Florida State University
Course Notes
MAD 3105 Discrete Mathematics II

Florida State University

Tallahassee, Florida 32306-4510

Contents

CHAPTER 1

Relations

1. Relations and Their Properties

1.1. Definition of a Relation.

DEFINITION 1.1.1. *A* **binary relation from a set** A **to a set** B *is a subset*

$$R \subseteq A \times B.$$

If $(a, b) \in R$ *we say* a *is* **Related** *to* b *by* R.

A *is the* **domain** *of* R, *and*

B *is the* **codomain** *of* R.

If $A = B$, R *is called a* **binary relation on the set** A.

Notation.

- If $(a, b) \in R$, then we write aRb.
- If $(a, b) \notin R$, then we write $a \not R b$.

Discussion

We recall the basic definitions and terminology associated with the concept of a relation from a set A to a set B. You should review the notes from MAD 2104 whenever you wish to see more examples or more discussion on any of the topics we review here.

A relation is a generalization of the concept of a function, so that much of the terminology will be the same. Specifically, if $f\colon A \to B$ is a function from a set A to a B, then the graph of f, $graph(f) = \{(x, f(x)) | x \in A\}$ is a relation from A to B. Recall, however, that a relation may differ from a function in two essential ways. If R is an arbitrary relation from A to B, then

- it is possible to have both $(a, b) \in R$ and $(a, b') \in R$, where $b' \neq b$; that is, an element a in A may be related to any number of elements of B; or

- it is possible to have some element a in A that is not related to any element in B at all.

EXAMPLE 1.1.1. *Suppose $R \subset \mathbb{Z} \times \mathbb{Z}^+$ is the relation defined by $(a, b) \in R$ if and only if $a|b$. (Recall that $\mathbb{Z}^+$ denotes the set of positive integers, and $a|b$, read "a divides b", means that $b = na$ for some integer n.) Then R fails to be a function on both counts listed above. Certainly, $-2|2$ and $-2|4$, so that -2 is related to more than one positive integer. (In fact, it is related to infinitely many integers.) On the other hand $0 \not| b$ for any positive integer b.*

REMARK 1.1.1. *It is often desirable in computer science to relax the requirement that a function be defined at* every *element in the domain. This has the beneficial effect of reducing the number of sets that must be named and stored. In order not to cause too much confusion with the usual mathematical definition of a function, a relation such as this is called a* **partial function**. *That is, a partial function $f\colon A \to B$ is a relation such that, for all $a \in A$, $f(a)$ is a uniquely determined element of B whenever it is defined. For example, the formula $f(x) = \frac{1}{1-x^2}$ defines a partial function $f\colon \mathbb{R} \to \mathbb{R}$ with domain $\mathbb{R}$ and codomain $\mathbb{R}$: f is not defined at $x = -1$ and $x = 1$, but, otherwise, $f(x)$ is uniquely determined by the formula.*

EXERCISE 1.1.1. *Let n be a positive integer. How many binary relations are there on a set A if $|A| = n$? [Hint: How many elements are there in $|A \times A|$?]*

1.2. Directed Graphs.

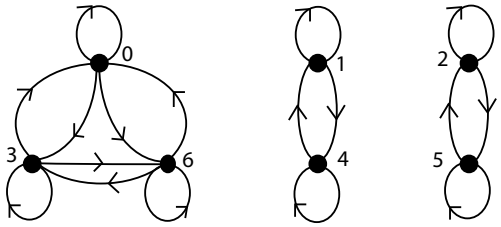
DEFINITIONS 1.2.1.

- *A* **directed graph** *or a* **digraph** *D from A to B is a collection of* **vertices** *$V \subseteq A \cup B$ and a collection of* **edges** *$R \subseteq A \times B$.*
- *If there is an ordered pair $e = (x, y)$ in R then there is an* **arc** *or* **edge** *from x to y in D.*
- *The elements x and y are called the* **initial** *and* **terminal** *vertices of the edge $e = (x, y)$, respectively.*

Discussion

A digraph can be a useful device for representing a relation, especially if the relation isn't "too large" or complicated. If R is a relation on a set A, we simplify the digraph G representing R by having only one vertex for each $a \in A$. This results, however, in the possibility of having **loops**, that is, edges from a vertex to itself, and having more than one edge joining distinct vertices (but with opposite orientations). A digraph for the relation R in Example 1.1.1 would be difficult to illustrate, and impossible to draw completely, since it would require infinitely many vertices and edges. We could draw a digraph for some finite subset of R.

EXAMPLE 1.2.1. *Suppose R is the relation on $\{0,1,2,3,4,5,6\}$ defined by mRn if and only if $m \equiv n (mod\ 3)$. The digraph that represents R is*



1.3. Representing Relations with Matrices.

DEFINITION 1.3.1. *Let R be a relation from $A = \{a_1, a_2, \ldots, a_m\}$ to $B = \{b_1, b_2, \ldots, b_n\}$. An $m \times n$* **connection matrix** *$M_R = \{m_{ij}\}$ for R is defined by*

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

1.4. Example 1.4.1.

EXAMPLE 1.4.1. *Let $A = \{a, b, c\}$, $B = \{e, f, g, h\}$, and $R = \{(a, e), (c, g)\}$.*

then the connection matrix

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

Discussion

Recall the connection matrix for a finite relation is a method for representing a relation using a matrix.

REMARK 1.4.1. *The* **ordering** *of the elements in A and B is important. If the elements are rearranged the matrix would be different! If there is a natural order to the elements of the sets (like numerical or alphabetical) you would expect to use this order when creating connection matrices.*

To find this matrix we may use a table as follows. First we set up a table labeling the rows and columns with the vertices.

	e	f	g	h
a				
b				
c				

Since $(a, e) \in R$ we put a 1 in the row a and column e and since $(c, g) \in R$ we put a 1 in row c and column g.

	e	f	g	h
a	1			
b				
c			1	

Fill in the rest of the entries with 0's. The matrix may then be read straight from the table.

1.5. Inverse Relation.

DEFINITION 1.5.1. *Let R be a relation from A to B. Then $R^{-1} = \{(b, a)|(a, b) \in R\}$ is a relation from B to A.*

R^{-1} is called the **inverse of the relation** *R.*

Discussion

The inverse of a relation R is the relation obtained by simply reversing the ordered pairs of R. The inverse of a relation is also called the **converse** of a relation.

EXAMPLE 1.5.1. *Let $A = \{a, b, c\}$ and $B = \{1, 2, 3, 4\}$ and let $R = \{(a, 1), (a, 2), (c, 4)\}$. Then $R^{-1} = \{(1, a), (2, a), (4, c)\}$.*

EXERCISE 1.5.1. *Suppose A and B are sets and $f\colon A \to B$ is a function. The graph of f, $graph(f) = \{(x, f(x))|x \in A\}$ is a relation from A to B.*

(a) What is the inverse of this relation?
(b) Does f have to be invertible (as a function) for the inverse of this relation to exist?
(c) If f is invertible, find the inverse of the relation $graph(f)$ in terms of the inverse function f^{-1}.

1.6. Special Properties of Binary Relations. Definitions. Let A be a set, and let R be a binary relation on A.

(1) R is **reflexive** if
$\forall x[(x \in A) \to ((x, x) \in R)]$.
(2) R is **irreflexive** if
$\forall x[(x \in A) \to ((x, x) \notin R)]$.
(3) R is **symmetric** if
$\forall x \forall y[((x, y) \in R) \to ((y, x) \in R)]$.

(4) R is **antisymmetric** if
$\forall x \forall y[((x, y) \in R \wedge (y, x) \in R) \rightarrow (x = y)]$.
(5) R is **asymmetric** if
$\forall x \forall y[((x, y) \in R) \rightarrow ((y, x) \notin R)]$.
(6) R is **transitive** if
$\forall x \forall y \forall z[((x, y) \in R \wedge (y, z) \in R) \rightarrow ((x, z) \in R)]$.

Discussion

The definition above recalls six special properties that a relation may (or may not) satisfy. Notice that the definitions of reflexive and irreflexive relations are not complementary. That is, a relation on a set may be both reflexive and irreflexive or it may be neither. The same is true for the symmetric and antisymmetric properties, as well as the symmetric and asymmetric properties.

The terms reflexive, symmetric, and transitive is generally consistent from author to author. The rest are not as consistent in the literature

EXERCISE 1.6.1. *Before reading further, find a relation on the set* $\{a, b, c\}$ *that is neither*

(a) reflexive nor irreflexive.
(b) symmetric nor antisymmetric.
(c) symmetric nor asymmetric.

1.7. Examples of Relations and their Properties.

EXAMPLE 1.7.1. *Suppose A is the set of all residents of Florida and R is the relation given by aRb if a and b have the same last four digits of their Social Security Number. This relation is...*

- *reflexive*
- *not irreflexive*
- *symmetric*
- *not antisymmetric*
- *not asymmetric*
- *transitive*

EXAMPLE 1.7.2. *Suppose T is the relation on the set of integers given by xTy if $2x - y = 1$. This relation is...*

- *not reflexive*
- *not irreflexive*

- *not symmetric*
- *antisymmetric*
- *not asymmetric*
- *not transitive*

EXAMPLE 1.7.3. *Suppose* $A = \{a, b, c, d\}$ *and* R *is the relation* $\{(a, a)\}$. *This relation is...*

- *not reflexive*
- *not irreflexive*
- *symmetric*
- *antisymmetric*
- *not asymmetric*
- *transitive*

Discussion

When proving a relation, R, on a set A *has* a particular property, the property must be shown to hold for all appropriate combinations of members of the set. When proving a relation R does *not* have a property, however, it is enough to give a counterexample.

EXAMPLE 1.7.4. *Suppose* T *is the relation in Example 1.7.2 in Section 1.7. This relation is*

- *not reflexive*

  PROOF. 2 is an integer and $2 \cdot 2 - 2 = 2 \neq 1$. This shows that $\forall x[x \in \mathbb{Z} \rightarrow (x, x) \in T]$ is **not true**. □

- *not irreflexive*

  PROOF. 1 is an integer and $2 \cdot 1 - 1 = 1$. This shows that $\forall x[x \in \mathbb{Z} \rightarrow (x, x) \notin T]$ is **not true**. □

- *not symmetric*

  PROOF. Both 2 and 3 are integers, $2 \cdot 2 - 3 = 1$, and $2 \cdot 3 - 2 = 4 \neq 1$. This shows $2R3$, but $3 \not R 2$; that is, $\forall x \forall y[(x, y) \in \mathbf{Z} \rightarrow (y, x) \in T]$ is **not true**. □

- *antisymmetric*

  PROOF. Let $m, n \in \mathbb{Z}$ be such that $(m, n) \in T$ and $(n, m) \in T$. By the definition of T, this implies both equations $2m - n = 1$ and $2n - m = 1$ must hold. We may use the first equation to solve for n, $n = 2m - 1$, and substitute this in for n in the second equation to get $2(2m-1) - m = 1$. We

may use this equation to solve for m and we find $m = 1$. Now solve for n and we get $n = 1$.

This shows that the only integers, m and n, such that both equations $2m - n = 1$ and $2n - m = 1$ hold are $m = n = 1$. This shows that $\forall m \forall n[((m, n) \in T \wedge (n, m) \in T) \rightarrow m = n]$. □

- *not asymmetric*

  PROOF. 1 is an integer such that $(1, 1) \in T$. Thus $\forall x \forall y[((x, y) \in T \rightarrow (b, a) \notin T]$ is **not true** (counterexample is $a = b = 1$). □

- *not transitive*

  PROOF. 2, 3, and 5 are integers such that $(2, 3) \in T$, $(3, 5) \in T$, but $(2, 5) \notin T$. This shows $\forall x \forall y \forall z[(x, y) \in T \wedge (y, z) \in T \rightarrow (x, z) \in T]$ is **not true**. □

EXERCISE 1.7.1. *Verify the assertions made about the relation in Example 1.7.1 in Section 1.7.*

EXERCISE 1.7.2. *Verify the assertions made about the relation in Example 1.7.3 in Section 1.7.*

EXERCISE 1.7.3. *Suppose $R \subset \mathbb{Z}^+ \times \mathbb{Z}^+$ is the relation defined by $(m, n) \in R$ if and only if $m|n$. Prove that R is*

(a) reflexive.
(b) not irreflexive.
(c) not symmetric.
(d) antisymmetric.
(e) not asymmetric.
(f) transitive.

1.8. Theorem 1.8.1: Connection Matrices v.s. Properties.

THEOREM 1.8.1. *Let R be a binary relation on a set A and let M_R be its connection matrix. Then*

- *R is reflexive iff $m_{ii} = 1$ for all i.*
- *R is irreflexive iff $m_{ii} = 0$ for all i.*
- *R is symmetric iff M is a symmetric matrix.*
- *R is antisymmetric iff for each $i \neq j$, $m_{ij} = 0$ or $m_{ji} = 0$.*
- *R is asymmetric iff for every i and j, if $m_{ij} = 1$, then $m_{ji} = 0$.*

Discussion

Connection matrices may be used to test if a finite relation has certain properties and may be used to determine the composition of two finite relations.

EXAMPLE 1.8.1. *Determine which of the properties: reflexive, irreflexive, symmetric, antisymmetric, asymmetric, the relation on* $\{a, b, c, d\}$ *represented by the following matrix has.*

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

Solution. *The relation is irreflexive, asymmstric and antisymmetric only.*

EXERCISE 1.8.1. *Determine which of the properties: reflexive, irreflexive, symmetric, antisymmetric, asymmetric, the relation on* $\{a, b, c, d\}$ *represented by the following matrix has.*

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

1.9. Combining Relations. Suppose property P is one of the properties listed in Section 1.6, and suppose R and S are relations on a set A, each having property P. Then the following questions naturally arise.

(1) Does $\overline{R}$ (necessarily) have property P?
(2) Does $R \cup S$ have property P?
(3) Does $R \cap S$ have property P?
(4) Does $R - S$ have property P?

Discussion

Notice that when we combine two relations using one of the binary set operations we are combining *sets of ordered pairs.*

1.10. Example 1.10.1.

EXAMPLE 1.10.1. *Let* R *and* S *be transitive relations on a set* A*. Does it follow that* $R \cup S$ *is transitive?*

Solution. No. Here is a counterexample:

$$A = \{1, 2\}, \quad R = \{(1, 2)\}, \quad S = \{(2, 1)\}$$

$$\text{Therefore,} \qquad R \cup S = \{(1, 2), (2, 1)\}$$

Notice that R and S are both transitive (vacuously, since there are no two elements satisfying the hypothesis of the conditions of the property). However $R \cup S$ is not transitive. If it were it would have to have $(1, 1)$ and $(2, 2)$ in $R \cup S$.

Discussion

The solution to Example 1.10.1 gives a counterexample to show that the union of two transitive relations is not necessarily transitive. Note that you could find an example of two transitive relations whose union *is* transitive. The question, however, asks if the given property holds for two relations must it hold for the binary operation of the two relations. This is a general question and to give the answer "yes" we must know it is true for *every* possible pair of relations satisfying the property.

Here is another example:

EXAMPLE 1.10.2. *Suppose R and S are transitive relations on the set A. Is $R \cap S$ transitive?*

Solution. *Yes.*

PROOF. Assume R and S are both transitive and suppose $(a, b), (b, c) \in R \cap S$. Then $(a, b), (b, c) \in R$ and $(a, b), (b, c) \in S$. It is given that both R and S are transitive, so $(a, c) \in R$ and $(a, c) \in S$. Therefore $(a, c) \in R \cap S$. This shows that for arbitrary $(a, b), (b, c) \in R \cap S$ we have $(a, c) \in R \cap S$. Thus $R \cap S$ is transitive. □

As it turns out, the intersection of any two relations satisfying one of the properties in Section 1.6 also has that property. As the following exercise shows, sometimes even more can be proved.

EXERCISE 1.10.1. *Suppose R and S are relations on a set A.*

(a) Prove that if R and S are reflexive, then so is $R \cap S$.
(b) Prove that if R is irreflexive, then so is $R \cap S$.

EXERCISE 1.10.2. *Suppose R and S are relations on a set A.*

(a) Prove that if R and S are symmetric, then so is $R \cap S$.
(b) Prove that if R is antisymmetric, then so is $R \cap S$.
(c) Prove that if R is asymmetric, then so is $R \cap S$.

EXERCISE 1.10.3. *Suppose R and S are relations on a set A. Prove or disprove:*

(a) If R and S are reflexive, then so is $R \cup S$.
(b) If R and S are irreflexive, then so is $R \cup S$.

EXERCISE 1.10.4. *Suppose R and S are relations on a set A. Prove or disprove:*

(a) If R and S are symmetric, then so is $R \cup S$.
(b) If R and S are antisymmetric, then so is $R \cup S$.
(c) If R and S are asymmetric, then so is $R \cup S$.

1.11. Definition of Composition.

DEFINITION 1.11.1.
(1) Let
- *R be a relation from A to B, and*
- *S be a relation from B to C.*

Then the **composition** *of R with S, denoted $S \circ R$, is the relation from A to C defined by the following property:*
$(x, z) \in S \circ R$ if and only if there is a $y \in B$ such that $(x, y) \in R$ and $(y, z) \in S$.

(2) Let R be a binary relation on A. Then R^n is defined recursively as follows:
Basis: $R^1 = R$
Recurrence: $R^{n+1} = R^n \circ R$, if $n \geq 1$.

Discussion

The composition of two relations can be thought of as a generalization of the composition of two functions, as the following exercise shows.

EXERCISE 1.11.1. *Prove: If $f\colon A \to B$ and $g\colon B \to C$ are functions, then $graph(g \circ f) = graph(g) \circ graph(f)$.*

EXERCISE 1.11.2. *Prove the composition of relations is an associative operation.*

EXERCISE 1.11.3. *Suppose R is a relation on A. Using the previous exercise and mathematical induction, prove that $R^n \circ R = R \circ R^n$.*

EXERCISE 1.11.4. *Prove an ordered pair $(x, y) \in R^n$ if and only if, in the digraph D of R, there is a directed path of length n from x to y.*

Notice that if there is no element of B such that $(a, b) \in R_1$ and $(b, c) \in R_2$ for some $a \in A$ and $c \in C$, then the composition is empty.

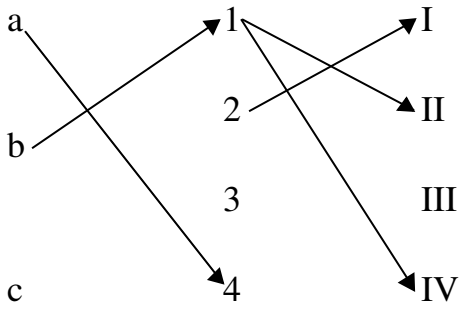
1.12. Example 1.12.1.

EXAMPLE 1.12.1. *Let* $A = \{a, b, c\}$, $B = \{1, 2, 3, 4\}$, *and* $C = \{I, II, III, IV\}$.

- *If* $R = \{(a, 4), (b, 1)\}$ *and*
- $S = \{(1, II), (1, IV), (2, I)\}$, *then*
- $S \circ R = \{(b, II), (b, IV)\}$.

Discussion

It can help to consider the following type of diagram when discussing composition of relations, such as the ones in Example 1.12.1 shown here.



EXAMPLE 1.12.2. *If* R *and* S *are transitive binary relations on* A, *is* $R \circ S$ *transitive?*

Solution. *No. Here is a counterexample: Let*

$$R = \{(1, 2), (3, 4)\}, \quad \text{and} \quad S = \{(2, 3), (4, 1)\}.$$

Both R *and* S *are transitive (vacuously), but*

$$R \circ S = \{(2, 4), (4, 2)\}$$

is not transitive. (Why?)

EXAMPLE 1.12.3. *Suppose* R *is the relation on* $\mathbb{Z}^+$ *defined by* aRb *if and only if* $a|b$. *Then* $R^2 = R$.

EXERCISE 1.12.1. *Suppose* R *is the relation on the set of real numbers given by* xRy *if and only if* $\frac{x}{y} = 2$.

(a) Describe the relation R^2.
(b) Describe the relation R^n, $n \geq 1$.

EXERCISE 1.12.2. *Suppose* R *and* S *are relations on a set* A *that are reflexive. Prove or disprove the relation obtained by combining* R *and* S *in one of the following ways is reflexive. (Recall:* $R \oplus S = (R \cup S) - (R \cap S)$.*)*

(a) $R \oplus S$
(b) $R - S$
(c) $R \circ S$
(d) R^{-1}
(e) R^n, $n \geq 2$

EXERCISE 1.12.3. *Suppose R and S are relations on a set A that are symmetric. Prove or disprove the relation obtained by combining R and S in one of the following ways is symmetric.*

(a) $R \oplus S$
(b) $R - S$
(c) $R \circ S$
(d) R^{-1}
(e) R^n, $n \geq 2$

EXERCISE 1.12.4. *Suppose R and S are relations on a set A that are transitive. Prove or disprove the relation obtained by combining R and S in one of the following ways is transitive.*

(a) $R \oplus S$
(b) $R - S$
(c) $R \circ S$
(d) R^{-1}
(e) R^n, $n \geq 2$

1.13. Theorem 1.13.1: Characterization of Transitive Relations.

THEOREM 1.13.1. *Let R be a binary relation on a set A. R is transitive if and only if $R^n \subseteq R$, for $n \geq 1$.*

PROOF. To prove (R transitive) $\rightarrow$ ($R^n \subseteq R$) we assume R is transitive and prove $R^n \subseteq R$ for $n \geq 1$ by induction.

Basis Step, $n = 1$. $R^1 = R$, so the statement is vacuously true when $n = 1$, since $R^1 = R \subseteq R$ whether or not R is transitive.

Induction Step. Prove $R^n \subseteq R \rightarrow R^{n+1} \subseteq R$.

Assume $R^n \subseteq R$ for some $n \geq 1$. Suppose $(x, y) \in R^{n+1}$. By definition, $R^{n+1} = R^n \circ R$, so there must be some $a \in A$ such that $(x, a) \in R$ and $(a, y) \in R^n$. By the induction hypothesis, $R^n \subseteq R$, so $(a, y) \in R$. Since R is transitive, $(x, a), (a, y) \in R$ implies $(x, y) \in R$. Since (x, y) was an arbitrary element of R^{n+1}, we have shown $R^{n+1} \subseteq R$.

We now prove the reverse implication: $R^n \subseteq R$, for $n \geq 1$, implies R is transitive. We prove this directly using only the hypothesis for $n = 2$.

Assume $(x, y), (y, z) \in R$. The definition of composition implies $(x, z) \in R^2$. But $R^2 \subseteq R$, so $(x, z) \in R$. Thus R is transitive.

□

Discussion

Theorem 1.13.1 gives an important characterization of the transitivity property. Notice that, since the statement of the theorem was a property that was to be proven for all positive integers, induction was a natural choice for the method of proof.

EXERCISE 1.13.1. *Prove that a relation R on a set A is transitive if and only if $R^2 \subseteq R$. [Hint: Examine not only the statement, but the proof of the Theorem 1.13.1.]*

1.14. Connection Matrices v.s. Composition. Recall: The **boolean product** of an $m \times k$ matrix $A = [a_{ij}]$ and a $k \times n$ matrix $B = [b_{ij}]$, denoted $A \odot B = [c_{ij}]$, is defined by

$$c_{ij} = (a_{i1} \wedge b_{1j}) \vee (a_{i2} \wedge b_{2j}) \vee \cdots \vee (a_{ik} \wedge b_{kj}).$$

THEOREM 1.14.1. *Let X, Y, and Z be finite sets. Let R_1 be a relation from X to Y and R_2 be a relation from Y to Z. If M_1 is the connection matrix for R_1 and M_2 is the connection matrix for R_2, then $M_1 \odot M_2$ is the connection matrix for $R_2 \circ R_1$.*

We write $M_{R_2 \circ R_1} = M_{R_1} \odot M_{R_2}$.

COROLLARY 1.14.1.1. *$M_{R^n} = (M_R)^n$ (the boolean n^{th} power of M_R).*

EXERCISE 1.14.1. *Let the relation R on $\{a, b, c, d\}$ be represented by the matrix*

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

Find the matrix that represents R^3.

2. Closure of Relations

2.1. Definition of the Closure of Relations.

DEFINITION 2.1.1. *Given a relation R on a set A and a property P of relations, the* **closure** *of R with respect to property P, denoted $Cl_P(R)$, is smallest relation on A that contains R and has property P. That is, $Cl_P(R)$ is the relation obtained by adding the minimum number of ordered pairs to R necessary to obtain property P.*

Discussion

To say that $Cl_P(R)$ is the "smallest" relation on A containing R and having property P means that

- $R \subseteq Cl_P(R)$,
- $Cl_P(R)$ has property P, and
- if S is another relation with property P and $R \subseteq S$, then $Cl_P(R) \subseteq S$.

The following theorem gives an equivalent way to define the closure of a relation.

THEOREM 2.1.1. *If R is a relation on a set A, then $Cl_P(R) = \bigcap_{S \in \mathcal{P}} S$, where $\mathcal{P} = \{S | R \subseteq S$ and S has property $P\}$.*

EXERCISE 2.1.1. *Prove Theorem 2.1.1. [Recall that one way to show two sets, A and B, are equal is to show $A \subseteq B$ and $B \subseteq A$.]*

2.2. Reflexive Closure.

DEFINITION 2.2.1. *Let A be a set and let $\Delta = \{(x, x) | x \in A\}$. Δ is called the* **diagonal relation** *on A.*

THEOREM 2.2.1. *Let R be a relation on A. The* **reflexive closure** *of R, denoted $r(R)$, is the relation $R \cup \Delta$.*

PROOF. Clearly, $R \cup \Delta$ is reflexive, since $(a, a) \in \Delta \subseteq R \cup \Delta$ for every $a \in A$.

On the other hand, if S is a reflexive relation containing R, then $(a, a) \in S$ for every $a \in A$. Thus, $\Delta \subseteq S$ and so $R \cup \Delta \subseteq S$.

Thus, by definition, $R \cup \Delta \subseteq S$ is the reflexive closure of R.

□

Discussion

Theorem 2.2.1 in Section 2.2 gives a simple method for finding the reflexive closure of a relation R.

REMARKS 2.2.1.
(1) Sometimes Δ is called the **identity** *or* **equality** *relation on A.*
(2) Δ is the smallest reflexive relation on A.
(3) Given the digraph of a relation, to find the digraph of its reflexive closure, add a loop at each vertex (for which no loop already exists).
(4) Given the connection matrix M of a finite relation, the matrix of its reflexive closure is obtained by changing all zeroes to ones on the main diagonal of M. That is, form the Boolean sum $M \vee I$, where I is the identity matrix of the appropriate dimension.

2.3. Symmetric Closure.

THEOREM 2.3.1. *The* **symmetric closure** *of R, denoted $s(R)$, is the relation $R \cup R^{-1}$, where R^{-1} is the inverse of the relation R.*

Discussion

REMARKS 2.3.1.
(1) To get the digraph of the inverse of a relation R from the digraph of R, reverse the direction of each of the arcs in the digraph of R.
(2) To get the digraph of the symmetric closure of a relation R, add a new arc (if none already exists) for each (directed) arc in the digraph for R, but with the reverse direction.
(3) To get the connection matrix of the inverse of a relation R from the connection matrix M of R, take the transpose, M^t.
(4) To get the connection matrix of the symmetric closure of a relation R from the connection matrix M of R, take the Boolean sum $M \vee M^t$.
(5) The composition of a relation and its inverse is not necessarily equal to the identity. A bijective function composed with its inverse, however, is equal to the identity.

2.4. Examples.

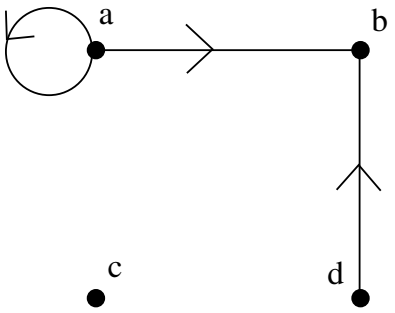
EXAMPLE 2.4.1. *If $A = \mathbb{Z}$, and R is the relation $(x, y) \in R$ iff $x \neq y$, then*

- $r(R) = \mathbb{Z} \times \mathbb{Z}$.
- $s(R) = R$.

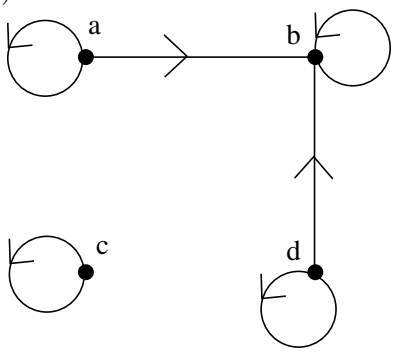
EXAMPLE 2.4.2. *If $A = \mathbb{Z}^+$, and R is the relation $(x, y) \in R$ iff $x < y$, then*

- *$r(R)$ is the relation $(x, y) \in r(R)$ iff $x \leq y$.*
- *$s(R)$ is the relation $(x, y) \in s(R)$ iff $x \neq y$.*

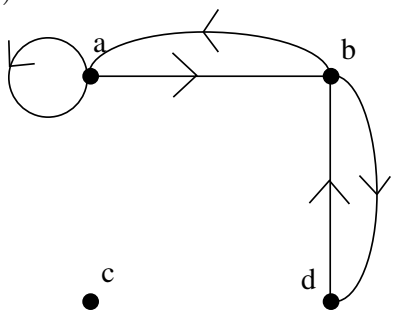
EXAMPLE 2.4.3. *Let R be represented by the digraph*



- *The digraph of $r(R)$ is*



- *The digraph of $s(R)$ is*



Discussion

In Section 2.4 we give several reflexive and symmetric closures of relations. Here is another example using the connection matrix of a relation.

EXAMPLE 2.4.4. *Suppose R is a relation with connection matrix*

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

- *The connection matrix of the reflexive closure is*

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

- *The connection matrix for the symmetric closure is*

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

2.5. Relation Identities.

THEOREM 2.5.1. *Suppose R and S are relations from A to B. Then*

(1) $(R^{-1})^{-1} = R$
(2) $(R \cup S)^{-1} = R^{-1} \cup S^{-1}$
(3) $(R \cap S)^{-1} = R^{-1} \cap S^{-1}$
(4) $(A \times B)^{-1} = B \times A$
(5) $\emptyset^{-1} = \emptyset$
(6) $(\overline{R})^{-1} = \overline{R^{-1}}$
(7) $(R - S)^{-1} = R^{-1} - S^{-1}$
(8) If $A = B$ *then* $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$
(9) If $R \subseteq S$ *then* $R^{-1} \subseteq S^{-1}$

Discussion

The properties given in Theorem 2.5.1 may be useful when trying to find the inverse of a relation.

EXAMPLE 2.5.1. *Suppose R is the relation on the real numbers defined by xRy iff $x+y=1$ or $x-y=1$. R is the union of the two relations R_1 and R_2 defined by xR_1y iff $x+y=1$ and xR_2y iff $x-y=1$. Since R_1^{-1} is defined by $xR_1^{-1}y$ iff $y+x=1$ and R_2^{-1} is defined by $xR_2^{-1}y$ iff $y-x=1$, we apply the property $(R_1 \cup R_2)^{-1} = R_1^{-1} \cup R_2^{-1}$ to see $xR^{-1}y$ is defined by $y+x=1$ or $y-x=1$.*

EXERCISE 2.5.1. *Describe the symmetric closure of the relation, R, in Example 2.5.1.*

PROOF THAT $(R \cup S)^{-1} = R^{-1} \cup S^{-1}$. $(x,y) \in (R \cup S)^{-1}$

$\Leftrightarrow (y,x) \in R \cup S$, by the definition of the inverse relation,

$\Leftrightarrow (y,x) \in R$ or $(y,x) \in S$, by the definition of union,

$\Leftrightarrow (x, y) \in R^{-1}$ or $(x, y) \in S^{-1}$, by the definition of inverse relations,

$\Leftrightarrow (x, y) \in R^{-1} \cup S^{-1}$, by the definition of union.

