- An edge that connects a vertex to itself is called a *loop*.
- A *pseudograph* may include loops, as well as multiple edges connecting the same pair of vertices.

Example:

This pseudograph has both multiple edges and a loop.



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

Directed Graphs

Definition: 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:

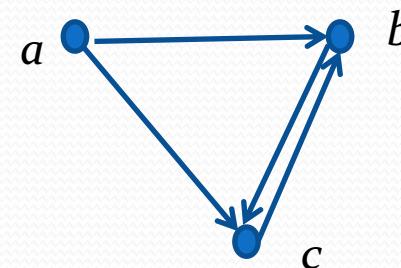
- Graphs where the end points of an edge are not ordered are said to be *undirected graphs*.

Some Terminology (*continued*)

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

Example:

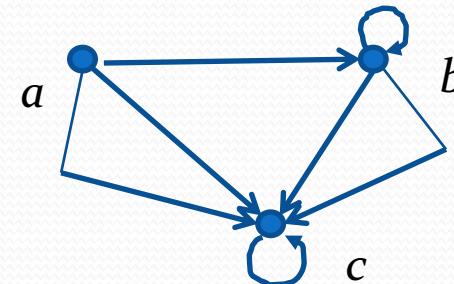
This is a directed graph with three vertices and four edges.



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

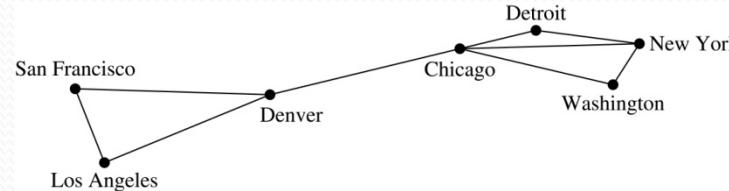
Example:

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



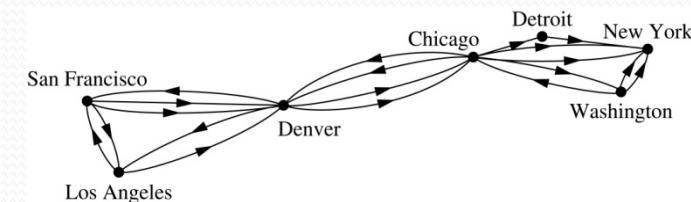
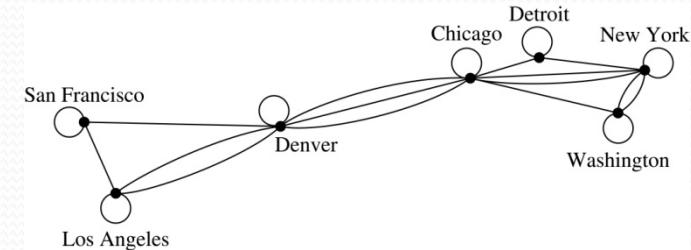
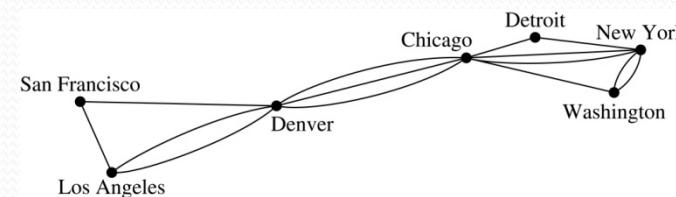
Graph Models: Computer Networks

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



Graph Models: Computer Networks (*continued*)

- To model a computer network where we care about the number of links between data centers, we use a multigraph.
- To model a computer network with diagnostic links at data centers, we use a pseudograph, as loops are needed.
- 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

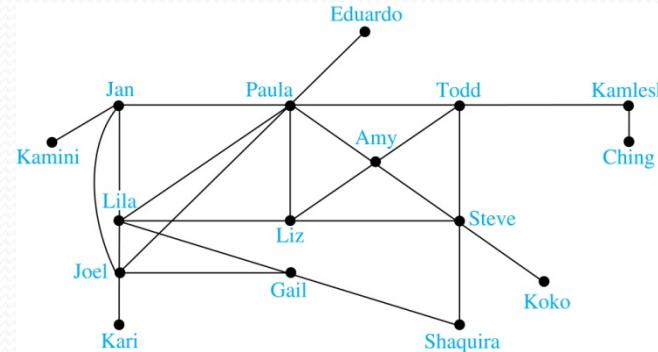
- We will illustrate how graph theory can be used in models of:
 - Social networks
 - Communications networks
 - Information networks
 - Software design
 - Transportation networks
 - Biological networks
- It's a challenge to find a subject to which graph theory has not yet been applied. Can you find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

Graph Models: Social Networks

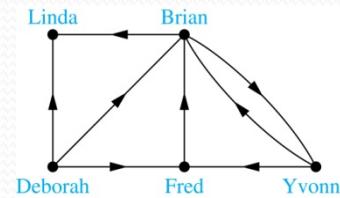
- Graphs can be used to model social structures based on different kinds of relationships between people or groups.
- In a *social network*, vertices represent individuals or organizations and edges represent relationships between them.
- Useful graph models of social networks include:
 - *friendship graphs* - undirected graphs where two people are connected if they are friends (in the real world, on Facebook, or in a particular virtual world, and so on.)
 - *collaboration graphs* - undirected graphs where two people are connected if they collaborate in a specific way
 - *influence graphs* - directed graphs where there is an edge from one person to another if the first person can influence the second person

Graph Models: Social Networks *(continued)*

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



Example: An influence graph



Next Slide: Collaboration Graphs

Examples of Collaboration Graphs

- The *Hollywood graph* models the collaboration of actors in films.
 - We represent actors by vertices and we connect two vertices if the actors they represent have appeared in the same movie.
 - We will study the Hollywood Graph in Section 10.4 when we discuss Kevin Bacon numbers.
- An *academic collaboration graph* models the collaboration of researchers who have jointly written a paper in a particular subject.
 - We represent researchers in a particular academic discipline using vertices.
 - We connect the vertices representing two researchers in this discipline if they are coauthors of a paper.
 - We will study the academic collaboration graph for mathematicians when we discuss *Erdős numbers* in Section 10.4.

Applications to Information Networks

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

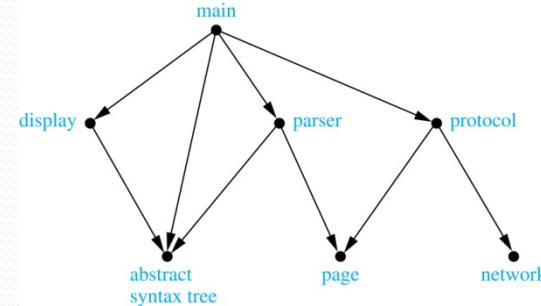
Transportation Graphs

- Graph models are extensively used in the study of transportation networks.
- Airline networks can be modeled using directed multigraphs where
 - airports are represented by vertices
 - each flight is represented by a directed edge from the vertex representing the departure airport to the vertex representing the destination airport
- Road networks can be modeled using graphs where
 - vertices represent intersections and edges represent roads.
 - undirected edges represent two-way roads and directed edges represent one-way roads.

Software Design Applications

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

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



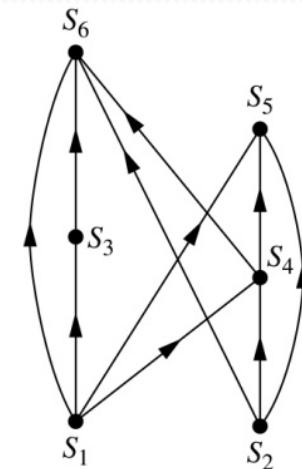
Software Design Applications

(continued)

- We can use a directed graph called a *precedence graph* to represent which statements must have already been executed before we execute each statement.
 - Vertices represent statements in a computer program
 - There is a directed edge from a vertex to a second vertex if the second vertex cannot be executed before the first

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

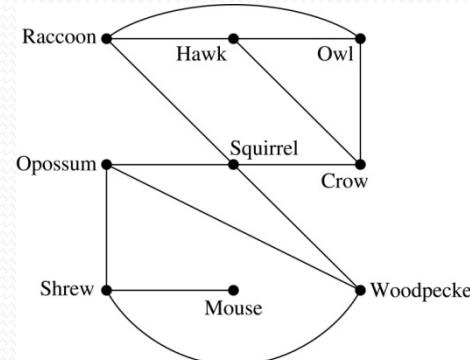
$S_1 \quad a := 0$
 $S_2 \quad b := 1$
 $S_3 \quad c := a + 1$
 $S_4 \quad d := b + a$
 $S_5 \quad e := d + 1$
 $S_6 \quad e := c + d$



Biological Applications

- Graph models are used extensively in many areas of the biological science. We will describe two such models, one to ecology and the other to molecular biology.
- *Niche overlap graphs* model competition between species in an ecosystem
 - Vertices represent species and an edge connects two vertices when they represent species who compete for food resources.

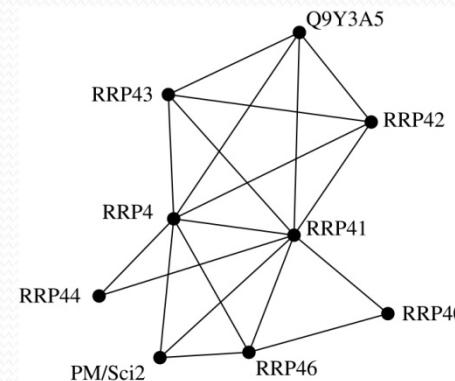
Example: This is the niche overlap graph for a forest ecosystem with nine species.



Biological Applications (*continued*)

- We can model the interaction of proteins in a cell using a *protein interaction network*.
- In a *protein interaction graph*, vertices represent proteins and vertices are connected by an edge if the proteins they represent interact.
- Protein interaction graphs can be huge and can contain more than 100,000 vertices, each representing a different protein, and more than 1,000,000 edges, each representing an interaction between proteins
- Protein interaction graphs are often split into smaller graphs, called *modules*, which represent the interactions between proteins involved in a particular function.

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



Graph Terminology and Special Types of Graphs

Section 10.2

Section Summary

- Basic Terminology
- Some Special Types of Graphs
- Bipartite Graphs
- Bipartite Graphs and Matchings (*not currently included in overheads*)
- Some Applications of Special Types of Graphs (*not currently included in overheads*)
- New Graphs from Old

Basic Terminology

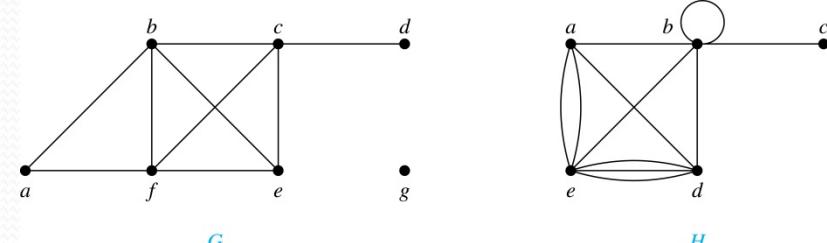
Definition 1. Two vertices u, v in an undirected graph G are called *adjacent* (or *neighbors*) in G if there is an edge e between u and v . Such an edge e is called *incident with* the vertices u and v and e is said to *connect* u and v .

Definition 2. The set of all neighbors of a vertex v of $G = (V, E)$, denoted by $N(v)$, is called the *neighborhood* of v . If A is a subset of V , we denote by $N(A)$ the set of all vertices in G that are adjacent to at least one vertex in A . So, $N(A) = \bigcup_{v \in A} N(v)$.

Definition 3. The *degree of a vertex in an undirected graph* is the number of edges incident with it, except that, by convention (but this is not always the case), a loop at a vertex contributes two to the degree of that vertex. The degree of the vertex v is denoted by $\deg(v)$.

Degrees and Neighborhoods of Vertices

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



Solution:

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

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

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

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

Degrees of Vertices

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

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

Proof:

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

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

Handshaking Theorem

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

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

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

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

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

Degree of Vertices (*continued*)

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

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

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



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

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

Directed Graphs

Recall the definition of a directed graph.

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

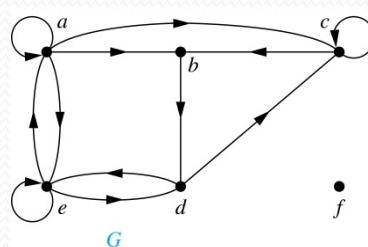
Definition: Let (u,v) be an edge in G . Then u is the *initial vertex* of this edge and is *adjacent to* v and v is the *terminal (or end) vertex* of this edge and is *adjacent from* u .

The initial and terminal vertices of a loop are the same.

Directed Graphs (*continued*)

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

Example: In the graph G we have



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

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

Directed Graphs (*continued*)

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

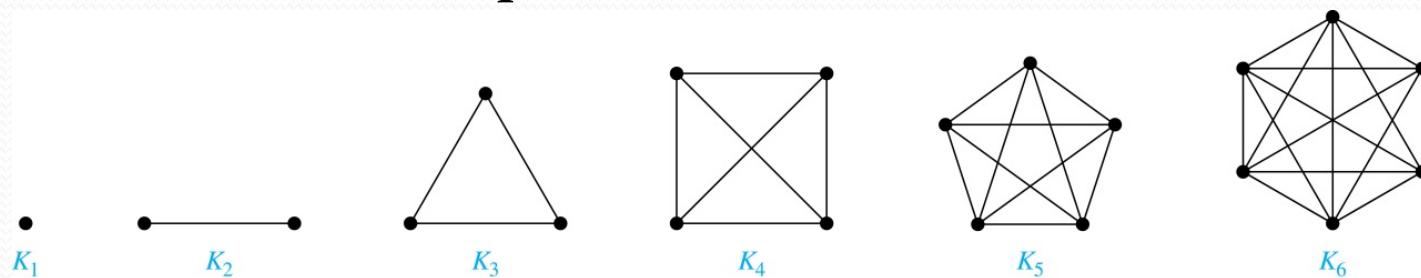
$$|E| = \sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v).$$

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

Special Types of Simple Graphs: Complete Graphs

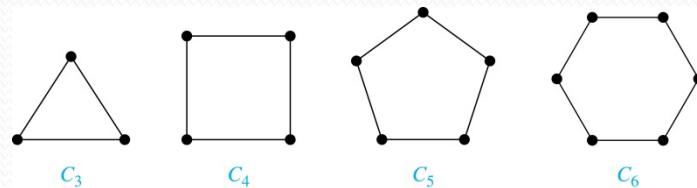
A *complete graph on n vertices*, denoted by K_n , is the simple graph that contains exactly one edge between each pair of distinct vertices.

It is also called a *clique*

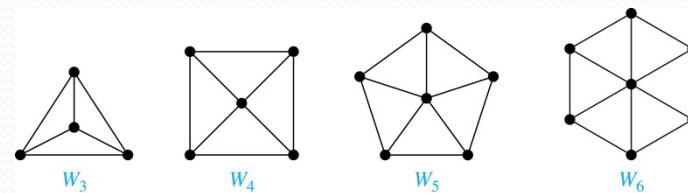


Special Types of Simple Graphs: Cycles and Wheels

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

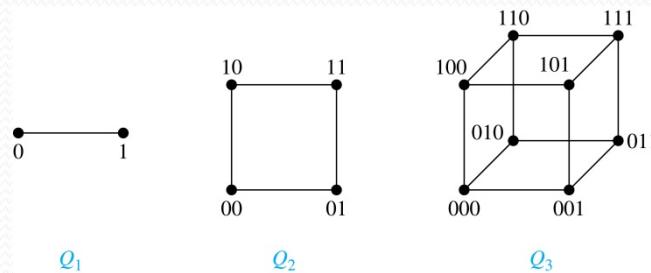


A *wheel* W_n is obtained by adding an additional vertex to a cycle C_n for $n \geq 3$ and connecting this new vertex to each of the n vertices in C_n by new edges.



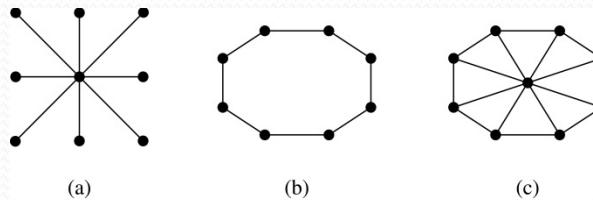
Special Types of Simple Graphs: n -Cubes

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

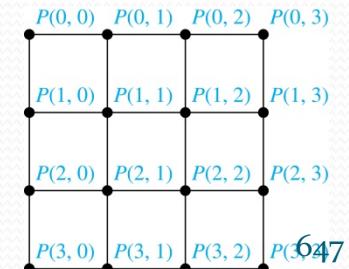


Special Types of Graphs and Computer Network Architecture

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



- Some local area networks use a *star topology*, which is a complete bipartite graph $K_{1,n}$, as shown in (a). All devices are connected to a central control device.
- Other local networks are based on a *ring topology*, where each device is connected to exactly two others using C_n , as illustrated in (b). Messages may be sent around the ring.
- Others, as illustrated in (c), use a W_n – based topology, combining the features of a star topology and a ring topology.
- Various special graphs also play a role in parallel processing where processors need to be interconnected as one processor may need the output generated by another.
 - The *n-dimensional hypercube*, or *n-cube*, Q_n , is a common way to connect processors in parallel, e.g., Intel Hypercube.
 - Another common method is the *mesh* network, illustrated here for 16 processors.