□

EXERCISE 2.5.2. *Prove Property 1 in Theorem 2.5.1.*

EXERCISE 2.5.3. *Prove Property 3 in Theorem 2.5.1.*

EXERCISE 2.5.4. *Prove Property 4 in Theorem 2.5.1.*

EXERCISE 2.5.5. *Prove Property 5 in Theorem 2.5.1.*

PROOF OF PROPERTY 6 IN THEOREM 2.5.1. Let $a \in A$ and $b \in B$. Then

$$\begin{array}{lll} (b,a) \in (\overline{R})^{-1} & \Leftrightarrow (a,b) \in (\overline{R}) & \text{by the definition of the inverse of a relation} \\ & \Leftrightarrow (a,b) \notin R & \text{by the definition of the complement} \\ & \Leftrightarrow (b,a) \notin R^{-1} & \text{by the definition of the inverse of a relation} \\ & \Leftrightarrow (b,a) \in \overline{R^{-1}} & \text{by the definition of the complement} \end{array}$$

□

EXERCISE 2.5.6. *Prove Property 7 in Theorem 2.5.1.*

EXERCISE 2.5.7. *Prove Property 8 in Theorem 2.5.1.*

EXERCISE 2.5.8. *Prove Property 9 in Theorem 2.5.1.*

2.6. Characterization of Symmetric Relations.

THEOREM 2.6.1. *Let R be a relation on a set A. Then R is symmetric iff $R = R^{-1}$.*

PROOF. First we show that if R is symmetric, then $R = R^{-1}$. Assume R is symmetric. Then

$$\begin{array}{lll} (x,y) \in R & \Leftrightarrow (y,x) \in R, & \text{since } R \text{ is symmetric} \\ & \Leftrightarrow (x,y) \in R^{-1}, & \text{by definition of } R^{-1} \end{array}$$

That is, $R = R^{-1}$.

Conversely, Assume $R = R^{-1}$, and show R is symmetric. To show R is symmetric we must show the implication "if $(x, y) \in R$, then $(y, x) \in R$" holds. Assume $(x, y) \in R$. Then $(y, x) \in R^{-1}$ by the definition of the inverse. But, since $R = R^{-1}$, we have $(y, x) \in R$.

□

Discussion

Theorem 2.6.1 in Section 2.6 gives us an easy way to determine if a relation is symmetric.

2.7. Paths.

Definition 2.7.1.

(1) *A* **path of length** n *in a digraph* G *is a sequence of edges*

$$(x_0, x_1)(x_1, x_2)(x_2, x_3)\cdots(x_{n-1}, x_n).$$

(2) *If* $x_0 = x_n$ *the path is called a* **cycle** *or a* **circuit**.

(3) *If* R *is a relation on a set* A *and* a *and* b *are in* A*, then a* **path of length** n *in* R *from* a *to* b *is a sequence of ordered pairs*

$$(a, x_1)(x_1, x_2)(x_2, x_3)\cdots(x_{n-1}, b)$$

from R.

Discussion

The reflexive and symmetric closures are generally not hard to find. Finding the transitive closure can be a bit more problematic. It is not enough to find $R \circ R = R^2$. R^2 is certainly contained in the transitive closure, but they are not necessarily equal. Defining the transitive closure requires some additional concepts.

Notice that in order for a sequence of ordered pairs or edges to be a path, the terminal vertex of an arc in the list must be the same as the initial vertex of the next arc.

Example 2.7.1. *If* $R = \{(a, a), (a, b), (b, d), (d, b), (b, a)\}$ *is a relation on the set* $A = \{a, b, c, d\}$*, then*

$$(d, b)(b, a)(a, a)(a, a)$$

is a path in R *of length 4.*

2.8. Paths v.s Composition.

Theorem 2.8.1. *If* R *be a relation on a set* A*, then there is a path of length* n *from* a *to* b *in* A *if and only if* $(a, b) \in R^n$.

Proof. (by induction on n):

Basis Step: An arc from a to b is a path of length 1, and it is a path of length 1 in R iff $(a, b) \in R = R^1$.

Induction Hypothesis: Assume that there is a path of length n from a to b in R iff $(a, b) \in R^n$ for some integer n.

Induction Step: Prove that there is a path of length $n + 1$ from a to b in R iff $(a, b) \in R^{n+1}$.

Assume

$$(a, x_1)(x_1, x_2) \cdots (x_{n-1}, x_n)(x_n, b)$$

is a path of length $n + 1$ from a to b in R. Then

$$(a, x_1)(x_1, x_2) \cdots (x_{n-1}, x_n)$$

is a path of length n from a to x_n in R. By the induction hypothesis, $(a, x_n) \in R^n$, and we also have $(x_n, b) \in R$. Thus by the definition of R^{n+1}, $(a, b) \in R^{n+1}$.

Conversely, assume $(a, b) \in R^{n+1} = R^n \circ R$. Then there is a $c \in A$ such that $(a, c) \in R$ and $(c, b) \in R^n$. By the induction hypothesis there is a path of length n in R from c to b. Moreover, (a, c) is a path of length 1 from a to c. By concatenating (a, c) onto the beginning of the path of length n from c to b, we get a path of length $n+1$ from a to b. This completes the induction step.

This shows by the principle of induction that There is a path of length n from a to b iff $(a, b) \in R^n$ for any positive integer n.

□

Discussion

Notice that since the statement is stated for all positive integers, induction is a natural choice for the proof.

2.9. Characterization of a Transitive Relation.

THEOREM 2.9.1. *Let R, S, T, U be relations on a set A.*

(1) If $R \subseteq S$ and $T \subseteq U$, then $R \circ T \subseteq S \circ U$.
(2) If $R \subseteq S$, then $R^n \subseteq S^n$ for every positive integer n.
(3) If R is transitive, then so is R^n for every positive integer n.
(4) If $R^k = R^j$ for some $j > k$, then $R^{j+m} = R^{k+m}$ for every positive integer m.
(5) R is transitive iff R^n is contained in R for every positive integer n.

Discussion

Theorem 2.9.1 in Section 2.9 give several properties that are useful in finding the transitive closure. The most useful are the last two. The second to the last property

stated in the theorem tells us that if we find a power of R that is the same as an earlier power, then we will not find anything new by taking higher powers. The very last tells us that we must include all the powers of R in the transitive closure.

PROOF OF PROPERTY 1. Assume T and U are relations from A to B, R and S are relations from B to C, and $R \subseteq S$ and $T \subseteq U$. Prove $R \circ T \subseteq S \circ U$.

Let $(x, y) \in R \circ T$. Then by the definition of composition, there exists a $b \in B$ such that $(x, b) \in T$ and $(b, y) \in R$. Since $R \subseteq S$ and $T \subseteq U$ we have $(x, b) \in U$ and $(b, y) \in S$. This implies $(x, y) \in S \circ U$. Since (x, y) was an arbitrary element of $R \circ T$ we have shown $R \circ T \subseteq S \circ U$.

□

EXERCISE 2.9.1. *Prove property 2 in Theorem 2.9.1. Hint: use induction and property 1.*

EXERCISE 2.9.2. *Prove property 3 in Theorem 2.9.1. Hint: use induction on n.*

EXERCISE 2.9.3. *Prove property 4 in Theorem 2.9.1.*

EXERCISE 2.9.4. *Prove property 5 in Theorem 2.9.1.*

2.10. Connectivity Relation.

DEFINITION 2.10.1. *The* **connectivity relation** *of the relation R, denoted R^*, is the set of ordered pairs (a, b) such that there is a path (in R) from a to b.*

$$R^* = \bigcup_{n=1}^{\infty} R^n$$

2.11. Characterization of the Transitive Closure.

THEOREM 2.11.1. *Given a relation R on a set A, the transitive closure of R, $t(R) = R^*$.*

PROOF. Let S be the transitive closure of R. We will prove $S = R^*$ by showing that each contains the other.

(1) Proof of $R^* \subseteq S$. By property 5 of Theorem 2.9.1, $S^n \subseteq S$ for all n, since S is transitive. Since $R \subseteq S$, property 2 of Theorem 2.9.1 implies $R^n \subseteq S^n$ for all n. Hence, $R^n \subseteq S$ for all n, and so

$$R^* = \bigcup_{n=1}^{\infty} R^n \subseteq S.$$

(2) Proof of $S \subseteq R^*$. We will prove this by first showing that R^* is transitive.
Suppose (a, b) and (b, c) are in R^*. Then $(a, b) \in R^m$ for some $m \geq 1$ and $(b, c) \in R^n$ for some $n \geq 1$. This means there is a path of length m in R from a to b and a path of length n in R from b to c. Concatenating these two paths gives a path of length $m + n$ in R from a to c. That is, $(a, c) \in R^{m+n} \subseteq R^*$. Thus, R^* is transitive. Since S is the transitive closure of R, it must be contained in any transitive relation that contains R. Thus, $S \subseteq R^*$.

□

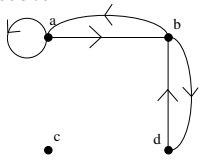
Discussion

Theorem 2.11.1 in Section 2.11 identifies the transitive closure, $t(R)$, of a relation R. In the second part of the proof of Theorem 2.11.1 we have used the definition of the closure of a relation with respect to a property as given in Section 2.1. (See also the discussion immediately following Section 2.1.)

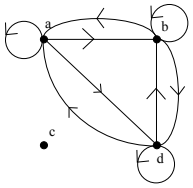
2.12. Example.

EXAMPLE 2.12.1. *Let R be the relation on the set of integers given by $\{(i, i+1)|i \in \mathbb{Z}\}$. Then the transitive closure of R is $R^* = \{(i, i + k)|i \in \mathbb{Z}$ and $k \in \mathbb{Z}^+\}$.*

EXAMPLE 2.12.2. *S is the relation*



The transitive closure is



Discussion

Here is another example:

EXAMPLE 2.12.3. *Let A be the set of all people and R the relation $\{(x, y)|$ person x is a parent of person $y\}$. Then R^* is the relation $\{(x, y)|$ person x is an ancestor of person $y\}$.*

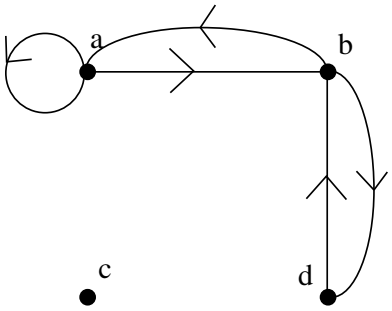
2.13. Cycles.

THEOREM 2.13.1. *If R is a relation on a set A and $|A| = n$, then any path of length greater than or equal to n must contain a cycle.*

PROOF. Suppose $x_0, x_1, x_2, ..., x_m$ is a sequence in A such that $(x_{i-1}, x_i) \in R$ for $i = 1, ..., m$, where $m \geq n$. That is, there is a path in R of length greater than or equal to n. Since $|A| = n$, the pigeon hole principle implies that $x_i = x_j$ for some $i \neq j$. Thus, (assuming $i < j$) the path contains a cycle from x_i to x_j $(= x_i)$. □

Discussion

An easy way to understand what Theorem 2.13.1 in Section 2.13 is saying is to look at a digraph. Take the digraph



Take *any* path of length greater than 4. Say, $(a,b)(b,d)(d,b)(b,a)(a,b)$. By the theorem the path must have a cycle contained within it (a path that begins and ends at the same vertex). In this particular example there are several cycles; e.g., $(a,b)(b,d)(d,b)(b,a)$. Notice if you eliminate this cycle you will have a shorter path with same beginning and ending vertex as the original path.

One of the points of Theorem 2.13.1 is that if $|A| = n$ then R^k for $k \geq n$ does not contain any arcs that did not appear in at least one of the first n powers of R.

2.14. Corollaries to Theorem 2.13.1.

COROLLARY 2.14.0.1. *If $|A| = n$, the transitive closure of a relation R on A is*

$$R^* = R \cup R^2 \cup R^3 \cup \cdots \cup R^n.$$

COROLLARY 2.14.0.2. *We can find the connection matrix of R^* by computing the join of the first n Boolean powers of the connection matrix of R.*

Discussion

The corollaries of Theorem 2.13.1 in Section 2.14 give us powerful algorithms for computing the transitive closure of relations on finite sets. Now, it is *possible* for $R^* = R \cup R^2 \cup \cdots \cup R^k$ for some $k < n$, but the point is that you are guaranteed that $R^* = R \cup R^2 \cup \cdots \cup R^n$ if $|A| = n$.

2.15. Example 10.

EXAMPLE 2.15.1. *Let R be the relation $\{(a,b),(c,b),(d,a),(c,c)\}$ in $A = \{a,b,c,d\}$. We find the transitive closure by using the corollary in Section 2.14.*

The connectivity matrix for R is

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \quad M^{[2]} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$M^{[3]} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{ and } M^{[4]} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Therefore the connectivity matrix for R^* is

$$M \vee M^{[2]} \vee M^{[3]} \vee M^{[4]} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Discussion

In this example we use the basic algorithm to find the transitive closure of a relation. Recall $M^{[n]} = \odot_{k=1}^{n} M$. After taking the Boolean powers of the matrix, we take the join of the four matrices. While it did not happen on this example, it is possible for a Boolean power of the matrix to repeat before you get to the cardinality of A. If this happens you may stop and take the join of the powers you have, since nothing new will be introduced after that.

2.16. Properties v.s Closure.

EXERCISE 2.16.1. *Suppose R is a relation on a set A that is reflexive. Prove or disprove*

(a) $s(R)$ is reflexive
(b) $t(R)$ is reflexive

EXERCISE 2.16.2. *Suppose R is a relation on a set A that is symmetric. Prove or disprove*

(a) $r(R)$ is symmetric
(b) $t(R)$ is symmetric

EXERCISE 2.16.3. *Suppose R is a relation on a set A that is transitive. Prove or disprove*

(a) $r(R)$ is transitive
(b) $s(R)$ is transitive

3. Equivalence Relations

3.1. Definition of an Equivalence Relations.

DEFINITION 3.1.1. *A relation R on a set A is an* **equivalence relation** *if and only if R is*

- *reflexive,*
- *symmetric, and*
- *transitive.*

Discussion

Section 3.1 recalls the definition of an equivalence relation. In general an equivalence relation results when we wish to "identify" two elements of a set that share a common attribute. The definition is motivated by observing that any process of "identification" must behave somewhat like the equality relation, and the equality relation satisfies the reflexive ($x = x$ for all x), symmetric ($x = y$ implies $y = x$), and transitive ($x = y$ and $y = z$ implies $x = z$) properties.

3.2. Example.

EXAMPLE 3.2.1. *Let R be the relation on the set $\mathbb{R}$ real numbers defined by xRy iff $x - y$ is an integer. Prove that R is an equivalence relation on $\mathbb{R}$.*

PROOF.

I. Reflexive: Suppose $x \in \mathbb{R}$. Then $x - x = 0$, which is an integer. Thus, xRx.
II. Symmetric: Suppose $x, y \in \mathbb{R}$ and xRy. Then $x - y$ is an integer. Since $y - x = -(x - y)$, $y - x$ is also an integer. Thus, yRx.
III. Suppose $x, y \in \mathbb{R}$, xRy and yRz. Then $x - y$ and $y - z$ are integers. Thus, the sum $(x - y) + (y - z) = x - z$ is also an integer, and so xRz.

Thus, R is an equivalence relation on $\mathbb{R}$. □

Discussion

EXAMPLE 3.2.2. *Let R be the relation on the set of real numbers $\mathbb{R}$ in Example 1. Prove that if xRx' and yRy', then $(x + y)R(x' + y')$.*

PROOF. Suppose xRx' and yRy'. In order to show that $(x+y)R(x'+y')$, we must show that $(x + y) - (x' + y')$ is an integer. Since

$$(x + y) - (x' + y') = (x - x') + (y - y'),$$

and since each of $x-x'$ and $y-y'$ is an integer (by definition of R), $(x-x')+(y-y')$ is an integer. Thus, $(x+y)R(x'+y')$.

□

EXERCISE 3.2.1. *In the example above, show that it is possible to have xRx' and yRy', but $(xy)\not R(x'y')$.*

EXERCISE 3.2.2. *Let V be the set of vertices of a simple graph G. Define a relation R on V by vRw iff v is adjacent to w. Prove or disprove: R is an equivalence relation on V.*

3.3. Equivalence Classes.

DEFINITION 3.3.1.

(1) *Let R be an equivalence relation on A and let $a \in A$. The set $[a] = \{x | aRx\}$ is called the* **equivalence class** *of a.*
(2) *The element in the bracket in the above notation is called the* **Representative** *of the equivalence class.*

THEOREM 3.3.1. *Let R be an equivalence relation on a set A. Then the following are equivalent:*

(1) aRb
(2) $[a] = [b]$
(3) $[a] \cap [b] \neq \emptyset$

PROOF. $1 \to 2$. Suppose $a, b \in A$ and aRb. We must show that $[a] = [b]$.

Suppose $x \in [a]$. Then, by definition of $[a]$, aRx. Since R is symmetric and aRb, bRa. Since R is transitive and we have both bRa and aRx, bRx. Thus, $x \in [b]$.

Suppose $x \in [b]$. Then bRx. Since aRb and R is transitive, aRx. Thus, $x \in [a]$.

We have now shown that $x \in [a]$ if and only if $x \in [b]$. Thus, $[a] = [b]$.

$2 \to 3$. Suppose $a, b \in A$ and $[a] = [b]$. Then $[a] \cap [b] = [a]$. Since R is reflexive, aRa; that is $a \in [a]$. Thus $[a] = [a] \cap [b] \neq \emptyset$.

$3 \to 1$. Suppose $[a] \cap [b] \neq \emptyset$. Then there is an $x \in [a] \cap [b]$. By definition, aRx and bRx. Since R is symmetric, xRb. Since R is transitive and both aRx and xRb, aRb. □

Discussion

The purpose of any identification process is to break a set up into subsets consisting of mutually identified elements. An equivalence relation on a set A does precisely this: it decomposes A into special subsets, called *equivalence classes*. Looking back at the example given in Section 3.2, we see the following equivalence classes:

- $[0] = \mathbb{Z}$, the set of integers.
- $[\frac{1}{2}] = \{\frac{m}{2} | m \text{ is an odd integer}\}$
- $[\pi] = \{\pi + n | n \text{ is an integer}\} = [\pi + n]$, for any integer n.

Notice that $[\frac{3}{4}] = [-\frac{37}{4}]$. The number $\frac{3}{4}$ is a representative of $[\frac{3}{4}]$, but $-\frac{37}{4}$ is also a representative of $[\frac{3}{4}]$. Indeed, any element of an equivalence class can be used to represent that equivalence class.

These ideas are summed up in Theorem 3.3.1 in Section 3.3. When we say several statements, such as P_1, P_2, and P_3 are equivalent, we mean $P_1 \leftrightarrow P_2 \leftrightarrow P_3$ is true. Notice that in order to prove that the statements are mutually equivalent, it is sufficient to prove a circle of implications, such as $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_1$. This is how we set up the proof of Theorem 3.3.1.

3.4. Partition.

DEFINITION 3.4.1. *A collection $\mathcal{S}$ of nonempty subsets of a set A is a* **partition** *of A if*

(1) $S \cap S' = \emptyset$, if S and S' are in $\mathcal{S}$ and $S \neq S'$, and
(2) $A = \bigcup\{S | S \in \mathcal{S}\}$.

THEOREM 3.4.1. *The equivalence classes of an equivalence relation on A form a partition of A. Conversely, given a partition on A, there is an equivalence relation with equivalence classes that are exactly the partition given.*

Discussion

The definition in Section 3.4 along with Theorem 3.4.1 describe formally the properties of an equivalence relation that motivates the definition. Such a decomposition is called a **partition**. For example, if we wish to identify two integers if they are either both even or both odd, then we end up with a partition of the integers into two sets, the set of even integers and the set of odd integers. The converse of Theorem 3.4.1 allows us to create or define an equivalence relation by merely partitioning a set into mutually exclusive subsets. The common "attribute" then might just be that elements belong to the same subset in the partition.

the notation used in the second part of Theorem 3.4.1 means that we take the union of all the sets that are members of the set to the far right and this union is defined to be set A.

DEFINITION 3.4.2. *If R is an equivalence relation on a set A, the set of equivalence classes of R is denoted A/R.*

Theorem 3.4.1 follows fairly easily from Theorem 3.3.1 in Section 3.3. Here is a proof of one part of Theorem 3.4.1.

PROOF. Suppose R is an equivalence relation on A and $\mathcal{S}$ is the set of equivalence classes of R. If S is an equivalence class, then $S = [a]$, for some $a \in A$; hence, S is nonempty, since aRa by the reflexive property of R.

By Theorem 3.3.1, if $S = [a]$ and $S' = [b]$ are in $\mathcal{S}$, then $[a] = [b]$ iff $[a] \cap [b] \neq \emptyset$. Since this is a biconditional, this statement is equivalent to $[a] \neq [b]$ iff $[a] \cap [b] = \emptyset$.

Since each equivalence class is contained in A, $\bigcup\{S|S \in \mathcal{S}\} \subseteq A$. But, as we just saw, every element in A is in the equivalence class it represents, so $A \subseteq \bigcup\{S|S \in \mathcal{S}\}$. This shows $\bigcup\{S|S \in \mathcal{S}\} = A$. □

EXERCISE 3.4.1. *Prove the converse statement in Theorem 3.4.1.*

3.5. Intersection of Equivalence Relations.

THEOREM 3.5.1. *If R_1 and R_2 are equivalence relations on a set A then $R_1 \cap R_2$ is also an equivalence relation on A.*

Discussion

To prove Theorem 3.5.1, it suffices to show the intersection of

- reflexive relations is reflexive,
- symmetric relations is symmetric, and
- transitive relations is transitive.

But these facts were established in the section on the Review of Relations.

3.6. Example.

EXAMPLE 3.6.1. *Let m be a positive integer. The relation $a \equiv b\,(mod\ m)$, is an equivalence relation on the set of integers.*

PROOF. Reflexive. If a is an arbitrary integer, then $a - a = 0 = 0 \cdot m$. Thus $a \equiv a \pmod{m}$.

Symmetric. If $a \equiv b \pmod{m}$, then $a - b = k \cdot m$ for some integer k. Thus, $b - a = (-k) \cdot m$ is also divisible by m, and so $b \equiv a \pmod{m}$.

Transitive. Suppose $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$. Then $a - b = k \cdot m$ and $b - c = \ell \cdot m$ for some integers k and ℓ. Then

$$a - c = (a - b) + (b - c) = k \cdot m + \ell \cdot m = (k + \ell)m$$

is also divisible by m. That is, $a \equiv c \pmod{m}$. □

Discussion

Recall the "congruence" relations on the set $\mathbb{Z}$ of integers: Given an positive integer m and integers a and b, $a \equiv b \pmod{m}$ (read "a is congruent to b modulo m) iff $m|(a - b)$; that is, $a - b = k \cdot m$ for some integer k.

EXERCISE 3.6.1. *What are the equivalence classes for the congruence relation*

(1) $a \equiv b \, (mod\ 2)$*?*
(2) $a \equiv b \, (mod\ 3)$*?*
(3) $a \equiv b \, (mod\ 5)$*?*

Given a positive integer m, the equivalence classes under the relation $a \equiv b \pmod{m}$ have *canonical* representatives. If we use the Division Algorithm to divide the integer a by the integer m, we get a quotient q and remainder r, $0 \leq r < m$, satisfying the equation $a = mq + r$. Recall that $r = a \, \mathbf{mod} \, m$ and that $a \equiv r \pmod{m}$. Thus $[a] = [r]$, and so there are exactly m equivalence classes

$$[0], [1], ...[m - 1].$$

If R is the congruence modulo m relation on the set $\mathbb{Z}$ of integers, the set of equivalence classes, $\mathbb{Z}/R$ is usually denoted by either $\mathbb{Z}/m$ or $\mathbb{Z}/m\mathbb{Z}$. That is,

$$\mathbb{Z}/m = \{[0], [1], ...[m - 1]\}.$$

REMARK 3.6.1. *If A is an infinite set and R is an equivalence relation on A, then A/R may be finite, as in the example above, or it may be infinite. As the following exercise shows, the set of equivalences classes may be very large indeed.*

EXERCISE 3.6.2. *Let R be the equivalence relation defined on the set of real numbers $\mathbb{R}$ in Example 3.2.1 (Section 3.2). That is, xRy iff $x - y$ is an integer. Prove that every equivalence class $[x]$ has a* unique *canonical representative r such that $0 \leq r < 1$. That is, for every x there is a unique r such that $[x] = [r]$ and $0 \leq r < 1$. [Hint: You might recall the "floor" function $f(x) = \lfloor x \rfloor$.]*

3.7. Example.

EXAMPLE 3.7.1. *Let R be the relation on the set of ordered pairs of positive integers such that $(a,b)R(c,d)$ if and only if $ad = bc$.*

- *R is an equivalence relation.*
- *The equivalence class of $(2,3)$:*
$$[(2,3)] = \{(2k, 3k) | k \in \mathbb{Z}^+\}.$$
- *There is a natural bijection between the equivalence classes of this relation and the set of positive rational numbers.*

Discussion

Notice that the relation R in Example 3.7.1 is a relation on the set $\mathbb{Z}^+ \times \mathbb{Z}^+$, and so $R \subseteq (\mathbb{Z}^+ \times \mathbb{Z}^+) \times (\mathbb{Z}^+ \times \mathbb{Z}^+)$.

PROOF R IN EXAMPLE 3.7.1 IS AN EQUIVALENCE RELATION. We must show that R is reflexive, symmetric, and transitive.

I. Reflexive: Let (a,b) be an ordered pair of positive integers. To show R is reflexive we must show $((a,b),(a,b)) \in R$. Multiplication of integers is commutative, so $ab = ba$. Thus $((a,b),(a,b)) \in R$.
II. Symmetric: Let (a,b) and (c,d) be ordered pairs of positive integers such that $(a,b)R(c,d)$ (recall this notation is equivalent to $((a,b),(c,d)) \in R$). Then $ad = bc$. This equation is equivalent to $cb = da$, so $(c,d)R(a,b)$. This shows R is symmetric.
III. Transitive: Let (a,b), (c,d), and (e,f) be ordered pairs of positive integers such that $(a,b)R(c,d)$ and $(c,d)R(e,f)$. Then $ad = bc$ and $cf = de$. Thus, $adf = bcf$ and $bcf = bde$, which implies $adf = bde$. Since $d \neq 0$, we can cancel it from both sides of this equation to get $af = be$. This shows $(a,b)R(e,f)$, and so R is transitive.

□

One of the points of this example is that there is a bijection between the equivalence classes of this relation and the set of positive rational numbers. In other words, the function

$$f\colon (\mathbb{Z}^+ \times \mathbb{Z}^+)/R = \{[(a,b)] | [(a,b)] \text{ is an equivalence class of } R\} \to \mathbb{Q}^+$$

defined by $f([(a,b)]) = a/b$ is well-defined and is a bijection. This follows from the fact that

$$[(a,b)] = [(c,d)] \Leftrightarrow (a,b)R(c,d) \Leftrightarrow ad = bc \Leftrightarrow \frac{a}{b} = \frac{c}{d}.$$

EXERCISE 3.7.1. *Let R be the relation defined on the set of ordered pairs $\mathbb{Z}^+ \times \mathbb{Z}^+$ of positive integers defined by*

$$(a, b)R(c, d) \Leftrightarrow a + d = b + c.$$

(1) Prove that R is an equivalence relation on $\mathbb{Z}^+ \times \mathbb{Z}^+$.
(2) List 5 different members of the equivalence class $[(1, 4)]$.

EXERCISE 3.7.2. *Let R be the relation defined on the set of ordered pairs $\mathbb{Z}^+ \times \mathbb{Z}^+$ of positive integers defined by*

$$(a, b)R(c, d) \Leftrightarrow a + d = b + c.$$

Prove that the function $f : \mathbb{Z}^+ \times \mathbb{Z}^+/R \to \mathbb{Z}$, defined by $f([a, b]) = a-b$, is well-defined and a bijection.

3.8. Isomorphism is an Equivalence Relation.

THEOREM 3.8.1. *Let S be a set of simple graphs. Define a relation R on S as follows:*

If $G, H \in S$, then $(G, H) \in R$ if and only if $G \simeq H$.

(This is equivalent to $(G, H) \in R$ if and only if there exists an isomorphism $f : V(G) \to V(H)$ that preserves adjacencies.)

Then R is an equivalence relation on S.

PROOF. Reflexive. Suppose $G \in S$. We need to show $(G, G) \in R$.

Define the function $f : V(G) \to V(G)$ by $f(v) = v$ for every vertex $v \in V(G)$. Then f is the identity function on $V(G)$; hence, f is a bijection. ($f^{-1} = f$!)

Clearly, v and u are adjacent in G if and only if $f(v) = v$ and $f(u) = u$ are adjacent.

Thus, $(G, G) \in R$.

Symmetric. Suppose $G, H \in S$ and $(G, H) \in R$. Then there exists an isomorphism $f : V(G) \to V(H)$. We need to find an isomorphism $g : V(H) \to V(G)$.

Since f is a bijection, f is invertible. Thus the map $f^{-1} : V(H) \to V(G)$ is defined, and we shall show it is an isomorphism. We know the inverse of a bijection is itself a bijection, so all we need to show is that f^{-1} preserves adjacency.

Suppose $u, v \in V(H)$. Then $f^{-1}(u) = x$ and $f^{-1}(v) = y$ are vertices of G.

Now, we know f preserves adjacency, so x and y are adjacent in G if and only if $f(x) = u$ and $f(y) = v$ are adjacent in H. Use the previous equations to rewrite this statement in terms of u and v: $f^{-1}(u)(= x)$ and $f^{-1}(v)(= y)$ are adjacent in G if and only if $u(= f(x))$ and $v(= f(y))$ are adjacent in H.

Thus f^{-1} preserves adjacency, and so $(H, G) \in R$.

Transitive. Suppose $G, H, K \in S$ are graphs such that $(G, H), (H, K) \in R$. We need to prove $(G, K) \in R$.

Since (G, H) and (H, K) are in R, there are isomorphisms $f\colon V(G) \to V(H)$ and $g\colon V(H) \to V(K)$. We need to find an isomorphism $h\colon V(G) \to V(K)$. *Notice that we have used different letters for the functions here. The function g is not necessarily the same as the function f, so we cannot call it f as well.*

Let $h = g \circ f$. We will show h is an isomorphism.

Since the composition of bijections is again a bijection, $g \circ f\colon V(G) \to V(K)$ is a bijection.

What we still need to show is that the composition preserves adjacency. Let u and v be vertices in G. Recall that f must preserve adjacency. Therefore, u and v are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H. But since g preserves adjacency, $f(u)$ and $f(v)$ are adjacent in H if and only if $g(f(u))$ and $g(f(v))$ are adjacent in K. Using the fact that "if and only if" is transitive, we see that u and v are adjacent in G if and only if $(g \circ f)(u)$ and $(g \circ f)(v)$ are adjacent in K. This implies that $g \circ f$ preserves adjacency, and so $g \circ f\colon V(G) \to V(K)$ is an isomorphism.

□

Discussion

Section 3.8 recalls the notion of graph isomorphism. Here we prove that graph isomorphism is an equivalence relation on any set of graphs. It is tempting to say that graph isomorphism is an equivalence relation on the "set of all graphs," but logic precludes the existence of such a set.

3.9. Equivalence Relation Generated by a Relation R.

Definition 3.9.1. *Suppose R is a relation on a set A. The* **equivalence relation on A generated by a R,** *denoted R_e, is the smallest equivalence relation on A that contains R.*

Discussion

There are occasions in which we would like to define an equivalence relation on a set by starting with a primitive notion of "equivalence", which, in itself, may not satisfy one or more of the three required properties. For example, consider the set of vertices V of a simple graph G and the adjacency relation R on V: uRv iff u is adjacent to v. You would have discovered while working through Exercise 3.2.2 that, for most graphs, R is neither reflexive nor transitive.

EXERCISE 3.9.1. *Suppose V is the set of vertices of a simple graph G and R is the adjacency relation on V: uRv iff u is adjacent to v. Prove that R_e is the relation*

$$uR_ev \text{ iff either } u = v \text{ or there is a path in } G \text{ from } u \text{ to } v.$$

3.10. Using Closures to find an Equivalence Relation.

THEOREM 3.10.1. *Suppose R is a relation on a set A. Then R_e, the equivalence relation on A generated by R, is the relation $t(s(r(R)))$. That is, R_e may be obtained from R by taking*

(1) the reflexive closure $r(R)$ of R, then
(2) the symmetric closure $s(r(R))$ of $r(R)$, and then
(3) the transitive closure $t(s(r(R)))$ of $s(r(R))$.

PROOF. Suppose R is a relation on a set A. We must show
(1) $t(s(r(R)))$ is an equivalence relation containing R, and
(2) if S is an equivalence relation containing R, then $t(s(r(R))) \subseteq S$.

Proof of (1).

I. Reflexive: If $a \in A$, then $(a, a) \in r(R)$; hence, $(a, a) \in t(s(r(R)))$, since $r(R) \subseteq t(s(r(R)))$.
II. Symmetric: Suppose $(a, b) \in t(s(r(R)))$. Then there is a chain $(a, x_1), (x_1, x_2), ..., (x_n, b)$ in $s(r(R))$. Since $s(r(R))$ is symmetric, $(b, x_n), ..., (x_2, x_1), (x_1, a)$ are in $s(r(R))$. Hence, $(b, a) \in t(s(r(R)))$, since $t(s(r(R)))$ is transitive.
III. Transitive: $t(s(r(R)))$, being the transitive closure of $s(r(R))$, is transitive, by definition.

Proof of (2). Suppose S is an equivalence relation containing R.

I. Since S is reflexive, S contains the reflexive closure of R. That is, $r(R) \subseteq S$.
II. Since S is symmetric and $r(R) \subseteq S$, S contains the symmetric closure of $r(R)$. That is, $s(r(R)) \subseteq S$.
III. Since S is transitive and $s(r(R)) \subseteq S$, S contains the transitive closure of $s(r(R))$. That is, $t(s(r(R))) \subseteq S$.

□

Discussion

Theorem 3.10.1 in Section 3.10 describes the process by which the equivalence relation generated by a relation R can be constructed using the closure operations discussed in the notes on Closure. As it turns out, it doesn't matter whether you take the reflexive closure before you take the symmetric and transitive closures, but it is important that the symmetric closure be taken *before* the transitive closure.

EXERCISE 3.10.1. *Given a relation R on a set A, prove that $R_e = (R \cup \Delta \cup R^{-1})^*$. [See the lecture notes on Closure for definitions of the terminology.]*

EXERCISE 3.10.2. *Suppose A is the set of all people (alive or dead) and R is the relation "is a parent of". Describe the relation R_e in words. What equivalence class do you represent?*

EXERCISE 3.10.3. *Give an example of a relation R on a set A such that $R_e \neq s(t(r(R)))$.*

4. Partial Orderings

4.1. Definition of a Partial Order.

DEFINITION 4.1.1.

(1) A relation R on a set A is a **partial order** *iff R is*
- *reflexive,*
- *antisymmetric, and*
- *transitive.*

(2) (A, R) is called a **partially ordered set** *or a* **poset**.

(3) If, in addition, either aRb or bRa, for every $a, b \in A$, then R is called a **total order** *or a* **linear order** *or a* **simple order**. *In this case (A, R) is called a* **chain**.

(4) The notation $a \preceq b$ is used for aRb when R is a partial order.

Discussion

The classic example of an order is the order relation on the set of real numbers: aRb iff $a \leq b$, which is, in fact, a total order. It is this relation that suggests the notation $a \preceq b$, but this notation is *not* used exclusively for total orders.

Notice that in a partial order on a set A it is *not required* that every pair of elements of A be related in one way or the other. That is why the word *partial* is used.

4.2. Examples.

EXAMPLE 4.2.1. *$(\mathbb{Z}, \leq)$ is a poset. Every pair of integers is comparable via $\leq$, so $\leq$ is a total order and $(\mathbb{Z}, \leq)$ is a chain.*

EXAMPLE 4.2.2. *If S is a set then $(P(S), \subseteq)$ is a poset.*

EXAMPLE 4.2.3. *$(\mathbb{Z}^+, |)$ is a poset. The relation $a|b$ means "a divides b."*

Discussion

Example 4.2.2 better illustrates the general nature of a partial order. This partial order is not necessarily a total order; that is, it is not always the case that either $A \subseteq B$ or $B \subseteq A$ for every pair of subsets of S. Can you think of the only occasions in which this would be a total order?

Example 4.2.3 is also only a partial order. There are many pairs of integers such that $a \not| b$ and $b \not| a$.

4.3. Pseudo-Orderings.

Definition 4.3.1.

A relation $\prec$ on S is called a **pseudo-order** *if*

- *the relation is irreflexive and*
- *transitive.*

Some texts call this a **quasi-order**. *Rosen uses quasi-order to mean a different type of relation, though.*

Theorem 4.3.1 (Theorems and Notation).

(1) *Given a poset $(S, \preceq)$, we define a relation $\prec$ on S by $x \prec y$ if and only if $x \preceq y$ and $x \neq y$. The relation $\prec$ is a pseudo-order.*
(2) *Given a set S and a pseudo-order $\prec$ on S, we define a relation $\preceq$ on S by $x \preceq y$ if and only if $x \prec y$ or $x = y$. The relation $\preceq$ is a partial order.*
(3) *Given a poset $(S, \preceq)$, we define the relation $\succeq$ on S by $x \succeq y$ iff $y \preceq x$. $(S, \succeq)$ is a poset and $\succeq$ is the inverse of $\preceq$.*
(4) *Given a set S and a pseudo-order $\prec$ on S, we define the relation $\succ$ on S by $x \succ y$ iff $y \prec x$. $\succ$ is a pseudo-order on S and $\succ$ is the inverse of $\prec$.*
(5) *In any discussion of a partial order relation $\preceq$, we will use the notations $\prec$, $\succeq$, and $\succ$ to be the relations defined above, depending on $\preceq$. Similarly, if we are given a a pseudo-order, $\prec$, then $\preceq$ will be the partial order defined in part 2.*

Discussion

The notation above is analogous to the usual $\leq$, $\geq$, $<$, and $>$ notations used with real numbers. We do not require that the orders above be total orders, though. Another example you may keep in mind that uses similar notation is $\subseteq$, $\supseteq$, $\subset$, $\supset$ on sets. These are also partial and pseudo-orders.

Exercise 4.3.1. *Prove a pseudo-order, $\prec$, is antisymmetric.*

Exercise 4.3.2. *Prove Theorem 4.3.1 part 1.*

Exercise 4.3.3. *Prove Theorem 4.3.1 part 2.*

4.4. Well-Ordered Relation.

Definition 4.4.1. *Let R be a partial order on A and suppose $S \subseteq A$.*

(1) *An element $s \in S$ is a* **least element** *of S iff sRb for every $b \in S$.*
(2) *An element $s \in S$ is a* **greatest element** *of S iff bSs for every $b \in S$.*

(3) A chain (A, R) is **well-ordered** *iff every nonempty subset of A has a least element.*

Discussion

Notice that if s is a least (greatest) element of S, s must be an element of S and s must precede (be preceded by) all the other elements of S.

Confusion may arise when we define a partial order on a well-known set, such as the set $\mathbb{Z}^+$ of positive integers, that already has a natural ordering. One such ordering on $\mathbb{Z}^+$ is given in Example 4.2.3. As another example, one could perversely impose the relation $\preceq$ on $\mathbb{Z}^+$ by defining $a \preceq b$ iff $b \leq a$. With respect to the relation $\preceq$, $\mathbb{Z}^+$ would have no least element and it's "greatest" element would be 1! This confusion may be alleviated somewhat by reading $a \preceq b$ as "a precedes b" instead of "a is less than or equal to b", especially in those cases when the set in question already comes equipped with a natural order different from $\preceq$.

4.5. Examples.

EXAMPLE 4.5.1. *$(\mathbb{Z}, \leq)$ is a chain, but it is not well-ordered.*

EXAMPLE 4.5.2. *$(\mathbb{N}, \leq)$ is well-ordered.*

EXAMPLE 4.5.3. *$(\mathbb{Z}^+, \geq)$ is a chain, but is not well-ordered.*

Discussion

In Example 4.5.1, the set of integers does not have a least element. If we look at the set of positive integers, however, every nonempty subset (including $\mathbb{Z}^+$) has a least element. Notice that if we reverse the inequality, the "least element" is now actually the one that is larger than the others – look back at the discussion in Section 4.4 – and there is no "least element" of $\mathbb{Z}^+$.

Pay careful to the definition of what it means for a chain to be well-ordered. It requires every nonempty subset to have a *least* element, but it does not require that every nonempty subset have a greatest element.

EXERCISE 4.5.1. *Suppose $(A, \preceq)$ is a poset such that every nonempty subset of A has a least element. Prove that $\preceq$ is a total ordering on A.*

4.6. Lexicographic Order.

DEFINITION 4.6.1. *Given two posets $(A_1, \preceq_1)$ and $(A_2, \preceq_2)$ we construct an* **induced** *or* **lexicographic** *partial order $\preceq_L$ on $A_1 \times A_2$ by defining $(x_1, y_1) \preceq_L (x_2, y_2)$ iff*

- $x_1 \prec_1 x_2$ *or*
- $x_1 = x_2$ *and* $y_1 \preceq_2 y_2$.

This definition is extended recursively to Cartesian products of partially ordered sets $A_1 \times A_2 \times \cdots \times A_n$.

Discussion

EXERCISE 4.6.1. *Prove that if each of the posets $(A_1, \preceq_1)$ and $(A_2, \preceq_2)$ is a chain and $\preceq_L$ is the lexicographic order on $A_1 \times A_2$, then $(A_1 \times A_2, \preceq_L)$ is also a chain.*

EXERCISE 4.6.2. *Suppose for each positive integer n, $(A_n, preceq_n)$ is a poset. Give a recursive definition for the lexicographic order on $A_1 \times A_2 \times \cdots \times A_n$ for all positive integers n.*

4.7. Examples 4.7.1 and 4.7.2.

EXAMPLE 4.7.1. *Let $A_1 = A_2 = \mathbb{Z}^+$ and $\preceq_1 = \preceq_2 = |$ ("divides"). Then*

- $(2, 4) \preceq_L (2, 8)$
- $(2, 4)$ *is not related under* $\preceq_L$ *to* $(2, 6)$.
- $(2, 4) \preceq_L (4, 5)$

EXAMPLE 4.7.2. *Let $A_i = \mathbb{Z}^+$ and $\preceq_i = |$, for $i = 1, 2, 3, 4$. Then*

- $(2, 3, 4, 5) \preceq_L (2, 3, 8, 2)$
- $(2, 3, 4, 5)$ *is not related under* $\preceq_L$ *to* $(3, 6, 8, 10)$.
- $(2, 3, 4, 5)$ *is not related under* $\preceq_L$ *to* $(2, 3, 5, 10)$.

Discussion

Notice that $(2, 4)$ does not precede $(2, 6)$: although their first entries are equal, 4 does not divide 6. In fact, the pairs $(2, 4)$ and $(2, 6)$ are not related in any way. On the other hand, since $2|4$ (and $2 \neq 4$), we do not need to look any further than the first place to see that $(2, 4) \preceq_L (4, 5)$.

Notice also in Example 4.7.2, the first non-equal entries determine whether or not the relation holds.

4.8. Strings. We extend the lexigraphic ordering to strings of elements in a poset $(A, \preceq)$ as follows:

$$a_1 a_2 \cdots a_m \preceq_L b_1 b_2 \cdots b_n$$

iff

- $(a_1, a_2, \ldots, a_t) \preceq_L (b_1, b_2, \ldots, b_t)$ where $t = \min(m, n)$, or
- $(a_1, a_2, \ldots, a_m) = (b_1, b_2, \ldots, b_m)$ and $m < n$.

Discussion

The ordering defined on strings gives us the usual alphabetical ordering on words and the usual order on bit string.

EXERCISE 4.8.1. *Put the bit strings* $0110, 10, 01$*, and* 010 *in increasing order using*

(1) numerical order by considering the strings as binary numbers,
(2) lexicographic order using $0 < 1$*,*

There are numerous relations one may impose on products. Lexicographical order is just one partial order.

EXERCISE 4.8.2. *Let* $(A_1, \preceq_1)$ *and* $(A_2, \preceq_2)$ *be posets. Define the relation* $\preceq$ *on* $A_1 \times A_2$ *by* $(a_1, a_2) \preceq (b_1, b_2)$ *if and only if* $a_1 \preceq_1 b_1$ *and* $a_2 \preceq_2 b_2$*. Prove* $\preceq$ *is a partial order. This partial order is called the* **product order**.

4.9. Hasse or Poset Diagrams.

DEFINITION 4.9.1. *To construct a* **Hasse** *or* **poset diagram** *for a poset* (A, R)*:*

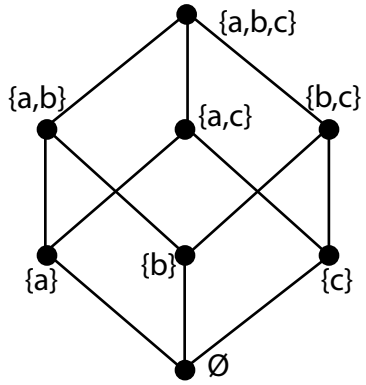
(1) Construct a digraph representation of the poset (A, R) *so that all arcs point up (except the loops).*
(2) Eliminate all loops.
(3) Eliminate all arcs that are redundant because of transitivity.
(4) Eliminate the arrows on the arcs.

Discussion

The Hasse diagram of a poset is a simpler version of the digraph representing the partial order relation. The properties of a partial order assure us that its digraph can be drawn in an oriented plane so that each element lies below all other elements it precedes in the order. Once this has been done, all redundant information can be removed from the digraph and the result is the Hasse diagram.

4.10. Example 4.10.1.

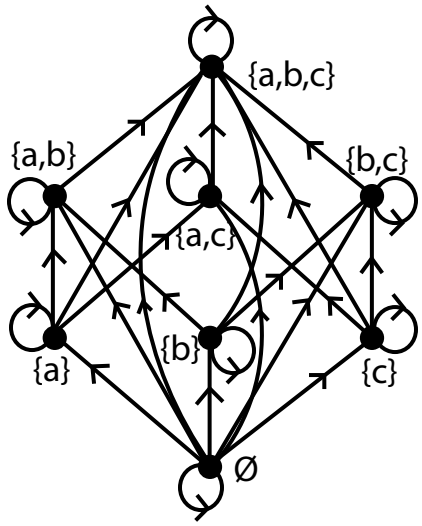
EXAMPLE 4.10.1. *The Hasse diagram for* $(P(\{a, b, c\}), \subseteq)$ *is*



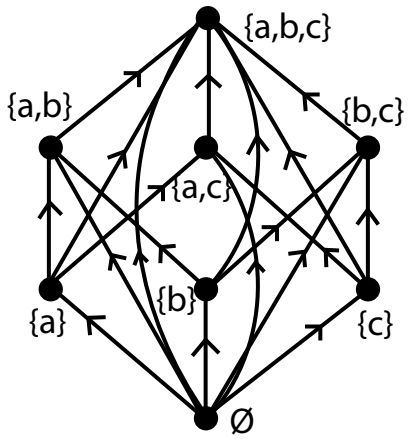
Discussion

The following steps could be used to get the Hasse diagram above.

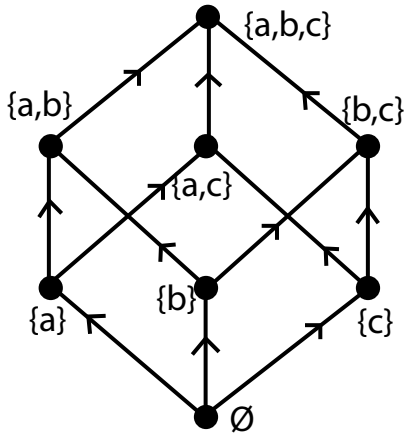
(1) You can see that even this relatively simple poset has a complicated digraph.



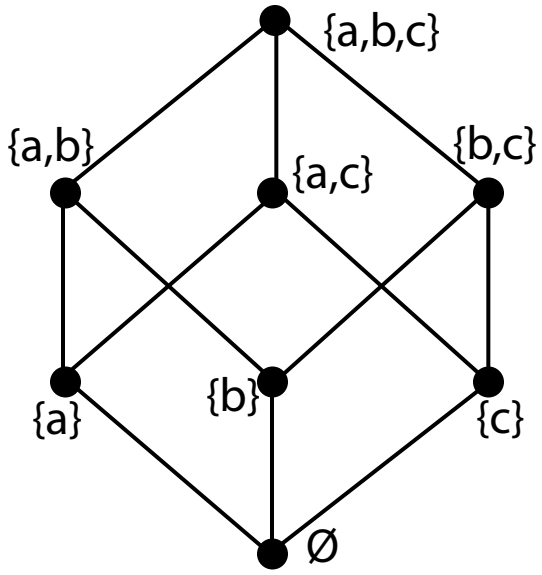
(2) Eliminate the loops.



(3) Now eliminate redundant arcs resulting from transitivity.



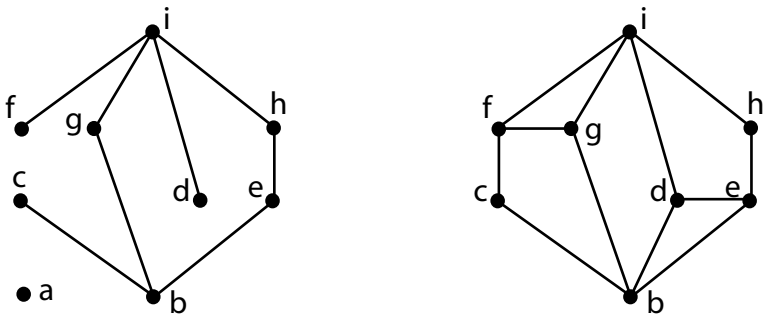
(4) Finally eliminate the arrows



EXERCISE 4.10.1. *Construct the Hasse diagram for the poset* $(\{1, 2, 3, 4, 6, 9, 12\}, |)$*, where* $|$ *is the "divides" relation.*

EXERCISE 4.10.2. *Is the diagram the Hasse diagram for a partial order? If so, give the partial order and if not explain why.*

(a) *(b)*



4.11. Maximal and Minimal Elements.

DEFINITION 4.11.1. *Let* (A, R) *be a poset.*

(1) *An element* $a \in A$ *is a* **minimal element** *if there does not exist an element* $b \in A$*,* $b \neq a$*, such that* $b \preceq a$*.*
(2) *An element* $a \in A$ *is a* **maximal element** *if there does not exist an element* $b \in A$*,* $b \neq a$*, such that* $a \preceq b$*.*

Discussion

Another useful way to characterize minimal elements of a poset $(A, \preceq)$ is to say that a is a minimal element of A iff $b \preceq a$ implies $b = a$. A similar characterization holds for maximal elements. It is possible for a poset to have more than one maximal and minimal element. In the poset in Exercise 4.10.1, for example, 1 is the only minimal element, but both 9 and 12 are maximal elements. These facts are easily observable from the Hasse diagram.

4.12. Least and Greatest Elements.

DEFINITION 4.12.1. *Let $(A, \preceq)$ be a poset.*

(1) An element $a \in A$ is the **least element** *of A if $a \preceq b$ for every element $b \in A$.*
(2) An element $a \in A$ is the **greatest element** *of A if $b \preceq a$ for every element $b \in A$.*

THEOREM 4.12.1. *A poset $(A, \preceq)$ can have at most one least element and at most one greatest element. That is, least and greatest elements are unique, if they exist.*

Discussion

We revisit the definition of greatest and least elements, which were defined in Section 4.4. As with minimal and maximal elements, Hasse diagrams can be helpful in illustrating least and greatest elements. Although a poset may have many minimal (or maximal) elements, Theorem 4.12.1 guarantees that it may have no more than one least (or greatest) element. We ask you to explore the relationship among these concepts in the following exercise.

EXERCISE 4.12.1. *Let $(A, \preceq)$ be a poset.*

(a) Prove that if a is the least element of A, then a is a minimal element of A.
(b) Prove that if b is the greatest element of A, then b is a maximal element of A.
(c) Prove that if A has more than one minimal element, then A does not have a least element.
(d) Prove that if A has more than one maximal element, then A does not have a greatest element.

4.13. Upper and Lower Bounds.

DEFINITION 4.13.1. *Let S be a subset of A in the poset $(A, \preceq)$.*

(1) If there exists an element $a \in A$ such that $s \preceq a$ for all $s \in S$, then a is called an **upper bound** *on S.*

(2) If there exists an element $a \in A$ such that $a \preceq s$ for all $s \in S$, then a is called an **lower bound** *on S.*

4.14. Least Upper and Greatest Lower Bounds.

DEFINITION 4.14.1. *Suppose $(A, \preceq)$ is a poset, S a is subset of A, and $a \in A$.*

(1) a is the **least upper bound** *of S if*
- *a is an upper bound of S and*
- *if s is another upper bound of S, then $a \preceq s$.*

(2) a is the **greatest lower bound** *of S if*
- *a is an lower bound of S and*
- *if s is another lower bound of S, then $s \preceq a$.*

Discussion

In Section 4.13 we extend the concepts upper and lower bound as well as least upper bound and greatest lower bound to subsets of a poset. The difference between a least element of a subset of A and a lower bound for the subset of A is that the least element is required to be *in* the subset and the lower bound is not. Here are a few facts about lower bounds and minimal elements to keep in mind. You should rephrase each statement replacing "lower" with "upper", etc.

- For a to be a lower bound for a subset S, a need not be in S, but it must precede every element of S.
- A minimal element, a, of a subset S is a lower bound for the set of all elements in S preceded by a.
- A subset may have more than one lower bound, or it may have none.
- A subset may have lower bounds, but no greatest lower bound.

EXAMPLE 4.14.1. *Suppose we are given the poset $(A, |)$, where $A = \{1, 2, 3, 4, 6, 8, 9, 12\}$.*

(1) The subset $\{2, 3, 4, 6\}$ has no greatest or least element.
(2) 1 is the greatest lower bound for $\{2, 3, 4, 6\}$ and 12 is its least upper bound.
(3) The subset $\{1, 2, 3, 8\}$ has no upper bound in A.
(4) Every subset of A has a greatest lower bound.

EXERCISE 4.14.1. *Consider the poset $(\mathcal{A}, \subseteq)$, where $\mathcal{A} = P(S)$ is the power set of a set S. Prove that every nonempty subset of $\mathcal{A}$ has a least upper bound and a greatest lower bound in $\mathcal{A}$.*

4.15. Lattices.

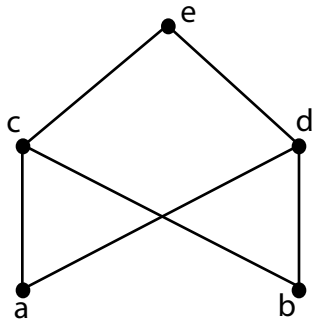
DEFINITION 4.15.1. *A poset $(A, \preceq)$ is a* **lattice** *if every pair of elements has a least upper bound and a greatest lower bound in A.*

Discussion

To check if a poset is a lattice you must check every pair of elements to see if they each have a greatest lower bound and least upper bound. If you draw its Hasse diagram, you can check to see whether some pair of elements has more than one upper (or lower) bound on the same level. If so, then the poset is not a lattice.

4.16. Example 4.16.1.

EXAMPLE 4.16.1. *The poset given by the following Hasse diagram is not a lattice.*



Discussion

In Example 4.16.1 notice that $\{a, b\}$ has upper bounds c, d and e. Since e is larger than either c or d, it cannot be the least upper bound. But c and d are not related in any way. Thus there is no least upper bound for the subset $\{a, b\}$.

EXERCISE 4.16.1. *Prove that the poset $(P(S), \subseteq)$ is a lattice.*

4.17. Topological Sorting.

DEFINITION 4.17.1.

(1) *A total ordering $\leq$ on a set A is said to be* **compatible** *with a partial ordering $\preceq$ on A, if $a \preceq b$ implies $a \leq b$ for all $a, b \in A$.*
(2) *A* **topological sorting** *is a process of constructing a compatible total order for a given partial order.*

Discussion

A topological sorting is a process of creating a total order from a partial order. Topological sorting has a number of applications. For example:

- It can be useful in PERT charts to determine an ordering of tasks.
- It can be useful in graphics to render objects from back to front to expose hidden surfaces.
- A painter often uses a topological sort when applying paint to a canvas. He/she paints parts of the scene furthest from the view first.

There may be several total orders that are compatible with a given partial order.

4.18. Topological Sorting Algorithm. procedure *topological sort* ($(A, \preceq)$: finite poset)
k:= 1
while $A \neq \emptyset$
begin
  a_k:= a minimal element of A
  A:= $A - \{a_k\}$
  k:= $k + 1$
end $\{a_1, a_2, ..., a_n$ is a compatible total ordering of $A\}$

Discussion

In order to justify the existence of a minimal element of S at each step in the topological sorting algorithm, we need to prove Theorem 4.19.1 in Section 4.19.

4.19. Existence of a Minimal Element.

THEOREM 4.19.1. *Suppose $(A, \preceq)$ is a finite nonempty poset. Then A has a minimal element.*

PROOF. Suppose $(A, \preceq)$ is a finite nonempty poset, where A has n elements, $n \geq 1$. We will prove that A has a minimal element by mathematical induction.

BASIS STEP. $n = 1$. Then $A = \{a\}$ and a is a minimal (least!) element of A.

INDUCTION STEP. Suppose that every poset having n elements has a minimal element. Suppose $(A, \preceq)$ is a poset having $n + 1$ elements. Let a be an arbitrary element of A, and let $S = A - \{a\}$. Then S, together with the partial order $\preceq$ restricted to S, is a poset with n elements. By the inductive hypothesis, S has a minimal element b. There are two possibilities.

(1) $a \preceq b$. Then a is a minimal element of A. Otherwise, there is an element c in A, different from a, such that $c \preceq a$. But then c is in S, c is different from b (why?), and $c \preceq b$, which contradicts the fact that b is a minimal element of S.

(2) $a \not\preceq b$. Suppose b is not a minimal element of A. Then there is a $c \in A$ with $c \preceq b$. Since $a \not\preceq b$ we have $c \neq a$. Thus $c \in S$ and we then conclude b is not minimal in S. This is a contradiction so there are no elements of A that preceed b. Hence b is a minimal element of A in this case.

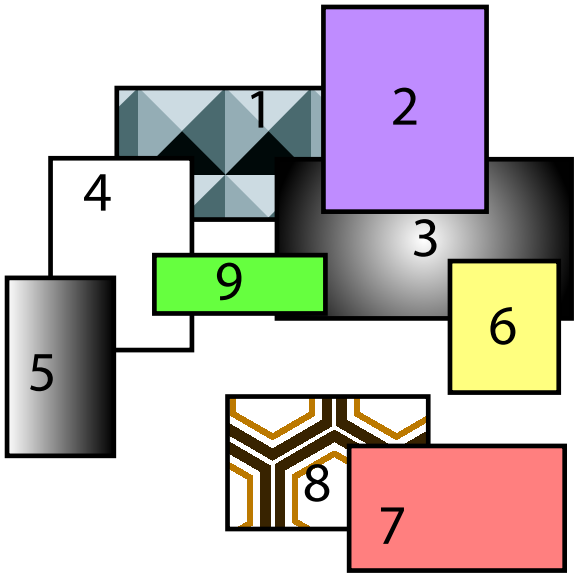
Thus, in any case A has a minimal element, and so by the principle of induction every finite poset has a minimal element. □

EXERCISE 4.19.1. *Let $(A, \preceq_A)$ be a poset and let $S \subseteq A$. Define $\preceq_S$ on S by*

$$\forall a, b \in S[a \preceq_S b \Leftrightarrow a \preceq_A b].$$

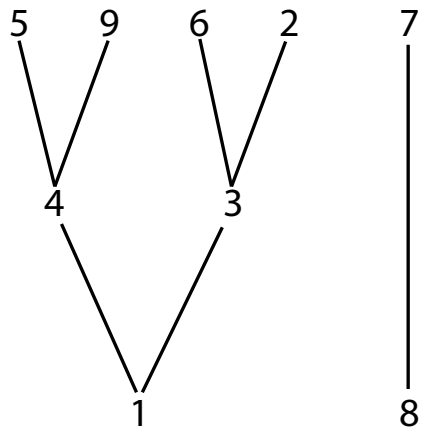
Prove $(S, \preceq_S)$ is a poset. The partial order $\preceq_S$ is the **Restriction** *of $\preceq_A$ to S and we usually use the same notation for both partial orders.*

EXAMPLE 4.19.1. *Consider the set of rectangles T and the relation R given by tRs if t is more distant than s from the viewer.*

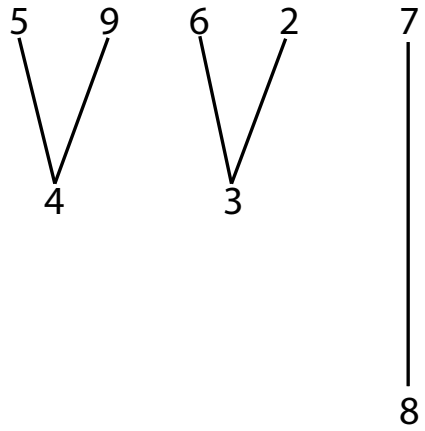


Here are some of the relations that we find from the figure: $1R2, 1R4, 1R3, 4R5, 3R2, 3R9, 3R6$.

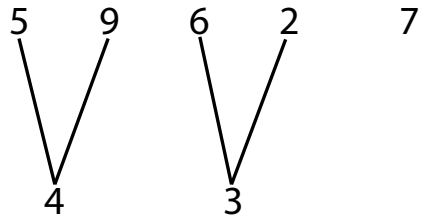
The Hasse diagram for R is



If we draw 1 (it would also be fine to use 8) out of the diagram and delete it we get



Then draw 8



and so on. By drawing minimal elements you may get the following total order (there are many other total orders compatible with the partial order):

CHAPTER 2

Graphs

1. Introduction to Graphs and Graph Isomorphism

1.1. The Graph Menagerie.

DEFINITION 1.1.1.

- *A* **simple graph** $G = (V, E)$ *consists of a set* V *of vertices and a set* E *of edges, represented by unordered pairs of elements of* V*.*
- *A* **multigraph** *consists of a set* V *of vertices, a set* E *of edges, and a function*

$$f : E \to \{\{u, v\} : u, v \in V \text{ and } u \neq v\}.$$

  If $e_1, e_2 \in E$ *are such that* $f(e_1) = f(e_2)$*, then we say* e_1 *and* e_2 *are* **multiple** *or* **parallel edges***.*
- *A* **pseudograph** *consists of a set* V *of vertices, a set* E *of edges, and a function* $f : E \to \{\{u, v\} : u, v \in V\}$*. If* $e \in E$ *is such that* $f(e) = \{u, u\} = \{u\}$*, then we say* e *is a* **loop***.*
- *A* **directed graph or digraph** $G = (V, E)$ *consists of a set* V *of vertices and a set* E *of directed edges, represented by* **ordered pairs** *of vertices.*

Discussion

In Section 1.1 we recall the definitions of the various types of graphs that were introduced in MAD 2104. In this section we will revisit some of the ways in which graphs can be represented and discuss in more detail the concept of a graph isomorphism.

1.2. Representing Graphs and Graph Isomorphism.

DEFINITION 1.2.1. *The* **adjacency matrix**, $A = [a_{ij}]$*, for a simple graph* $G = (V, E)$*, where* $V = \{v_1, v_2, ..., v_n\}$*, is defined by*

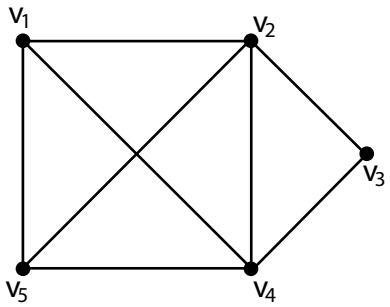
$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

Discussion

We introduce some alternate representations, which are extensions of connection matrices we have seen before, and learn to use them to help identify isomorphic graphs.