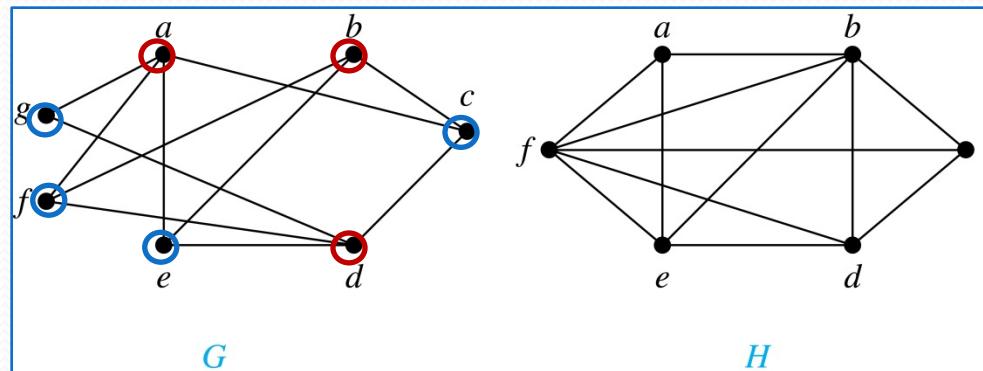


Bipartite Graphs

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

It is not hard to show that an equivalent definition of a bipartite graph is a graph where it is possible to color the vertices red or blue so that no two adjacent vertices are the same color.

G is bipartite

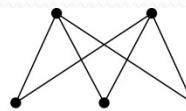


H is not bipartite since if we color a red, then the adjacent vertices f and b must both be blue.

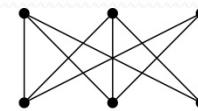
Complete Bipartite Graphs

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

Example: We display four complete bipartite graphs here.



$K_{2,3}$



$K_{3,3}$



$K_{3,5}$

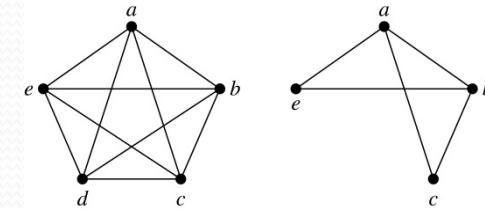


$K_{2,6}$

New Graphs from Old

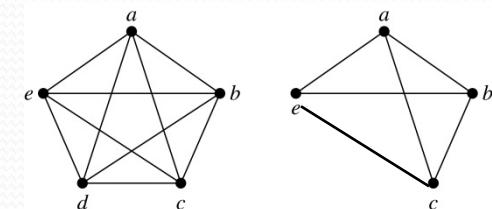
Definition: A proper *subgraph* of a graph $G = (V, E)$ is a graph (W, F) , where $W \subset V$ and $F \subset E$. Thus a subgraph H of G is a proper subgraph of G if $H \neq G$.

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



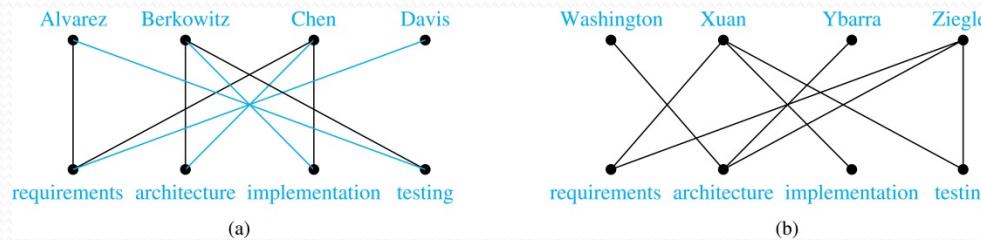
Definition: Let $G = (V, E)$ be a simple graph. The *subgraph induced* by a subset W of the vertex set V is the graph (W, F) , where the edge set F contains an edge in E if and only if both endpoints are in W .

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



Bipartite Graphs and Matchings

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



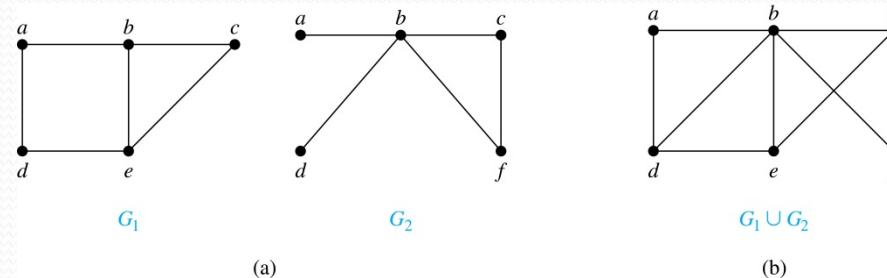
- *Marriage* - vertices represent the men and the women and edges link a man and a woman if they are an acceptable spouse. We may wish to find the largest number of possible marriages.

See the text for more about matchings in bipartite graphs.

New Graphs from Old (*continued*)

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

Example:



Representing Graphs and Graph Isomorphism

Section 10.3

Section Summary

- Adjacency Lists
- Adjacency Matrices
- Incidence Matrices
- Isomorphism of Graphs

Representing Graphs: Adjacency Lists

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

Example:

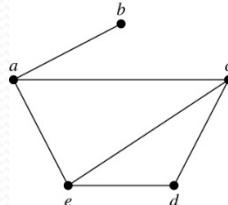


TABLE 1 An Adjacency List for a Simple Graph.

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

Example:

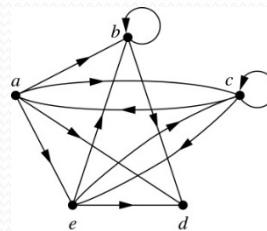


TABLE 2 An Adjacency List for a Directed Graph.

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

Representation of Graphs: Adjacency Matrices

Definition: Suppose that $G = (V, E)$ is a simple graph where $|V| = n$. Arbitrarily list the vertices of G as

v_1, v_2, \dots, v_n .

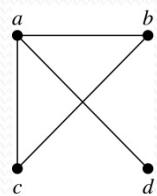
- The *adjacency matrix* A_G of G , with respect to the listing of vertices, is the $n \times n$ zero-one matrix with 1 as its (i, j) th entry when v_i and v_j are adjacent, and 0 as its (i, j) th entry when they are not adjacent.
- In other words, if the graphs adjacency matrix is

$A_G = [a_{ij}]$, then

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

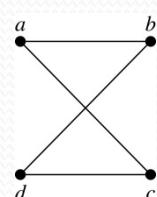
Adjacency Matrices (*continued*)

Example:



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

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



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

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

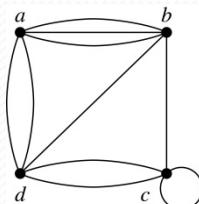
When a graph is sparse, that is, it has few edges relatively to the total number of possible edges, it is much more efficient to represent the graph using an adjacency list than an adjacency matrix. But for a dense graph, which includes a high percentage of possible edges, an adjacency matrix is preferable.

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

Adjacency Matrices (*continued*)

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

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



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

Adjacency Matrices (*continued*)

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

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

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

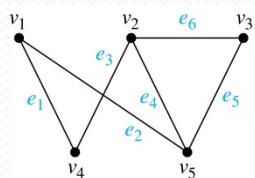
Representation of Graphs: Incidence Matrices

Definition: Let $G = (V, E)$ be an undirected graph with vertices where v_1, v_2, \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

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

Incidence Matrices (*continued*)

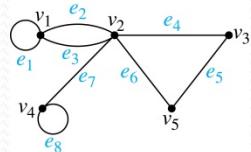
Example: Simple Graph and Incidence Matrix



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

The rows going from top to bottom represent v₁ through v₅ and the columns going from left to right represent e₁ through e₆.

Example: Pseudograph and Incidence Matrix



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

The rows going from top to bottom represent v₁ through v₅ and the columns going from left to right represent e₁ through e₈.

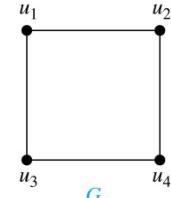
Isomorphism of Graphs

Definition: The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there is a bijection (permutation) f from V_1 to V_2 with the property that a and b are adjacent in G_1 if and only if $f(a)$ and $f(b)$ are adjacent in G_2 , for all a and b in V_1 .

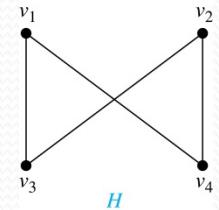
Such a function f is called an *isomorphism*. Two simple graphs that are not isomorphic are called *nonisomorphic*.

Isomorphism of Graphs (cont.)

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



Solution: The function f with $f(u_1) = v_1$, $f(u_2) = v_4$, $f(u_3) = v_3$, and $f(u_4) = v_2$ is a one-to-one correspondence between V and W . Note that adjacent vertices in G are u_1 and u_2 , u_1 and u_3 , u_2 and u_4 , and u_3 and u_4 . Each of the pairs $f(u_1) = v_1$ and $f(u_2) = v_4$, $f(u_1) = v_1$ and $f(u_3) = v_3$, $f(u_2) = v_4$ and $f(u_4) = v_2$, and $f(u_3) = v_3$ and $f(u_4) = v_2$ consists of two adjacent vertices in H .

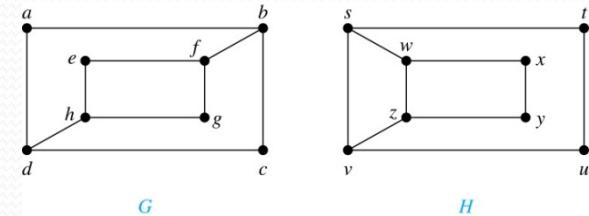


Isomorphism of Graphs (cont.)

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

Isomorphism of Graphs (cont.)

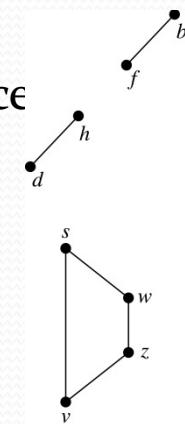
Example: Determine whether these two graphs are isomorphic.



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

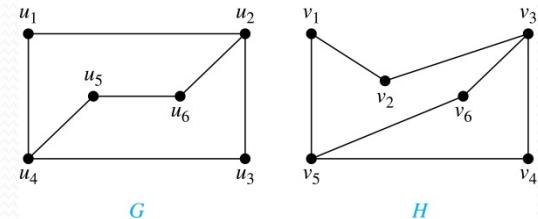
However, G and H are not isomorphic. Note that since $\deg(a) = 2$ in G , a must correspond to t , u , x , or y in H , because these are the vertices of degree 2. But each of these vertices is adjacent to another vertex of degree two in H , which is not true for a in G .

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



Isomorphism of Graphs (cont.)

Example: Determine whether these two graphs are isomorphic.



Solution: Both graphs have six vertices and seven edges. They also both have four vertices of degree two and two of degree three. The subgraphs of G and H consisting of all the vertices of degree two and the edges connecting them are isomorphic. So, it is reasonable to try to find an isomorphism f .

We define an injection f from the vertices of G to the vertices of H that preserves the degree of vertices. We will determine whether it is an isomorphism.

The function f with $f(u_1) = v_6$, $f(u_2) = v_3$, $f(u_3) = v_4$, and $f(u_4) = v_5$, $f(u_5) = v_1$, and $f(u_6) = v_2$ is a one-to-one correspondence between G and H . Showing that this correspondence preserves edges is straightforward, so we will omit the details here. Because f is an isomorphism, it follows that G and H are isomorphic graphs.

See the text for an illustration of how adjacency matrices can be used for this verification.

Algorithms for Graph Isomorphism

- The best algorithms known for determining whether two graphs are isomorphic have exponential worst-case time complexity (in the number of vertices of the graphs).
- However, there are algorithms with linear average-case time complexity.
- You can use a public domain program called NAUTY to determine in less than a second whether two graphs with as many as 100 vertices are isomorphic.
- Graph isomorphism is a problem of special interest because it is one of a few NP problems not known to be either tractable or NP-complete (see Section 3.3).

Applications of Graph Isomorphism

- The question whether graphs are isomorphic plays an important role in applications of graph theory. For example,
 - chemists use molecular graphs to model chemical compounds. Vertices represent atoms and edges represent chemical bonds. When a new compound is synthesized, a database of molecular graphs is checked to determine whether the graph representing the new compound is isomorphic to the graph of a compound that is already known.
 - Electronic circuits are modeled as graphs in which the vertices represent components and the edges represent connections between them. Graph isomorphism is the basis for
 - the verification that a particular layout of a circuit corresponds to the design's original schematics.
 - determining whether a chip from one vendor includes the intellectual property of another vendor.

Connectivity

Section 10.4

Section Summary

- Paths
- Connectedness in Undirected Graphs
- Vertex Connectivity and Edge Connectivity (*not currently included in overheads*)
- Connectedness in Directed Graphs
- Paths and Isomorphism (*not currently included in overheads*)
- Counting Paths between Vertices

Paths

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

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

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

Paths

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

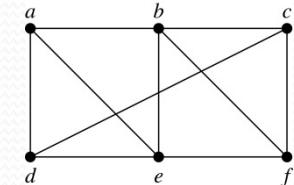
- When the graph is simple, we denote this path by its vertex sequence x_0, x_1, \dots, x_n (since listing the vertices uniquely determines the path).
- The path is a *circuit* if it begins and ends at the same vertex ($u = v$) and has length greater than zero.
- The path or circuit is said to *pass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

This terminology is readily extended to directed graphs. (see text)

Paths (*continued*)

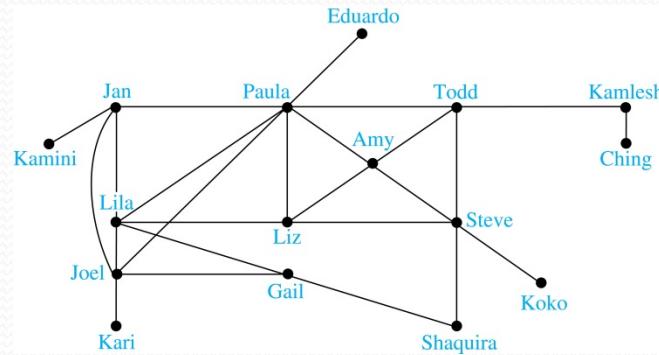
Example: In the simple graph here:

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



Degrees of Separation

Example: Paths in Acquaintanceship Graphs. In an acquaintanceship graph there is a path between two people if there is a chain of people linking these people, where two people adjacent in the chain know one another. In this graph there is a chain of six people linking Kamini and Ching.



Some have speculated that almost every pair of people in the world are linked by a small chain of no more than six, or maybe even, five people. The play *Six Degrees of Separation* by John Guare is based on this notion.

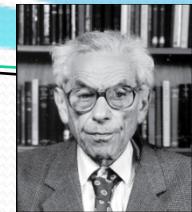
Erdős numbers

Example: *Erdős numbers.*

In a collaboration graph, two people a and b are connected by a path when there is a sequence of people starting with a and ending with b such that the endpoints of each edge in the path are people who have collaborated.

- In the academic collaboration graph of people who have written papers in mathematics, the *Erdős number* of a person m is the length of the shortest path between m and the prolific mathematician Paul Erdős.
- To learn more about Erdős numbers, visit

<http://www.ams.org/mathscinet/collaborationDistance.html>



Paul Erdős

TABLE 1 The Number of Mathematicians with a Given Erdős Number (as of early 2006).

Erdős Number	Number of People
0	1
1	504
2	6,593
3	33,605
4	83,642
5	87,760
6	40,014
7	11,591
8	3,146
9	819
10	244
11	68
12	23
13	5

Bacon Numbers

- In the Hollywood graph, two actors a and b are linked when there is a chain of actors linking a and b , where every two actors adjacent in the chain have acted in the same movie.
- The *Bacon number* of an actor c is defined to be the length of the shortest path connecting c and the well-known actor Kevin Bacon. (Note that we can define a similar number by replacing Kevin Bacon by a different actor.)
- The *oracle of Bacon* web site <http://oracleofbacon.org/how.php> provides a tool for finding Bacon numbers.

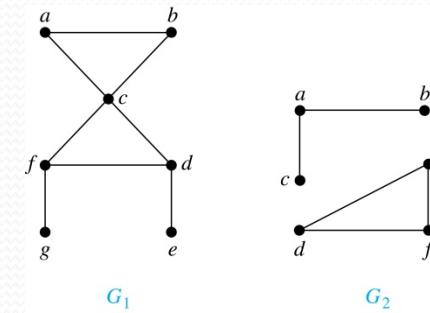
TABLE 2 The Number of Actors with a Given Bacon Number (as of early 2011).

Bacon Number	Number of People
0	1
1	2,367
2	242,407
3	785,389
4	200,602
5	14,048
6	1,277
7	114
8	16

Connectedness in Undirected Graphs

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

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

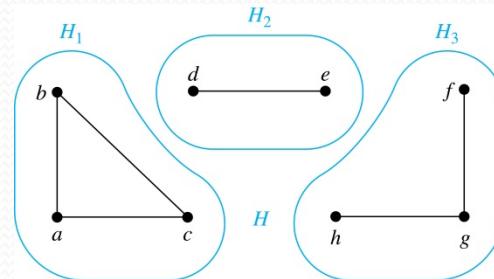


Connected Components

Definition: A *connected component* of a graph G is a connected subgraph of G that is not a proper subgraph of another connected subgraph of G .

A graph G that is not connected has two or more connected components that are disjoint and have G as their union.

Example: The graph H is the union of three disjoint subgraphs H_1 , H_2 , and H_3 , none of which are proper subgraphs of a larger connected subgraph of G . These three subgraphs are the connected components of H .



Connectedness in Directed Graphs

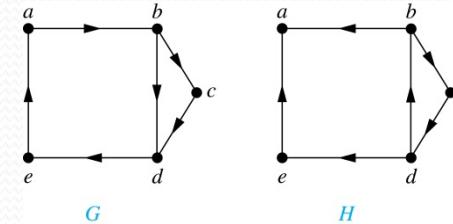
Definition: A directed graph is *strongly connected* if there is a path from a to b and a path from b to a whenever a and b are vertices in the graph.

Definition: A directed graph is *weakly connected* if there is a path between every two vertices in the underlying undirected graph, which is the undirected graph obtained by ignoring the directions of the edges of the directed graph.

Connectedness in Directed Graphs (continued)

Example: G is strongly connected because there is a path between any two vertices in the directed graph. Hence, G is also weakly connected.

The graph H is not strongly connected, since there is no directed path from a to b , but it is weakly connected.