REMARKS 1.2.1. *Here are some properties of the adjacency matrix of an undirected graph.*

(1) The adjacency matrix is always symmetric.
(2) The vertices must be ordered: and the adjacency matrix depends on the order chosen.
(3) An adjacency matrix can be defined for multigraphs by defining a_{ij} to be the **number** *of edges between vertices i and j.*
(4) If there is a natural order on the set of vertices we will use that order unless otherwise indicated.



EXAMPLE 1.2.1. *An adjacency matrix for this graph is*

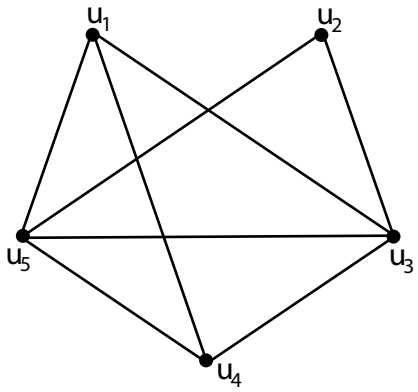
$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Discussion

As with connection matrices, an adjacency matrix can be constructed by using a table with the columns and rows labeled with the elements of the vertex set.

Here is another example

EXAMPLE 1.2.2. *The adjacency matrix for the graph*



is the matrix $M = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$

1.3. Incidence Matrices.

DEFINITION 1.3.1. *The* **incidence matrix**, $A = [a_{ij}]$, *for the undirected graph* $G = (V, E)$ *is defined by*

$$a_{ij} = \begin{cases} 1 & \text{if edge } j \text{ is incident with vertex } i \\ 0 & \text{otherwise.} \end{cases}$$

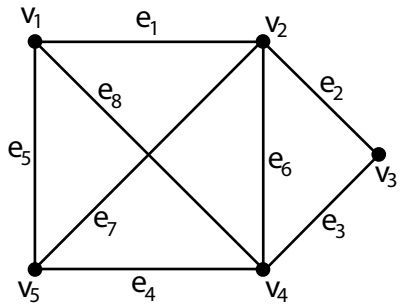
Discussion

The incidence matrix is another way to use matrices to represent a graph.

REMARKS 1.3.1.

(1) *This method requires the edges and vertices to be labeled and the matrix depends on the order in which they are written.*
(2) *Every column will have exactly two 1's.*
(3) *As with adjacency matrices, if there is a natural order for the vertices and edges that order will be used unless otherwise specified.*

1.4. Example 1.4.1.



EXAMPLE 1.4.1. *The incidence matrix for this graph is*

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

Discussion

Again you can use a table to get the matrix. List all the vertices as the labels for the rows and all the edges for the labels of the columns.

1.5. Degree.

DEFINITION 1.5.1.

(1) Let $G = (V, E)$ be an undirected graph.

- *Two vertices $u, v \in V$ are* **adjacent** *or* **neighbors** *if there is an edge e between u and v.*
  - *The edge e* **connects** *u and v.*
  - *The vertices u and v are* **endpoints** *of e.*
- *The* **degree** *of a vertex v, denoted $deg(v)$, is the number of edges for which it is an endpoint. A loop contributes twice in an undirected graph.*
  - *If $deg(v) = 0$, then v is called* **isolated**.
  - *If $deg(v) = 1$, then v is called* **pendant**.

(2) Let $G = (V, E)$ be a directed graph.

- *Let (u, v) be an edge in G. Then u is an* **initial vertex** *and is* **adjacent to** *v. The vertex v is a* **terminal vertex** *and is* **adjacent from** *u.*
- *The* **in degree** *of a vertex v, denoted $deg^-(v)$ is the number of edges which terminate at v.*

- *Similarly, the* **out degree** *of v, denoted $deg^+(v)$, is the number of edges which initiate at v.*

Discussion

We now recall from MAD 2104 the terminology we use with undirected and directed graphs. Notice that a loop contributes two to the degree of a vertex.

1.6. The Handshaking Theorem.

THEOREM 1.6.1 (The Handshaking Theorem). *Let $G = (V, E)$ be an undirected graph. Then*

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

PROOF. Each edge contributes twice to the sum of the degrees of all vertices. □

Discussion

The handshaking theorem is one of the most basic and useful combinatorial formulas associated to a graph. It lets us conclude some facts about the numbers of vertices and the possible degrees of the vertices. Notice the immediate corollary.

COROLLARY 1.6.1.1. *The sum of the degrees of the vertices in any graph must be an even number.*

In other words, it is impossible to create a graph so that the sum of the degrees of its vertices is odd (try it!).

1.7. Example 1.7.1.

EXAMPLE 1.7.1. *Suppose a graph has 5 vertices. Can each vertex have degree 3? degree 4?*

- *The sum of the degrees of the vertices would be $3 \cdot 5$ if the graph has 5 vertices of degree 3. This is an odd number, though, so this is not possible by the handshaking Theorem.*
- *The sum of the degrees of the vertices if there are 5 vertices with degree 4 is 20. Since this is even it is possible for this to equal $2|E|$.*

Discussion

If the sum of the degrees of the vertices is an even number then the handshaking theorem is not contradicted. In fact, you can create a graph with any even degree you want if multiple edges are permitted. However, if you add more restrictions it may not be possible. Here are two typical questions the handshaking theorem may help you answer.

EXERCISE 1.7.1. *Is it possible to have a graph S with 5 vertices, each with degree 4, and 8 edges?*

EXERCISE 1.7.2. *A graph with 21 edges has 7 vertices of degree 1, three of degree 2, seven of degree 3, and the rest of degree 4. How many vertices does it have?*

1.8. Theorem 1.8.1.

THEOREM 1.8.1. *Every graph has an even number of vertices of odd degree.*

PROOF. Let V_o be the set of vertices of odd degree, and let V_e be the set of vertices of even degree. Since $V = V_o \cup V_e$ and $V_o \cap V_e = \emptyset$, the handshaking theorem gives us

$$2|E| = \sum_{v \in V} deg(v) = \sum_{v \in V_o} deg(v) + \sum_{v \in V_e} deg(v)$$

or

$$\sum_{v \in V_o} deg(v) = 2|E| - \sum_{v \in V_e} deg(v).$$

Since the sum of any number of even integers is again an even integer, the right-hand-side of this equations is an even integer. So the left-hand-side, which is the sum of a collection of odd integers, must also be even. The only way this can happen, however, is for there to be an even number of odd integers in the collection. That is, the number of vertices in V_o must be even. □

Discussion

Theorem 1.8.1 goes a bit further than our initial corollary of the handshaking theorem. If you have a problem with the last sentence of the proof, consider the following facts:

- odd + odd = even
- odd + even = odd
- even + even = even

If we add up an odd number of odd numbers the previous facts will imply we get an odd number. Thus to get an even number out of $\sum_{v \in V_o} deg(v)$ there must be an even number of vertices in V_o (the set of vertices of odd degree).

While there must be an even number of vertices of odd degree, there is no restrictions on the parity (even or odd) of the number of vertices of even degree.

This theorem makes it easy to see, for example, that it is not possible to have a graph with 3 vertices each of degree 1 and no other vertices of odd degree.

1.9. Handshaking Theorem for Directed Graphs.

THEOREM 1.9.1. *For any directed graph $G = (E, V)$,*

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

Discussion

When considering directed graphs we differentiate between the number of edges going into a vertex verses the number of edges coming out from the vertex. These numbers are given by the **in degree** and the **out degree**.

Notice that each edge contributes one to the in degree of some vertex and one to the out degree of some vertex. This is essentially the proof of Theorem 1.9.1.

1.10. Graph Invariants. The following are invariants under isomorphism of a graph G:

(1) G has r vertices.
(2) G has s edges.
(3) G has degree sequence $(d_1, d_2, ..., d_n)$.
(4) G is a bipartite graph.
(5) G contains r complete graphs K_n (as a subgraphs).
(6) G contains r complete bipartite graphs $K_{m,n}$.
(7) G contains r n-cycles.
(8) G contains r n-wheels.
(9) G contains r n-cubes.

Discussion

Recall that two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there is a bijection

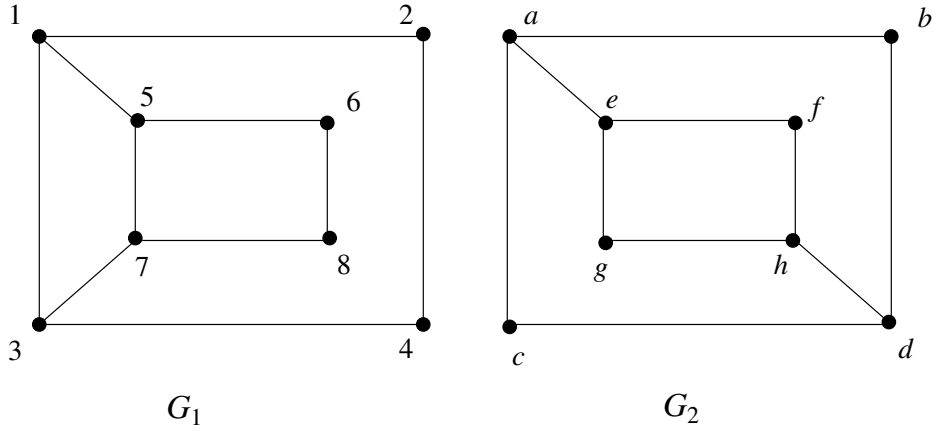
$$f : V_1 \to V_2$$

such that vertices u and v in V_1 are adjacent in G_1 if and only if $f(u)$ and $f(v)$ are adjacent in G_2. If there is such a function, we say f is an **isomorphism** and we write $G_1 \simeq G_2$.

It is often easier to determine when two graphs are *not* isomorphic. This is sometimes made possible by comparing *invariants* of the two graphs to see if they are different. We say a property of graphs is a **graph invariant** (or, just invariant) if, whenever a graph G has the property, any graph isomorphic to G also has the property. The **degree sequence** a graph G with n vertices is the sequence $(d_1, d_2, ..., d_n)$, where $d_1, d_2, ..., d_n$ are the degrees of the vertices of G and $d_1 \geq d_2 \geq \cdots \geq d_n$. Note that a graph could conceivably have infinitely many vertices. If the vertices are *countable* then the degree sequence would be an infinite sequence. If the vertices are not countable, then this degree sequence would not be defined.

The invariants in Section 1.10 may help us determine fairly quickly in some examples that two graphs are **not** isomorphic.

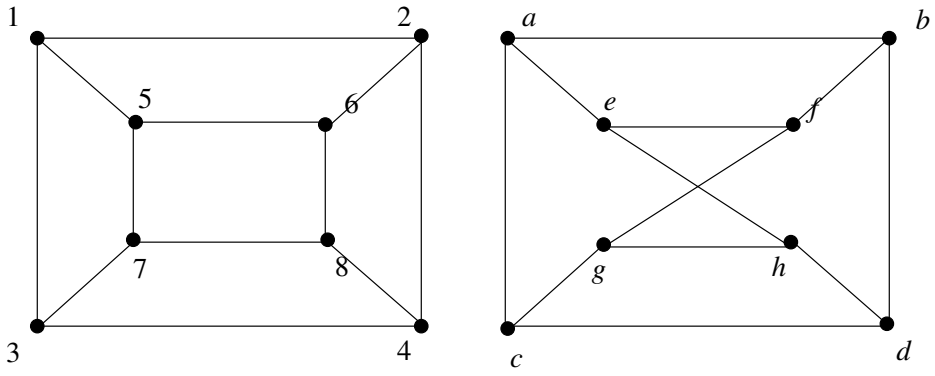
EXAMPLE 1.10.1. *Show that the following two graphs are not isomorphic.*



The two graphs have the same number of vertices, the same number of edges, and same degree sequences $(3, 3, 3, 3, 2, 2, 2, 2)$. *Perhaps the easiest way to see that they are not isomorphic is to observe that* G_1 *has three 4-cycles, whereas* G_2 *has two 4-cycles. In fact, the four vertices of* G_1 *of degree 3 lie in a 4-cycle in* G_1*, but the four vertices of* G_2 *of degree 3 do not. Either of these two discrepancies is enough to show that the graphs are not isomorphic.*

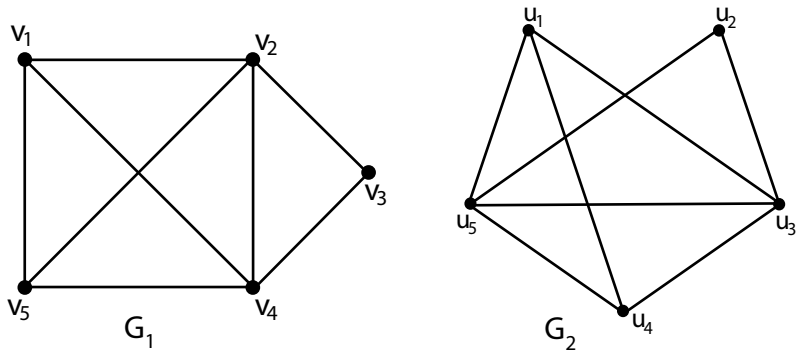
Another way we could recognize the graphs above are not isomorphic is to consider the adjacency relationships. Notice in G_1 *all the vertices of degree 3 are adjacent to 2 vertices of degree 3 and 1 of degree 2. However, in graph* G_2 *all of the vertices of degree 3 are adjacent to 1 vertex of degree 3 and 2 vertices of degree 2. This discrepancy indicates the two graphs cannot be isomorphic.*

EXAMPLE 1.10.2. *The following two graphs are not isomorphic. Can you find an invariant that is different on the graphs.*



1.11. Example 1.11.1.

EXAMPLE 1.11.1. *Determine whether the graphs G_1 and G_2 are isomorphic.*



Solution

We go through the following checklist that might tell us immediately if the two are *not* isomorphic.

- They have the same number of vertices, 5.
- They have the same number of edges, 8.
- They have the same degree sequence $(4, 4, 3, 3, 2)$.

Since there is no obvious reason to think they are not isomorphic, we try to construct an isomorphism, f.
Note that the above does *not* tell us there *is* an isomorphism, only that there might be one.

The only vertex on each that have degree 2 are v_3 and u_2, so we must have $f(v_3) = u_2$.

Now, since $deg(v_1) = deg(v_5) = deg(u_1) = deg(u_4)$, we must have either

- $f(v_1) = u_1$ and $f(v_5) = u_4$, or
- $f(v_1) = u_4$ and $f(v_5) = u_1$.

It is possible only one choice would work or both choices may work (or neither choice may work, which would tell us there is no isomorphism).

We try $f(v_1) = u_1$ and $f(v_5) = u_4$.

Similarly we have two choices with the remaining vertices and try $f(v_2) = u_3$ and $f(v_4) = u_5$. This defines a bijection from the vertices of G_1 to the vertices of G_2. We still need to check that adjacent vertices in G_1 are mapped to adjacent vertices in G_2. To check this we will look at the adjacency matrices.

The adjacency matrix for G_1 (when we list the vetices of G_1 by v_1, v_2, v_3, v_4, v_5) is

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

We create an adjacency matrix for G_2, using the bijection f as follows: since $f(v_1) = u_1$, $f(v_2) = u_3$, $f(v_3) = u_2$, $f(v_4) = u_5$, and $f(v_5) = u_4$, we rearrange the order of the vertices of G_2 to u_1, u_3, u_2, u_5, u_4. With this ordering, the adjacency matrix for G_2 is

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

Since $A = B$, adjacency is preserved under this bijection. Hence the graphs are isomorphic.

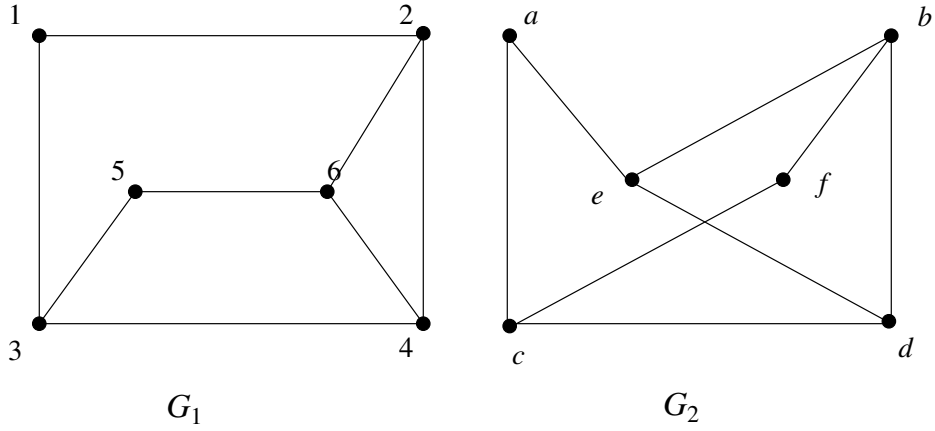
Discussion

In this example we show that two graphs are isomorphic. Notice that it is *not* enough to show they have the same number of vertices, edges, and degree sequence. In fact, if we knew they were isomorphic and we were asked to prove it, we would

proceed directly to try and find a bijection that preserves adjacency. That is, the check list is not necessary if you already know they are isomorphic. On the other hand, having found a bijection between two graphs that doesn't preserve adjacency doesn't tell us the graphs are not isomorphic, because some other bijection might work. If we go down this path, we would have to show that *every* bijection fails to preserve adjacency.

The advantage of the checklist is that it will give you a quick and easy way to show two graphs are *not* isomorphic if some invariant of the graphs turn out to be different. If you examine the logic, however, you will see that if two graphs have all of the same invariants we have listed so far, we still wouldn't have a proof that they are isomorphic. Indeed, there is no known list of invariants that can be efficiently checked to determine when two graphs are isomorphic. The best algorithms known to date for determining graph isomorphism have exponential complexity (in the number n of vertices).

EXERCISE 1.11.1. *Determine whether the following two graphs are isomorphic.*



EXERCISE 1.11.2. *How many different isomorphism (that is, bijections that preserve adjacencies) are possible from G_2 to itself in Example 1.10.1.*

EXERCISE 1.11.3. *There are 14 nonisomorphic pseudographs with 3 vertices and 3 edges. Draw all of them.*

EXERCISE 1.11.4. *Draw all nonisomorphic simple graphs with 6 vertices, 5 edges, and no cycles.*

EXERCISE 1.11.5. *Recall the equivalence relation on a set, $\mathbb{S}$, of graphs given by G_1 is related to G_2 if and only if $G_1 \simeq G_2$. How many equivalence classes are there if $\mathbb{S}$ is the set of all simple graphs with 6 vertices, 5 edges, and no cycles? Explain.*

1.12. Proof of Section 1.10 Part 3 for simple graphs.

PROOF. Let G_1 and G_2 be isomorphic simple graphs having degree sequences. By part 1 of Section 1.10 the degree sequences of G_1 and G_2 have the same number

of elements (finite or infinite). Let $f : V(G_1) \to V(G_2)$ be an isomorphism and let $v \in V(G_1)$. We claim $deg_{G_1}(v) = deg_{G_2}(f(v))$. If we show this, then f defines a bijection between the vertices of G_1 and G_2 that maps vertices to vertices of the same degree. This will imply the degree sequences are the same.

Proof of claim: Suppose $deg_{G_1}(v) = k$. Then there are k vertices adjacent to v, say $u_1, u_2, \ldots, u_k$. The isomorphism maps each of the vertices to k distinct vertices adjacent to $f(v)$ in G_2 since the isomorphism is a bijection and preserves adjacency. Moreover, $f(v)$ will not be adjacent to any vertices other than the k vertices $f(u_1), f(u_2), \ldots f(u_k)$. Otherwise, v would be adjacent to the preimage of such a vertex and this preimage would not be one of the vertices $u_1, u_2, \ldots, u_k$ since f is an isomorphism. This would contradict that the degree of v is k. This shows the degree of $f(v)$ in G_2 must be k as well, proving our claim.

□

EXERCISE 1.12.1. *Prove the remaining properties listed in Section 1.10 for simple graphs using only the properties listed before each and the definition of isomorphism.*

2. Connectivity

2.1. Connectivity.

Definition 2.1.1.

(1) *A* **path** *in a graph $G = (V, E)$ is a sequence of vertices $v_0, v_1, v_2, \ldots, v_n$ such that $\{v_{i-1}, v_i\}$ is an edge of G for $i = 1, ..., n$. The edge $\{v_{i-1}, v_i\}$ is an* **edge of the path**.
(2) *A path with n edges is said to have* **length** n.
(3) *A path beginning and ending with same vertex (that is, $v_0 = v_n$) is a* **circuit**.
(4) *A path is* **simple** *if no vertex or edge is repeated, with the possible exception that the first vertex is the same as the last.*
(5) *A simple path that begins and ends with the same vertex is a* **simple circuit** *or a* **cycle**.

Discussion

This section is devoted to defining what it means for a graph to be connected and the theorems about connectivity.

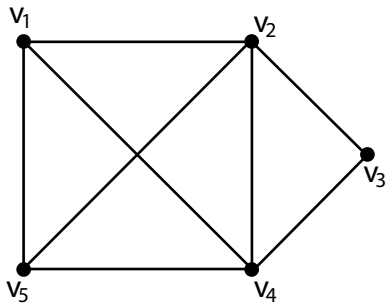
In the definition above we use a *vertex sequence* to define a path. We could also use an *edge sequence* to define a path as well. In fact, in a multigraph a path may not be well-defined by a vertex sequence. In this case an edge sequence must be used to clearly define a path.

A *circuit* must begin and end at the same vertex, but this is the only requirement for a circuit. A path that goes up one vertex and then right back is a circuit. Our definition of a simple path may be different than that found in some texts: some writers merely require that the same edge not be traversed more than once. In addition, our definition of a *simple* circuit does not include the circuit that goes up an edge and travels back by the same edge.

Some authors also allow the possibility of a path having length 0 (the path consists of a single vertex and no edges). We will require that paths have length at least 1. Notice that a path must also be finite. It is possible for a graph to have infinitely many vertices and/or edges and one could also imagine a kind of path with infinitely many edges but our definition of a path requires the path be finite.

2.2. Example 2.2.1.

Example 2.2.1. *Let G_1 be the graph below.*



(1) v_1, v_4, v_2, v_3 *is a simple path of length 3 from* v_1 *to* v_3.
(2) $\{v_1, v_4\}, \{v_4, v_2\}, \{v_2, v_3\}$ *is the edge sequence that describes the same path in part 1*
(3) $v_1, v_5, v_4, v_1, v_2, v_3$ *is a path of length 5 from* v_1 *to* v_3.
(4) v_1, v_5, v_4, v_1 *is a simple circuit of length 3.*
(5) $v_1, v_2, v_3, v_4, v_2, v_5, v_1$ *is a circuit of length 6, but it is not simple.*

Discussion

This example gives a variety of paths and circuits. You can certainly come up with many more.

EXERCISE 2.2.1. *In this exercise consider two cycles different if they begin at a different vertex and/or if they traverse vertices in a different direction. Explain your answer:*

(a) How many different cycles are there in the graph K_4*?*
(b) How many different circuits are there in the graph K_4*?*

EXERCISE 2.2.2. *In this exercise consider two cycles different if they begin at a different vertex and/or if they traverse vertices in a different direction. How many different cycles are there in the graph* K_n *where* n *is some integer greater than 2.*

EXERCISE 2.2.3. *(Uses combinations from counting principles) In this exercise consider two cycles are the same if they begin at a different vertex and/or if they traverse vertices in a different direction, but they use the same vertices and edges. Explain your answer:*

(a) How many different cycles are there in the graph K_4*?*
(b) How many different circuits are there in the graph K_4*?*

EXERCISE 2.2.4. *(Uses combinations from counting principles) In this exercise consider two cycles are the same if they begin at a different vertex and/or if they traverse vertices in a different direction, but they use the same vertices and edges. How many different cycles are there in the graph K_n where n is some integer greater than 2.*

EXERCISE 2.2.5. *Prove a finite graph with all vertices of degree at least 2 contains a cycle.*

EXERCISE 2.2.6. *Prove a graph with n vertices and at least n edges contains a cycle for all positive integers n. You may use Exercise 2.2.5.*

2.3. Connectedness.

DEFINITION 2.3.1. *A simple graph is* **connected** *if there is a path between every pair of distinct vertices.*

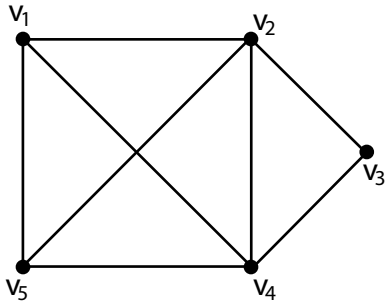
Discussion

When looking at a sketch of a graph just look to see if each vertex is connected to each of the other vertices by a path. If so, this graph would be connected. If it has two or more distinct pieces with no edge connecting them then it is *disconnected*.

2.4. Examples.

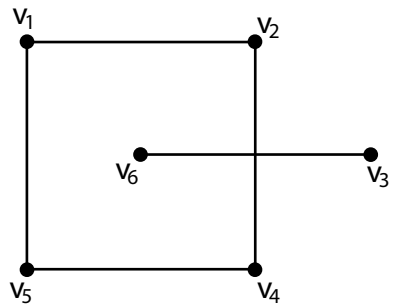
EXAMPLE 2.4.1.

This graph is connected

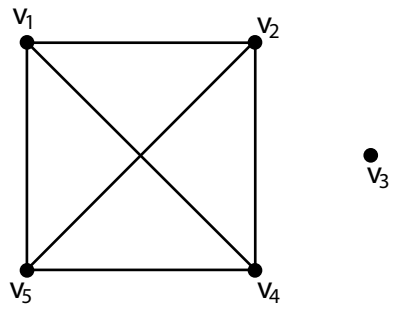


EXAMPLE 2.4.2.

This graph is not connected



EXAMPLE 2.4.3. *The following graph is also not connected. There is no edge between v_3 and any of the other vertices.*



2.5. Theorem 2.5.1.

THEOREM 2.5.1. *There is a simple path between every pair of distinct vertices in a connected graph.*

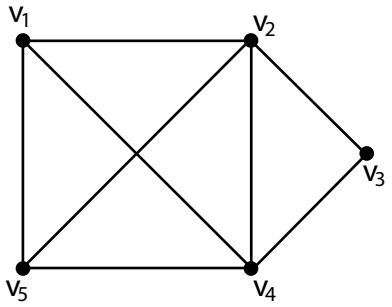
PROOF. Suppose u and v are arbitrary, distinct vertices in a connected graph, G. Because the graph is connected there is a path between u and v. Among all paths between u and v, choose a path $u = v_0, v_1, ..., v_n = v$ of shortest length. That is, there are no paths in G of length $< n$. Suppose this path contains a circuit starting and ending with, say, v_i. This circuit must use at least one edge of the path; hence, after removing the circuit we will have a path from u to v of length $< n$, contradicting the minimality of our initial path.

□

Discussion

Theorem 2.5.1 implies that if we need a path between two vertices in a connected graph we may use a simple path. This really simplifies (no pun intended) the types of paths we need to consider when examining properties of connected graphs. Certainly there are many paths that are not simple between any two vertices in a connected graph, but this theorem guarantees there are nicer paths to work with.

2.6. Example 2.6.1.



EXAMPLE 2.6.1. *The path $v_1, v_5, v_4, v_1, v_2, v_3$ is a path between v_1 and v_3. However, v_1, v_5, v_4, v_1 is a circuit. Remove the circuit (except the endpoint) to get from the original path v_1, v_2, v_3. This is still a path between v_1 and v_3, but this one is simple. There are no edges in this last path that are used more than one time.*

Discussion

In many examples it is possible to find more than one circuit that could be removed to create a simple path. Depending on which circuit is chosen there may be more than one simple path between two given vertices. Let us use the same graph in Example 2.6.1, but consider the path $v_1, v_2, v_5, v_1, v_4, v_2$. We could either remove the circuit v_1, v_2, v_5, v_1 or the circuit v_2, v_5, v_1, v_4, v_2. If we removed the first we would be left with v_1, v_4, v_2, while if we removed the latter we would get v_1, v_2. Both of these are parts of the original path between v_1 and v_2 that are simple.

2.7. Connected Component.

DEFINITION 2.7.1. *The maximally connected subgraphs of G are called the* **connected components** *or just the* **components**.

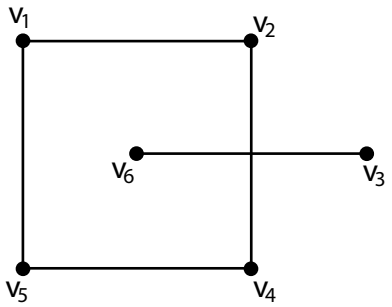
Discussion

Another way we could express the definition of a component of G is: A is a component of G if

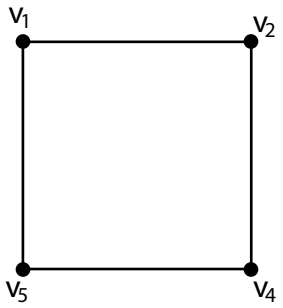
(1) A is a connected subgraph of G and
(2) if B is another subgraph of G containing A then either $B = A$ or B is disconnected.

2.8. Example 2.8.1.

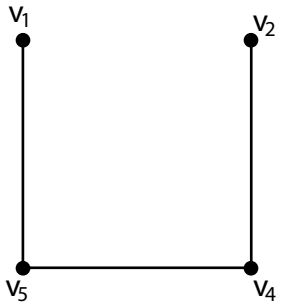
EXAMPLE 2.8.1. *In the graph below the vertices v_6 and v_3 are in one component while the vertices v_1, v_2, v_4, and v_5 are in the other component.*



If we looked at just one of the components and consider it as a graph by itself, it would be a connected graph. If we try to add any more from the original graph, however, we no longer have a connected graph. This is what we mean by "largest". Here are pictures that may help in understanding the components of the graph in Example 2.8.1



Above is a connected component of the original graph.



Above is *not* a connected component of the original. We are missing an edge that should have been in the component.

2.9. Cut Vertex and Edge.

Definition 2.9.1.

(1) If one can remove a vertex and all incident edges from a graph and produce a graph with more components than the original graph, then the vertex that was removed is called a **cut vertex** *or an* **articulation point**.

(2) If one can remove an edge from a graph and create more components than the original graph, then the edge that was removed is called a **cut edge** *or* **bridge**.

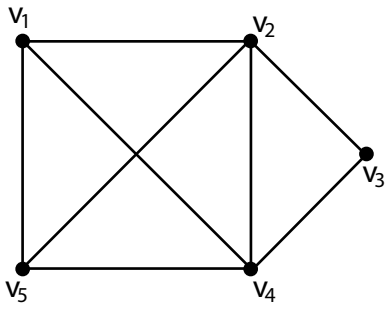
Note: When removing a vertex, you must remove all the edges with that vertex as an endpoint. When removing an edge we do *not* remove any of the vertices. Remember, edges depend on vertices, but vertices may stand alone.

EXERCISE 2.9.1. *Prove that every connected graph has at least two non-cut vertices. [Hint: Use the second principle of mathematical induction on the number of vertices.]*

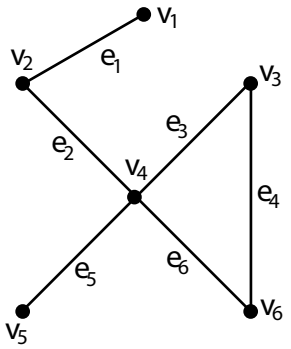
EXERCISE 2.9.2. *Prove that if a simple connected graph has exactly two non-cut vertices, then the graph is a simple path between these two non-cut vertices. [Hint: Use induction on the number of vertices and Exercise 2.9.1.]*

2.10. Examples.

EXAMPLE 2.10.1. *There are no cut vertices nor cut edges in the following graph.*



EXAMPLE 2.10.2. *v_2 and v_4 are cut vertices. e_1, e_2, and e_5 are cut edges in the following graph.*



Discussion

EXERCISE 2.10.1. *In each case, find how many cut edges and how many cut vertices there are for each integer n for which the graph is defined.*

(1) Star Network
(2) Cycle
(3) Complete Graphs.