Definition: The subgraphs of a directed graph G that are strongly connected but not contained in larger strongly connected subgraphs, that is, the maximal strongly connected subgraphs, are called the *strongly connected components* or *strong components* of G .

Example (continued): The graph H has three strongly connected components, consisting of the vertex a ; the vertex e ; and the subgraph consisting of the vertices b , c , d and edges (b,c) , (c,d) , and (d,b) .

The Connected Components of the Web Graph

- Recall that at any particular instant the web graph provides a snapshot of the web, where vertices represent web pages and edges represent links. According to a 1999 study, the Web graph at that time had over 200 million vertices and over 1.5 billion edges. (The numbers today are several orders of magnitude larger.)
- The underlying undirected graph of this Web graph has a connected component that includes approximately 90% of the vertices.
- There is a *giant strongly connected component (GSCC)* consisting of more than 53 million vertices. A Web page in this component can be reached by following links starting in any other page of the component. There are three other categories of pages with each having about 44 million vertices:
 - pages that can be reached from a page in the GSCC, but do not link back.
 - pages that link back to the GSCC, but can not be reached by following links from pages in the GSCC.
 - pages that cannot reach pages in the GSCC and can not be reached from pages in the GSCC.

Counting Paths between Vertices

- We can use the adjacency matrix of a graph to find the number of paths between two vertices in the graph.

Theorem: Let G be a graph with adjacency matrix \mathbf{A} with respect to the ordering v_1, \dots, v_n of vertices (with directed or undirected edges, multiple edges and loops allowed). The number of different paths of length r from v_i to v_j , where $r > 0$ is a positive integer, equals the (i,j) th entry of \mathbf{A}^r .

Proof by mathematical induction:

Basis Step: By definition of the adjacency matrix, the number of paths from v_i to v_j of length 1 is the (i,j) th entry of \mathbf{A} .

Inductive Step: For the inductive hypothesis, we assume that the (i,j) th entry of \mathbf{A}^r is the number of different paths of length r from v_i to v_j .

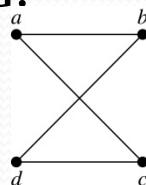
- Because $\mathbf{A}^{r+1} = \mathbf{A}^r \mathbf{A}$, the (i,j) th entry of \mathbf{A}^{r+1} equals $b_{i1}a_{1j} + b_{i2}a_{2j} + \dots + b_{in}a_{nj}$, where b_{ik} is the (i,k) th entry of \mathbf{A}^r ($\mathbf{B} = \mathbf{A}^r$). By the inductive hypothesis, b_{ik} is the number of paths of length r from v_i to v_k .
- A path of length $r + 1$ from v_i to v_j is made up of a path of length r from v_i to some v_k , and an edge from v_k to v_j . By the product rule for counting, the number of such paths is the product of the number of paths of length r from v_i to v_k (i.e., b_{ik}) and the number of edges from v_k to v_j (i.e., a_{kj}). The sum over all possible intermediate vertices v_k is $b_{i1}a_{1j} + b_{i2}a_{2j} + \dots + b_{in}a_{nj}$.



Counting Paths between Vertices (continued)

Example: How many paths of length four are there from a to d in the graph G .

G



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

adjacency
matrix of G

Solution: The adjacency matrix of G (ordering the vertices as a, b, c, d) is given above. Hence the number of paths of length four from a to d is the $(1, 4)$ th entry of A^4 . The eight paths are as:

$$\begin{array}{ll} a, b, a, b, d & a, b, a, c, d \\ a, b, d, b, d & a, b, d, c, d \\ a, c, a, b, d & a, c, a, c, d \\ a, c, d, b, d & a, c, d, c, d \end{array}$$

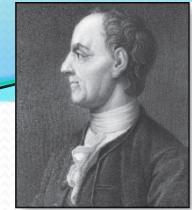
$$A^4 = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

Euler and Hamiltonian Graphs

Section 10.5

Section Summary

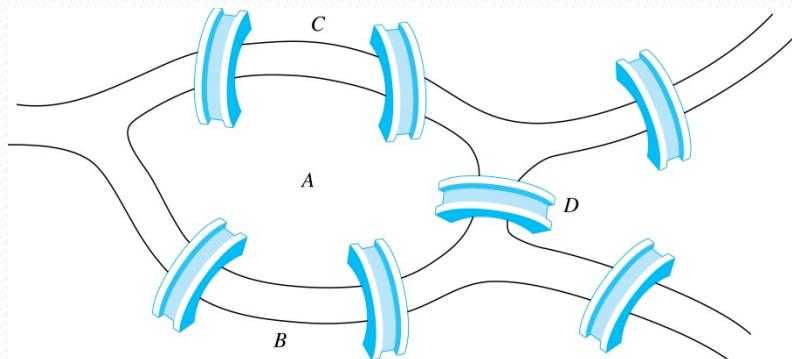
- Euler Paths and Circuits
- Hamilton Paths and Circuits
- Applications of Hamilton Circuits



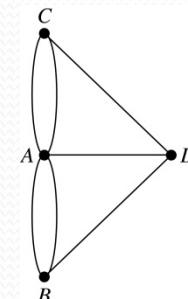
Euler Paths and Circuits

Leonard Euler
(1707-1783)

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



The 7 Bridges of Königsberg

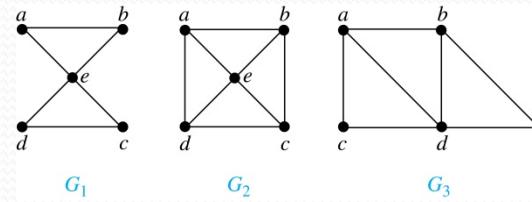


Multigraph
Model of the
Bridges of
Königsberg

Euler Paths and Circuits (*continued*)

Definition: An *Euler circuit* in a graph G is a simple circuit containing every edge of G . An *Euler path* in G is a simple path containing every edge of G .

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

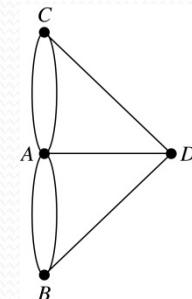


Solution: The graph G_1 has an Euler circuit (e.g., a, e, c, d, e, b, a). But, as can easily be verified by inspection, neither G_2 nor G_3 has an Euler circuit. Note that G_3 has an Euler path (e.g., a, c, d, e, b, d, a, b), but there is no Euler path in G_2 , which can be verified by inspection.

Necessary and Sufficient Conditions for Euler Circuits and Paths

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

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



Necessary Conditions for Euler Circuits and Paths

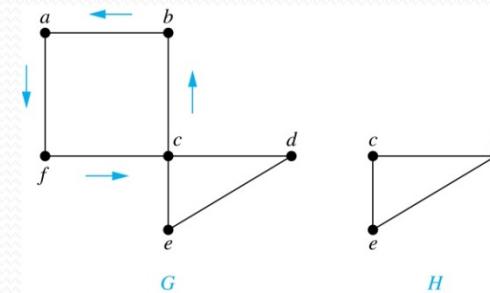
- There exists an Euler circuit \rightarrow all degrees are even
- An Euler circuit begins with a vertex a and continues with an edge incident with a , say $\{a, b\}$. The edge $\{a, b\}$ contributes one to $\deg(a)$.
- Each time the circuit passes through a vertex it contributes two to the vertex's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

Sufficient Conditions for Euler Circuits and Paths

All degrees are even \rightarrow there exists an Euler circuit

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

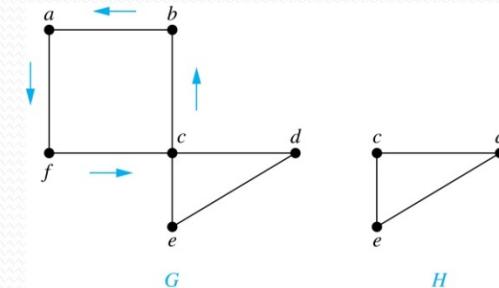
We illustrate this idea in the graph G here.
We begin at a and choose the edges $\{a, f\}, \{f, c\}, \{c, b\}$, and $\{b, a\}$ in succession.



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

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

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



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

In the example, the vertex is c .

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

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

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

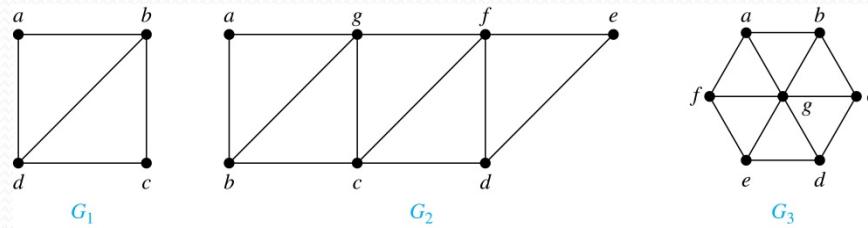
Algorithm for Constructing an Euler Circuits

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

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

Euler Circuits and Paths

Example:



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

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

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

Applications of Euler Paths and Circuits

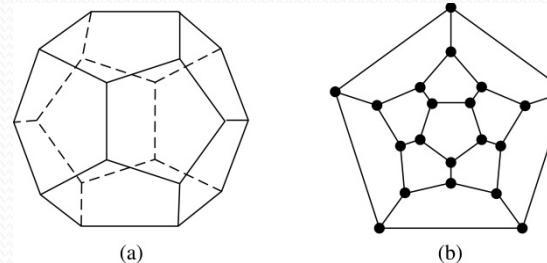
- Euler paths and circuits can be used to solve many practical problems such as finding a path or circuit that traverses each
 - street in a neighborhood,
 - road in a transportation network,
 - connection in a utility grid,
 - link in a communications network.
- Other applications are found in the
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



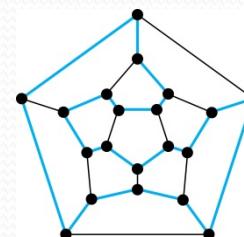
William Rowan
Hamilton
(1805- 1865)

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.
- 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 Paths and Circuits

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

That is, a simple path $x_0, x_1, \dots, x_{n-1}, x_n$ 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

(continued)

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

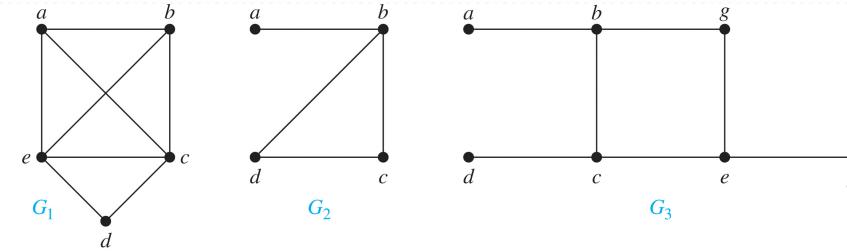


FIGURE 10 Three Simple Graphs.

Solution: G_1 has a Hamilton circuit: a, b, c, d, e, a .

G_2 does not have a Hamilton circuit (Why?), but does have a Hamilton path : a, b, c, d .

G_3 does not have a Hamilton circuit, or a Hamilton path. Why?

Necessary Conditions for Hamilton Circuits



Gabriel Andrew Dirac
(1925-1984)

- Unlike for an Euler circuit, no simple necessary and sufficient conditions are known for the existence of a Hamilton circuit.
- However, there are some useful necessary conditions. We describe two of these now.

Dirac's Theorem: If G is a simple graph with $n \geq 3$ vertices such that the degree of every vertex in G is $\geq n/2$, then G has a Hamilton circuit.

Ore's Theorem: If G is a simple graph with $n \geq 3$ vertices such that $\deg(u) + \deg(v) \geq n$ for every pair of nonadjacent vertices, then G has a Hamilton circuit.



Øysten Ore
(1899-1968)

Applications of Hamilton Paths and Circuits

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

Shortest-Path Problems

- Many problems can be modeled using graphs with weights assigned to their edges.
- Graphs that have a number assigned to each edge are called **weighted graphs**.
 - The weight between u and v is denoted as $w(u, v)$
 - Weights have to be **non-negative**, $w(u, v) \geq 0$
- As an illustration, consider how an airline system can be modeled.

Shortest-Path Problems

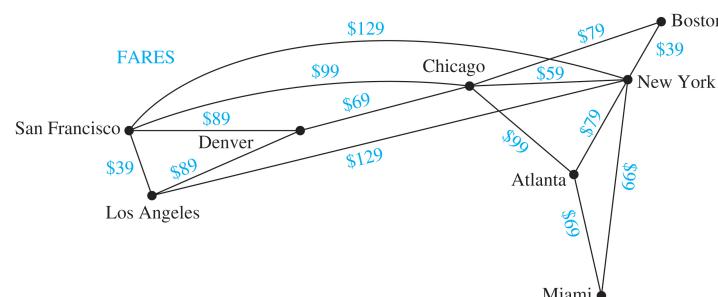
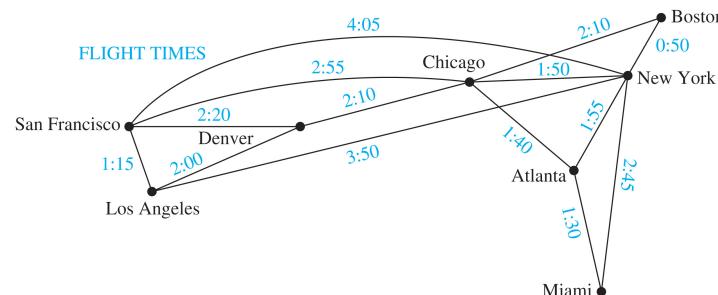
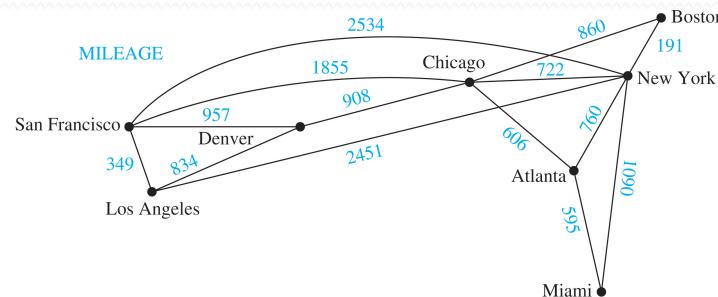


FIGURE 1 Weighted Graphs Modeling an Airline System.

Shortest-Path Problems



EDSGER WYBE DIJKSTRA (1930–2002)

- Determining a path of least length between two vertices in a network is an important problem !
 - let the *length* of a path in a weighted graph be the sum of the weights of the edges of this path.
- We will study *Dijkstra's algorithm* for finding such a shortest path.
- **Theorem:** Dijkstra's algorithm finds the length of a shortest path between two vertices in a connected simple undirected weighted graph.

Dijkstra's algorithm

- Assume we want to find the shortest path from a to z .
- It begins by labeling the starting vertex a with 0 and the other vertices with ∞ . We use the notation $L_0(a) = 0$ and $L_0(v) = \infty$ for these initial labels.
 - Before any iterations have taken place (the subscript 0 stands for the “0th” iteration).
- Dijkstra’s algorithm proceeds by forming a distinguished set of vertices. Let S_k denote this set after k iterations of the labeling procedure.
- We begin with $S_0 = \emptyset$. The set S_k is formed from S_{k-1} by adding a vertex u not in S_{k-1} with the *smallest label*.

Dijkstra's algorithm

- Let v be a vertex not in S_k . To update the label of v , note that $L_k(v)$ is the length of a shortest path from a to v *containing only vertices in S_k* as intermediate nodes (to be proved later).
- A shortest path from a to v containing only elements of S_k when u is added to S_{k-1} is either
 - a shortest path from a to v that contains only elements of S_{k-1} (that is, the distinguished vertices not including u),
 - or it is a shortest path from a to u at the $(k - 1)$ st stage with the edge $\{u, v\}$ added.
 - In this case,

$$L_k(v) = \min\{L_{k-1}(v), L_{k-1}(u) + w(u, v)\}$$

Dijkstra's algorithm

- This procedure is iterated by successively adding vertices to the distinguished set S until z is added.
- When z is added to the distinguished set, its label is the length of a shortest path from a to z .

Dijkstra's algorithm

ALGORITHM 1 Dijkstra's Algorithm.

```
procedure Dijkstra( $G$ : weighted connected simple graph, with  
all weights positive)  
{ $G$  has vertices  $a = v_0, v_1, \dots, v_n = z$  and lengths  $w(v_i, v_j)$   
where  $w(v_i, v_j) = \infty$  if  $\{v_i, v_j\}$  is not an edge in  $G$ }  
for  $i := 1$  to  $n$   
     $L(v_i) := \infty$   
 $L(a) := 0$   
 $S := \emptyset$   
{the labels are now initialized so that the label of  $a$  is 0 and all  
other labels are  $\infty$ , and  $S$  is the empty set}  
while  $z \notin S$   
     $u :=$  a vertex not in  $S$  with  $L(u)$  minimal  
     $S := S \cup \{u\}$   
    for all vertices  $v$  not in  $S$   
        if  $L(u) + w(u, v) < L(v)$  then  $L(v) := L(u) + w(u, v)$   
    {this adds a vertex to  $S$  with minimal label and updates the  
labels of vertices not in  $S$ }  
return  $L(z)$  { $L(z) =$  length of a shortest path from  $a$  to  $z$ }
```

Dijkstra's algorithm

- **Theorem:** Any sub-path of a shortest path has to be itself a shortest path.
- This can be proved by contradiction.
 - Assume the full path is a shortest path.
 - Indeed, if a sub-path is not a shortest path, the length of the full path could be decreased by replacing this sub-path by a shorter sub-path.
 - Which contradicts the fact that the full path is a shortest path.
- Thus, any sub-path of a shortest path has to be itself a shortest path.

Dijkstra's algorithm

- Moreover, a node cannot appear more than once in a shortest path.
- This can also be proved by contradiction.

Dijkstra's algorithm

- Dijkstra's algorithm: proof by induction
- Theorem: At the end of any iteration k ,
 1. for any u in S_k , $L_k(u)$ contains the length of the shortest path from a to u , $L^*(u)$.
 2. for any u not in S_k , $L_k(u)$ holds the length of the shortest path from a to u containing only intermediate nodes in S_k (except the ending node u itself).
 - Sources: Rosen's book + Kevin Wayne's presentation on Dijkstra's algorithm and Bellman-Ford algorithm (Princeton University, www.princeton.edu/~cos226).
- **Initialization:** At $k = 0$, $S_0 = \emptyset$ with $L_0(a) = 0$ and $L_0(v) = \infty$ for the other labels v .
 - It therefore holds the length of a shortest path from a *containing only intermediate vertices in $S_0 = \emptyset$* .
 - The property 2 is therefore true for $k = 0$, the basis case.
 - Moreover, initially, a is chosen to be included in S and 0 is indeed the length of the shortest path from a to a . The property 1 is therefore true at $k = 1$.

Dijkstra's algorithm

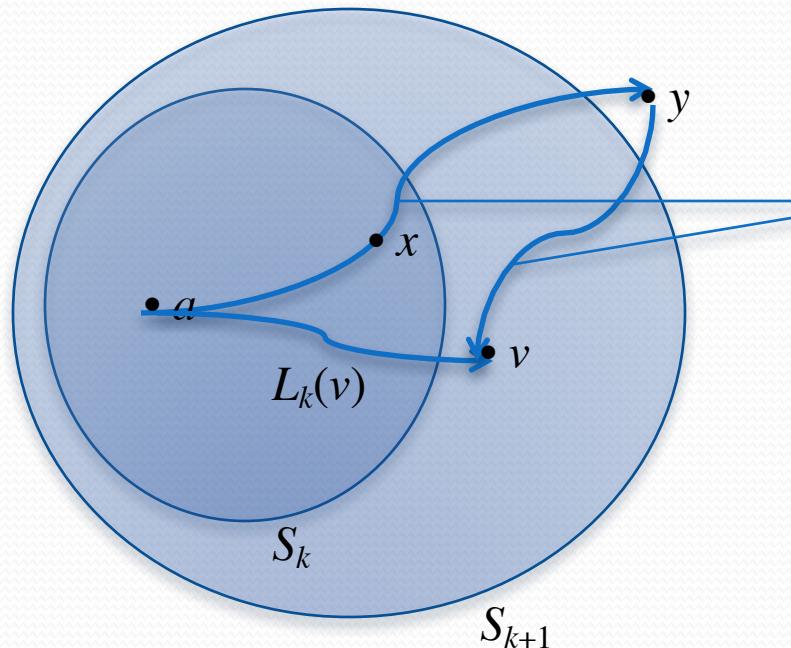
Part 1 of the theorem

- Assume the hypothesis 1 and 2 are true at the end of step k .
- Let v be a vertex not in S_k . Note that $L_k(v)$ is the length of a shortest path from a to v *containing only nodes in S_k* .
 - Assume v is the node not in S_k with *lowest L_k* .
 - This node will be added to S_k to form S_{k+1} .
 - We have to prove that $L_k(v)$ is also the length of the overall shortest path from a to v .

Dijkstra's algorithm

- By contradiction: assume there is a shorter path from a to v . It must contain nodes not in S_k .
 - Consider the node y which is *the first node not in S_k* on this shortest path from a to v (the first node leaving S_k).
 - Moreover, node x is the node in S_k *immediately preceding* y on this shortest path.

Dijkstra's algorithm



shortest path from a to v
with length $L^*(v)$

- We have $L_k(v) \stackrel{(1)}{>} L^*(v) \stackrel{(2)}{\geq} L^*(y) = L_k(y) \stackrel{(3)}{=} L_k(v) \stackrel{(4)}{\geq} L_k(v)$

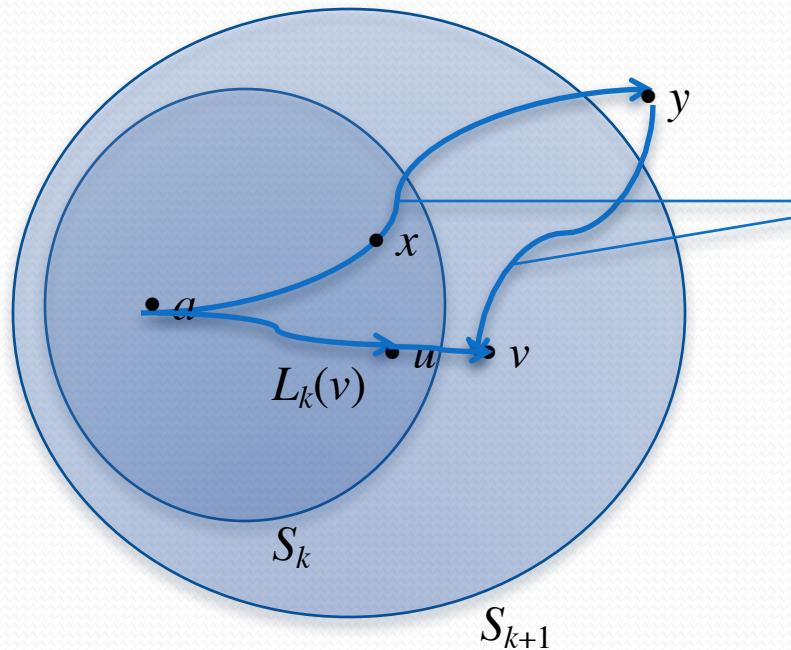
Dijkstra's algorithm

1. (1) follows from the fact that $L^*(v)$ is the length of the shortest path.
2. (2) follows from the fact that any sub-path of a shortest path is itself a shortest sub-path whose length is lower than the shortest-path length.
3. (3) follows from the induction hypothesis 2: all nodes preceding y are in S_k
4. (4) follows from the fact that v is the node not in S_k with lowest L_k .

Dijkstra's algorithm

- Here is an alternative proof of **part 1 of the theorem**.
- Assume hypotheses 1 and 2 are true at the end of step k .
- Let v be a vertex not in S_k . Note that, by induction hypothesis, $L_k(v)$ is the length of a shortest path from a to v containing *only nodes in S_k* .
 - Assume v is the node not in S_k with *the lowest L_k* .
 - This node will be added to S_k to form S_{k+1} .
 - We thus have to show that $L_k(v)$ must also be the overall shortest path from a to v :
 - By contradiction, assume there is an overall **shorter path** from a to v of length $L^*(v)$. This shorter path must pass through at least one node not in S_k .
 - Assume it is going through y (which is not in S_k), which is the first node not in S_k on this shorter path.
 - Since y is on the shorter path from a to v (a sub-path), we have $L_k(y) \leq L^*(v) < L_k(v)$, which breaks our assumption that v is the node not in S_k with *the lowest L_k* .
 - Therefore, there is no such node y .

Dijkstra's algorithm



shortest path from a to v
with length $L^*(v)$

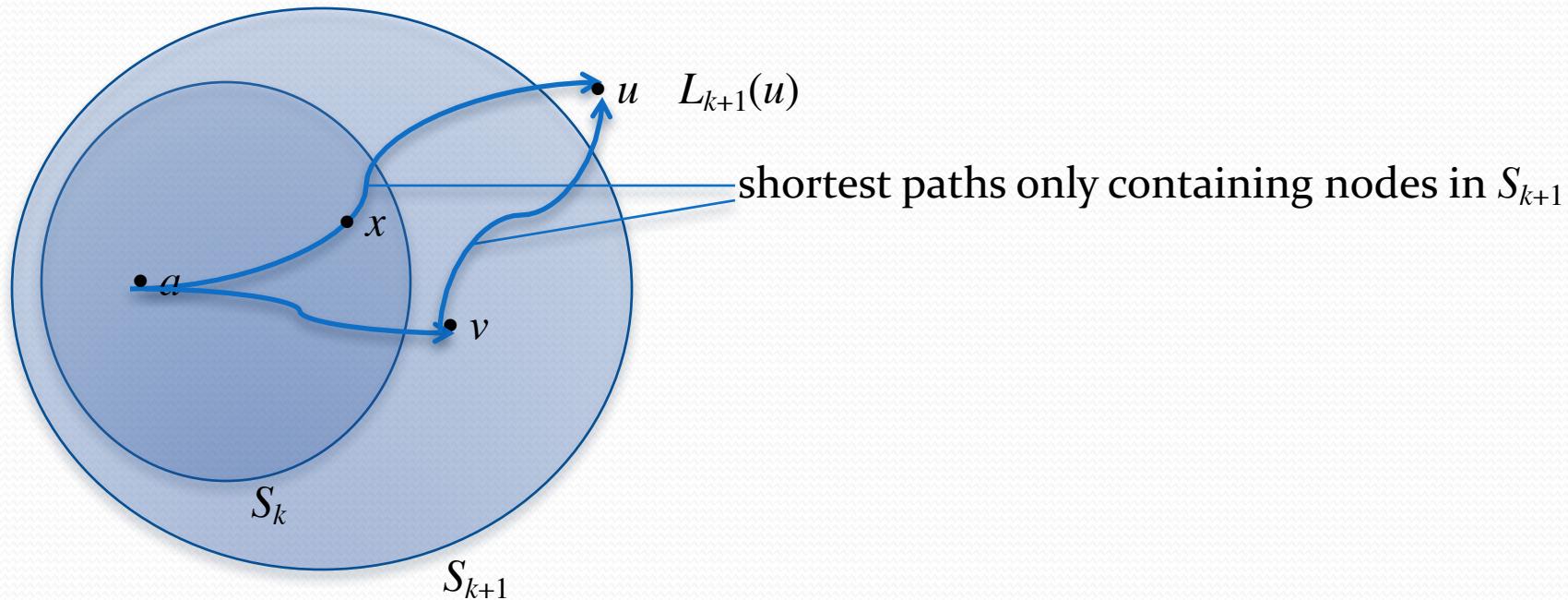
Dijkstra's algorithm

Part 2 of the theorem

- Assume the hypothesis 1 and 2 are true at step k . Moreover, we just showed that hypothesis 1 is true at step $k+1$.
- Let u be a vertex not in S_{k+1} . To update the label of u , note that, by induction hypothesis, $L_k(u)$ is the length of a shortest path from a to u containing only vertices in S_k .
 - We now have to compute $L_{k+1}(u)$ for all u not in S_{k+1} .
- A shortest path from a to u containing only elements of S_{k+1} either comes from
 - a node of S_k (that is, the distinguished vertices not including v),
 - or the new node v , added at step $k+1$.
- In this case, since the sub-path of a shortest path must itself be a shortest path,

$$L_{k+1}(u) = \min\{L_k(u), L_k(v) + w(v, u)\}$$

Dijkstra's algorithm



The last node on the shortest path to u is either a node in S_k or the node v .

Dijkstra's algorithm

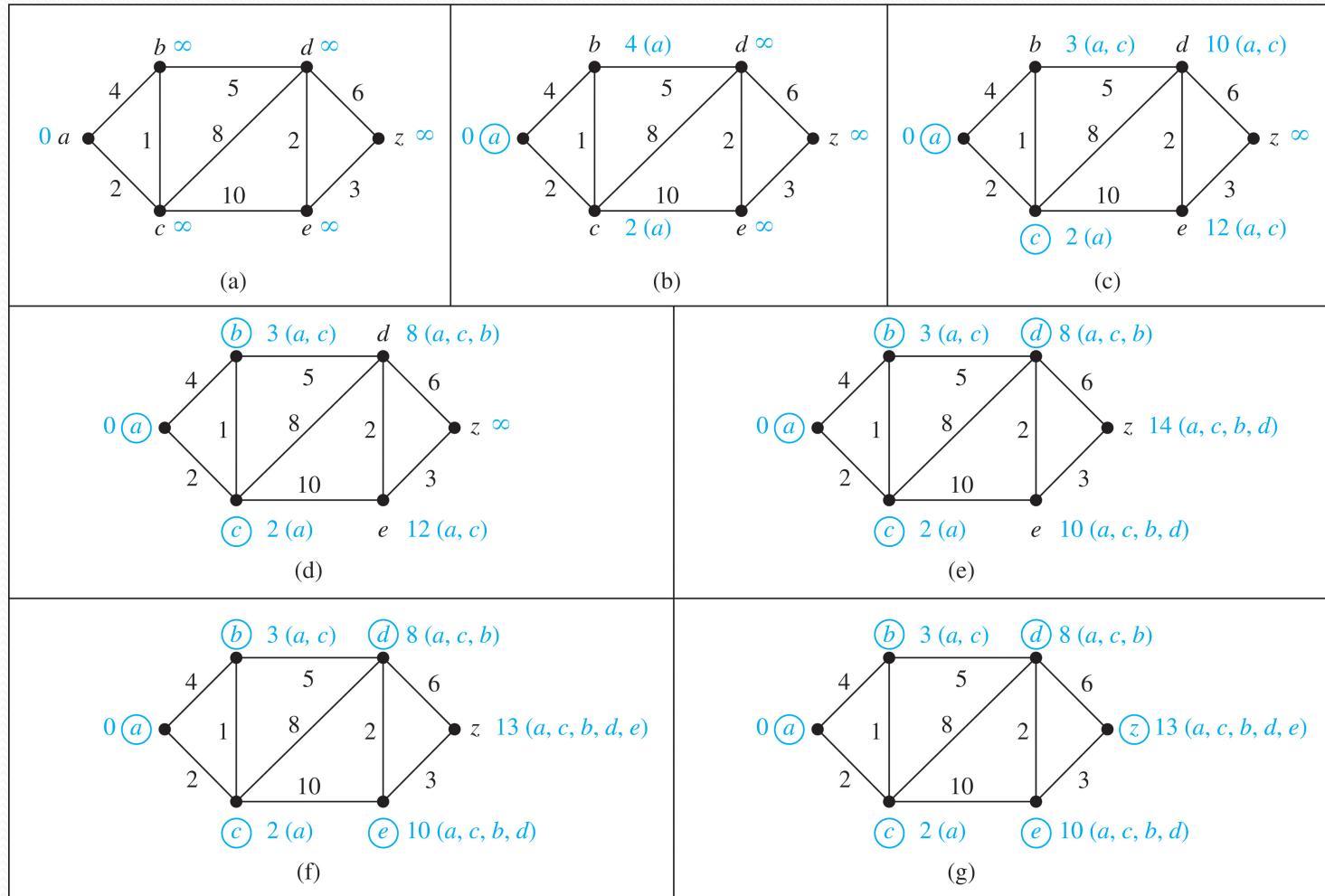


FIGURE 4 Using Dijkstra's Algorithm to Find a Shortest Path from a to z .

Planar graphs

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

Planar graphs

- Euler's formula
 - Let G be a connected planar simple graph with e edges and v vertices.
 - Let r be the number of regions in a planar representation of G .
- Then $r = e - v + 2$.
 - The proof is in the book.

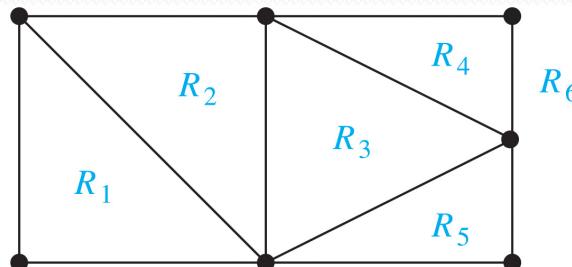


FIGURE 8 The Regions of the Planar Representation of a Graph.

Graph coloring

- A *coloring* of a simple graph is the assignment of a color to each vertex of the graph
 - so that no two adjacent vertices are assigned the same color.
- The *chromatic number* of a graph is the least number of colors needed for a coloring of this graph.
 - The chromatic number of a graph G is denoted by $\chi(G)$ (here χ is the Greek letter *chi*.).

Graph coloring

- A simple example of translating a map into a graph.
 - Two regions are linked if they have a border.

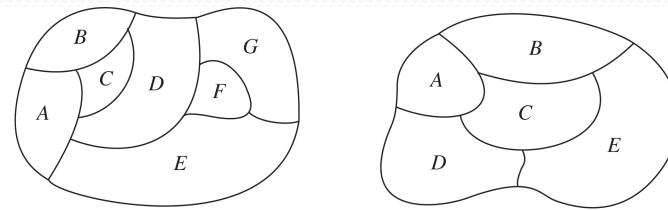


FIGURE 1 Two Maps.

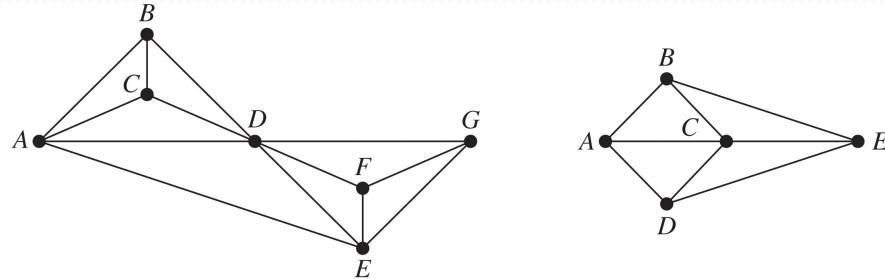


FIGURE 2 Dual Graphs of the Maps in Figure 1.

Graph coloring

- **The four color problem.**
 - The chromatic number of a planar graph is no greater than four.
 - The proof is quite difficult.
 - It involved a computer enumerating many different cases.
- Many different heuristics have been developed for graph coloring.

Graph coloring

- Here is a simple example.

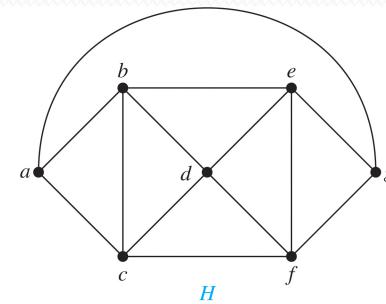
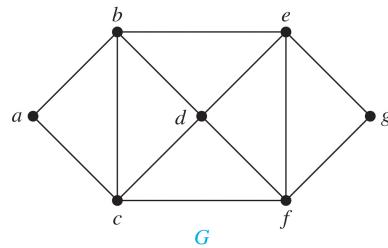


FIGURE 3 The Simple Graphs G and H .