2.11. Counting Edges.

THEOREM 2.11.1. *A connected graph with n vertices has at least $n-1$ edges.*

Discussion

Notice the Theorem states there are *at least* $n-1$ edges, not exactly $n-1$ edges. In a proof of this theorem we should be careful not to assume equality. Induction is the natural choice for a proof of this statement, but we need to be cautious of how we form the induction step.

Recall in the induction step we must show that a connected graph with $n+1$ vertices has at least n edges if we know every connected graph with n vertices has at least $n-1$ edges. It may seem like a good idea to begin with an arbitrary graph with n vertices and add a vertex and edge(s) to get one with $n+1$ vertices. However, the graph with $n+1$ vertices would depend on the one we started with. We want to make sure we have covered every possible connected graph with $n+1$ vertices, so we would have to prove *every* connected graph with $n+1$ vertices may be obtained this way to approach the proof this way. On the other hand, if we begin with an arbitrary graph with $n+1$ vertices and remove some vertex and adjacent edges to create a graph with n vertices the result may no longer be connected and we have to consider this possibility.

The proof of this theorem is a graded exercise.

2.12. Connectedness in Directed Graphs.

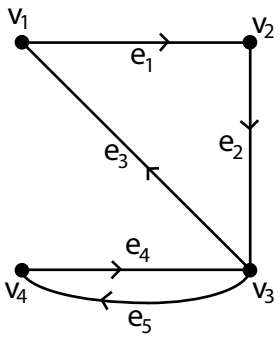
DEFINITION 2.12.1.
(1) A directed graph is **strongly connected** *if there is a directed path between every pair of vertices.*
(2) A directed graph is **weakly connected** *if the underlying undirected graph is connected.*

Discussion

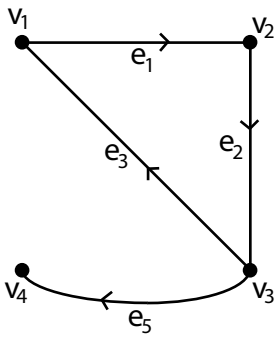
Recall that the underlying graph of a directed graph is the graph obtained by eliminating all the arrows. So the weakly connected means you can ignore the direction of the edges when looking for a path. Strongly directed means you must respect the direction when looking for a path between vertices. To relate this to something more familiar, if you are a pedestrian you do not have to worry about the direction of one way streets. This is not the case, however, if you are driving a car.

EXERCISE 2.12.1. *Are the following graphs strongly connected, weakly connected, both or neither?*

(a)



(b)



2.13. Paths and Isomorphism.

THEOREM 2.13.1. *Let M be the adjacency matrix for the graph G. Then the (i,j)th entry of M^r is the number of paths of length r from vertex i to vertex j, where M^r is the standard matrix product of M by itself r times (not the Boolean product).*

PROOF. The proof is by induction on the length of the path, r. Let p be the number of vertices in the graph (so the adjacency matrix is $p \times p$).

Basis: The adjacency matrix represents paths of length one by definition, so the basis step is true.

Induction Hypothesis: Assume each entry, say $m_{ij}^{[n]}$, in $M^n = [m_{ij}^{[n]}]$ equals the number of paths of length n from the i-th vertex to the j-th vertex.

Inductive Step: Prove each entry, say $m_{ij}^{[n+1]}$, in $M^{n+1} = [m_{ij}^{[n+1]}]$ equals the number of paths of length $n+1$ from the i-th vertex to the j-th vertex.

We begin by recalling $M^{n+1} = M^n \cdot M$ and by the definition of matrix multiplication the entry $m_{ij}^{[n+1]}$ in M^{n+1} is

$$m_{ij}^{[n+1]} = \sum_{k=1}^{p} m_{ik}^{[n]} \cdot m_{kj}$$

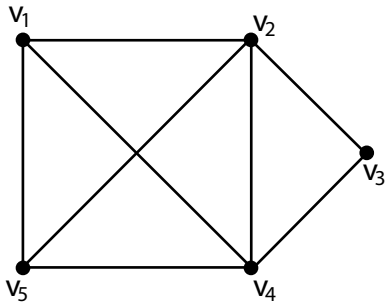
where $M^n = [m_{ij}^{[n]}]$ and $M = [m_{ij}]$.

By the induction hypothesis, $m_{ik}^{[n]}$ is the number of paths of length n between the i-th vertex and the k-th vertex, while m_{kj} is the number of paths of length 1 from the k-th vertex to the j-th vertex. Each of these paths may be combined to create paths of length $n+1$ from the i-th vertex to the j-th vertex. Using counting principles we see that the number of paths of length $n+1$ that go through the k-th vertex just before reaching the j-th vertex is $m_{ik}^{[n]} \cdot m_{kj}$.(1)

The above sum runs from $k = 1$ to $k = p$ which covers all the possible vertices in the graph. Therefore the sum counts all the paths of length $n+1$ from the i-th vertex to the j-the vertex.

□

2.14. Example 2.14.1.



Example 2.14.1. *The adjacency matrix for the graph above is*

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

We get the following powers of M:

$$M^2 = \begin{bmatrix} 3 & 2 & 2 & 2 & 2 \\ 2 & 4 & 1 & 3 & 2 \\ 2 & 1 & 2 & 1 & 2 \\ 2 & 3 & 1 & 4 & 2 \\ 2 & 2 & 2 & 2 & 3 \end{bmatrix}$$

$$M^3 = \begin{bmatrix} 6 & 9 & 4 & 9 & 7 \\ 9 & 8 & 7 & 9 & 9 \\ 4 & 7 & 2 & 7 & 4 \\ 9 & 9 & 7 & 8 & 9 \\ 7 & 9 & 4 & 9 & 6 \end{bmatrix}$$

The last matrix tells us there are 4 paths of length 3 between vertices v_3 and v_1. Find them and convince yourself there are no more.

Discussion

If you recall that the adjacency matrix and all its powers are symmetric, you will cut your work in half when computing powers of the matrix.

EXERCISE 2.14.1. *Find the page(s) in the text that covers the counting principal(s) used in the sentence referenced as* (1) *in the proof of Theorem 2.13.1. Explain how the conclusion of this gives us the result of the sentence.*

2.15. Theorem 2.15.1.

THEOREM 2.15.1. *If G is a disconnected graph, then the compliment of G, $\overline{G}$, is connected.*

Discussion

The usual approach prove a graph is connected is to choose two arbitrary vertices and show there is a path between them. For the Theorem 2.15.1 we need to consider the two cases where the vertices are in different components of G and where the vertices are in the same component of G.

EXERCISE 2.15.1. *Prove Theorem 2.15.1.*

EXERCISE 2.15.2. *The compliment of a connected graph may or may not be connected. Find two graphs such that the compliment is (a) connected and (b) disconnected.*

3. Euler and Hamilton Paths

3.1. Euler and Hamilton Paths.

DEFINITIONS 3.1.1.

(1) *An* **Euler Circuit** *in a graph G is a path in G that uses every edge exactly once and begins and ends at the same vertex.*
(2) *An* **Euler path** *in G is a path in G that uses every edge exactly once, but does not necessarily begin and end at the same vertex.*

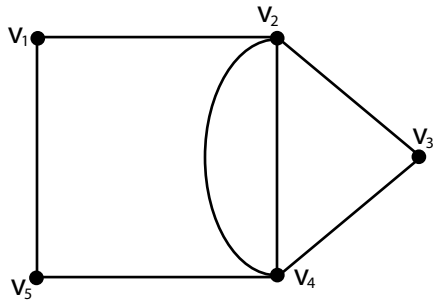
Discussion

There are several special types of paths in graphs that we will study in this section.

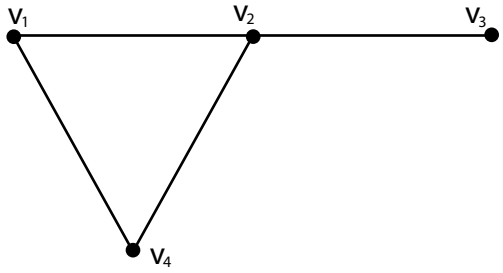
An Euler path or circuit should use every single *edge* exactly one time. The difference between and Euler path and Euler circuit is simply whether or not the path begins and ends at the same vertex. Remember a circuit begins and ends at the same vertex. If the graph is a directed graph then the path must use the edges in the direction given.

3.2. Examples.

EXAMPLE 3.2.1. *This graph has the Euler circuit (and hence Euler path) v_1, v_2, v_3, v_4, v_2, v_4, v_5, v_1.*



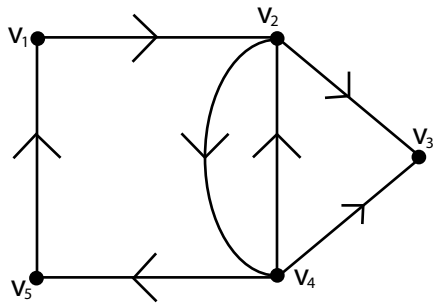
EXAMPLE 3.2.2. *This graph does not have an Euler circuit, but has the Euler path v_2, v_4, v_1, v_2, v_3.*



Discussion

Not all graphs have Euler circuits or Euler paths. See page 578, Example 1 G_2, in the text for an example of an undirected graph that has no Euler circuit nor Euler path.

In a directed graph it will be less likely to have an Euler path or circuit because you must travel in the correct direction. Consider, for example,



This graph has neither an Euler circuit nor an Euler path. It is impossible to cover both of the edges that travel to v_3.

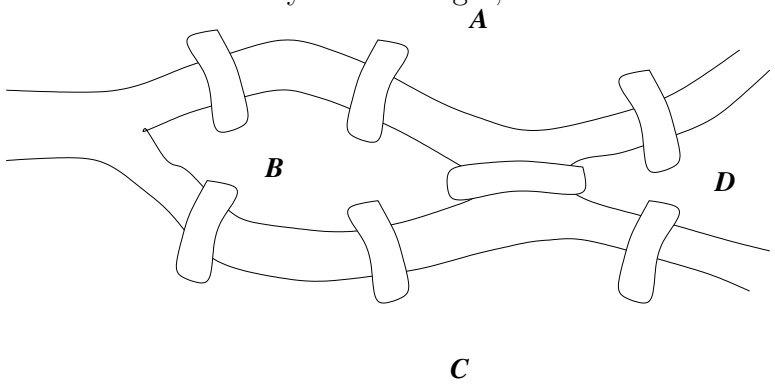
3.3. Necessary and Sufficient Conditions for an Euler Circuit.

THEOREM 3.3.1. *A connected, undirected multigraph has an Euler circuit if and only if each of its vertices has even degree.*

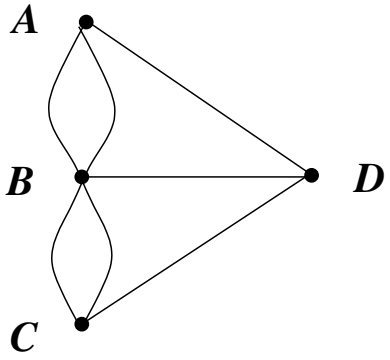
Discussion

This is a wonderful theorem which tells us an easy way to check if an undirected, connected graph has an Euler circuit or not. There is an extension for Euler paths which we will soon see.

This theorem allows us to solve the famous Königsberg problem: The town, once called Königsberg, Prussia, now Kaliningrad, Russia), was divided by the river Pregel into parts, which were connected by seven bridges, as illustrated below.



When the people of Königsberg would walk through the town, they wondered whether they could plan their walk so that they would cross each bridge exactly once and end up at their starting point. Leonhard Euler, a Swiss mathematician, solved the problem in the negative, by discovering and proving a theorem, which is essentially Theorem 3.3.1. The problem can be modelled by the multigraph below, and the solution depends upon whether the graph has an Euler circuit.



Proof of Theorem 3.3.1. Assume G is a connected, undirected multigraph with an Euler circuit. The degree of any given vertex may be counted by considering this circuit, since the circuit traverses every edge exactly once. While traveling the circuit we move into a vertex by one edge and leave by another edge, so there must be an even number of edges adjacent to each vertex.

Conversely, if the graph G is such that every edge has an even degree, then we can build an Euler circuit by the following algorithm: We begin at some arbitrary vertex and travel an edge out from that vertex to another. Then travel to another

vertex using an unused edge. Since each vertex has even degree there will always be an unused edge to travel out if we have traveled into the vertex until we reach the beginning vertex and have used all the edges.

□

Try your hand at the following exercise before you read further.

EXERCISE 3.3.1. *Is it possible for the people in Königsberg to plan a walk that crosses each bridge exactly once and ends up in a part of town different from where they started? (That is, is there an Euler path?) Either show that this is possible or explain why it is not.*

There is another algorithm one may use to find an Euler circuit given a graph with all vertices of even degree. The algorithm is written in pseudocode in the text, but the general idea is to start with some arbirtary circuit which we consider the "main" circuit. Now find a circuit that uses edges not used in the main circuit but begins and ends at some vertex in the main circuit. Insert this circuit into the main circuit. Repeat until all edges are used.

3.4. Necessary and Sufficient Conditions for an Euler Path.

THEOREM 3.4.1. *A connected, undirected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree.*

Discussion

Now you can determine precisely when a graph has an Euler path. If the graph has an Euler circuit, then it has an Euler path (why?). If it does not have an Euler circuit, then we check if there are exactly two vertices of odd degree.

PROOF OF THEOREM 3.4.1. Suppose G is a connected multigraph that does not have an Euler circuit. If G has an Euler path, we can make a new graph by adding on one edge that joins the endpoints of the Euler path. If we add this edge to the Euler path we get an Euler circuit. Thus there is an Euler circuit for our new graph. By the previous theorem, this implies every vertex in the new graph has even degree. However, this graph was obtained from G by adding the one edge between distinct vertices. This edge added one to the degrees of these two vertices. Thus in G these vertices must have odd degree and are the only vertices in G with odd degree.

Conversely, suppose G has exactly two vertices with odd degree. Again, add an edge joining the vertices with odd degree. The previous theorem tells us there is an Euler circuit. Since it is a circuit, we could consider the circuit as one which begins and ends at one of these vertices where the degree is odd in G. Now, remove the edge

we added earlier and we get G back and an Euler path in G.

□

3.5. Hamilton Circuits.

Definitions 3.5.1.

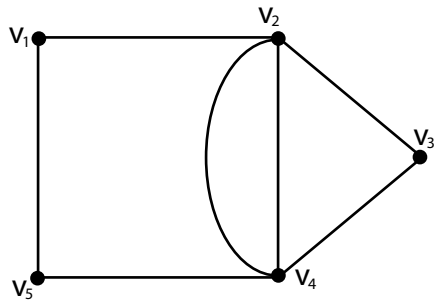
(1) *A* **Hamilton path** *is a path in a graph G that passes through every vertex exactly once.*
(2) *A* **Hamilton circuit** *is a Hamilton path that is also a circuit.*

Discussion

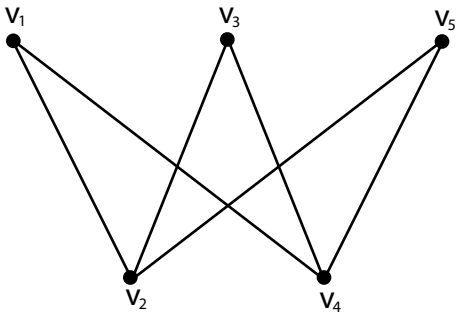
The difference between a Hamilton path and an Euler path is the Hamilton path must pass through each *vertex* exactly once and we do not worry about the edges, while an Euler path must pass through every *edge* exactly once and we do not worry about the vertices.

3.6. Examples.

Example 3.6.1. *The circuit $v_1, v_2, v_3, v_4, v_5, v_1$ is a Hamilton circuit (and so a path too).*



Example 3.6.2. *This graph has no Hamilton circuit, but v_1, v_2, v_3, v_4, v_5 is a Hamilton path.*



3.7. Sufficient Condition for a Hamilton Circuit.

THEOREM 3.7.1. *Let G be a connected, simple graph with N vertices, where $N \geq 3$. If the degree of each vertex is at least $n/2$, then G has a Hamilton circuit.*

Discussion

Unfortunately, there are no necessary and sufficient conditions to determine if a graph has a Hamilton circuit and/or path. Fortunately, there are theorems that give sufficient conditions for the existence of a Hamilton circuit. Theorem 3.7.1 above is just one example.

4. Introduction to Trees

4.1. Definition of a Tree.

DEFINITION 4.1.1. *A* **tree** *is a connected, undirected graph with no simple circuits.*

Discussion

For the rest of this chapter, unless specified, a *graph* will be understood to be undirected and simple. Recall that a simple circuit is also called a cycle. A graph is **acyclic** if it does not contain any cycles. A tree imposes two conditions on a (simple) graph: that is be connected and acyclic.

4.2. Examples.

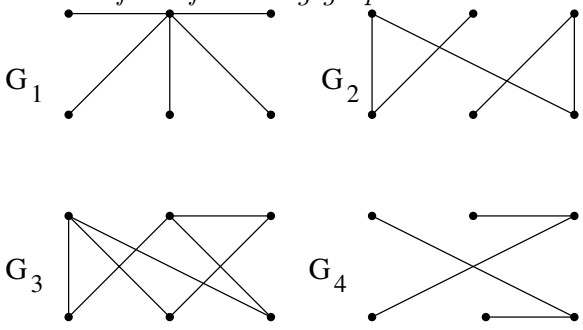
EXAMPLE 4.2.1. *The following are examples of trees*

- *Family tree*
- *File/directory tree*
- *Decision tree*
- *Organizational charts*

Discussion

You have most likely encountered examples of trees: your family tree, the directory or folder tree in your computer. You have likely encountered trees in other courses as well. When you covered counting principals and probability in precalculus you probably used trees to demonstrate the possible outcomes of some experiment such as a coin toss.

EXERCISE 4.2.1. *Which of the following graphs are trees?*



THEOREM 4.2.1. *A graph is a tree iff there is a unique simple path between any two of its vertices.*

PROOF. Suppose T is a tree and suppose u and v are distinct vertices in T. T is connected since it is a tree, and so there is a simple path between u and v. Suppose there are two different simple paths between u and v, say

$$P_1 : u = u_0, u_1, u_2, \ldots u_m = v$$

and

$$P_2 : u = v_0, v_1, v_2, \ldots v_n = v.$$

(Which type of proof do you think we are planning to use?)

Since the paths are different and since P_2 is a simple path, P_1 must contain an edge that isn't in P_2. Let $j \geq 1$ be the first index for which the edge $\{u_{j-1}, u_j\}$ of P_1 is not an edge of P_2. Then $u_{j-1} = v_{j-1}$. Let u_k be the first vertex in the path P_1 after u_{j-1} (that is, $k \geq j$) that is in the path P_2. Then $u_k = v_\ell$ for some $\ell \geq j$. We now have two simple paths, $Q_1 : u_{j-1}, ..., u_k$ using edges from P_1 and $Q_2 : v_{j-1}, ..., v_\ell$ using edges from P_2, between $u_{j-1} = v_{j-1}$ and $u_k = v_\ell$. The paths Q_1 and Q_2 have no vertices in common, other than the first and last, and no edges in common. Thus, the path from u_{j-1} to u_k along Q_1 followed by the path from v_ℓ to v_{j-1} along the reverse of Q_2 is a simple circuit in T, which contradicts the assumption that T is a tree. Thus, the path from u to v must be unique proving a tree has a unique path between any pair of vertices.

Conversely, assume G is not a tree.

(What kind of proof are we setting up for the reverse direction?)

Then either (a) G is not connected, so there is no path between some pair of vertices, or (b) G contains a simple circuit.

(a) Suppose G is not connected. Then there are two vertices u and v that can not be joined by a path, hence, by a simple path.
(b) Suppose G contains a simple circuit $C : v_0, ..., v_n$, where $v_0 = v_n$. If $n = 1$, then C would be a loop which is not possible since G is simple. Thus we have $n \geq 2$. But, since $n \geq 2$, $v_1 \neq v_0$, and so we have two different simple paths from v_0 to v_1: one containing the single edge $\{v_0, v_1\}$, and the other the part of the reverse of C from $v_n = v_0$ back to v_1.

Thus we have proved the statement "If a graph G is not a tree, then either there is no simple path between some pair of vertices of G or there is more than one simple path between some pair of vertices of G." This, is the contrapositive of the statement "If there is a unique simple path between any two vertices of a graph G, then G is a tree."

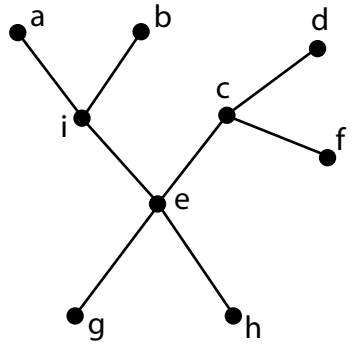
□

4.3. Roots.

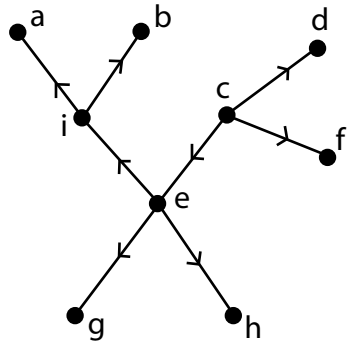
DEFINITION 4.3.1. *A* **rooted tree** *is a tree T together with a particular vertex designated as it* **root**. *Any vertex may a priori serve as the root. A rooted tree provides each edge with a direction by traveling away from the root.*

4.4. Example 4.4.1.

EXAMPLE 4.4.1. *Consider the tree below.*



We choose c to be the root. Then we have the directed tree:



Discussion

A rooted tree has a natural direction on its edges: given the root, v, an edge $e = \{x, y\}$ lies in the unique simple path from either v to x or from v to y, but *not both*. Say e is in the simple path from v to y. Then we can direct e from x to y.

4.5. Isomorphism of Directed Graphs.

DEFINITION 4.5.1. *Let G and H be a directed graphs. G and H are* **isomorphic** *if there is a bijection $f : V(G) \to V(H)$ such that (u, v) is an edge in G if and only if $(f(u), f(v))$ is an edge in H. We call the map f and isomorphism and write $G \simeq H$*

Discussion

Notice the notation for *ordered pairs* is used for the edges in a directed graph and that the isomorphism must preserve the direction of the edges.

EXERCISE 4.5.1. *Let $\mathcal{G}$ be a set of directed graphs. Prove that isomorphism of directed graphs defines an equivalence relation on $\mathcal{G}$.*

4.6. Isomorphism of Rooted Trees.

DEFINITION 4.6.1. *Rooted trees T_1 and T_2 are* **isomorphic** *if they are isomorphic as directed graphs.*

Discussion

EXERCISE 4.6.1. *Give an example of rooted trees that are isomorphic as (undirected) simple graphs, but not isomorphic as rooted trees.*

EXERCISE 4.6.2. *How many different isomorphism types are there of (a) trees with four vertices? (b) rooted trees with four vertices?*

4.7. Terminology for Rooted Trees.

DEFINITION 4.7.1.

1. *If $e = (u, v)$ is a directed edge then u is the* **parent** *of v and v is the* **child** *of u. The root has no parent and some of the vertices do not have a child.*
2. *The vertices with no children are called the* **leaves** *of the (rooted) tree.*
3. *If two vertices have the same parent, they are* **siblings**.
4. *If there is a directed, simple path from u to v then u is an* **ancestor** *of v and v is a* **descendant** *of u.*
5. *Vertices that have children are* **internal vertices**.

Discussion

Much of the terminology you see on this slide comes directly from a family tree. There are a few exceptions. For example, on a family tree you probably would not say cousin Freddy, who has never had kids, is a "leaf."

4.8. m-ary Tree.

DEFINITION 4.8.1.
1. *An m-**ary tree** is one in which every internal vertex has no more than m children.*
2. *A **full** m-**ary tree** is a tree in which every internal vertex has exactly m children.*

Discussion

Notice the distinction between an m-ary tree and a full m-ary tree. The first may have fewer than m children off of some internal vertex, but a latter must have exactly m children off of each internal vertex. A full m-ary tree is always an m-ary tree. More generally, if $k \leq m$ then every k-ary tree is also an m-ary tree. Caution: terminology regarding m-ary trees differs among authors.

4.9. Counting the Elements in a Tree.

THEOREM 4.9.1. *A tree with n vertices has $n - 1$ edges.*

THEOREM 4.9.2.
1. *A full m-ary tree with i internal vertices has $n = mi + 1$ vertices.*
2. *If a full m-ary tree has n vertices, i internal vertices, and L leaves then*
   (a) $i = (n - 1)/m$
   (b) $L = [n(m - 1) + 1]/m$
   (c) $L = i(m - 1) + 1$
   (d) $n = (mL - 1)/(m - 1)$
   (e) $i = (L - 1)/(m - 1)$

Discussion

PROOF OF THEOREM 4.9.1. Select a root v and direct the edges away from v. Then there are as many edges as there are terminal vertices of the edges and every vertex except v is a terminal vertex of some edge. □

PROOF OF THEOREM 4.9.2 PART 1. Every internal vertex has m children; hence, there are mi children. Only the root is not counted, since it is the only vertex that is not a child. □

EXERCISE 4.9.1. *Prove Theorem 4.9.2 Part 2 [Hint: What is $L + i$?]*

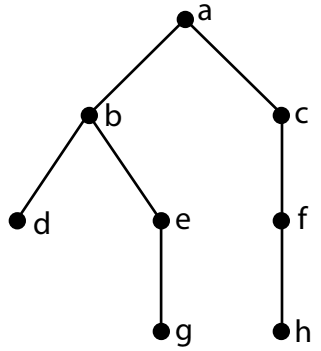
4.10. Level.

Definition 4.10.1.

1. *The* **level** *of a vertex in a rooted tree is the length of the shortest path to the root. The root has level 0.*
2. *The* **height** *of a rooted tree is the maximal level of any vertex.*
3. *A rooted tree of height h is* **balanced** *if each leaf has level h or $h-1$.*

Discussion

Example 4.10.1. *Consider the rooted tree below*



If a is the root of the tree above, then

- *the level of b and c is 1.*
- *The level of g and h is 3.*
- *The height of the tree is 3.*
- *This is also a balanced tree since the level of every leaf is either 2 or 3.*

4.11. Number of Leaves.

Theorem 4.11.1. *An m-ary tree of height h has at most m^h leaves.*

Corollary 4.11.1.1.

1. *If T is an m-ary tree with L leaves and height h, then*
$$h \geq \lceil \log_m L \rceil.$$
2. *If T is full and balanced, then*
$$h = \lceil \log_m L \rceil.$$

Discussion

Notice the *at most* in Theorem 4.11.1.

PROOF OF THEOREM 4.11.1. We prove the theorem by induction on the height h, $h \geq 0$.

Basis: If the height of the tree is 0, then the tree consists of a single vertex, which is also a leaf, and $m^0 = 1$.

Induction hypothesis: Assume any m-ary tree of height h has at most m^h leaves for some positive integer m.

Induction step: Prove that an m-ary tree of height $h + 1$ has at most m^{h+1} leaves.

Let T be an m-ary tree of height $h + 1$, $h \geq 0$. Remove *all* the leaves of T to get a tree T'. T' is an m-ary tree of height h, and so by the induction hypothesis it has at most m^h leaves. We recover T from T' by adding back the deleted edges, and there are at most m edges added to each leaf of T'. Thus, T has at most $m \cdot m^h = m^{h+1}$ leaves, since no leaf of T' is a leaf of T.

By the principle of mathematical induction any m-ary tree with height h has at most m^h leaves for any positive integers m and h.

□

EXERCISE 4.11.1. *Prove Corollary 4.11.1.1.*

4.12. Characterizations of a Tree.

THEOREM 4.12.1. *Let G be a graph with at least two vertices. Then the following statements are equivalent.*

1. *G is a tree*
2. *For each pair of distinct vertices in G, there is a unique simple path between the vertices.*
3. *G is connected, but if one edge is removed the resulting graph is disconnected.*
4. *G is acyclic, but if an edge is added between two vertices in G the resulting graph contains a cycle.*
5. *G is connected and the number of edges is one less than the number of vertices.*
6. *G is acyclic and the number of edges is one less than the number of vertices.*

Discussion

Theorem 4.12.1 gives us many different tools for recognizing trees. Once we prove this Theorem, it is enough to prove a graph satisfies any one of the statements to show the graph is a tree. Equivalently we could show a graph fails to satisfy any of the conditions to show it is not a tree.

The equivalence 1⇔2 is actually Theorem 4.2.1 which we have already proven. The equivalence 1⇔3 will be part of your graded assignment.

PROOF 1⇔4. First we show 1⇒4.

Let G be a tree with at least two vertices. By the definition of a tree G is acyclic, so what we must show is if an edge is added between two vertices of G then resulting graph contains a cycle. Let u and v be two vertices in G and add an edge, e, between these vertices. Let G' be the resulting graph. Note that G' is not necessarily simple.

By part 2 of this Theorem there must be a unique simple path between v and u in G. Let $e_1, e_2, e_3, \ldots, e_k$ be the edge sequence defining this path. The edge sequence $e_1, e_2, e_3, \ldots, e_k, e$ will define a path from v to itself in G'. Moreover, this is a simple circuit since $e_1, e_2, e_3, \ldots, e_k$ defined a simple path in G and e is not an edge in G. This shows G' contains a cycle.

Now we show 4⇒1.

Let G be an acyclic simple graph that satisfies the property given in part 4 of Theorem 4.12.1. Since we already know G is acyclic, what we need to show to complete the proof that G is a tree is that it is connected. To show a graph is connected we show two arbitrary vertices are connected by a path in G.

Let u and v be vertices in G. Add an edge, $e = \{u, v\}$, to G and call the resulting graph G'. By our assumption, there must be a cycle in G' now. In fact, the edge e must be part of the cycle because without this edge we would not have a cycle. Suppose $e, e_1, e_2, \ldots, e_k$ is the cycle that begins and ends at u. Notice that k must be at least 1 for this edges sequence to define a cycle. Moreover, the edge sequence $e_1, e_2, \ldots, e_k$ defines a path between v and u in G since all of these edges must be in G. This shows G is connected and so is a tree. □

EXERCISE 4.12.1. *Prove 1⇔5 in Theorem 4.12.1.*

EXERCISE 4.12.2. *Prove 1⇔6 in Theorem 4.12.1.*

5. Spanning Trees

5.1. Spanning Trees.

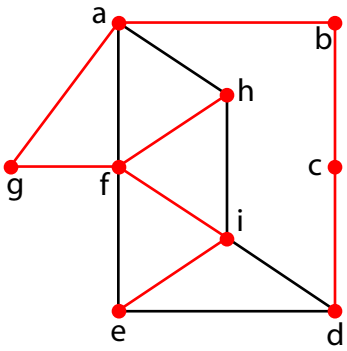
DEFINITION 5.1.1. *Given a connected graph G, a connected subgraph that is both a tree and contains all the vertices of G is called a* **spanning tree** *for G.*

Discussion

Given a connected graph G, one is often interested in constructing a connected subgraph, which contains all of the vertices of G, but has as few the edges of G as possible. For example, one might wish to find a minimal graph in a connected circuit in which each pair of nodes is connected. Such a subgraph will be a spanning tree for G. As we will soon see, if G is not already a tree, then it will have more than one spanning tree.

5.2. Example 5.2.1.

EXAMPLE 5.2.1. *In the figure below we have a graph drawn in red and black and a spanning tree of the graph in red.*



Discussion

The spanning tree of a graph is *not* unique.

EXERCISE 5.2.1. *Find at least two other spanning trees of the graph given in Example 5.2.1.*

5.3. Example 5.3.1.

EXAMPLE 5.3.1. *K_5 has 3 nonisomorphic spanning trees.*

Discussion

The three nonisomorphic spanning trees would have the following characteristics. One would have 3 vertices of degree 2 and 2 of degree 1, another spanning tree would have one vertex of degree three, and the third spanning tree would have one vertex of degree four. There *are* more than 3 spanning trees, but any other will be isomorphic to one of these three.