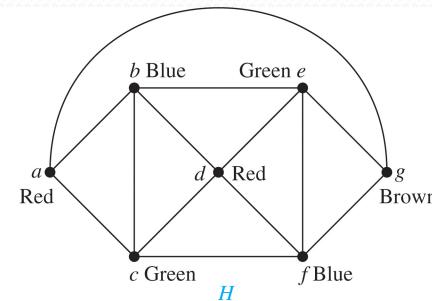
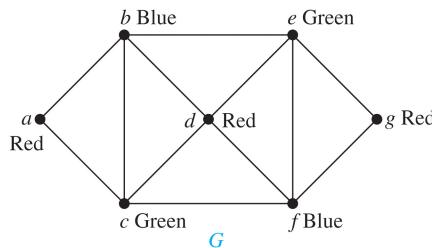
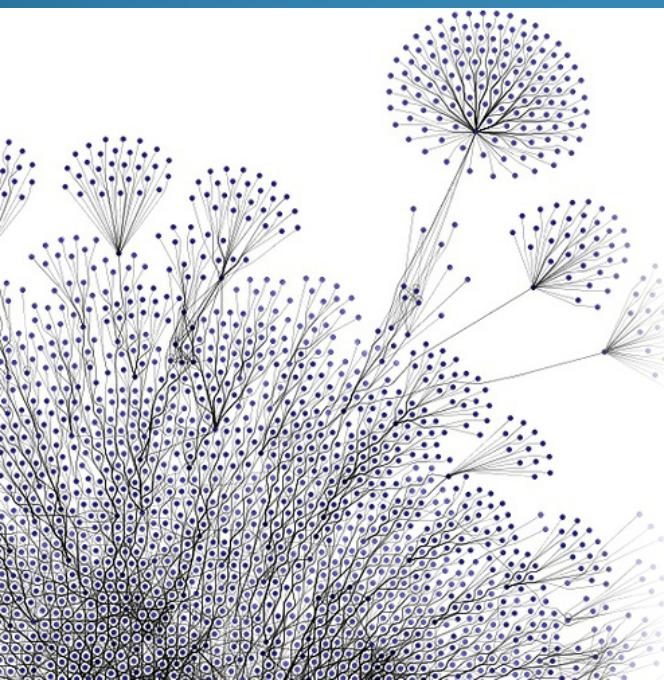


FIGURE 4 Colorings of the Graphs G and H .

Graph coloring

- An interesting application of graph coloring.
- How can the final exams at a university be scheduled so that no student has two exams at the same time?
- *Solution:* This scheduling problem can be solved using a graph model, with vertices representing courses and with an edge between two vertices if there is a common student in the courses they represent.
 - Each time slot for a final exam is represented by a different color.
 - A scheduling of the exams corresponds to a coloring of the associated graph.

The PageRank algorithm



The basic PageRank algorithm

- Introduced by Page, Brin, Motwani & Winograd in 1998
- Partly implemented in Google
- Corresponds to a measure of « prestige » in a directed graph

Web link analysis

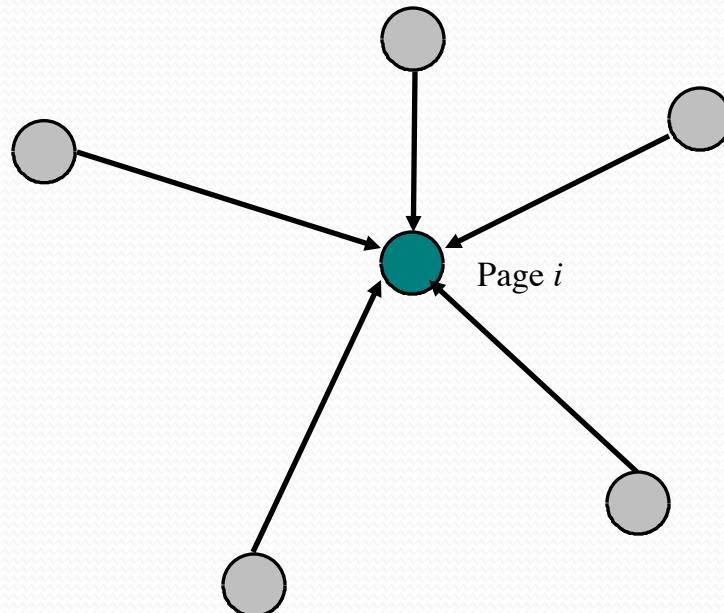
- A set of techniques
 - Applied to: Hyperlink document repositories
 - Typically web pages
- Objective:
 - To exploit the link structure of the documents
 - In order to extract interesting information
 - Viewing the document repository as a graph where
 - Nodes are documents
 - Edges are directed links between documents
 - It does not exploit the content of the pages !!

Web link analysis

- Suppose we performed a search with a search engine
- **Objective:** to improve the (content-based) ranking of the search engine
 - Based on the graph structure of the web hyperlinks
 - PageRank is computed off-line

The basic PageRank algorithm

- To each web page we associate a score, x_i
 - The score of page i , x_i , is proportional to the weighted averaged score of the pages pointing to page i



The basic PageRank algorithm

- Let w_{ij} be the weight of the link connecting page i to page j
 - Usually, it is simply 0 or 1
 - Thus, $w_{ij} = 1$ if page i has a link to page j ; $w_{ij} = 0$ otherwise
- Let \mathbf{W} be the matrix made of the elements w_{ij}
 - Notice that this matrix is not symmetric
 - We suppose that the graph is strongly connected

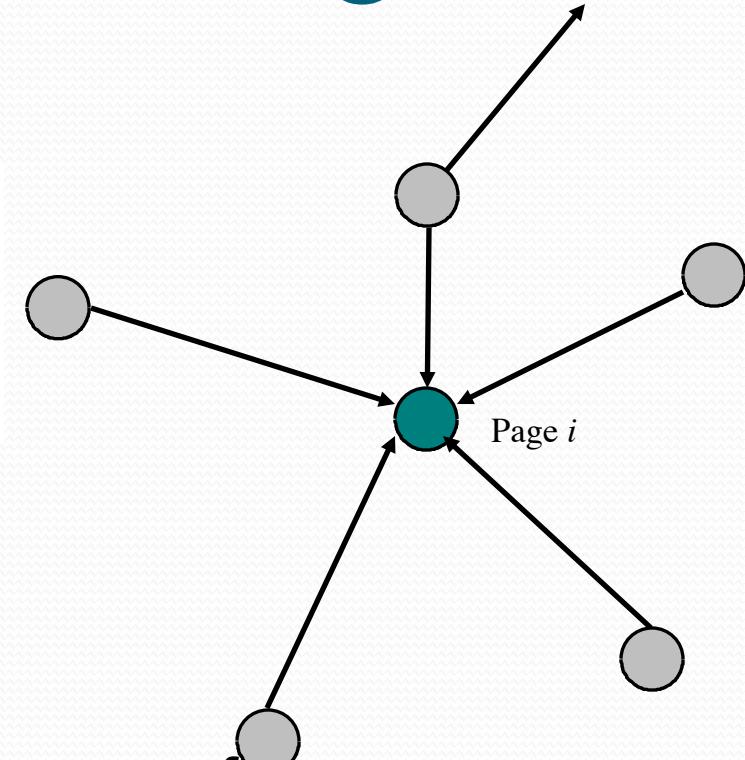
The basic PageRank algorithm

- In other words

$$x_i \propto \sum_{j=1}^n \frac{w_{ji} x_j}{w_{j.}}$$

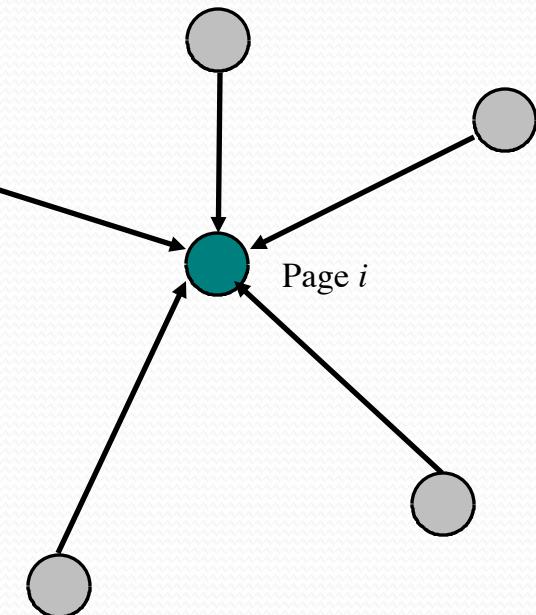
$$w_{j.} = \sum_{i=1}^n w_{ji}$$

- Where $w_{j.}$ is the outdegree of page j



The basic PageRank algorithm

- In other words,
- A page with a **high score** is a page that is pointed by
 - Many pages
 - Having each a high score
- Thus a page is an **important** page if
 - It is pointed by **many, important**, pages



The basic PageRank algorithm

- These equations can be updated **iteratively** until convergence
- In order to obtain the scores, x_i
 - We normalize the vector \mathbf{x} at each iteration
- The pages are then **ranked** according to their score

The basic PageRank algorithm

- This definition has a nice interpretation in terms of **random surfing**
- If we define the probability of following the link from page j to page i as

$$P(\text{page}(k+1) = i | \text{page}(k) = j) = \frac{w_{ji}}{w_j}.$$

$$w_{j.} = \sum_{i=1}^n w_{ji}$$

The basic PageRank algorithm

- We can write the updating equation as

$$\begin{aligned}x_i(k+1) &= \text{P}(page(k+1) = i) \\&= \sum_{j=1}^n \text{P}(page(k+1) = i | page(k) = j) x_j(k)\end{aligned}$$

- And thus we can define a **random surfer** following the links according to the transition probabilities

$$\text{P}(page(k+1) = i | page(k) = j) = \frac{w_{ji}}{w_j}.$$

The basic PageRank algorithm

- This is the equation of a **Markov model** of **random surf** through the web
- This is exactly the same equation as before:

$$x_i \propto \sum_{j=1}^n \frac{w_{ji} x_j}{w_{j.}}$$

$$w_{j.} = \sum_{i=1}^n w_{ji}$$

The basic PageRank algorithm

- If we denote element i, j of the transition probability matrix \mathbf{P} as p_{ij}
- We thus have

$$[\mathbf{P}]_{ij} = p_{ij} = \text{P}(page(k+1) = j | page(k) = i)$$

- And the equation can be rewritten as

$$x_i(k+1) = \sum_{j=1}^n p_{ji} x_j(k)$$

The basic PageRank algorithm

- In matrix form, if the vector \mathbf{x} has elements x_i

$$\mathbf{x}(k+1) = \mathbf{P}^T \mathbf{x}(k)$$

- The **stationary distribution** is given by $\mathbf{x}(k+1) = \mathbf{x}(k)$, and thus

$$\mathbf{x} = \mathbf{P}^T \mathbf{x}$$

The basic PageRank algorithm

- x_i can then be viewed as the probability of being at page i
 - The solution to these equations is the **stationary distribution** of the random surf
 - Which is the probability of finding the surfer on page i on the **long-term behaviour**
- The « most probable page » is the best ranked

The basic PageRank algorithm

- The PageRank scores can be obtained
 - By computing **the left eigenvector** of the matrix \mathbf{P} corresponding to **eigenvalue 1**
 - Which is the right eigenvector of \mathbf{P}^T
 - Where \mathbf{P} is **the transition probabilities matrix** of the Markov process
 - Containing the transition probabilities
- If the graph is undirected, the scores are simply the **indegrees** of the nodes

The basic PageRank algorithm

- For that purpose, we can use the well-known power method
- The computation of the dominant right eigenvector \mathbf{x} of a matrix \mathbf{M} iterates

$$\begin{cases} \mathbf{x} \leftarrow \mathbf{Mx} \\ \mathbf{x} \leftarrow \frac{\mathbf{x}}{\|\mathbf{x}\|} \end{cases}$$

- And, in our case, $\mathbf{M} = \mathbf{P}^T$

Adjustments to the basic model

- However, there is a problem with
 - Dangling nodes
 - That is, nodes without any outgoing link
- In this case, the **P** matrix is no more stochastic
 - Rows do not sum to one
- Moreover, the graph could have separate components

Adjustments to the basic model

- One potential solution is to allow to jump to any node of the graph
 - With some non-zero probability (teleportation)
- Thus.

$$\mathbf{G} = \alpha \mathbf{P} + (1 - \alpha) \frac{\mathbf{e}\mathbf{e}^T}{n}$$

where \mathbf{G} is called the Google matrix, \mathbf{e} is a column vector full of 1's and $0 < \alpha < 1$

Adjustments to the basic model

- Moreover, the dangling nodes are made absorbing
 - That is, if i is a dangling node, $p_{ii} = 1$ and $p_{ij} = 0$ for $i \neq j$
- So that the row sum is equal to 1
- And since there is a transportation probability, random walkers will be able to escape from these dangling nodes

Adjustments to the basic model

- In this case,
 - The matrix is stochastic
 - The matrix is irreducible (no separate component)
 - The matrix is aperiodic
- Then, there is a unique eigenvector associated to eigenvalue 1
- However, G is no more sparse !

Computing PageRank

- The problem is thus to find the **left eigenvector** of \mathbf{G}

- corresponding to the eigenvalue 1
 - instead of \mathbf{P}

$$\mathbf{x}^T \mathbf{G} = \mathbf{x}^T$$

- with the normalization
- One can use the standard $\textcolor{teal}{p} \|\mathbf{x}\|_1 = 1$ or $\mathbf{x}^T \mathbf{e} = 1$

Computing PageRank

- Fortunately, the power method results in sparse matrix multiplication only:

$$\begin{aligned}\mathbf{x}^T(k+1) &= \mathbf{x}^T(k)\mathbf{G} \\ &= \alpha \mathbf{x}^T(k)\mathbf{P} + \frac{(1-\alpha)}{n} \mathbf{x}^T(k)\mathbf{e}\mathbf{e}^T \\ &= \alpha \mathbf{x}^T(k)\mathbf{P} + \frac{(1-\alpha)}{n} \mathbf{e}^T\end{aligned}$$

Personalization in PageRank

- How can we **favour** some pages in a natural way (advertising, etc) ?
- Rather than using $\mathbf{e}\mathbf{e}^T/n$,
- use $\mathbf{e}\mathbf{v}^T$ where
 - $\mathbf{v} > 0$ is a **probability vector** ($\mathbf{v}^T\mathbf{e} = 1$)
 - Which is called the **personalization vector**
- It contains the **a priori probability** of jumping to any page

Personalization in PageRank

- The Google matrix thus becomes

$$\mathbf{G} = \alpha \mathbf{P} + (1 - \alpha) \mathbf{e} \mathbf{v}^T$$

- Where \mathbf{v} is provided a priori

The PageRank problem as a sparse linear system

- Here is an alternative formulation of the PageRank problem
 - As a sparse linear system

$$\mathbf{x}^T \mathbf{G} = \mathbf{x}^T$$

$$\Rightarrow \mathbf{x}^T (\alpha \mathbf{P} + (1 - \alpha) \mathbf{e} \mathbf{v}^T) = \mathbf{x}^T$$

$$\Rightarrow \alpha \mathbf{x}^T \mathbf{P} + (1 - \alpha) \mathbf{v}^T = \mathbf{x}^T$$

$$\Rightarrow \mathbf{x}^T (\mathbf{I} - \alpha \mathbf{P}) = (1 - \alpha) \mathbf{v}^T$$

$$\Rightarrow (\mathbf{I} - \alpha \mathbf{P})^T \mathbf{x} = (1 - \alpha) \mathbf{v}$$

The PageRank problem as a sparse linear system

- In other words, the problem is

$$\text{Solve } (\mathbf{I} - \alpha \mathbf{P})^T \mathbf{x} = (1 - \alpha) \mathbf{v} \text{ with } \mathbf{x}^T \mathbf{e} = 1$$

- Which has been shown (Del Corso, Gulli & Romani, 2005; Langville & Meyer, 2006) to be equivalent to

$$\text{Solve } (\mathbf{I} - \alpha \mathbf{P})^T \mathbf{x}' = \mathbf{v} \text{ and compute } \mathbf{x} = \mathbf{x}' / \|\mathbf{x}'\|_1$$

Trees

Chapter 11

Chapter Summary

- Introduction to Trees
- Applications of Trees (*not currently included in overheads*)
- Tree Traversal
- Spanning Trees
- Minimum Spanning Trees (*not currently included in overheads*)

Introduction to Trees

Section 11.1

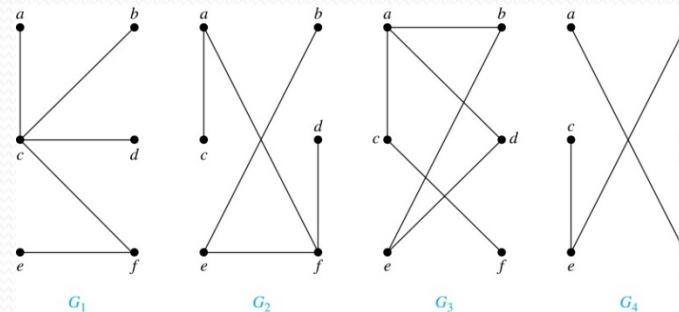
Section Summary

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

Trees

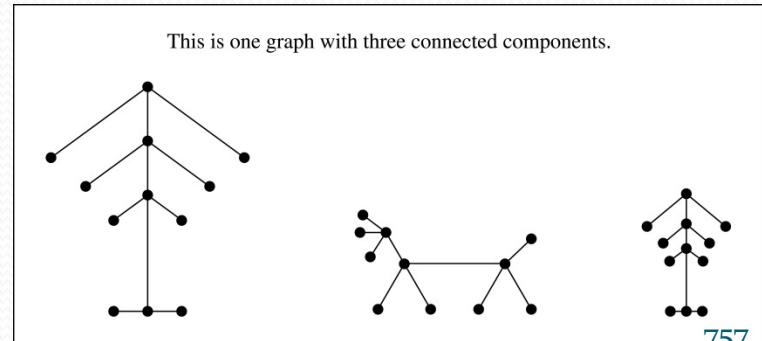
Definition: A *tree* is a connected undirected graph with no simple circuits.

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.

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 (*continued*)

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

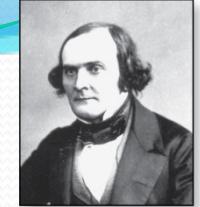
Proof: Assume that T is a tree. Then T is connected with no simple circuits. Hence, if x and y are distinct vertices of T , there is a simple path between them. This path must be unique - for if there were a second path, there would be a simple circuit in T . Hence, there is a unique simple path between any two vertices of a tree.

Now assume that there is a unique simple path between any two vertices of a graph T . Then T is connected because there is a path between any two of its vertices. Furthermore, T can have no simple circuits since if there were a simple circuit, there would be two paths between some two vertices.

Hence, a graph with a unique simple path between any two vertices is a tree. ◀

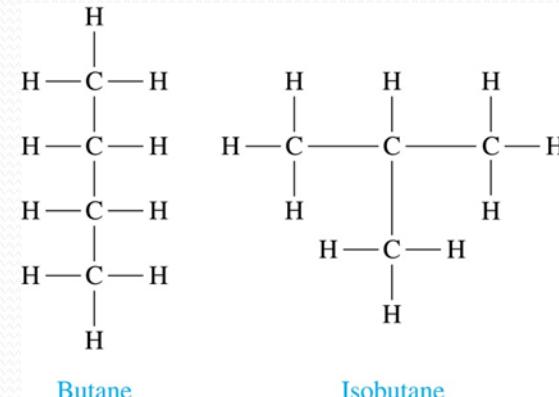
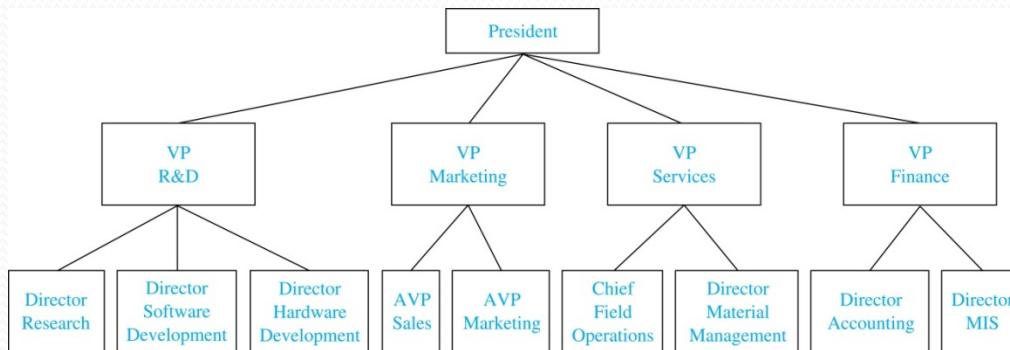


Arthur Cayley
(1821-1895)



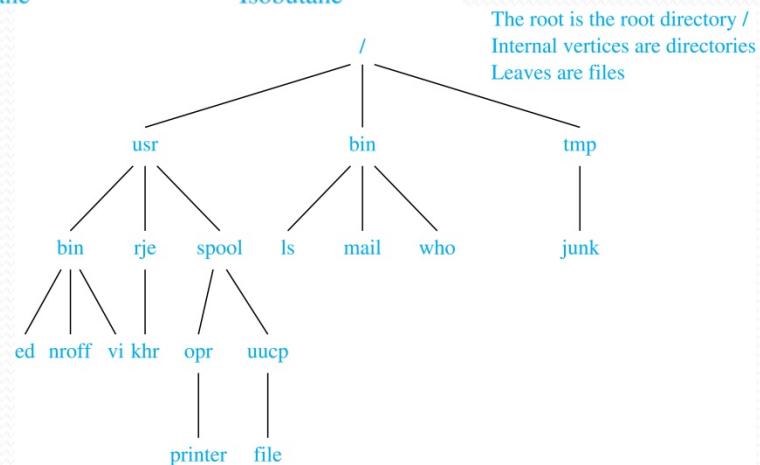
Trees as Models

- 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.
- 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.



Butane

Isobutane



Trees as Models

- Trees are also widely used in game theory. Assume there are two opponents.
- We model such games using **game trees**.
- The vertices of these trees represent the positions that a game can be in as it progresses.
- The edges represent legal moves between these positions.

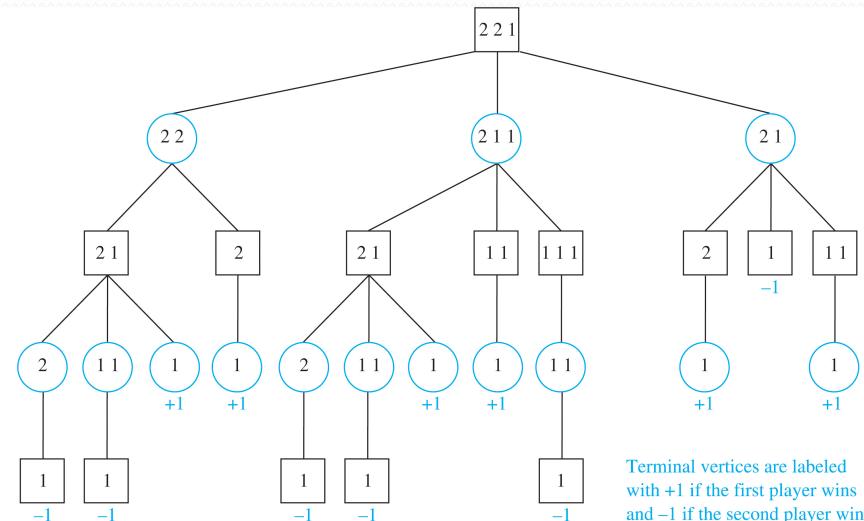


FIGURE 7 The Game Tree for a Game of Nim.

Trees as Models

- Examples: game of nim and tic-tac-toe.

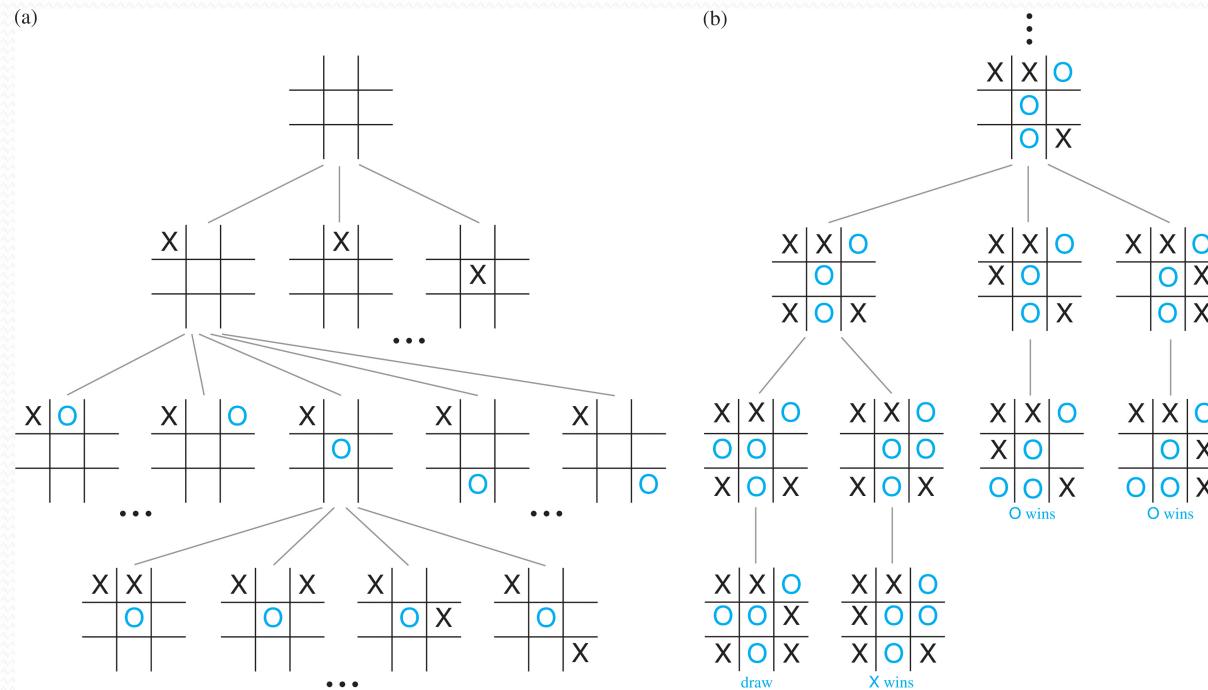
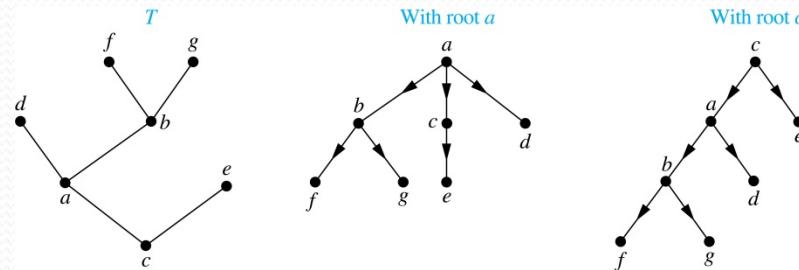


FIGURE 8 Some of the Game Tree for Tic-Tac-Toe.

Rooted Trees

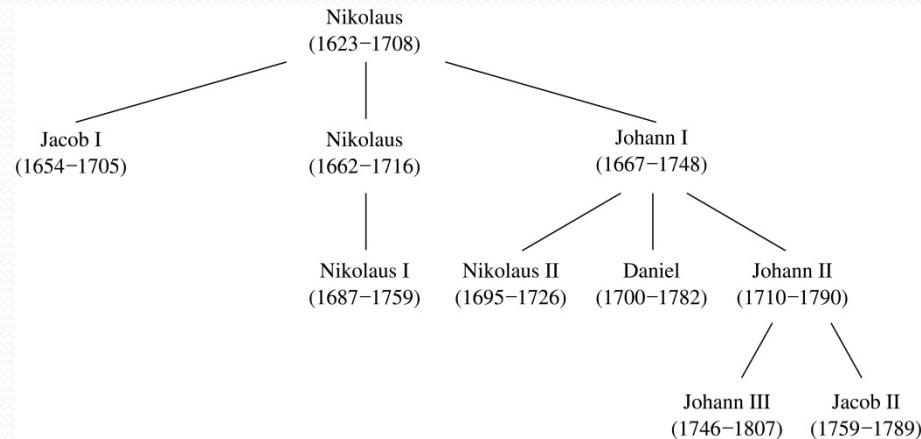
Definition: A *rooted tree* is a tree in which one vertex has been designated as the *root* and every edge is directed away from the root.

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).

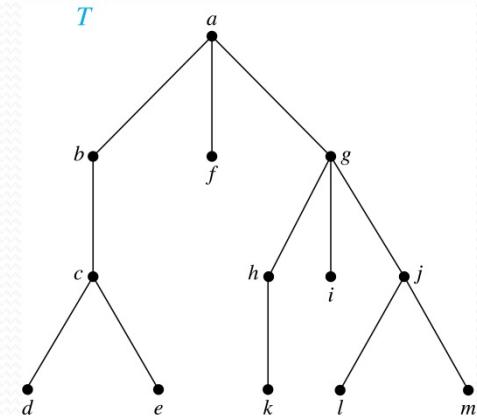


- If v is a vertex of a rooted tree other than the root, the *parent* of v is the unique vertex u such that there is a directed edge from u to v . When u is a parent of v , v is called a *child* of u . Vertices with the same parent are called *siblings*.
- The *ancestors* of a vertex are the vertices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

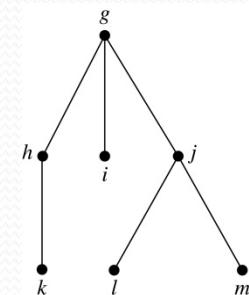
Example: In the rooted tree T (with root a):

- (i) Find the parent of c , the children of g , the siblings of h , 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:

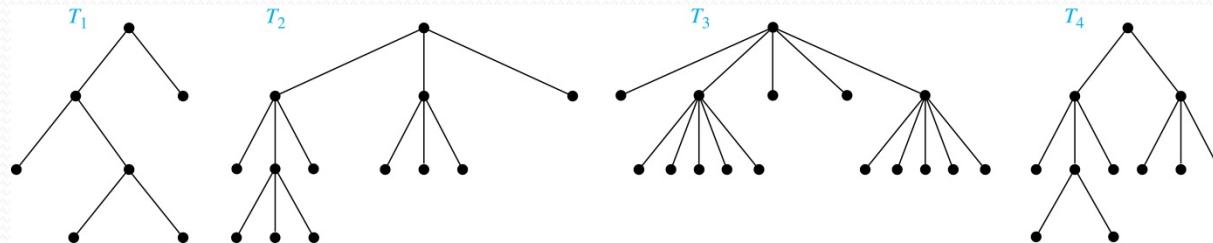
- (i) The parent of c is b . The children of g are h , i , and j . The siblings of h are i and j . The ancestors of e are c , b , and a . The descendants of b are c , d , and e .
- (ii) The internal vertices are a , b , c , g , h , and j . The leaves are d , e , f , i , k , l , and m .
- (iii) We display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

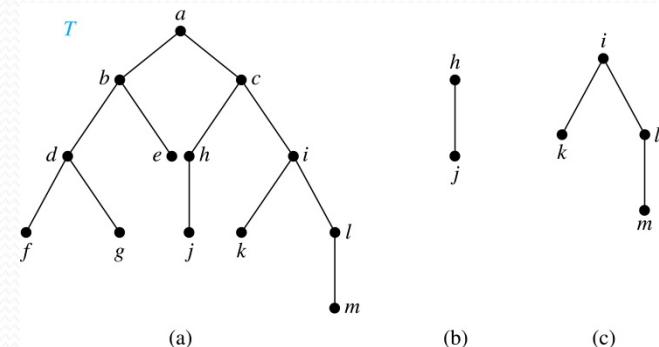
Definition: A *binary tree* is an ordered rooted where each internal vertex has at most two children. If an internal vertex of a binary tree has two children, the first is called the *left child* and the second the *right child*. The tree rooted at the left child of a vertex is called the *left subtree*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

Example: Consider the binary tree T .

- What are the left and right children of d ?
- What are the left and right subtrees of c ?

Solution:

- The left child of d is f and the right child is g .
- The left and right subtrees of c are displayed in (b) and (c).



Properties of Trees

Theorem 2: A tree with n vertices has $n - 1$ edges.

Proof (by mathematical induction):

BASIS STEP: When $n = 1$, a tree with one vertex has no edges. Hence, the theorem holds when $n = 1$.

INDUCTIVE STEP: Assume that every tree with k vertices has $k - 1$ edges.

Suppose that a tree T has $k + 1$ vertices and that v is a leaf of T . Let w be the parent of v . Removing the vertex v and the edge connecting w to v produces a tree T' with k vertices. By the inductive hypothesis, T' has $k - 1$ edges. Because T has one more edge than T' , we see that T has k edges. This completes the inductive step. ◀

Counting Vertices in Full m -Ary Trees

Theorem 3: A full m -ary tree with i internal vertices has $n = m i + 1$ vertices.

Proof: Every vertex, except the root, is the child of an internal vertex. Because each of the i internal vertices has m children, there are $m i$ vertices in the tree other than the root. Hence, the tree contains $n = mi + 1$ vertices. ◀

Counting Vertices in Full m -Ary Trees (*continued*)

Theorem 4: A full m -ary tree with

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

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

(iii) l leaves has $n = (ml - 1)/(m - 1)$ vertices and
 $i = (l - 1)/(m - 1)$ internal vertices.

proofs of parts (ii) and (iii) are left as exercises

Proof (of part i): Solving for i in $n = mi + 1$ (from Theorem 3) gives $i = (n - 1)/m$. Since each vertex is either a leaf or an internal vertex, $n = l + i$. By solving for l and using the formula for i , we see that

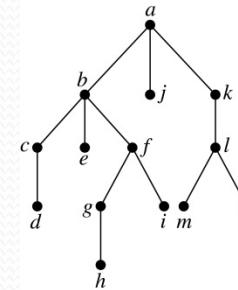
$$l = n - i = n - (n - 1)/m = [(m - 1)n + 1]/m.$$

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.

Example:

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



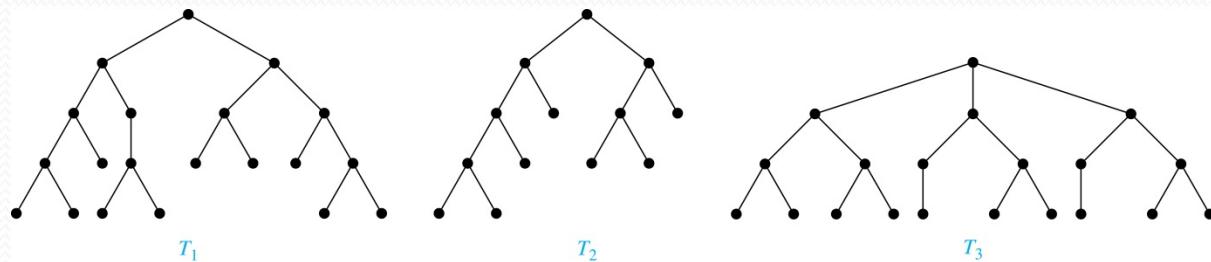
Solution:

- (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.