EXERCISE 5.3.1. *How many nonisomorphic (unrooted) spanning trees are there of the graph in Example 5.2.1?*

5.4. Existence.

THEOREM 5.4.1. *A graph is connected if and only if it has a spanning tree.*

Discussion

One of the key points of Theorem 5.4.1 is that any connected graph has a spanning tree. A spanning tree of a connected graph can be found by removing any edge that forms a simple circuit and continuing until no such edge exists. This algorithm turns out to be very inefficient, however, since it would be time-consuming to look for circuits each time we wish to consider removing an edge. There are two recursive algorithms, the depth-first search and the breadth-first search algorithms, that are fairly efficient, but we will not discuss them here, since they will be covered in depth in your computer science courses.

PROOF OF THEOREM 5.4.1. First we let G be a graph that contains a spanning tree, T. Let u and v be vertices in G. Since T is a spanning tree of G, u and v are also vertices in T. Now, T is a tree, so it is connected. Thus there is a path from u to v in T. T is, by definition of a spanning tree, a subgraph of G, so the path in T from u to v is also a path in G. Since u and v were arbitrary vertices in G we see that G is connected.

Conversely, we assume G is a connected graph. Notice that if G is not simple we may remove all loops and all but one edge between any pair of vertices that has more than one edge between them and the resulting graph will be a simple connected graph. Thus we may assume without loss of generality that G is simple.

Let $n = |V(G)|$. If G has fewer than $n-1$ edges then it would not be connected as a result of an exercise in *Connectivity*, thus $|E(G)| \geq n-1$. Also from *Introduction*

to Trees we recall that G is a tree if and only if $|E(G)| = n - 1$, so we assume G has at least $n - 1$ edges, say $|E(G)| = (n - 1) + k$.

We will prove that a connected graph with n vertices and $(n-1)+k$ edges contains a spanning tree by induction on k, $k \geq 0$.

Basis Step: $k = 0$. Then G is already a tree by our observation above.

Induction Step: Assume that every connected graph with n vertices and $(n - 1) + k$ edges, $k \geq 0$, contains a spanning tree. Suppose G is a connected graph with n vertices and $(n - 1) + (k + 1)$ edges. Since $|E(G)| > n - 1$, G is not a tree so there must be a simple circuit in G. Removing any edge from this circuit will result in a connected subgraph, G_1, of G with the same vertex set and $n - 1 + k$ edges. By our induction hypothesis, G_1 contains a spanning tree. But since G_1 is a subgraph of G containing all of the vertices of G, any spanning tree of G_1 is a spanning tree of G. Thus, G contains a spanning tree.

Therefore, by the principle of mathematical induction, every simple connected graph contains a spanning tree.

□

This proof can be used as a basis of a recursive algorithm for constructing spanning trees. It is, however, nothing more than the inefficient algorithm we alluded to above.

COROLLARY 5.4.1.1. *A connected subgraph of G that has the minimum number of edges and still contains all the vertices of G must be a spanning tree for G. Moreover, a spanning tree of a graph with n vertices must have exactly $n - 1$ edges.*

5.5. Spanning Forest.

DEFINITION 5.5.1. *A* **spanning forest** *of a graph G is the union of a collection of one spanning tree from each connected component of G.*

THEOREM 5.5.1. *Every finite simple graph has a spanning forest.*

Discussion

A graph that is not connected does not have a spanning tree. It does, however, have a spanning forest.

EXERCISE 5.5.1. *Prove Theorem 5.5.1.*

5.6. Distance.

DEFINITION 5.6.1. *Let T_1 and T_2 be spanning trees of the graph G. The* **distance**, *$d(T_1, T_2)$, between T_1 and T_2 is the number of edges in $E(T_1) \oplus E(T_2)$. That is,*

$$d(T_1, T_2) = |E(T_1) \oplus E(T_2)|.$$

Discussion

Recall the notation $\oplus$ from sets is the symmetric difference of two sets: $A \oplus B = A \cup B - A \cap B$.

Consider the spanning tree in Example 5.2.1. It is possible to find another spanning tree of the given graph by removing one edge from the given spanning tree and adding an edge not already in use. (See Exercise 5.6.4 below.) The distance between this new spanning tree and the old one is 2, since there is 1 edge in the first that is not in the second and one edge in the second that is not in the first.

EXERCISE 5.6.1. *What is the parity of the distance between any two spanning trees of a graph G? Explain. (Parity is even or odd).*

EXERCISE 5.6.2. *Prove that if A, B, and C are arbitrary finite sets, then*

$$|A \cap C| \geq |A \cap B| + |B \cap C| - |B|.$$

[Hint: If X and Y are finites sets, $|X \cup Y| = |X| + |Y| - |X \cap Y|$.]

EXERCISE 5.6.3. *This one is in your homework. Do not post the solution on the discussion board.*

Prove that if T_1, T_2, and T_3 are spanning trees of a simple connected graph G then

$$d(T_1, T_3) \leq d(T_1, T_2) + d(T_2, T_3).$$

[Hint: First, reduce the inequality to the one in Exercise 5.6.2.]

EXERCISE 5.6.4. *Suppose T and T' are spanning trees of a simple connected graph G and $T \neq T'$. Prove that there is an edge e in T that is not in T' and an edge e' in T' that is not in T such that if we remove e from T and then add e' to $T - e$, the resulting subgraph T'' of G is a spanning tree. [Hint: First show that $T \cup T'$ is a connected subgraph of G that is not a tree.]*

EXERCISE 5.6.5. *In Exercise 5.6.4 what is the relationship between $d(T', T'')$ and $d(T, T')$?*

EXERCISE 5.6.6. *Prove that if T and T' are spanning trees of a simple connected graph G and $T \neq T'$, then there are spanning trees $T_0, T_1, T_2, ..., T_k$ of G, $k \geq 1$, such that*

(a) $T_1 = T$,

(b) $T_k = T'$, *and*
(c) $d(T_{i-1}, T_i) = 2$, *for* $i = 1, ..., k$.

[Hint: Use induction on n, *where* $d(T, T') = 2n$, *and Exercise 5.6.4.]*

EXERCISE 5.6.7. *Prove that if* $T_0, T_1, T_2, ..., T_k$ *are are spanning trees of a simple connected graph* G, $k \geq 0$, *such that* $d(T_{i-1}, T_i) = 2$, *for* $i = 1, ..., k$, *then* $d(T_0, T_k) \leq 2k$.

6. Search and Decision Trees

6.1. Binary Tree.

DEFINITION 6.1.1. *A* **binary tree** *is a rooted 2-ary tree. In a binary tree a child of a parent may be designated as a* **left child** *or a* **right child**, *but each parent has at most one of each.*

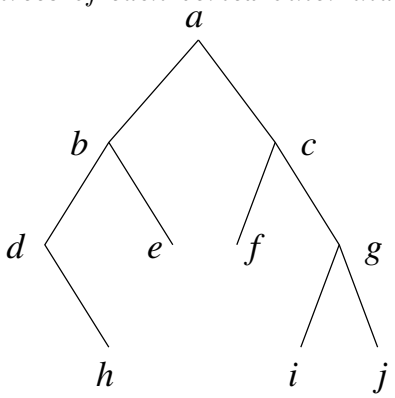
The rooted subtree consisting of the right (left) child of a vertex and all of its descendants is the **right** *(***left***)* **subtree** *at that vertex.*

A binary tree is often called a **binary search tree**.

Discussion

A binary tree is a rooted tree in which we may impose additional structure; namely, the designation of each child of a vertex as either a left or right child. This designation may be fairly arbitrary in general, although it may be natural in a particular application. A tree could be drawn with the left and right subtrees of a vertex reversed, so it may be unclear which is the left and right without a sketch of the tree or a clear designation from the beginning. If a tree has been sketched in the plane with the root at the top and the children of a vertex below the vertex, then the designation of left or right child should be inferred naturally from the sketch.

EXERCISE 6.1.1. *For the binary tree below, identify left and right children and sketch the left and right subtrees of each vertex other than the leaves.*



The tree shown above could have been drawn with the vertices b and c (and their subtrees) reversed. The resulting tree would be isomorphic to the original as rooted trees, but the isomorphism would not preserve the additional structure. The point is that the left and right descendents are not preserved under an isomorphism of rooted trees.

Searching items in a list can often be accomplished with the aid of a binary search tree. For example, suppose we are given a list of elements, $X_1, X_2, ..., X_n$, from an ordered set $(S, <)$, but the elements are not necessarily listed according to their preferred ordering. We can establish a recursive procedure for constructing a binary search tree with vertices labeled or *keyed* by the elements of the list as demonstrated by Example 6.2.1. This tree will allow us to search efficiently for any particular item in the list.

6.2. Example 6.2.1.

EXAMPLE 6.2.1. *Suppose $X_1, X_2, ..., X_n$ are elements from an ordered set $(S, <)$. Form a binary search tree recursively as follows.*

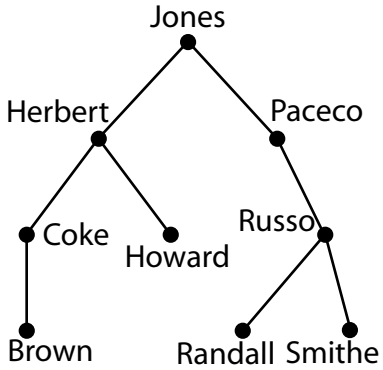
(1) Basis: Let X_1 be the label or key for the root.
(2) Recursion: Assume we have constructed a binary search tree with vertices keyed by $X_1, ..., X_i$, $1 \leq i < n$. Starting with the root, keyed X_1, compare X_{i+1} with the keys of the vertices already in the tree, moving to the left if the vertex key is greater than X_{i+1} and to the right otherwise. We eventually reach a leaf with key X_j for some j between 1 and i. We add a vertex with key X_{i+1} and edge (X_j, X_{i+1}) and designate X_{i+1} to be either a left child of X_j if $X_{i+1} < X_j$ or a right child of X_j if $X_j < X_{i+1}$.

Discussion

The main characteristic of the binary search tree constructed is that the key of any vertex is greater than the key of any vertex in its left subtree and is less than the key of any vertex in its right subtree.

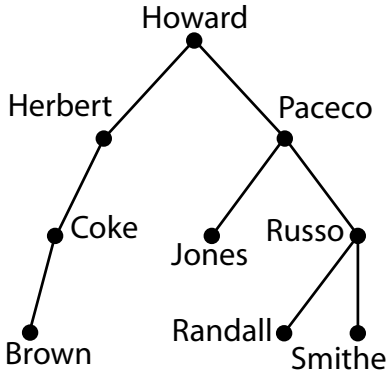
EXAMPLE 6.2.2. *Construct a binary search tree with vertices keyed by the names in the list {Jones, Paceco, Hebert, Howard, Russo, Coke, Brown, Smithe, Randall} using alphabetical order.*

Solution:



Discussion

Example 6.2.2 is one in which we have assigned the vertices keys from a list of names, ordered alphabetically. Notice that if we rearrange the list of names, say {Howard, Paceco, Jones, Hebert, Russo, Coke, Brown, Randall, Smithe}, we might get a different tree:



6.3. Decision Tree.

Definition 6.3.1. *A* **decision tree** *is a rooted tree in which each internal vertex corresponds to a decision and the leaves correspond to the possible outcomes determined by a sequence of decisions (a path).*

Discussion

Decision trees may be used to determine the complexity of a problem or an algorithm. Notice that a decision tree need *not* be a binary tree.

6.4. Example 6.4.1.

Example 6.4.1 (Has been featured as a puzzler on *Car Talk*). *Suppose you are given a collection of identical looking coins and told one of them is counterfeit. The counterfeit coin does not weigh the same as the real coins. You may or may not be told the counterfeit coin is heavier or lighter. The only tool you have to determine which is counterfeit is a balance scale. What is the fewest number of weighings required to find the counterfeit coin? Your answer does depend on the number of coins you are given and whether or not you are told the counterfeit is heavier or lighter than the rest.*

Solution

The solution will be obtained by a sequence of weighings as follows:

- Choose two subsets of the coins of equal number and compare them on the balance.
- There are three possibilities: one of the two sets of coins weighs more than, less than, or is the same as the other set of coins.
- Depending on the outcome of the first weighing, choose another two subsets of the coins and compare them.
- This problem can be modeled with a ternary (3-ary) tree where an internal vertex corresponds to a weighing, and edge corresponds to an outcome of that weighing, and a leaf corresponds to a coin found to be counterfeit.

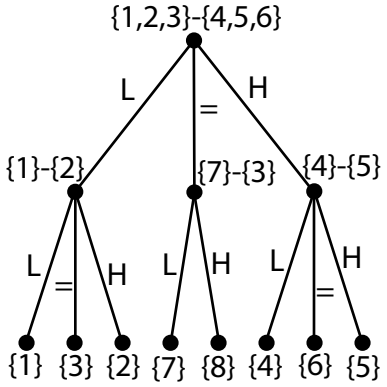
In order to determine the minimal number of weighings for a given problem, you must be clever in choosing the sets of coins to weigh in order not to require any redundant comparisons.

Discussion

Just think, you could have won a t-shirt from *Car Talk* if you had known about this!

EXAMPLE 6.4.2. *Suppose there are 8 coins,* $\{1, 2, 3, 4, 5, 6, 7, 8\}$*, and you know the counterfeit coin is lighter than the rest.*

Solution: Use a ternary tree to indicate the possible weighings and their outcomes. A vertex labeled with the notation $\{a, ..., b\} - \{x, ..., y\}$ *stands for the act of comparing the set of coins* $\{a, ..., b\}$ *to* $\{x, ..., y\}$*. An edge from that vertex will have one of the labels* L*,* $=$*, or* H*, depending on whether the first of the two sets is lighter than, equal in weight to, or heavier than the second set.*



With careful choices of the sets we see that only two weighings are necessary.

Notice there is a leaf for every possible outcome, that is, one for each coin. It is not difficult, but perhaps tedious, to see that we cannot get by with only one weighing. You would have to argue cases. This problem was made a bit easier since we knew the counterfeit coin is lighter.

EXERCISE 6.4.1. *In the example above, how many weighings are necessary if you don't know whether the counterfeit is lighter or heaver? [Notice that in this case there would be 16 leaves.]*

7. Tree Traversal

7.1. Ordered Trees.

DEFINITION 7.1.1. *An* **ordered tree** *is a rooted tree in which the set of children of each vertex is assigned a total order. In a drawing of an ordered rooted tree, with the root shown at the top, the children of a vertex are shown from left to right.*

Discussion

The concept of an ordered tree in some way generalizes the idea of a binary tree in which children of a vertex have been designated as left or right, if we think of the left child of a vertex as being "less than" the right child. The analogy only breaks down for binary trees that are not complete, however, since some vertex may have only a left child, whereas another may only have a right child.

7.2. Universal Address System.

DEFINITION 7.2.1. *The set of vertices of an ordered tree T can be provided with a total order, called a* **universal address system**, *using the following recursive process.*

Basis: *Label the root 0, and label the n children of the root $1, 2, ..., n$ from left to right.*

Recursion: *Given a vertex v with label L at level $k \geq 1$, label its children*

$$L.1, L.2, ..., L.n_v$$

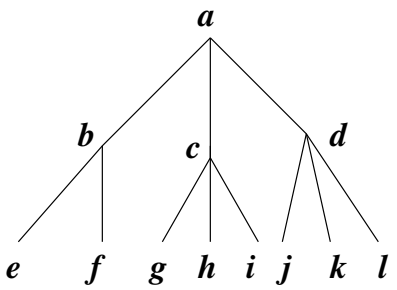
from left to right.

Totally order the vertices of T using the lexicographic ordering.

Discussion

EXAMPLE 7.2.1. *The following table gives the universal address system and the resulting ordering of the vertices for the ordered tree shown below.*

a	$<$	b	$<$	e	$<$	f	$<$	c	$<$	g		
0	$<$	1	$<$	1.1	$<$	1.2	$<$	2	$<$	2.1		
g	$<$	h	$<$	i	$<$	d	$<$	j	$<$	k	$<$	ℓ
2.1	$<$	2.2	$<$	2.3	$<$	3	$<$	3.1	$<$	3.2	$<$	3.3



7.3. Tree Traversal. Suppose we have information stored in an ordered rooted tree. How do we recover information from the tree? That is, how do we visit the vertices in the tree? We shall look at several procedures for visiting, or listing, the vertices of the tree. Each procedure is defined recursively on the subtrees, and each is based on a path that proceeds to the leftmost child of a vertex that has not occurred in the path before moving to a vertex to the right. These algorithms only differ as to *when* the root of a subtree is visited (or listed) relative to the vertices of its subtrees.

DEFINITION 7.3.1. *A procedure used to systematically visit each vertex in a tree is called a* **traversal algorithm***, or a* **traversal***.*

Notice that a subtree of T that does not include the root must have fewer vertices. Therefore, the recursive definition makes sense. Just keep applying the recursion step until you get to the leaves. Once you are down to a subtree that consists only of a leaf apply the basis step.

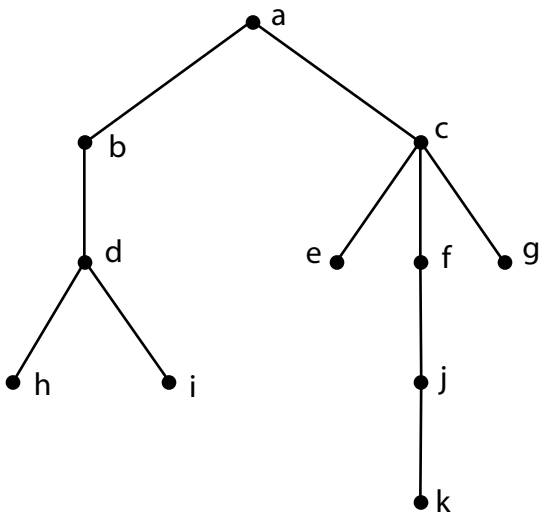
7.4. Preorder Traversal.

DEFINITION 7.4.1. *The* **preorder traversal** *of a rooted tree, T, with n vertices is defined recursively as follows:*

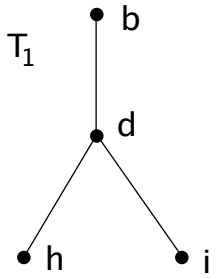
Basis: *If $n = 1$, then the root is the only vertex, so we visit the root.*
Recursive Step: *When $n > 1$ consider the subtrees, $T_1, T_2, T_3, \ldots, T_k$ of T whose roots are all the children of the root of T. Visit each of these subtrees from left to right.*

Discussion

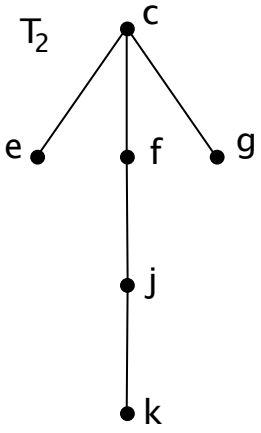
EXAMPLE 7.4.1. *In the Preorder Traversal of the the vertices in the following tree the vertices are visited in the following order: a, b, d, h, i, c, e, f, j, k, g.*



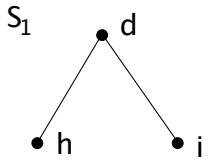
To begin finding the preorder traversal we begin with a. Next find the preorder traversal for the subtree



List that traversal and then find and list the prerorder traversal for the subtree



Now to find the preorder traversal of the subtree, T_1 we start with b and find the preorder traversal of



The preorder traversal of this one is *d, h, i.* The basis step was finally used to find the preorder traversal of S_1's subtrees.

The preorder traversal of T_2 still needs to be found. It is *c, e, f, j, k, g.* By putting all the vertices together in the order in which they were listed we get the preorder traversal *a, b, d, h, i, c, e, f, j, k, g.*

One recommendation is to take the original graph and point to each vertex in the order listed in the preorder traversal to help coordinate the order they are given in the list to their place in the tree.

Notice that if a tree is ordered using the universal address system, then a listing of the vertices in "increasing" order is a preorder traversal of the tree.

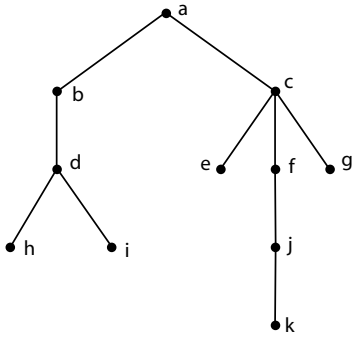
7.5. Inorder Traversal.

DEFINITION 7.5.1. *In an* **inorder traversal** *of a tree the root of a tree or subtree is visited after the vertices of the leftmost subtree, but before the vertices of all other subtrees.*

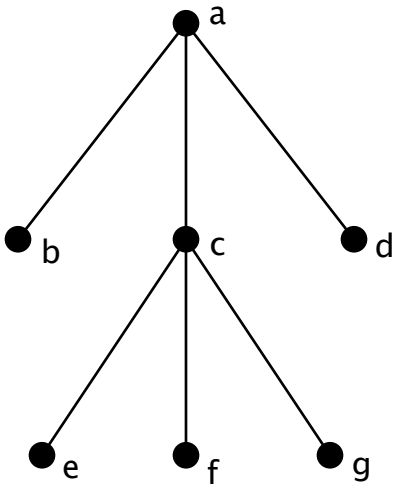
Note that in the inorder traversal, if a vertex, v, has multiple subtrees originating from it v is placed between the vertices of the leftmost subtree and the vertices of all the other subtrees.

EXERCISE 7.5.1. *Give a careful, recursive definition of an inorder traversal.*

EXAMPLE 7.5.1. *The inorder traversal of this tree below is h, d, i, b, a, e, c, k, j, f, g.*



EXAMPLE 7.5.2. *Here is another example to look at. Again, point to each vertex in the order it is listed to develop an understanding of the relationship between inorder traversal and the graph.*

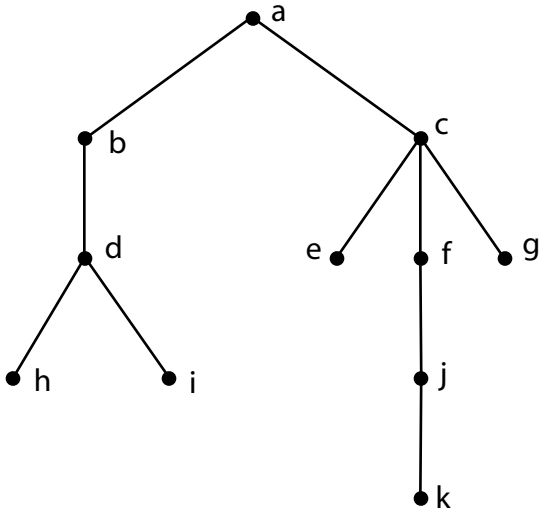


The inorder traversal is b, a, e, c, f, g, d.

7.6. Postorder Traversal.

DEFINITION 7.6.1. *In a* **postorder traversal** *of a tree, the root of a tree/subtree is visited after all of the vertices of its subtrees are visited.*

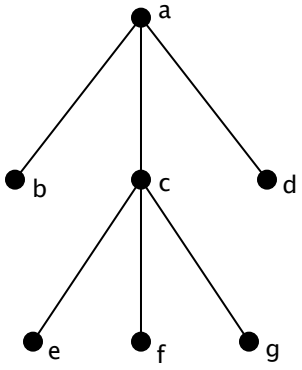
EXAMPLE 7.6.1. *The postorder traversal of the tree below is h, i, d, b, e, k, j, f, g, c, a.*



Discussion

The *postorder traversal* is when the root of a subtree is visited after all its vertices have been visited.

EXAMPLE 7.6.2. *We can find the postorder traversal of the tree*



Postorder Traversal: b, e, f, g, c, d, a.

EXERCISE 7.6.1. *Give a careful, recursive definition of an postorder traversal.*

7.7. Infix Form.

DEFINITIONS 7.7.1.

(1) *A fully parenthesized expression is in* **infix form**.
(2) *The expression obtained by traversing the ordered rooted tree by prefix traversal is* **prefix form** *or* **Polish notation**.
(3) *The expression obtained by traversing the ordered rooted tree by postfix traversal is* **postfix form** *or* **reverse Polish notation**.

EXAMPLE 7.7.1.

$$3(x-y)+\frac{(x+y)^2}{4}$$

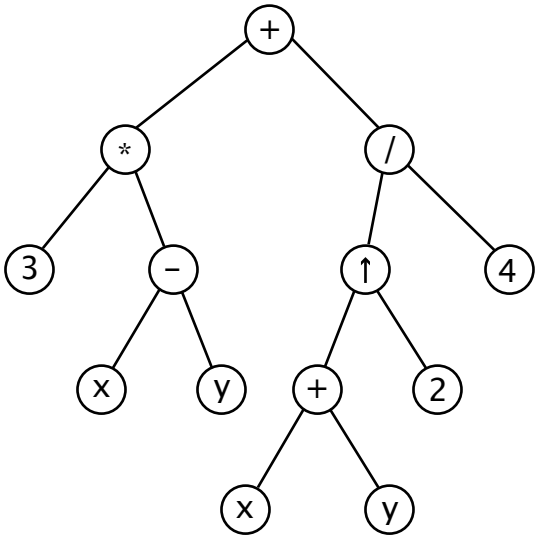
(1) *Find the infix form.*
(2) *Draw the rooted tree for the algebraic expression.*
(3) *Find the prefix form.*
(4) *Find the postfix form.*

Solutions

(1) *The infix form is*

$$(3*(x-y))+(((x+y)\uparrow 2)/4)$$

(2) The rooted tree for the algebraic expression is



(3) The prefix form is

$$+ \quad * \quad 3 \quad - \quad x \quad y \quad / \quad \uparrow \quad + \quad x \quad y \quad 2 \quad 4$$

(4) The postfix form is

$$3 \quad x \quad y \quad - \quad * \quad x \quad y \quad + \quad 2 \quad \uparrow \quad 4 \quad / \quad +$$

Discussion

An algebraic representation consists of a finite number of

- variables and numbers
- binary operations: addition $+$, subtraction $-$, multiplication $*$, division $/$, exponentiation $\uparrow$.

Pay attention to the fact that the symbol $\uparrow$ is used for exponentiation in this material. For example, $2^3 = 2 \uparrow 3$.

In these notes we define the infix, prefix, and postfix forms of an algebraic expression and give an example. The tree referred to in example 7.7.1 is found as follows.

A binary ordered tree can be built having internal vertices labeled by the operators and leaves labeled by the variables and numbers. We start from the innermost expressions and work our way outward constructing subtrees from the innermost expressions. These are joined together to form larger subtrees using the operations that join the innermost expressions. The *inorder* listing of the vertices then reproduces the

expression provided the parentheses are inserted as follows: as you begin to traverse a subtree from an internal vertex (operation) insert an open parenthesis; as you leave the subtree insert a closed parenthesis.

Prefix and postfix notation can be found from the tree. Notice that the prefix and postfix notations do *not* require parenthesis. A algebraic expression in prefix or postfix notation is unambiguous. Infix form requires parentheses, however, in order to resolve ambiguities. Despite this fact, infix notation is pretty much the universally accepted form for writing algebraic expressions. We have adopted a convention, so well known that we take it for granted, that allows us to reduce, but not eliminate, the number of parentheses needed without creating ambiguities. Namely, we agree that, in the absence of parentheses, the binary operations are performed in the following order: exponentiation, multiplication, division, addition, subtraction.

EXERCISE 7.7.1. *Find the prefix and postfix forms for the algebraic expressions* $((a * b) + (c/(d \uparrow 3)))$ *and* $(a * (((b + c)/d) \uparrow 3))$.

EXAMPLE 7.7.2. *Find the infix form of the expression given in prefix form.*

$$- \quad \uparrow \quad + \quad x \quad y \quad 5 \quad / \quad * \quad 2 \quad + \quad x \quad y \quad 3$$

Solution

(1) $- \quad \uparrow \quad + \quad x \quad y \quad 5 \quad / \quad * \quad 2 \quad (+ \quad x \quad y) \quad 3$
(2) $- \quad \uparrow \quad + \quad x \quad y \quad 5 \quad / \quad * \quad 2 \quad (x + y) \quad 3$
(3) $- \quad \uparrow \quad + \quad x \quad y \quad 5 \quad / \quad (* \quad 2 \quad (x + y)) \quad 3$
(4) $- \quad \uparrow \quad + \quad x \quad y \quad 5 \quad / \quad (2 * (x + y)) \quad 3$
(5) $- \quad \uparrow \quad + \quad x \quad y \quad 5 \quad (/ \quad (2 * (x + y)) \quad 3)$
(6) $- \quad \uparrow \quad + \quad x \quad y \quad 5 \quad ((2 * (x + y))/3)$
(7) $- \quad \uparrow \quad (+ \quad x \quad y) \quad 5 \quad ((2 * (x + y))/3)$
(8) $- \quad \uparrow \quad (x + y) \quad 5 \quad ((2 * (x + y))/3)$
(9) $- \quad (\uparrow \quad (x + y) \quad 5) \quad ((2 * (x + y))/3)$
(10) $- \quad ((x + y) \uparrow 5) \quad ((2 * (x + y))/3)$
(11) $(- \quad ((x + y) \uparrow 5) \quad ((2 * (x + y))/3))$
(12) $(((x + y) \uparrow 5) - ((2 * (x + y))/3))$

To go from prefix form to infix form, we read the expression right from left. Look for the rightmost operation and the variables or numbers immediately to the right of it. Put parenthesis around these and change to infix notation. Move to the next right most operation and put parenthesis around it and the expressions just to the right of it. Change to infix notation, and so on.

EXAMPLE 7.7.3. *Find the value of the postfix expression*

$$6 \quad 2 \quad \uparrow \quad 9 \quad / \quad 7 \quad 2 \quad 3 \quad \uparrow \quad + \quad 5 \quad / \quad +$$

Solution

(1) $(6 \quad 2 \quad \uparrow) \quad 9 \quad / \quad 7 \quad 2 \quad 3 \quad \uparrow \quad + \quad 5 \quad / \quad +$
(2) $(6^2) \quad 9 \quad / \quad 7 \quad 2 \quad 3 \quad \uparrow \quad + \quad 5 \quad / \quad +$
(3) $36 \quad 9 \quad / \quad 7 \quad 2 \quad 3 \quad \uparrow \quad + \quad 5 \quad / \quad +$
(4) $(36 \quad 9 \quad /) \quad 7 \quad 2 \quad 3 \quad \uparrow \quad + \quad 5 \quad / \quad +$
(5) $(36/9) \quad 7 \quad 2 \quad 3 \quad \uparrow \quad + \quad 5 \quad / \quad +$
(6) $4 \quad 7 \quad (2 \quad 3 \quad \uparrow) \quad + \quad 5 \quad / \quad +$
(7) $4 \quad 7 \quad 8 \quad + \quad 5 \quad / \quad +$
(8) $4 \quad (7 \quad 8 \quad +) \quad 5 \quad / \quad +$
(9) $4 \quad 15 \quad 5 \quad / \quad +$
(10) $4 \quad (15 \quad 5 \quad /) \quad +$
(11) $4 \quad 3 \quad +$
(12) 7

This time we look for the left most operation and the numbers immediately to the left of it.

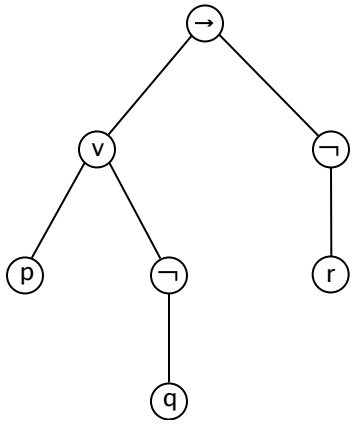
The exact same procedures may be applied to logical expressions as well as set operation expressions.