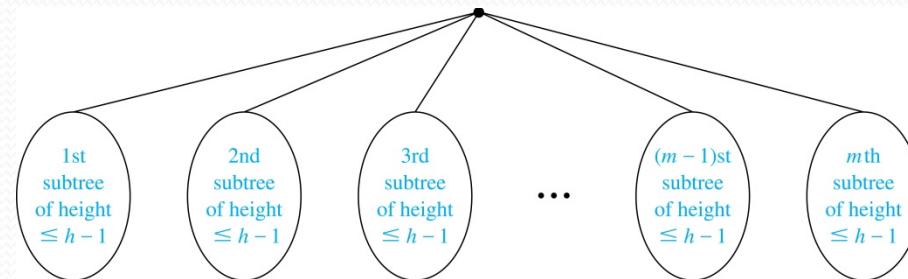
The Bound for the Number of Leaves in an m -Ary Tree

Theorem 5: There are at most m^h leaves in an m -ary tree of height h .

Proof (by mathematical induction on height):

BASIS STEP: Consider an m -ary trees of height 1. The tree consists of a root and no more than m children, all leaves. Hence, there are no more than $m^1 = m$ leaves in an m -ary tree of height 1.

INDUCTIVE STEP: Assume the result is true for all m -ary trees of height $< h$. Let T be an m -ary tree of height h . The leaves of T are the leaves of the subtrees of T we get when we delete the edges from the root to each of the vertices of level 1.



Each of these subtrees has height $\leq h - 1$. By the inductive hypothesis, each of these subtrees has at most m^{h-1} leaves. Since there are at most m such subtrees, there are at most $m \cdot m^{h-1} = m^h$ leaves in the tree. ◀

Corollary 1: If an m -ary tree of height h has l leaves, then $h \geq \lceil \log_m l \rceil$. If the m -ary tree is full and balanced, then $h = \lceil \log_m l \rceil$. (see text for the proof)

Huffman coding

- A *prefix code* is used to encode letters so that the bit string for a letter never occurs as the first part of the bit string for another letter.
- This ensures that no bit string corresponds to more than one sequence of letters.
 - For instance, the encoding of *e* as 0, *a* as 10, and *t* as 11 is a prefix code.
 - A prefix code can be represented using a binary tree, where the characters are the labels of the leaves in the tree.

Huffman coding

- Here is an example
- Consider the word encoded by 11111011100.
- The original word is *sane*.

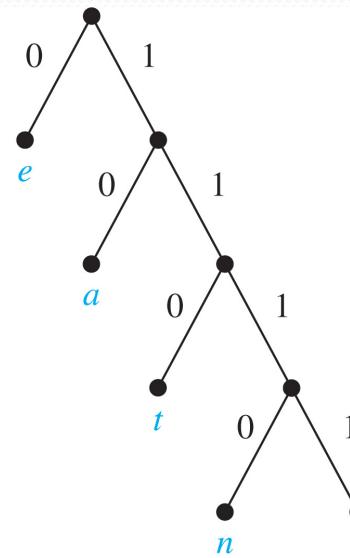


FIGURE 5 A Binary Tree with a Prefix Code.

Huffman coding



DAVID A. HUFFMAN (1925–1999)

- Huffman coding is a fundamental algorithm in *data compression*.
- It takes as input the *frequencies* of the symbols in a string,
- and produces as output a prefix code that encodes the string
 - using the fewest possible bits, among all possible binary prefix codes for these symbols.
- The idea is to obtain the smallest binary code for the most frequent symbols.

Huffman coding

- The algorithm begins with a forest of trees each consisting of one vertex,
 - where each vertex has a symbol as its label
 - and where the weight of this vertex equals the frequency of the symbol that is its label.
- At each step, we combine two trees having the least total weight into a single tree by
 - introducing a new root
 - and placing the tree with larger weight as its left subtree and the tree with smaller weight as its right subtree.
- Furthermore, we assign the sum of the weights of the two subtrees of this tree as the total weight of the tree.

Huffman coding

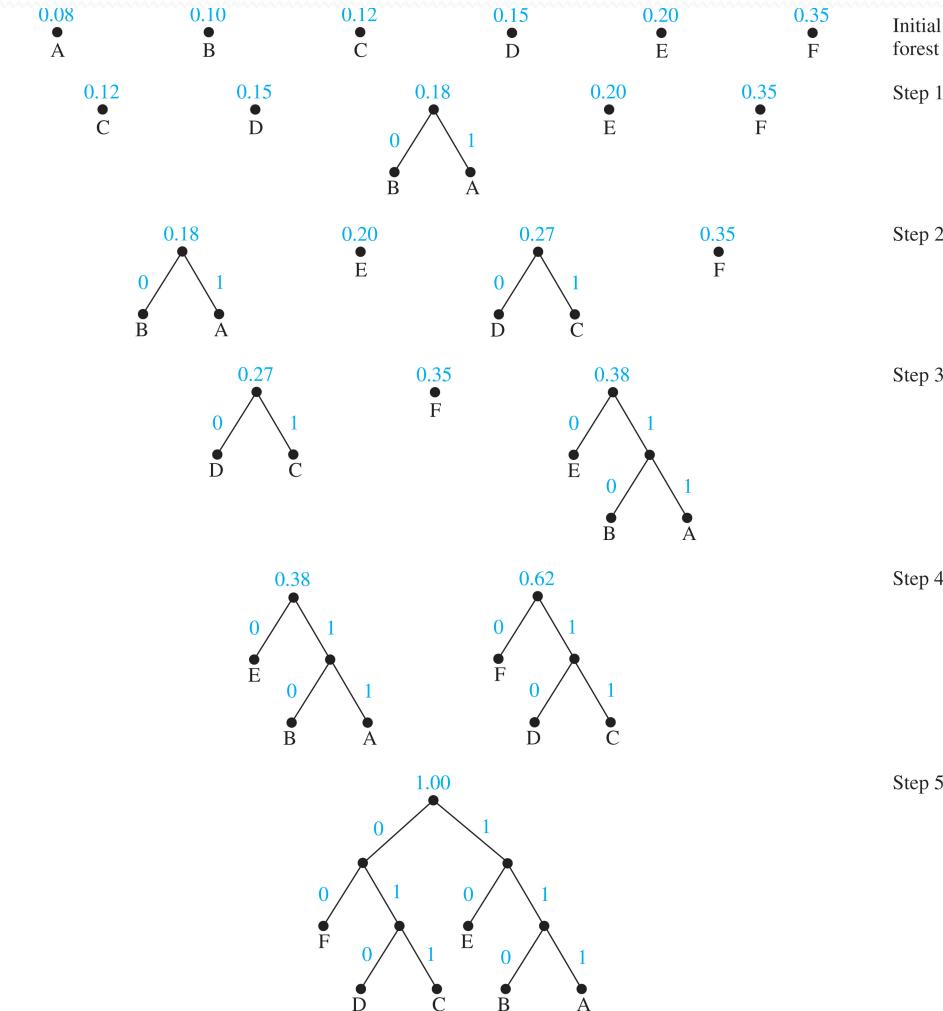


FIGURE 6 Huffman Coding of Symbols in Example 4.

Huffman coding

- **procedure** $Huffman(\text{symbols } a_i \text{ with frequencies } w_i, i = 1, \dots, n)$
- $F :=$ forest of n rooted trees, each consisting of the single vertex a_i and assigned weight w_i
- **while** F is not a tree
 - Replace the rooted trees T and T' of least weights from F with $w(T) \geq w(T')$ with a tree having a new root that has T as its left subtree and T' as its right subtree.
 - Label the new edge to T with 0 and the new edge to T' with 1.
Assign $w(T) + w(T')$ as the weight of the new tree.

Tree Traversal

Section 11.3

Section Summary

- Universal Address Systems
- Traversal Algorithms
- Infix, Prefix, and Postfix Notation

The universal address system

- This technique allows to order the vertices of an ordered rooted tree.
 1. Label the root with the integer 0. Then label its k children (at level 1) from left to right with $1, 2, 3, \dots, k$.
 2. For each vertex v at level n with label A , label its k_v children, as they are drawn from left to right, with $A.1, A.2, \dots, A.k_v$.

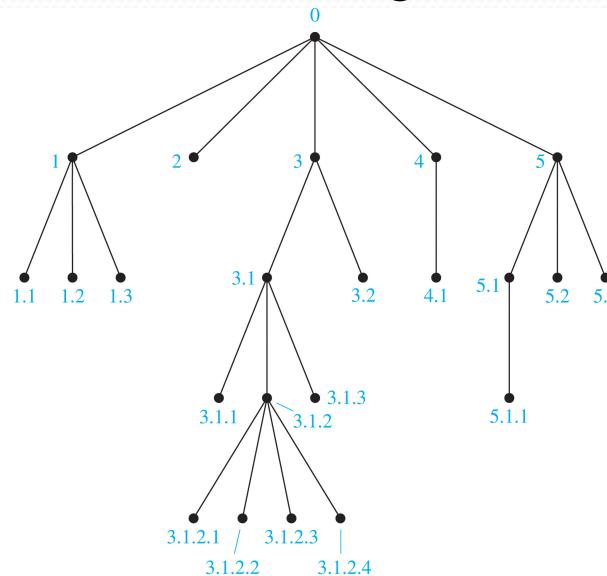


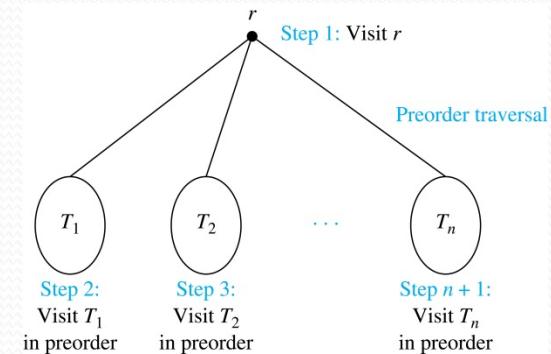
FIGURE 1 The Universal Address System of an Ordered Rooted Tree.

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)
```

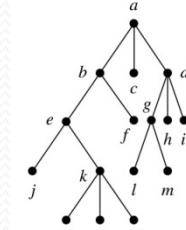
$r :=$ root of T

list r

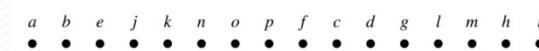
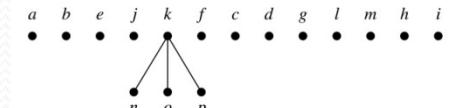
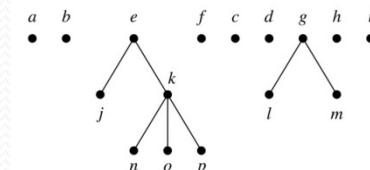
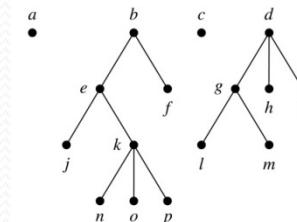
for each child c of r from left to right

$T(c) :=$ subtree with c as root

preorder($T(c)$)

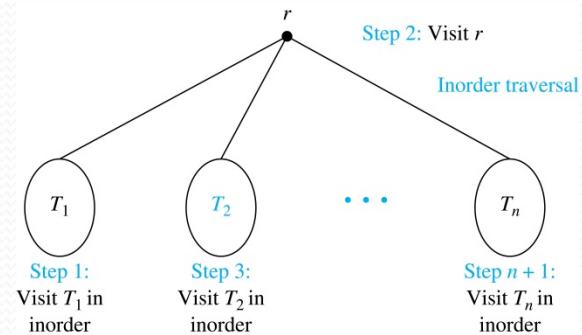


Preorder traversal: Visit root, visit subtrees left to right



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 (*continued*)

procedure *inorder* (T : ordered rooted tree)

$r :=$ root of T

if r is a leaf **then** list r

else

$l :=$ first child of r from left to right

$T(l) :=$ subtree with l as its root

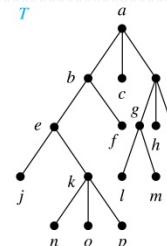
inorder($T(l)$)

 list(r)

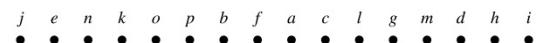
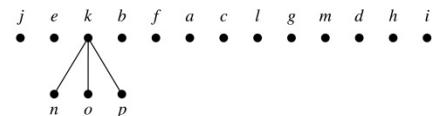
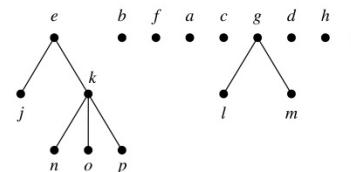
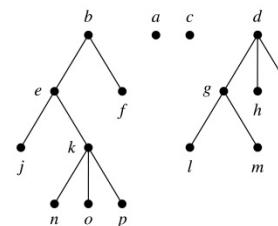
for each child c of r from left to right

$T(c) :=$ subtree with c as root

inorder($T(c)$)

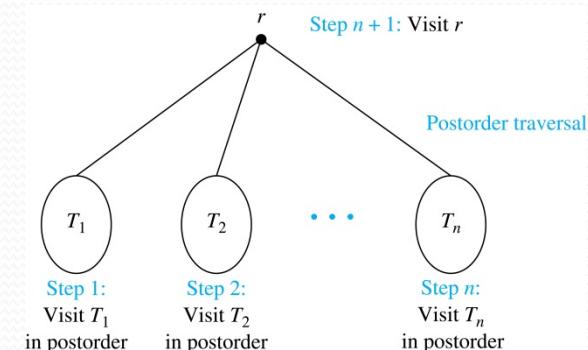


Inorder traversal: Visit leftmost subtree, visit root, visit other subtrees left to right



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 (*continued*)

procedure *postordered* (T : ordered rooted tree)

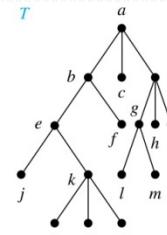
$r :=$ root of T

for each child c of r from left to right

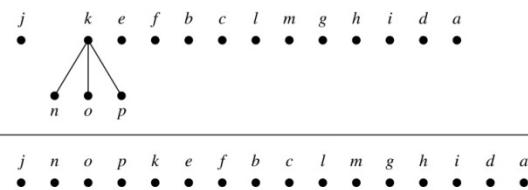
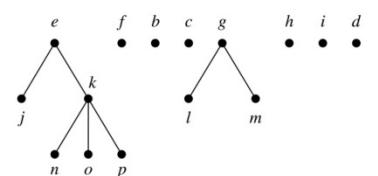
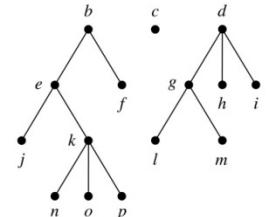
$T(c) :=$ subtree with c as root

postorder($T(c)$)

list r

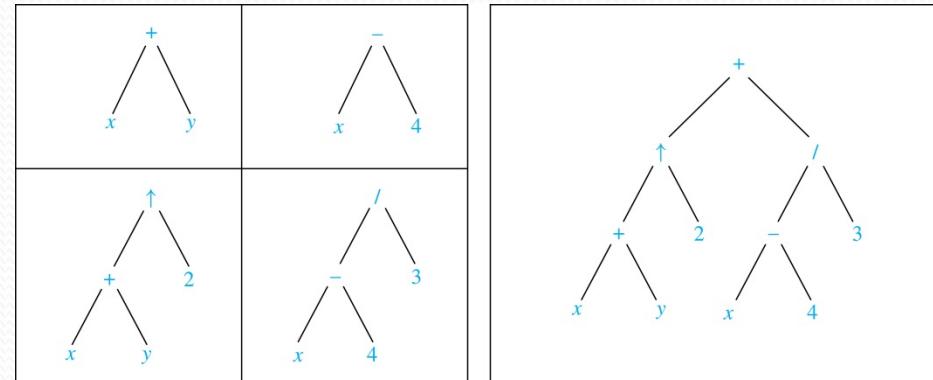


Postorder traversal: Visit subtrees left to right; visit root



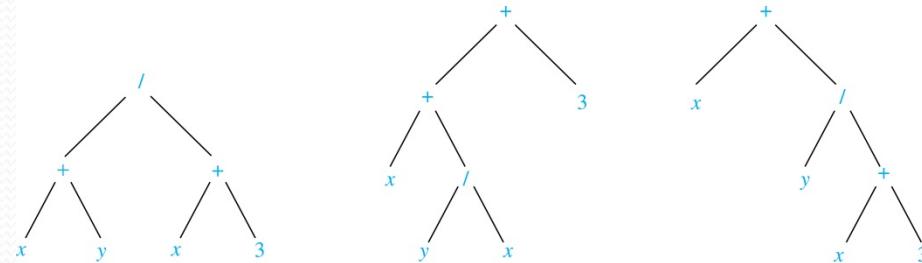
Expression Trees

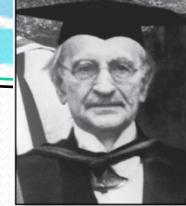
- Complex expressions can be represented using ordered rooted trees.
- Consider the expression $((x + y) \uparrow 2) + ((x - 4)/3)$.
- A binary tree for the expression can be built from the bottom up, as is illustrated here.



Infix Notation

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





Jan Łukasiewicz (1878-1956)

Prefix Notation

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

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

$$\begin{array}{ccccccccc}
 + & - & * & 2 & 3 & 5 & / & \uparrow & 2 & 3 \\
 & & & & & & & \underline{\hspace{2cm}} & \\
 & & & & & & & 2 \uparrow 3 = 8 \\
 \\
 + & - & * & 2 & 3 & 5 & / & 8 & 4 \\
 & & & & & & & \underline{\hspace{2cm}} & \\
 & & & & & & & 8 / 4 = 2 \\
 \\
 + & - & * & 2 & 3 & 5 & 2 \\
 & & & \underline{\hspace{2cm}} & & & \\
 & & & 2 * 3 = 6 \\
 \\
 + & - & 6 & 5 & 2 \\
 & & \underline{\hspace{2cm}} & & \\
 & & 6 - 5 = 1 \\
 \\
 + & 1 & 2 \\
 & \underline{\hspace{2cm}} & \\
 & 1 + 2 = 3
 \end{array}$$

Value of expression: 3

Postfix Notation

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

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

The diagram illustrates the evaluation of the postfix expression $7\ 2\ 3\ *\ -\ 4\ \uparrow\ 9\ 3\ /\ +$ through five stages:

- Stage 1:** $7\ 2\ 3\ *\ -\ 4\ \uparrow\ 9\ 3\ /\ +$
Annotation: $2 * 3 = 6$
- Stage 2:** $7\ 6\ -\ 4\ \uparrow\ 9\ 3\ /\ +$
Annotation: $7 - 6 = 1$
- Stage 3:** $1\ 4\ \uparrow\ 9\ 3\ /\ +$
Annotation: $1^4 = 1$
- Stage 4:** $1\ 9\ 3\ /\ +$
Annotation: $9 / 3 = 3$
- Stage 5:** $1\ 3\ +$
Annotation: $1 + 3 = 4$

Value of expression: 4

Spanning Trees

Section 11.4

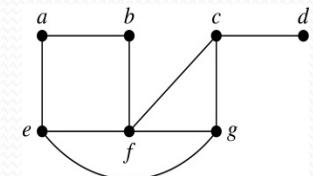
Section Summary

- Spanning Trees
- Depth-First Search
- Breadth-First Search
- Backtracking Applications
- Depth-First Search in Directed Graphs

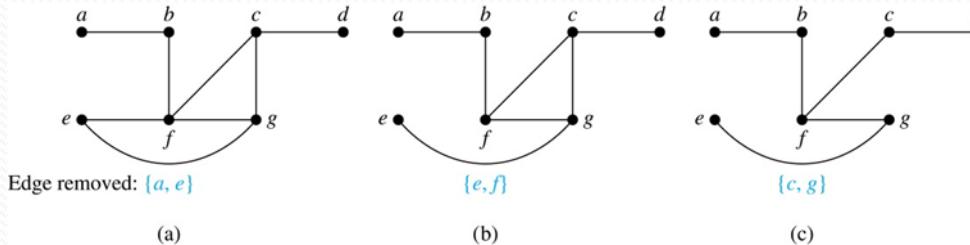
Spanning Trees

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

Example: Find the spanning tree of this graph:



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



Spanning Trees (*continued*)

Theorem: A simple graph is connected if and only if it has a spanning tree.

Proof: Suppose that a simple graph G has a spanning tree T . T contains every vertex of G and there is a path in T between any two of its vertices. Because T is a subgraph of G , there is a path in G between any two of its vertices. Hence, G is connected.

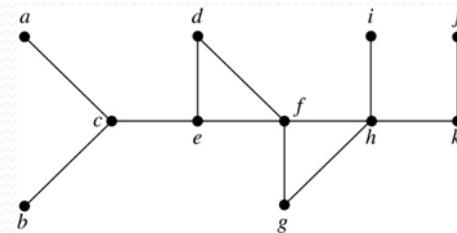
Now suppose that G is connected. If G is not a tree, it contains a simple circuit. Remove an edge from one of the simple circuits. The resulting subgraph is still connected because any vertices connected via a path containing the removed edge are still connected via a path with the remaining part of the simple circuit. Continue in this fashion until there are no more simple circuits. A tree is produced because the graph remains connected as edges are removed. The resulting tree is a spanning tree because it contains every vertex of G .

Depth-First Search

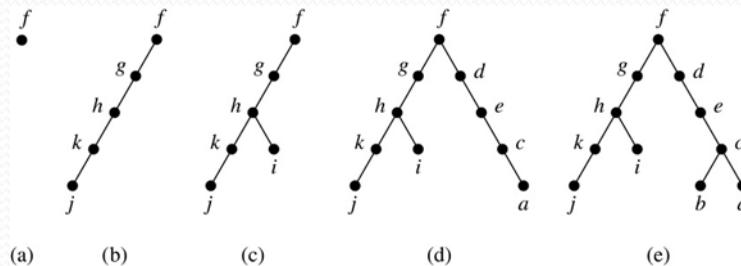
- To use *depth-first search* to build a spanning tree for a connected simple graph first arbitrarily choose a vertex of the graph as the root.
 - Form a path starting at this vertex by successively adding vertices and edges, where each new edge is incident with the last vertex in the path and a vertex not already in the path. Continue adding vertices and edges to this path as long as possible.
 - If the path goes through all vertices of the graph, the tree consisting of this path is a spanning tree.
 - Otherwise, move back to the next to the last vertex in the path, and if possible, form a new path starting at this vertex and passing through vertices not already visited. If this cannot be done, move back another vertex in the path.
 - Repeat this procedure until all vertices are included in the spanning tree.

Depth-First Search (*continued*)

Example: Use depth-first search to find a spanning tree of this graph.

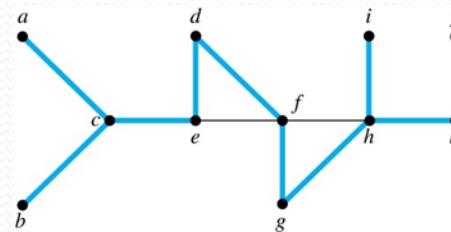


Solution: We start arbitrarily with vertex f . We build a path by successively adding an edge that connects the last vertex added to the path and a vertex not already in the path, as long as this is possible. The result is a path that connects f, g, h, k , and j . Next, we return to k , but find no new vertices to add. So, we return to h and add the path with one edge that connects h and i . We next return to f , and add the path connecting f, d, e, c , and a . Finally, we return to c and add the path connecting c and b . We now stop because all vertices have been added.



Depth-First Search (*continued*)

- The edges selected by depth-first search of a graph are called *tree edges*. All other edges of the graph must connect a vertex to an ancestor or descendant of the vertex in the graph. These are called *back edges*.
- In this figure, the tree edges are shown with heavy blue lines. The two thin black edges are back edges.



Depth-First Search Algorithm

- We now use pseudocode to specify depth-first search. In this recursive algorithm, after adding an edge connecting a vertex v to the vertex w , we finish exploring w before we return to v to continue exploring from v .

```
procedure DFS( $G$ : connected graph with vertices  $v_1, v_2, \dots, v_n$ )
```

```
     $T :=$  tree consisting only of the vertex  $v_1$ 
```

```
    visit( $v_1$ )
```

```
procedure visit( $v$ : vertex of  $G$ )
```

```
    for each vertex  $w$  adjacent to  $v$  and not yet in  $T$ 
```

```
        add vertex  $w$  and edge  $\{v,w\}$  to  $T$ 
```

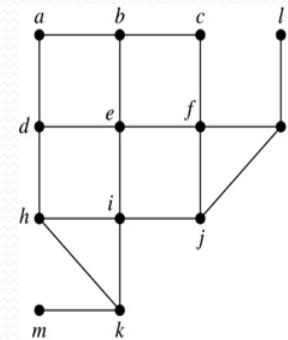
```
        visit( $w$ )
```

Breadth-First Search

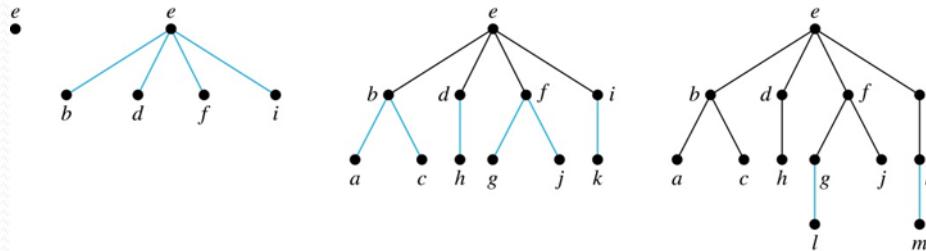
- We can construct a spanning tree using *breadth-first search*. We first arbitrarily choose a root from the vertices of the graph.
 - Then we add all of the edges incident to this vertex and the other endpoint of each of these edges. We say that these are the vertices at level 1.
 - For each vertex added at the previous level, we add each edge incident to this vertex, as long as it does not produce a simple circuit. The new vertices we find are the vertices at the next level.
 - We continue in this manner until all the vertices have been added and we have a spanning tree.

Breadth-First Search (*continued*)

Example: Use breadth-first search to find a spanning tree for this graph.



Solution: We arbitrarily choose vertex e as the root. We then add the edges from e to b, d, f , and i . These four vertices make up level 1 in the tree. Next, we add the edges from b to a and c , the edges from d to h , the edges from f to j and g , and the edge from i to k . The endpoints of these edges not at level 1 are at level 2. Next, add edges from these vertices to adjacent vertices not already in the graph. So, we add edges from g to l and from k to m . We see that level 3 is made up of the vertices l and m . This is the last level because there are no new vertices to find.



Breadth-First Search Algorithm

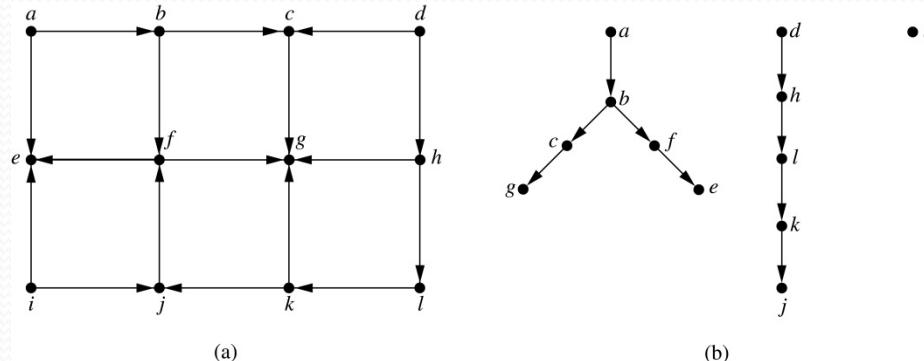
- We now use pseudocode to describe breadth-first search.

```
procedure BFS(G: connected graph with vertices  $v_1, v_2, \dots, v_n$ )  
  T := tree consisting only of the vertex  $v_1$   
  L := empty list visit( $v_1$ )  
  put  $v_1$  in the list L of unprocessed vertices  
  while L is not empty  
    remove the first vertex, v, from L  
    for each neighbor w of v  
      if w is not in L and not in T then  
        add w to the end of the list L  
        add w and edge  $\{v,w\}$  to T
```

Depth-First Search in Directed Graphs

- Both depth-first search and breadth-first search can be easily modified to run on a directed graph. But the result is not necessarily a spanning tree, but rather a spanning forest.

Example: For the graph in (a), if we begin at vertex a , depth-first search adds the path connecting a, b, c , and g . At g , we are blocked, so we return to c . Next, we add the path connecting f to e . Next, we return to a and find that we cannot add a new path. So, we begin another tree with d as its root. We find that this new tree consists of the path connecting the vertices d, h, l, k , and j . Finally, we add a new tree, which only contains i , its root.



- To index websites, search engines such as Google systematically explore the web starting at known sites. The programs that do this exploration are known as *Web spiders*. They may use both breath-first search or depth-first search to explore the Web graph.

Minimum spanning tree

- A *minimum spanning tree* in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

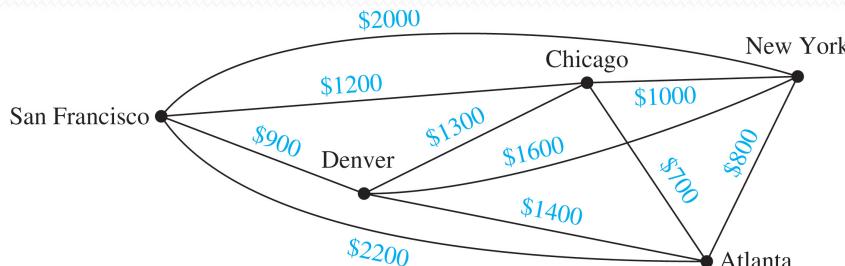


FIGURE 1 A Weighted Graph Showing Monthly Lease Costs for Lines in a Computer Network.

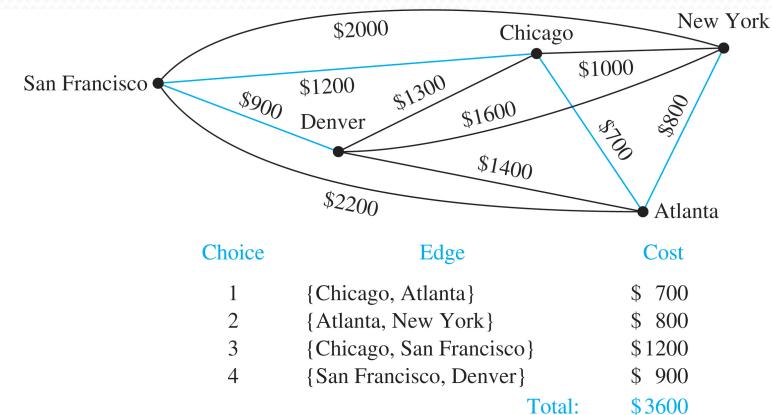


FIGURE 2 A Minimum Spanning Tree for the Weighted Graph in Figure 1.

Minimum spanning tree

- Prim's algorithm.
 - Begin by choosing any edge with smallest weight, putting it into the spanning tree.
 - Then successively add to the tree edges of minimum weight that are incident to a vertex already in the tree, never forming a simple circuit with those edges already in the tree.
 - Stop when $n - 1$ edges have been added.

Minimum spanning tree

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

Minimum spanning tree

- A simple example.

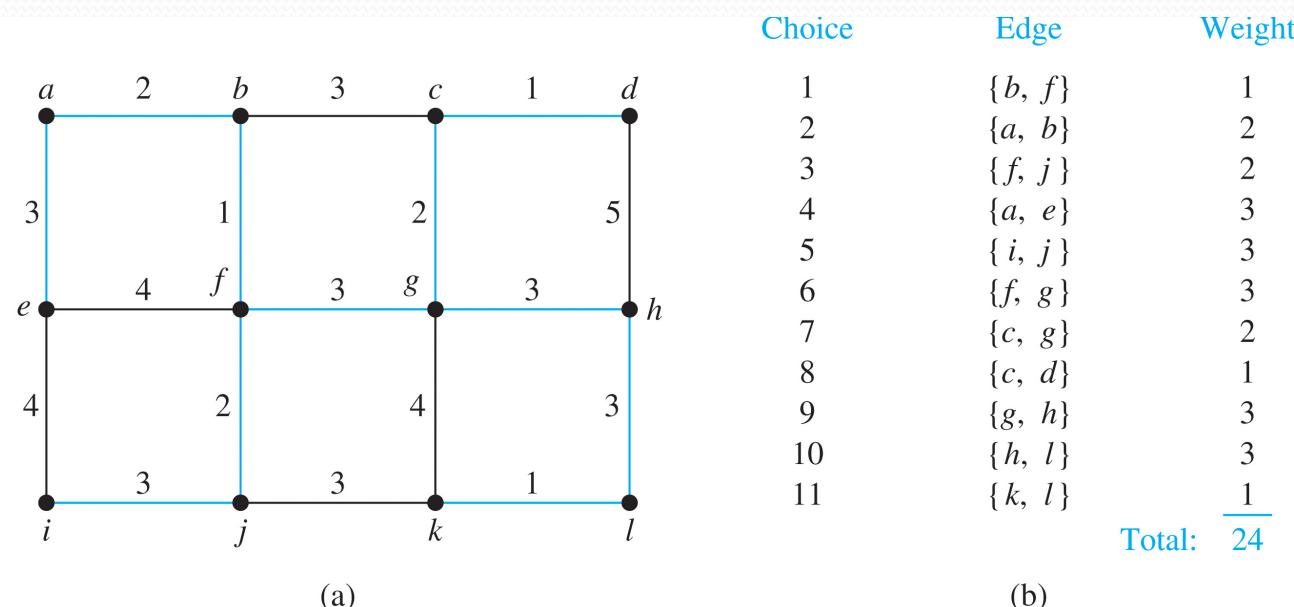


FIGURE 4 A Minimum Spanning Tree Produced Using Prim's Algorithm.

Minimum spanning tree

- We prove by induction that after k edges are added to T , T forms a spanning tree of H .
 - As a base case, after 0 edges are added, T is empty and H is the single node $\{v\}$. It is the minimum spanning tree of H_0 .
 - If at iteration k , $H_k = G$, the algorithm terminates, and since T is a spanning tree of H_k , T is a spanning tree of G .
 - If at iteration k an edge with minimal weight is added without forming a cycle, the result is itself a tree.
 - Since, by the induction hypothesis, it was a minimum spanning tree of H_{k-1} at iteration $k-1$, it remains a minimum spanning tree at iteration k (the weight is the lowest possible).

Minimum spanning tree

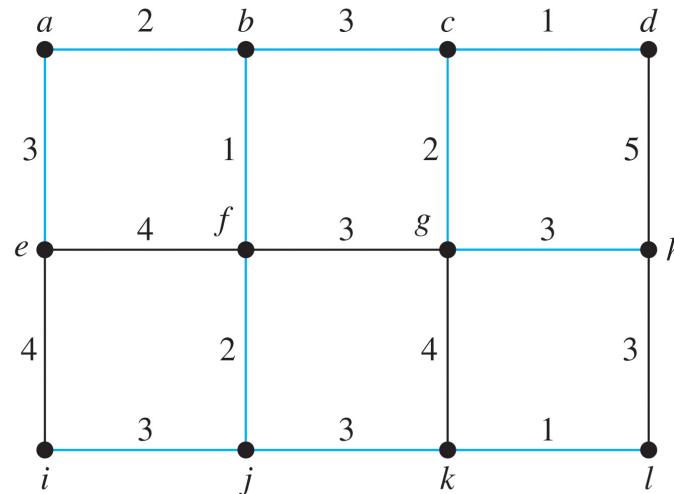
- Kruskal's algorithm.
 - To start, choose an edge in the graph with minimum weight.
 - Successively add edges with minimum weight that do not form a simple circuit with those edges already chosen.
 - Stop when $n - 1$ edges have been selected.

Minimum spanning tree

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

Minimum spanning tree

- A simple example.



Choice	Edge	Weight
1	{c, d}	1
2	{k, l}	1
3	{b, f}	1
4	{c, g}	2
5	{a, b}	2
6	{f, j}	2
7	{b, c}	3
8	{j, k}	3
9	{g, h}	3
10	{i, j}	3
11	{a, e}	3
Total:		24

(a)

(b)

FIGURE 5 A Minimum Spanning Tree Produced by Kruskal's Algorithm.