EXAMPLE 7.7.4. *Given the logical expression* $(p \vee \neg q) \rightarrow \neg r$

1. *Find the infix notation.*
2. *Represent using an ordered rooted tree.*
3. *Find the preorder form.*
4. *Find the postorder form.*

Solution:

1. $(p \vee (\neg q)) \rightarrow (\neg r)$
2. *Tree:*



3. *Preorder:* $\rightarrow \vee p \neg q \neg r$
4. *Postorder:* $pq\neg \vee r\neg \rightarrow$

CHAPTER 3

Boolean Algebra

1. Boolean Functions

1.1. Boolean Functions.

DEFINITIONS 1.1.1.

1. *A* **Boolean variable** *is a variable that may take on values only from the set* $B = \{0, 1\}$.
2. *A* **Boolean function of degree** n *or* **of order** n *is a function with domain* $B^n = \{(x_1, x_2, ..., x_n) | x_i \in B\}$ *and codomain* B. *In other words, Boolean functions of degree* n *are functions of the form* $F : B^n \to B$.
3. *Two Boolean functions,* F *and* G *are* **equivalent** *if*

$$F(x_1, x_2, x_3, \ldots, x_n) = G(x_1, x_2, x_3, \ldots, x_n)$$

for every ordered n*-tuple* $(x_1, x_2, x_3, \ldots, x_n) \in B^n$.

Discussion

We have already encountered Boolean variables, namely, propositional variables, which must have value 1 ("true") or 0 ("false"). Indeed, the propositional calculus is the motivating concept for the material in this chapter, and we shall refer to it frequently.

1.2. Example 1.2.1.

EXAMPLE 1.2.1. *A Boolean function of degree 2,* $F(x, y) : B^2 \to B$, *may be defined by a chart. For example, this function may be defined as follows:*

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

Discussion

There are several ways we may define a Boolean function. A table of values is one way. After we define addition, multiplication, and other operations on B, we may also use these operations to define functions.

Notice a Boolean function of two variables must assign to each of the four ordered pairs a value from B. This means there are $2^4 = 16$ different Boolean functions of order 2.

EXERCISE 1.2.1. *Use a table of values to define all 16 Boolean functions of order 2.*

EXERCISE 1.2.2. *How many Boolean functions of degree n are there?*

1.3. Binary Operations.

DEFINITIONS 1.3.1. *Let $x, y \in B$.*

1. **Addition** *is defined by the table*

x	y	$x+y$
1	1	1
1	0	1
0	1	1
0	0	0

2. **Multiplication** *is defined by the table*

x	y	$x \cdot y$
1	1	1
1	0	0
0	1	0
0	0	0

3. *The* **compliment** *is defined by the table*

x	$\overline{x}$
1	0
0	1

Discussion

If you think of the 1 as "true" and the 0 as "false", as we used in *Logic*, you should notice that Boolean addition corresponds to the logical "or", Boolean multiplication corresponds to the logical "and", and complementation corresponds to the logical "not". In fact, many authors use the notation $\vee$, $\wedge$, and $\neg$ for $+$, $\cdot$, and $\overline{\ }$, respectively. The notation $\overline{x} = x'$ is another common alternative for complimentation. We

will use this alternative on the discussion board and it may be used in homework. When there would be no confusion, we drop the $\cdot$ when denoting a Boolean product, just as is done in ordinary algebra.

Notice that Boolean addition defined here on $\{0, 1\}$ is NOT the same as the addition on the set of integers modulo 2.

1.4. Example 1.4.1.

EXAMPLE 1.4.1. *The following functions are equivalent.*

(1) $F(x, y) = x + y$
(2) $G(x, y) = xy + \overline{x}y + x\overline{y}$

PROOF. We show the two functions have the same values for every possible ordered pair in B^2.

x	y	xy	$\overline{x}$	$\overline{y}$	$\overline{x}y$	$x\overline{y}$	$xy + \overline{x}y + x\overline{y}$	$x + y$
1	1	1	0	0	0	0	1	1
1	0	0	0	1	0	1	1	1
0	1	0	1	0	1	0	1	1
0	0	0	1	1	0	0	0	0

□

Discussion

Example 1.4.1 gives an example of equivalent functions that are defined quite differently, although both representations are in terms of the algebra we have defined on $\{0, 1\}$.

1.5. Boolean Identities. Below is a table of the Boolean Identities you should know.

Identity	Name
$\overline{\overline{x}} = x$	Law of Double Complement
$x + x = x$ and $x \cdot x = x$	Idempotent Laws
$x + 0 = x$ and $x \cdot 1 = x$	Identity Laws
$x + 1 = 1$ and $x \cdot 0 = 0$	Dominance Laws
$x + y = y + x$ $xy = yx$	Commutative Laws
$x + (y + z) = (x + y) + z$ $x(yz) = (xy)z$	Associative Laws
$x + yz = (x + y)(x + z)$ $x(y + z) = xy + xz$	Distributive Laws
$\overline{x \cdot y} = \overline{x} + \overline{y}$ $\overline{x + y} = \overline{x} \cdot \overline{y}$	DeMorgan's Laws

Discussion

The distributive law for addition over multiplication and the DeMorgan's Laws may seem somewhat unusual to you at this stage, since they have no counterpart in the operations of addition and multiplication used in our familiar number systems. Indeed, you should avoid any analogies with ordinary arithmetic and, instead, use the propositional calculus as your model whenever you feel the need to use a familiar setting in which to exemplify these or any other properties you might encounter.

EXERCISE 1.5.1. *Verify the distributive law $x + yz = (x + y)(x + z)$.*

1.6. Dual.

DEFINITION 1.6.1. *The* **dual** *of a Boolean expression is the expression one obtains by interchanging addition and multiplication and interchanging 0's and 1's. The dual of the function F is denoted F^d.*

THEOREM 1.6.1 (Duality Principle). *If F and G are Boolean functions such that $F = G$, then $F^d = G^d$.*

Discussion

EXAMPLE 1.6.3. *The dual of $x\overline{y} + \overline{x}z$ is $(x + \overline{y}) \cdot (\overline{x} + z)$.*

Notice that we have implicitly assumed an order of operations for a Boolean expression: unless grouping is present to indicate otherwise, complements are evaluated first, then products, and then sums. This order conforms to the convention we established earlier for the order of operations in the predicate calculus. As the example above shows, you must be careful to preserve the correct order of operation when taking the dual of an expression, using parentheses wherever necessary.

2. Representing Boolean Functions

2.1. Representing Boolean Functions.

DEFINITIONS 2.1.1.

1. *A* **literal** *is a Boolean variable or the complement of a Boolean variable.*
2. *A* **minterm** *is a product of literals. More specifically, if there are n variables, $x_1, x_2, \ldots x_n$, a minterm is a product $y_1 y_2 \cdots y_n$ where y_i is x_i or $\overline{x}_i$.*
3. *A* **sum-of-products expansion** *or* **disjunctive normal form** *of a Boolean function is the function written as a sum of minterms.*

Discussion

Consider a particular element, say $(0, 0, 1)$, in the Cartesian product B^3. There is a unique Boolean product that uses each of the variables x, y, z or its complement (but not both) and has value 1 at $(0, 0, 1)$ and 0 at every other element of B^3. This product is $\overline{x}\,\overline{y}z$.

This expression is called a *minterm* and the factors, $\overline{x}$, $\overline{y}$, and z, are *literals*. This observation makes it clear that one can represent *any* Boolean function as a *sum-of-products* by taking Boolean sums of all minterms corresponding to the elements of B^n that are assigned the value 1 by the function. This sum-of-products expansion is analogous to the *disjunctive normal form* of a propositional expressions discussed in *Propositional Equivalences* in MAD 2104.

2.2. Example 2.2.1.

EXAMPLE 2.2.1. *Find the disjunctive normal form for the Boolean function F defined by the table*

$x,$	y	z	$F(x, y, z)$
0	*0*	*0*	*0*
0	*0*	*1*	*0*
0	*1*	*0*	*1*
0	*1*	*1*	*0*
1	*0*	*0*	*1*
1	*0*	*1*	*1*
1	*1*	*0*	*0*
1	*1*	*1*	*0*

Solution: $F(x, y, z) = \overline{x}y\overline{z} + x\overline{y}\,\overline{z} + x\overline{y}z$

Discussion

The disjunctive normal form should have three minterms corresponding to the three triples for which F takes the value 1. Consider one of these: $F(0,1,0) = 1$. In order to have a product of literals that will equal 1, we need to multiply literals that have a value of 1. At the triple $(0,1,0)$ the literals we need are $\overline{x}$, y, and $\overline{z}$, since $\overline{x} = y = \overline{z} = 1$ when $x = 0$, $y = 1$, and $z = 0$. The corresponding minterm, $\overline{x}y\overline{z}$, will then have value 1 at $(0,1,0)$ and 0 at every other triple in B^3. The other two minterms come from considering $F(1,0,0) = 1$ and $F(1,0,1) = 1$. The sum of these three minterms will have value 1 at each of $(1,0,0)$, $(0,1,0)$, $(1,0,1)$ and 0 at all other triples in B^3.

2.3. Example 2.3.1.

EXAMPLE 2.3.1. *Simply the expression*

$$F(x,y,z) = \overline{x}y\overline{z} + x\overline{y}\,\overline{z} + x\overline{y}z$$

using properties of Boolean expressions.

Solution.

$$\begin{aligned} \overline{x}y\overline{z} + x\overline{y}\,\overline{z} + x\overline{y}z &= \overline{x}y\overline{z} + x\overline{y}(\overline{z} + z) \\ &= \overline{x}y\overline{z} + x\overline{y} \cdot 1 \\ &= \overline{x}y\overline{z} + x\overline{y} \end{aligned}$$

Discussion

Example 2.3.1 shows how we might simplify the function we found in Example 2.2.1. Often sum-of-product expressions may be simplified, but any nontrivial simplification will produce an expression that is *not* in sum-of-product form. A sum-of-products form must be a sum of *minterms* and a minterm must have each variable or its compliment as a factor.

EXAMPLE 2.3.2. *The following are examples of "simplifying" that changes a sum-of-products to an expression that is not a sum-of-products:*

$$\begin{aligned} \textit{sum-of-product form:} \quad & x\overline{y}z + x\overline{y}\,\overline{z} + xyz \\ \textit{NOT sum-of-product form:} \quad & = x\overline{y} + xyz \\ \textit{NOT sum-of-product form:} \quad & = x(\overline{y} + yz) \end{aligned}$$

EXERCISE 2.3.1. *Find the disjunctive normal form for the Boolean function, G, of degree 4 such that $G(x_1, x_2, x_3, x_4) = 0$ if and only if at least 3 of the variables are 1.*

2.4. Functionally Complete.

DEFINITION 2.4.1. *A set of operations is called* **functionally complete** *if every Boolean function can be expressed using only the operations in the set.*

Discussion

Since every Boolean function can be expressed using the operations $\{+, \cdot, \overline{\ }\}$, the set $\{+, \cdot, \overline{\ }\}$ is *functionally complete.* The fact that every function may be written as a sum-of-products demonstrates that this set is functionally complete.

There are many other sets that are also functionally complete. If we can show each of the operations in $\{+, \cdot, \overline{\ }\}$ can be written in terms of the operations in another set, S, then the set S is functionally complete.

2.5. Example 2.5.1.

EXAMPLE 2.5.1. *Show that the set of operations* $\{\cdot, \overline{\ }\}$ *is functionally complete.*

PROOF. Since $\cdot$ and $\overline{\ }$ are already members of the set, we only need to show that $+$ may be written in terms of $\cdot$ and $\overline{\ }$.

We claim

$$x + y = \overline{\overline{x} \cdot \overline{y}}.$$

Proof of Claim Version 1

$\overline{\overline{x} \cdot \overline{y}} = \overline{\overline{x}} + \overline{\overline{y}}$	De Morgan's Law
$= x + y$	Law of Double Complement

Proof of Claim Version 2

x	y,	$\overline{x}$	$\overline{y}$	$\overline{x} \cdot \overline{y}$	$\overline{\overline{x} \cdot \overline{y}}$	$x + y$
1	1	0	0	0	1	1
1	0	0	1	0	1	1
0	1	1	0	0	1	1
0	0	1	1	1	0	0

□

Discussion

EXERCISE 2.5.1. *Show that* $\{+, \overline{\ }\}$ *is functionally complete.*

EXERCISE 2.5.2. *Prove that the set* $\{+, \cdot\}$ *is not functionally complete by showing that the function* $F(x) = \overline{x}$ *(of order 1) cannot be written using only* x *and addition and multiplication.*

2.6. NAND and NOR.

DEFINITIONS 2.6.1.

1. *The binary operation* **NAND**, *denoted* $|$, *is defined by the table*

x	y	$x\|y$
1	1	0
1	0	1
0	1	1
0	0	1

2. *The binary operation* **NOR**, *denoted* $\downarrow$, *is defined by the table*

x	y	$x \downarrow y$
1	1	0
1	0	0
0	1	0
0	0	1

Discussion

Notice the NAND operator may be thought of as "not and" while the NOR may be thought of as "not or."

EXERCISE 2.6.1. *Show that* $x|y = \overline{x \cdot y}$ *for all* x *and* y *in* $B = \{0, 1\}$.

EXERCISE 2.6.2. *Show that* $\{|\}$ *is functionally complete.*

EXERCISE 2.6.3. *Show that* $x \downarrow y = \overline{x + y}$ *for all* x *and* y *in* $B = \{0, 1\}$.

EXERCISE 2.6.4. *Show that* $\{\downarrow\}$ *is functionally complete.*

3. Abstract Boolean Algebras

3.1. Abstract Boolean Algebra.

DEFINITION 3.1.1. *An* **abstract Boolean algebra** *is defined as a set* $\mathcal{B}$ *containing two distinct elements* 0 *and* 1*, together with binary operations* $+$, $\cdot$ *, and a unary operation* $\overline{\ }$ *, having the following properties:*

$x + 0 = x$ $x \cdot 1 = x$	*Identity Laws*
$x + \overline{x} = 1$ $x \cdot \overline{x} = 0$	*Compliments Laws*
$(x + y) + z = x + (y + z)$ $(x \cdot y) \cdot z = x \cdot (y \cdot z)$	*Associative Laws*
$x + y = y + x$ $x \cdot y = y \cdot x$	*Commutative Laws*
$x + (y \cdot z) = (x + y) \cdot (x + z)$ $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	*Distributive Laws*

Discussion

The definition of an abstract Boolean algebra gives the **axioms** for an abstract Boolean algebra. The unary operation $\overline{\ }$ is called **complementation**. Named after the English mathematician George Boole (1815-1864), Boolean algebras are especially important in computer science because of their applications to switching theory and design of digital computers.

3.2. Examples of Boolean Algebras. Examples.

1. $B = \{0, 1\}$ together with the operations $+$, $\cdot$, $\overline{\ }$ described in *Boolean Functions* is a Boolean Algebra.
2. B^k together with the operations
   (a) $(x_1, x_2, x_3, \ldots, x_k) + (y_1, y_2, y_3, \ldots, y_k) = (x_1 + y_1, x_2 + y_2, x_3 + y_3, \ldots, x_k + y_k)$
   (b) $(x_1, x_2, x_3, \ldots, x_k) \cdot (y_1, y_2, y_3, \ldots, y_k) = (x_1 \cdot y_1, x_2 \cdot y_2, x_3 \cdot y_3, \ldots, x_k \cdot y_k)$
   (c) $\overline{(x_1, x_2, x_3, \ldots, x_k)} = (\overline{x_1}, \overline{x_2}, \overline{x_3}, \ldots, \overline{x_k})$

   is a Boolean Algebra.

   We can find the element of B^k that is considered the "one element" by asking which element of B^k will satisfy the properties: $x \cdot$ "one" $= x$ and $x + \overline{x} =$ "one" for all $x \in B^k$? In other words, using the definition of the operations in B^k we need to find the element of B^k so that for all $(x_1, x_2, x_3, \ldots, x_k) \in B^k$ we have

$(x_1, x_2, x_3, \ldots, x_k)\cdot$"one" $= (x_1, x_2, x_3, \ldots, x_k)$ and $(x_1+\overline{x_1}, x_2+\overline{x_2}, x_3+\overline{x_3}, \ldots, x_k+\overline{x_k})$ = "one". Notice that the ordered k-tuple of all 1's satisfies these properties, so the "one" element is $(1, 1, 1, \ldots, 1)$.

3. $BOOL(k)$ defined to be the set of all Boolean functions of degree k together with the operations
   (a) $F + G$ (or $F \vee G$) defined by $(F + G)(u) = F(u) + G(u)$ for any $u \in B^k$,
   (b) $F \cdot G$ (or $F \wedge G$) defined by $(F \cdot G)(u) = F(u) \cdot G(u)$ for any $u \in B^k$,
   (c) and $\overline{F}$ defined by $\overline{F}(u) = \overline{F(u)}$ for any $u \in B^k$.

   is a Boolean Algebra.
4. Let S be a set and let $FUN(S, \{0, 1\})$ be the set of all functions with domain S and codomain $\{0, 1\}$. Define the Boolean operations on $FUN(S, \{0, 1\})$ as follows: Let $F, G \in FUN(S, \{0, 1\})$, then
   (a) $F + G : S \to \{0, 1\}$ is the function defined by $(F + G)(x) = F(x) + G(x)$ for all $x \in S$,
   (b) $F \cdot G : S \to \{0, 1\}$ is the function defined by $(F \cdot G)(x) = F(x) \cdot G(x)$ for all $x \in S$,
   (c) $\overline{F} : S \to \{0, 1\}$ is the function defined by $\overline{F}(x) = \overline{F(x)}$ for all $x \in S$,

   $FUN(S, \{0, 1\})$ together with these operations is a Boolean Algebra.
5. Let S be a set. The power set $P(S)$ together with the operations
   (a) $A + B = A \cup B$ for all $A, B \in P(S)$
   (b) $A \cdot B = A \cap B$ for all $A, B \in P(S)$
   (c) $\overline{A}$ is the complement of A for all $A \in P(S)$

   is a Boolean Algebra.

   We can find the element of $P(S)$ that is the "one" element by asking which element of $P(S)$ will satisfy the identity and compliments properties of a Boolean algebra. Interpreting this in terms of the way the operations are defined on this set we see the set S is the element in $P(S)$ that satisfies the properties since $A \cup \overline{A} = S$ and $A \cap S = A$ for any set A in $P(S)$.
6. The set $D_6 = \{1, 2, 3, 6\}$ along with the operations
   (a) $a + b = lcm(a, b)$ for all $a, b \in D_6$
   (b) $a \cdot b = gcd(a, b)$ for all $a, b \in D_6$
   (c) $\overline{a} = 6/a$ for all $a \in D_6$

   is a Boolean algrebra.

   The element 1 of D_6 is the "zero" element of D_6 since it satisfies the identity and compliments properties for this Boolean algebra. That is "zero" $= a \cdot \overline{a} = gcd(a, 6/a) = 1$ and $a +$ "zero" $= a + 1 = lcm(a, 1) = a$ for all $a \in D_6$.

Discussion

The set $B = \{0, 1\}$, together with the Boolean operations defined earlier, is the simplest example of a Boolean algebra, but there are many others, some of which do not involve Boolean operations on the set $\{0, 1\}$, at least overtly. The examples above exhibits six examples of abstract Boolean algebras, including $\{0, 1\}$ and the Boolean

algebra of Boolean functions discussed in the lectures on Boolean Functions and their Representations.

Let us examine example 3 a bit closer. The set $BOOL(2)$ is the set of all Boolean functions of degree 2. In the lecture notes *Boolean Functions* we determined there were 16 different Boolean functions of degree 2. In fact, in an exercise following this observation you created a table representing all 16 Boolean functions of degree 2. Notice $BOOL(2)$ is the set of the 16 functions represented by this table.

EXERCISE 3.2.1. *Write a table representing all the elements of* $BOOL(2)$ *and name the elements (functions)* $F_1, F_2, F_3, \ldots, F_{16}$.

(a) *Find* $F_3 + F_4$, $F_3 \cdot F_4$, *and* $\overline{F_3}$ *(your answers will depend on how you labeled your functions).*
(b) *Which of the functions is the 0 element of the abstract Boolean algebra?*
(c) *Which of the functions is the 1 element of the abstract Boolean algebra?*

The following table gives some of the identity elements, 0 and 1, of the Boolean algebras given in the previous examples of abstract Boolean algebras.

EXERCISE 3.2.2. *Fill in the rest of the information in the table.*

Boolean Algebra	*0 element*	*1 element*	*an element that is neither 0 nor 1*
B	*0*	*1*	*none*
B^5	*?*	$(1,1,1,1,1)$	$(1,0,0,0,0)$
$FUN(\{a,b,c\},\{0,1\})$	$\chi_\emptyset$	*?*	$f:\{a,b,c\}\to B$ *defined by* $f(a)=0, f(b)=1, f(c)=1$
$P(S)$	*?*	S	*?*
D_6	*1*	*?*	*?*

The function $\chi_A : S \to B$, where A a subset of S is called the *characteristic function of* A and is defined by $\chi_A(x) = 1$ if $x \in A$, and $\chi_A(x) = 0$ if $x \in S - A$. (χ is the lower case Greek letter "chi".)

Here is another important example that we discussed in some detail in MAD 2104.

EXAMPLE 3.2.7. *Let* $\mathcal{B}$ *be a nonempty set of propositions satisfying the conditions:*

(1) *if* p *is in* $\mathcal{B}$, *then so is* $\neg p$, *and*
(2) *if* p *and* q *are in* $\mathcal{B}$, *then so are* $p \vee q$ *and* $p \wedge q$.

Then $\mathcal{B}$ *together with the operations* $\vee$ *(for* $+$*),* $\wedge$ *(for* $\cdot$ *), and* $\neg$ *(for* $\overline{\ }$ *) is a Boolean algebra.*

PROOF. Since $\mathcal{B} \neq \emptyset$, $\mathcal{B}$ contains a proposition p. By (1), $\neg p$ is also in $\mathcal{B}$. By (2), $\mathcal{B}$ contains the tautology $p \vee \neg p = \mathbf{1}$ and the contradiction $p \wedge \neg p = \mathbf{0}$. The remaining properties were established in the materials covered in MAD 2104. □

As these examples illustrate, the names for addition and multiplication in a particular Boolean algebra may be idiomatic to that example. Addition may be called *sum*, *union*, *join*, or *disjunction*; whereas, multiplication may be called *product*, *intersection*, *meet*, or *conjunction*. Because the addition and multiplication operations in a Boolean algebra behave so differently from addition and multiplication in the more familiar algebraic systems, such as the integers or real numbers, alternative notation, such as $\vee$ for $+$ and $\wedge$ for $\cdot$, are often used instead. At the risk of creating confusion we shall use $+$ and $\cdot$ when working in an abstract Boolean algebra, but, when working with a particular example, such as the one above, we will use conventional notation associated with that example.

3.3. Duality.

DEFINITION 3.3.1 (**Duality**). *Notice how the axioms of an abstract Boolean algebra in the definition of a Boolean algebra have been grouped in pairs. It is possible to get either axiom in a pair from the other by interchanging the operations $+$ and $\cdot$, and interchanging the elements* 0 *and* 1. *This is called the* **principle of duality**. *As a consequence, any property of a Boolean algebra has a dual property (which is also true) obtained by performing these interchanges.*

3.4. More Properties of a Boolean Algebra.

THEOREM 3.4.1 (Properties). *Let $\mathcal{B}$ be an abstract Boolean algebra. Then for any $x, y \in \mathcal{B}$...*

(1) Idempotent Laws: $x + x = x$ and $x \cdot x = x$
(2) Domination Laws: $x + 1 = 1$ and $x \cdot 0 = 0$
(3) Absorption Laws: $(x \cdot y) + x = x$ and $(x + y) \cdot x = x$
(4) $x + y = 1$ and $x \cdot y = 0$ if and only if $y = \overline{x}$
(5) Double Complements Law: $\overline{\overline{x}} = x$
(6) DeMorgan's Laws: $\overline{x \cdot y} = \overline{x} + \overline{y}$ and $\overline{x + y} = \overline{x} \cdot \overline{y}$

Discussion

The properties in Theorem 3.4.1 are all consequences of the axioms of a Boolean algebra. When proving any property of an abstract Boolean algebra, we may only use the axioms and previously proven results. In particular, we may *not* assume we are working in any one particular example of a Boolean algebra, such as the Boolean algebra $\{0, 1\}$. In the following examples and exercises, $x, y, z, ...$ represent elements

of an arbitrary Boolean algebra $\mathcal{B}$. Notice that these arbitrary elements may or *may not* be the zero or one elements of the Boolean algebra.

EXAMPLE 3.4.1. *For any x in $\mathcal{B}$, $0 + x = x$ and $1 \cdot x = x$.*

PROOF. These follow directly from the Identity Laws and the Commutative Laws. Notice that the second property is the dual of the first. □

3.5. Proof of Idempotent Laws.

PROOF OF FIRST IDEMPOTENT LAW. Let $\mathcal{B}$ be a Boolean algebra and let $x \in \mathcal{B}$.

$$\begin{array}{lll} x + x & = (x + x) \cdot 1 & \text{Identity Law} \\ & = (x + x) \cdot (x + \overline{x}) & \text{Compliments Law} \\ & = x + (x \cdot \overline{x}) & \text{Distributive Law} \\ & = x + 0 & \text{Compliments Law} \\ & = x & \text{Identity Law} \end{array}$$

□

Discussion

EXERCISE 3.5.1. *Interpret the Idempotent Laws for the Boolean algebra $P(S)$ of subsets of a set S (Example 5).*

EXERCISE 3.5.2. *Prove the other Idempotent Law, for any x in $\mathcal{B}$, $x \cdot x = x$, in two ways: (a) using the principle of duality, and (b) directly (without invoking the duality principle).*

3.6. Proof of Dominance Laws.

PROOF OF THE DOMINANCE LAW $x + 1 = 1$. Let $\mathcal{B}$ be a Boolean algebra and let $x \in \mathcal{B}$.

$$\begin{array}{lll} x + 1 & = (x + 1) \cdot 1 & \text{Identity Law} \\ & = (x + 1) \cdot (x + \overline{x}) & \text{Compliments Law} \\ & = x + 1 \cdot \overline{x} & \text{Distributive Law} \\ & = x + \overline{x} & \text{Identity Law} \\ & = 1 & \text{Compliments Law} \end{array}$$

□

Discussion

One of the Dominance Laws, Property 2 of Theorem 3.4.1, is proved above. This "Dominance Law" may look a little peculiar, since there is no counterpart in the algebra of the integers or real numbers. It is, however, a familiar property of the Boolean algebra $P(S)$ of subsets of a set S. It merely says that $A \cup S = S$ for every subset A of S.

EXERCISE 3.6.1. *Prove the other Dominance Law (Theorem 3.4.1 Property 2), $x \cdot 0 = 0$ for every x in $\mathcal{B}$, in two ways: (a) using the principle of duality, and (b) directly (without invoking the duality principle).*

EXERCISE 3.6.2. *Prove the Absorption Laws (Theorem 3.4.1 Property 3): $(x \cdot y) + x = x$ and $(x + y) \cdot x = x$ for all x, y in $\mathcal{B}$. [Hint: Use Property 2.]*

EXERCISE 3.6.3. *Interpret the Absorption Laws for the Boolean algebra $P(S)$ of subsets of a set S (Example 5).*

3.7. Proof of Theorem 3.4.1 Property 4. Recall Theorem 3.4.1 Property 4: For all x and y in $\mathcal{B}$, $x + y = 1$ and $x \cdot y = 0$ if and only if $y = \overline{x}$.

PROOF OF THEOREM 3.4.1 PROPERTY 4. Let $x, y \in \mathcal{B}$ and suppose that $x + y = 1$ and $x \cdot y = 0$. Then,

y	$= y \cdot 1$	Identity Law
	$= y \cdot (x + \overline{x})$	Compliments Law
	$= y \cdot x + y \cdot \overline{x}$	Distributive Law
	$= 0 + y \cdot \overline{x}$	Hypothesis
	$= y \cdot \overline{x}$	Identity Law

On the other hand

$\overline{x}$	$= \overline{x} \cdot 1$	Identity Law
	$= \overline{x} \cdot (x + y)$	Hypothesis
	$= \overline{x} \cdot x + \overline{x} \cdot y$	Distributive Law
	$= x \cdot \overline{x} + y \cdot \overline{x}$	Commutative Law
	$= 0 + y \cdot \overline{x}$	Compliments Law
	$= y \cdot \overline{x}$	Identity Law

Thus, $y = y \cdot \overline{x} = \overline{x}$. □

Discussion

One direction of the "if and only if" statement of Property 4 Theorem 3.4.1 is just a restatement of the Compliments Laws; hence, we need only prove that if $u + v = 1$ and $u \cdot v = 0$, then $v = \overline{u}$. Since $u \cdot \overline{u} = 0$ and $u \cdot v = 0$, it is tempting to take the resulting equation $u \cdot \overline{u} = u \cdot v$ and simply "cancel" the u from each side to conclude that $\overline{u} = v$. However, **there is no cancellation law for multiplication**

(or addition) in a Boolean algebra as there is in the algebra of the integers or the real numbers. Thus, we must be a little more clever in constructing a proof.

EXERCISE 3.7.1. *Give an example of a Boolean algebra $\mathcal{B}$ and elements x, y, z in $\mathcal{B}$ such that $x + z = y + z$, but $x \neq y$.*

Property 4 shows that the complement of an element u in a Boolean algebra is the *unique* element that satisfies the Compliments Laws relative to u. Such uniqueness results can provide very powerful strategies for showing that two elements in a Boolean algebra are equal. Here is another example of uniqueness, this time of the additive identity element 0.

THEOREM 3.7.1. *Suppose u is an element of a Boolean algebra $\mathcal{B}$ such that $x+u = x$ for all x in $\mathcal{B}$. Then $u = 0$.*

PROOF. Since $x + u = x$ for all x in $\mathcal{B}$, $0 + u = 0$. But $0 + u = u + 0 = u$ by the Commutative and Identity Laws; hence, $u = 0$. □

EXERCISE 3.7.2. *Suppose v is an element of a Boolean algebra $\mathcal{B}$ such that $x \cdot v = x$ for all x in $\mathcal{B}$. Prove that $v = 1$.*

EXERCISE 3.7.3. *Prove that $\overline{1} = 0$ and $\overline{0} = 1$. [Hint: Use Theorem 3.4.1 Property 4 and duality.]*

EXERCISE 3.7.4. *Prove the Double Complements Law: $\overline{\overline{x}} = x$.*

3.8. Proof of DeMorgan's Law. Recall one of DeMorgan's Laws: $\overline{xy} = \overline{x} + \overline{y}$ for all x, y in a Boolean algebra, $\mathcal{B}$.

PROOF. Let $x, y \in \mathcal{B}$.

$$\begin{aligned} xy + (\overline{x} + \overline{y}) &= [x + (\overline{x} + \overline{y})][y + (\overline{x} + \overline{y})] \\ &= [(x + \overline{x}) + \overline{y}][(y + \overline{y}) + \overline{x}] \\ &= (1 + \overline{y})(1 + \overline{x}) \\ &= 1 \cdot 1 \\ &= 1 \end{aligned}$$

$$\begin{aligned} (xy)(\overline{x} + \overline{y}) &= (xy)\overline{x} + (xy)\overline{y} \\ &= (x\overline{x})y + x(y\overline{y}) \\ &= 0 \cdot y + x \cdot 0 \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

Now apply Property 4 with $u = xy$ and $v = \overline{x} + \overline{y}$ to conclude that $\overline{x} + \overline{y} = \overline{xy}$. □

Discussion

As in ordinary algebra we may drop the $\cdot$ and indicate multiplication by juxtaposition when there is no chance for confusion. We adopt this convention in the previous proof, wherein we give the steps in the proof of one of DeMorgan's Laws. The proof invokes the uniqueness property of complements, Property 4 in Theorem 3.4.1, by showing that $\overline{x} + \overline{y}$ behaves like the complement of xy.

EXERCISE 3.8.1. *Give reasons for each of the steps in the proof of the DeMorgan's Law proven above. (Some steps may use more than one property.)*

EXERCISE 3.8.2. *Prove the other DeMorgan's Law,* $\overline{x + y} = \overline{x}\,\overline{y}$ *using the principle of duality.*

EXERCISE 3.8.3. *Prove the other DeMorgan's Law,* $\overline{x + y} = \overline{x}\,\overline{y}$ *directly (without invoking the duality principle).*

Notice. One of the morals from DeMorgan's Laws is that you must be careful to distinguish between $\overline{x \cdot y}$ and $\overline{x} \cdot \overline{y}$ (or between $\overline{xy}$ and $\overline{x}\,\overline{y}$), since they may represent different elements in the Boolean algebra.

3.9. Isomorphism.

DEFINITION 3.9.1. *Two Boolean algebras* B_1 *and* B_2 *are* **isomorphic** *if there is a bijection* $f\colon B_1 \to B_2$ *that preserves Boolean operations. That is, for all* x *and* y *in* B_1,

(1) $f(x + y) = f(x) + f(y)$,
(2) $f(x \cdot y) = f(x) \cdot f(y)$, *and*
(3) $f(\overline{x}) = \overline{f(x)}$.

The bijection f *is called an* **isomorphism** *between* B_1 *and* B_2.

Discussion

The adjective *isomorphic* was used earlier to describe two graphs that are the *same* in the sense that they share all of the same graph invariants. A graph *isomorphism* was defined to be a one-to-one correspondence (bijection) between the vertices of two (simple) graphs that preserves incidence. The terms isomorphic and isomorphism are used throughout mathematics to describe two mathematical systems are essentially the *same*.

EXAMPLE 3.9.1. *Let* B *be the Boolean algebra* $\{0, 1\}$, *and let* $P(S)$ *be the Boolean algebra of subsets of the set* $S = \{a\}$ *(having just one element). Prove that* B *and* $P(S)$ *are isomorphic.*

PROOF. $P(S) = \{\emptyset, S\}$ has exactly two elements as does B. Thus, there are two bijections from B to $P(S)$. Only one of these, however, is an isomorphism of Boolean algebras, namely, the bijection $f\colon B \to P(S)$ defined by $f(0) = \emptyset$ and $f(1) = S$. We can check that the three properties of an isomorphism hold by using tables to check all possible cases:

x	$f(x)$	$\overline{x}$	$f(\overline{x})$	$\overline{f(x)}$
0	$\emptyset$	1	S	S
1	S	0	$\emptyset$	$\emptyset$

x	y	$f(x)$	$f(y)$	$x+y$	$x\cdot y$	$f(x+y)$	$f(x)\cup f(y)$	$f(x\cdot y)$	$f(x)\cap f(y)$
0	0	$\emptyset$	$\emptyset$	0	0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
0	1	$\emptyset$	S	1	0	S	S	$\emptyset$	$\emptyset$
1	0	S	$\emptyset$	1	0	S	S	$\emptyset$	$\emptyset$
1	1	S	S	1	1	S	S	S	S

□

EXERCISE 3.9.1. *Given B and $P(S)$ as in Example 3.9.1, show that the bijection $g\colon B \to P(S)$ defined by $g(0) = S$ and $g(1) = \emptyset$ does not define an isomorphism.*

3.10. Atoms.

DEFINITION 3.10.1. *An element a in a Boolean algebra B is called an* **atom** *if*

(1) $a \neq 0$, and
(2) for every x in B, either $ax = a$ or $ax = 0$.

THEOREM 3.10.1. *A nonzero element a in a Boolean algebra B is an atom if and only if for every $x, y \in B$ with $a = x+y$ we must have $a = x$ or $a = y$. In otherwords, a is* **indecomposable**.

Discussion

The method of exhausting all possible cases used in Example 3.9.1 to prove that a given bijection is an isomorphism is clearly not feasible for Boolean algebras that contain very many elements. The concept of an **atom** can be used to simplify the problem considerably. Atoms are in some sense *minimal* nonzero elements of a Boolean algebra, and, in the case of a finite Boolean algebra, they *generate* the algebra; that is, every nonzero element of the algebra can be written as a (finite) sum of atoms.

EXAMPLE 3.10.1. *Let* $B^2 = \{(0,0),(0,1),(1,0),(1,1)\}$ *be the Boolean algebra described in Example 2 with* $k = 2$. *The elements* $(0,1)$ *and* $(1,0)$ *are the atoms of* B^2. *(Why?) Notice that the only other nonzero element is* $(1,1)$, *and* $(1,1) = (0,1)+(1,0)$.

EXERCISE 3.10.1. *Let* $B^n = \{(x_1, x_2, ..., x_n) | x_i = 0 \text{ or } 1\}$ *be the Boolean algebra described in Example 2.*

(a) Show that the atoms of B^n *are precisely the elements*

$$a_i = (0, 0, ..., 0, \underset{\substack{\uparrow \\ i^{th}\ coordinate}}{1}, 0, ..., 0),$$

$i = 1, 2, ..., n$. *[Hint: Show (i) each* a_i *is an atom, and (ii) if* $x \in B^n$ *has two nonzero coordinates, then* x *is not an atom.]*

(b) Show that every nonzero element of B^n *is a sum of atoms.*

PROOF OF THEOREM 3.10.1. Let $a \in B$.

First we show

$$\{\forall u \in B[(au = 0) \wedge (au = a)]\} \Rightarrow \{\forall x, y \in B[(a = x + y) \rightarrow ((a = x) \vee (a = y))]\}.$$

Assume a is such that $au = 0$ or $au = a$ for all $u \in B$ and let $x, y \in B$ be such that $x + y = a$. Then

$$\begin{aligned} ax = \ & x \cdot x + yx \\ & = x + yx \\ & = x(1 + y) \\ & = x \cdot 1 \\ & = x \end{aligned}$$

But by our assumption, $ax = 0$ or $ax = a$, so $x = 0$ or $x = a$. If $x = 0$ then we would have $y = a$ proving $\forall x, y \in B[(a = x + y) \rightarrow ((a = x) \vee (a = y))]$

We now will show

$$\{\forall x, y \in B[(a = x + y) \rightarrow ((a = x) \vee (a = y))]\} \Rightarrow \{\forall u \in B[(au = 0) \wedge (au = a)]\}.$$

Assume a is such that $\forall x, y \in B[(a = x+y) \rightarrow ((a = x) \vee (a = y))]$ and let $u \in B$. Then

$$\begin{aligned} au + a\overline{u} \ & = a(u + \overline{u}) \\ & = a \cdot 1 \\ & = a \end{aligned}$$

Thus $au = a$ or $a\overline{u} = a$. Suppose $a\overline{u} = a$. Then

$$\begin{aligned} au &= (a\overline{u})u \\ &a(\overline{u} \cdot u) \\ &a \cdot 0 \\ &0 \end{aligned}$$

□

3.11. Theorem 3.11.1.

THEOREM 3.11.1. *If a and b are atoms in a Boolean algebra $\mathcal{B}$, then either $a = b$ or $ab = 0$.*

Discussion

Theorem 3.11.1 gives a property of atoms that we will find very useful. It says that, in some sense, atoms in a Boolean algebra are *disjoint*. When applied to the example $P(S)$ of subsets of a set S, this is precisely what it is saying.

EXERCISE 3.11.1. *Prove Theorem 3.11.1. [Hint: Use the definition to prove the logically equivalent statement: If $ab \neq 0$, then $a = b$.]*

3.12. Theorem 3.12.1.

THEOREM 3.12.1. *Suppose that an element x in a Boolean algebra $\mathcal{B}$ can be expressed as a sum of distinct atoms $a_1, ..., a_m$. Then $a_1, ..., a_m$ are unique except for their order.*

Discussion

Theorem 3.12.1 provides the rather strong result that an element of a Boolean algebra cannot be expressed as a sum of atoms in more than one way, except by reordering the summands. In particular, it shows that *each individual atom is indecomposable* in the sense that it cannot be written as a sum of two or more atoms in a nontrivial way.

PROOF OF THEOREM 3.12.1. Suppose x can be expressed as sums

$$x = a_1 + a_2 + \cdots + a_m = b_1 + b_2 + \cdots + b_n,$$

where each a_i and each b_j is an atom of $\mathcal{B}$, the a_i's are distinct, and the b_j's are distinct. Then, by the Distributive Law, for each $i = 1, ..., m$,

$$\begin{aligned} a_i x &= a_i(a_1 + a_2 + \cdots + a_m) &&= a_i(b_1 + b_2 + \cdots + b_n) \\ &= a_i a_1 + a_i a_2 + \cdots a_i a_m &&= a_i b_1 + a_i b_2 + \cdots + a_i b_n. \end{aligned}$$

By Theorem 3.11.1, $a_i a_j = 0$, if $i \neq j$, so that

$$a_i a_1 + a_i a_2 + \cdots a_i a_m = a_i a_i = a_i.$$

If $a_i \neq b_j$ for all j, then, again by Theorem 3.11.1, $a_i b_j = 0$ for all j, so that $a_i b_1 + a_i b_2 + \cdots + a_i b_n = 0$. This is not possible, however, since $a_i \neq 0$ and $a_i = a_i b_1 + a_i b_2 + \cdots + a_i b_n$. Thus, $a_i = b_j$ for some j.

By interchanging the roles of the a's and the b's, the same argument shows that for each j, $b_j = a_i$ for some i. Thus, $m = n$, and the sets $\{a_1, ..., a_m\}$ and $\{b_1, ..., b_m\}$ are equal.

□

3.13. Basis.

THEOREM 3.13.1. *Suppose $\mathcal{B}$ is a finite Boolean algebra. Then there is a set of atoms $\mathcal{A} = \{a_1, a_2, ..., a_k\}$ in $\mathcal{B}$ such that every nonzero element of $\mathcal{B}$ can be expressed uniquely as a sum of elements of $\mathcal{A}$ (up to the order of the summands).*

DEFINITION 3.13.1. *Given a Boolean algebra $\mathcal{B}$, a set $\mathcal{A}$ of atoms of $\mathcal{B}$ is called a* **basis** *if every nonzero element of $\mathcal{B}$ can be written as a finite sum of atoms in $\mathcal{A}$.*

Discussion

Theorem 3.13.1 shows that every **finite** Boolean algebra has a basis, as defined above. The finiteness condition is necessary as the following exercise makes clear.

EXERCISE 3.13.1. *Let $\mathbb{Z}$ denote the set of integers. Prove:*

(a) The atoms of $P(\mathbb{Z})$ are the sets $\{n\}$ for $n \in \mathbb{Z}$.
(b) $P(\mathbb{Z})$ does not contain a basis.

PROOF OF THEOREM 3.13.1. Suppose $\mathcal{B} = \{x_1, x_2, ..., x_m\}$, where the x_i's are distinct. As in *Representing Boolean Functions*, define a **minterm** in the x_i's as a product $y_1 y_2 \cdots y_m$, where each y_i is either x_i or $\overline{x_i}$. Using the Compliments Law, $x + \overline{x} = 1$, one can prove, by mathematical induction, that the sum of all possible minterms is 1:

$$\sum_{y_i = x_i \text{ or } y_i = \overline{x_i}} y_1 y_2 \cdots y_m = 1.$$

(See Exercise 3.13.2.)

If a minterm $y_1 y_2 \cdots y_m$ is not 0, then it must be an atom:

- $x_i(y_1y_2\cdots y_m) = y_1y_2\cdots(x_iy_i)\cdots y_m = 0$, if $y_i = \overline{x_i}$, and
- $x_i(y_1y_2\cdots y_m) = y_1y_2\cdots(x_iy_i)\cdots y_m = y_1y_2\cdots y_m$, if $y_i = x_i$.

Thus, for any i,

$$x_i = x_i \cdot 1 = x_i \cdot \sum_{y_i=x_i \text{ or } y_i=\overline{x_i}} y_1y_2\cdots y_m = \sum_{y_i=x_i \text{ or } y_i=\overline{x_i}} x_i(y_1y_2\cdots y_m).$$

As observed above, each product $x_i(y_1y_2\cdots y_m)$ is either 0 or is equal to the minterm $y_1y_2\cdots y_m$ itself. Thus, if $x_i \neq 0$, then x_i is a sum of nonzero minterms; hence, a sum of atoms. Thus, we have shown that each nonzero minterm in the x_i's is an atom and each nonzero element of $\mathcal{B}$ is a sum of nonzero minterms.

The theorem is now proved by letting $\mathcal{A} = \{a_1, a_2, ..., a_k\}$ be the set of all nonzero minterms in the x_i's. Every nonzero element of $\mathcal{B}$ can be expressed as a sum of elements of $\mathcal{A}$, and the uniqueness of this representation follows from Theorem 3.12.1.

□

EXERCISE 3.13.2. *Use mathematical induction to prove that if $x_1, x_2, ..., x_r$ are arbitrary elements of a Boolean algebra $\mathcal{B}$, then the sum of all minterms $y_1y_2\cdots y_r$ in the x_i's is equal to 1. [Hint: Notice that the terms in the sum of all minterms*

$$\sum_{y_i=x_i \text{ or } y_i=\overline{x_i}} y_1y_2\cdots y_r$$

fall into two classes, those for which $y_1 = x_1$ and those for which $y_1 = \overline{x_1}$.]

EXERCISE 3.13.3. *Suppose $I_n = \{1, 2, ..., n\}$. Show that the set $\{\{1\}, \{2\}, ..., \{n\}\}$ is a basis for the Boolean algebra $P(I_n)$ of subsets of I_n.*

3.14. Theorem 3.14.1.

THEOREM 3.14.1. *Suppose that $\mathcal{B}$ is a finite Boolean algebra having a basis consisting of n elements. Then $\mathcal{B}$ is isomorphic to the Boolean algebra $P(I_n)$ of subsets of $I_n = \{1, 2, ..., n\}$.*

Discussion

Theorem 3.14.1, together with Theorem 3.13.1, provides the main characterization of finite Boolean algebras. Putting these two theorems together, we see that every finite Boolean algebra has a basis and, hence, is isomorphic to the Boolean algebra, $P(I_n)$, of subsets of the set $I_n = \{1, 2, ..., n\}$ for some positive integer n. This characterization puts a severe constraint on the number of elements in a finite Boolean algebra, since the Boolean algebra $P(I_n)$ has 2^n elements.

PROOF OF THEOREM 3.14.1. Let $\{a_1, a_2, ..., a_n\}$ be the basis for $\mathcal{B}$. Recall that in the Boolean algebra $P(I_n)$, "addition" is union, $\cup$, "multiplication" is intersection, $\cap$, and "complementation" is set-theoretic complementation, $^-$.

A nonempty subset J of I_n determines an element of $\mathcal{B}$ by taking the sum of the atoms of $\mathcal{B}$ having indices in the set J. For example, if $J = \{1, 3, 5\}$, then we get the element

$$x = a_1 + a_3 + a_5$$

of $\mathcal{B}$. In general, we can denote the element of $\mathcal{B}$ determined by an arbitrary subset J of I_n by

$$x = \sum_{i \in J} a_i,$$

provided we adopt the convention that the "empty sum" adds to 0:

$$\sum_{i \in \emptyset} a_i = 0.$$

Define an isomorphism

$$f \colon \mathcal{B} \to P(I_n)$$

as follows: Given a subset J of I_n, and an element

$$x = \sum_{i \in J} a_i$$

of $\mathcal{B}$, set

$$f(x) = J.$$

f is well-defined, since, by Theorem 3.13.1, each element of $\mathcal{B}$ can be expressed uniquely as a sum of atoms. (By the convention, this now includes 0.) f has an inverse

$$g \colon P(I_n) \to \mathcal{B}$$

defined by

$$g(J) = \sum_{i \in J} a_i.$$

Thus, f is a bijection, since it has an inverse. In order to see that f is an isomorphism, we must show that f preserves sums, products, and complements.

Two key observations are that if J and K are subsets of I_n, then

$$\sum_{j \in J} a_j + \sum_{k \in K} a_k = \sum_{i \in J \cup K} a_i$$

and

$$\left(\sum_{j \in J} a_j \right) \cdot \left(\sum_{k \in K} a_k \right) = \sum_{i \in J \cap K} a_i.$$

The first follows from the Idempotent Law $x+x=x$, since, in the left-hand sum, if $i \in J \cap K$, then, after combining like terms, we get the summand $a_i + a_i = a_i$. That is, after simplifying using the Idempotent Law, we get a term a_i for each i in $J \cup K$, and no others.

The second follows from the Idempotent Law $x \cdot x = x$ and Theorem 3.11.1. After using the Distributive and Associative Laws, the left-hand side is a sum of terms $a_j a_k$, where j is in J and k is in K. Since the a_j's are atoms, Theorem 3.11.1 says that the only nonzero terms are terms in which $j = k$. This only occurs when $j = k \in J \cap K$, and, in this case, $a_j a_j = a_j$.

Thus, if $x = \sum_{j \in J} a_j$, $y = \sum_{k \in K} a_k$ are arbitrary elements of $\mathcal{B}$, then

$$x + y = \sum_{i \in J \cup K} a_i \quad \text{and} \quad xy = \sum_{i \in J \cap K} a_i,$$

so that

$$f(x+y) = J \cup K = f(x) \cup f(y) \quad \text{and} \quad f(xy) = J \cap K = f(x) \cap f(y).$$

In order to see that f preserves complements, we need one further observation: If $x = \sum_{j \in J} a_j$, and if $\overline{J}$ is the complement of J in I_n, then

$$\overline{x} = \sum_{j' \in \overline{J}} a_{j'}.$$

For example, if $n = 5$ and $x = a_1 + a_4$, then $\overline{x} = a_2 + a_3 + a_5$. This follows from Property 4 in Theorem 3.4.1: After using the Distributive Law, the terms in the product

$$\left(\sum_{j \in J} a_j \right) \cdot \left(\sum_{j' \in \overline{J}} a_{j'} \right)$$

are each of the form $a_j a_{j'}$, where j is in J and j' in $\overline{J}$. Since J and $\overline{J}$ are disjoint, $j \neq j'$ for any such term, and so, by Theorem 3.11.1, the product $a_j a_{j'} = 0$. That is,

$$\left(\sum_{j \in J} a_j \right) \cdot \left(\sum_{j' \in \overline{J}} a_{j'} \right) = 0.$$

On the other hand, we have already shown in the proof of Theorem 3.13.1 (and Exercise 3.13.2) that the sum of all of the atoms is 1, so that

$$\left(\sum_{j \in J} a_j \right) + \left(\sum_{j' \in \overline{J}} a_{j'} \right) = \sum_{i \in J \cup \overline{J}} a_i = \sum_{i \in I_n} a_i = a_1 + a_2 + \cdots + a_n = 1.$$

Property 4, Theorem 3.4.1, now guarantees that if $x = \sum_{j \in J} a_j$, then

$$\overline{x} = \sum_{j' \in \overline{J}} a_i,$$

and so

$$f(\overline{x}) = \overline{J} = \overline{f(x)}.$$

□

COROLLARY 3.14.1.1. *Suppose that $\mathcal{B}_1$ and $\mathcal{B}_2$ are finite Boolean algebras having bases of the same size. Then $\mathcal{B}_1$ and $\mathcal{B}_2$ are isomorphic.*

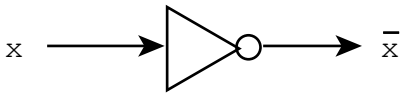
EXERCISE 3.14.1. *Show that if $\mathcal{B}$ is a finite Boolean algebra, then $\mathcal{B}$ is isomorphic to B^n for some positive integer n.*

4. Logic Gates

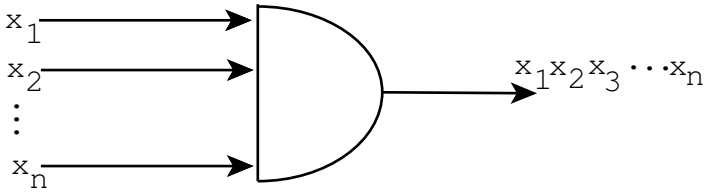
4.1. Logic Gates.

DEFINITION 4.1.1. *Boolean algebra can be used to model circuit design in an electronic device. Basic elements are* **gates**. *The three most common gates are*

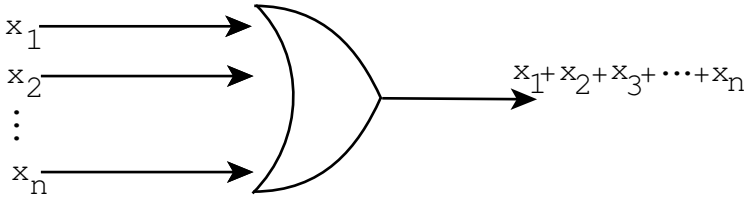
(1) Inverter:



(2) AND:



(3) OR:



Discussion

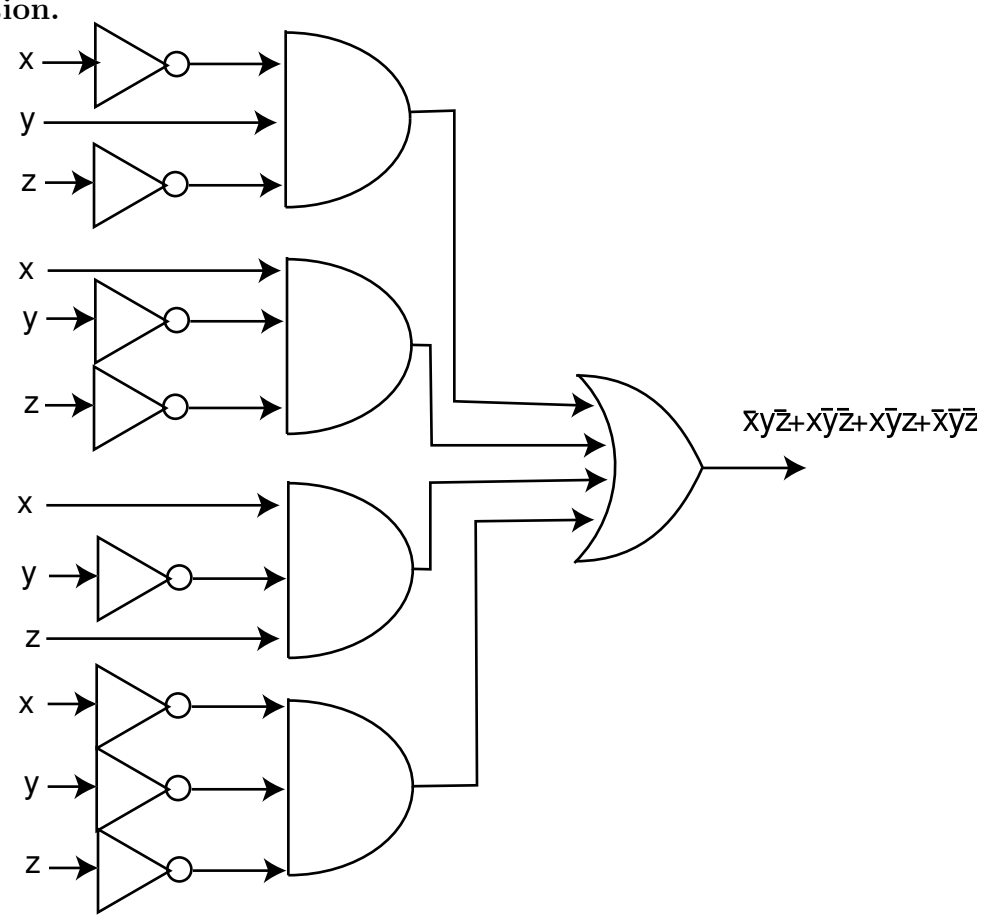
Gates are the basic building blocks of circuits. We combine gates to construct *combinatorial circuits* or *gating networks* that have as output a given Boolean expression.

4.2. Example 4.2.1.

EXAMPLE 4.2.1. *Construct a combinatorial circuit for the function F given by the following table:*

x	y	z	$F(x, y, z)$
0	*0*	*0*	*1*
0	*0*	*1*	*0*
0	*1*	*0*	*1*
0	*1*	*1*	*0*
1	*0*	*0*	*1*
1	*0*	*1*	*1*
1	*1*	*0*	*0*
1	*1*	*1*	*0*

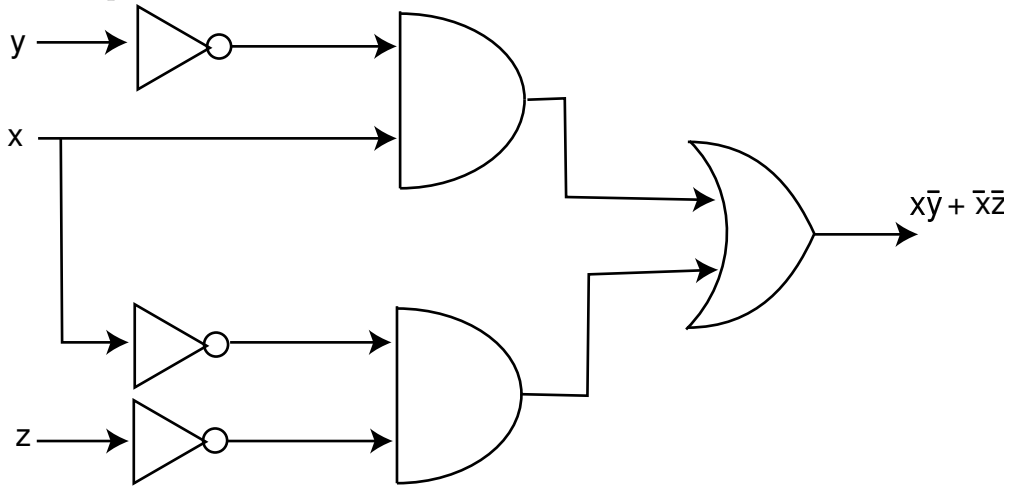
Solution.



Discussion

The figure given in the solution to example 4.2.1 comes from the disjunctive normal form for the function. The function is equivalent to $\overline{x}y\overline{z} + x\overline{y}\,\overline{z} + x\overline{y}z + \overline{x}\,\overline{y}\,\overline{z}$. This

function is also equivalent to $x\overline{y} + \overline{x}\,\overline{z}$, so the combinatorial network below also has the same output.

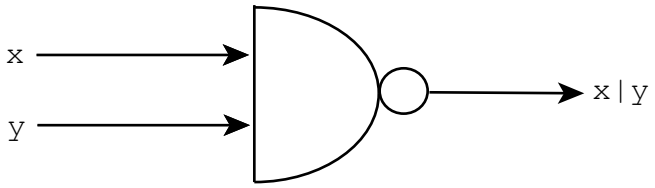


Clearly the network given here is simpler than the one given in the solution for example 4.2.1. In this section we will not be concerned with simplifying the networks, only with getting the correct output using the gates discussed. Simplification processes will be addressed in the next set of lecture notes.

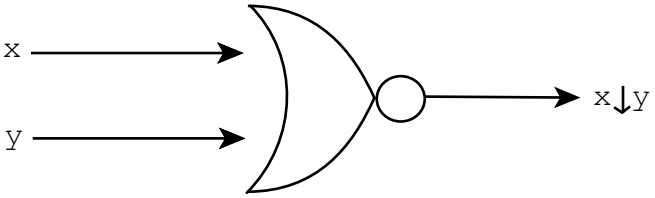
4.3. NOR and NAND gates.

Definition 4.3.1. *The NOR and NAND gates are given by*

NAND



NOR



Discussion

Recall the definition of NAND and NOR from *Representing Boolean Functions*. It was proven in that lecture that the sets $\{|\}$ and $\{\downarrow\}$ are both functionally complete. So a combinatorial circuit can always be created so that it consists only of NAND gates or only of NOR gates.

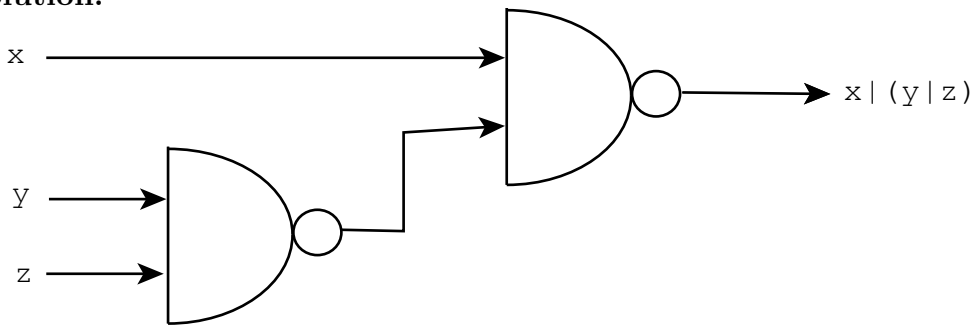
4.4. Example 4.4.1.

EXAMPLE 4.4.1. *Construct a circuit for the output of the expression*

$$\overline{x} + yz$$

using only NAND gates.

Solution.



Discussion

We use the following to get the expression in terms of NAND operators only.

$\overline{x} + yz = \overline{(x)\overline{(yz)}}$ by DeMorgan's Laws

$= (x)|\overline{(yz)}$ by the definition of NAND

$= x|(y|z)$ by the definition of NAND.

We then use this expression to make the circuit.

The expression used for the solution of Example 4.4.1 is not the only possible expression. We could have also found an expression for $\overline{x} + yz$ using the equivalences $\overline{x} = x|x$, $xy = (x|y)|(x|y)$, and $x + y = (x|x)|(y|y)$. Using these equivalences we get $\overline{x} + yz = (x|x) + (y|z)|(y|z) = \{(x|x)|(x|x)\}|\{[(y|z)|(y|z)]|[(y|z)|(y|z)]\}$. This expression is much more complex than the one used for the solution in Example 4.4.1, though!

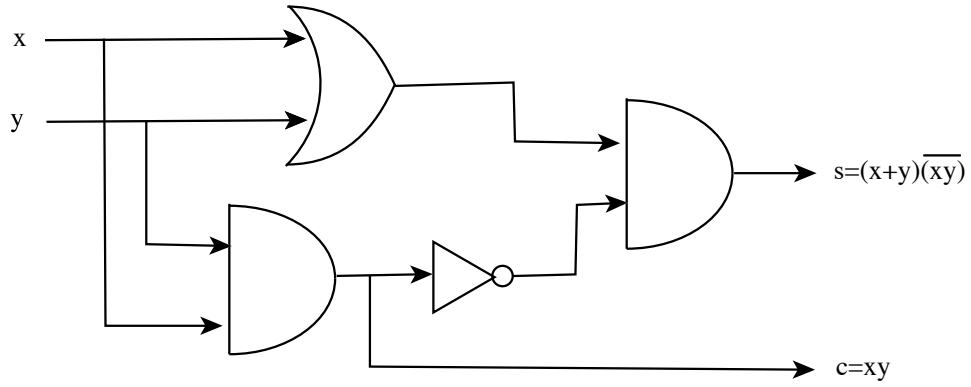
EXERCISE 4.4.1. *Construct a circuit for the output of the expression in Example 4.4.1 using only NOR gates.*

4.5. Half Adder.

DEFINITION 4.5.1. *The* **half adder** *circuit has as input the variables x and y and has as output the variables s and c given by the table.*

Input		*Output*	
x	y	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The functions $c = xy$ and $s = (x + y)\overline{(xy)}$ give the correct output for these functions. Therefore, the circuit for the half adder follows:

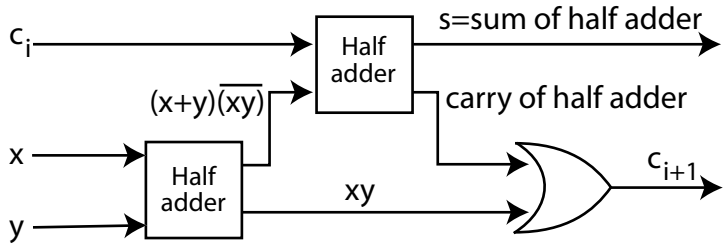


4.6. Full Adder.

DEFINITION 4.6.1. *The* **full adder** *circuit has as input the variables x, y, and c_i and has as output the variables s and c_{i+1} given by the table.*

Input			*Output*	
x	y	c_i	s	c_{i+1}
0	*0*	*0*	*0*	*0*
0	*0*	*1*	*1*	*0*
0	*1*	*0*	*1*	*0*
0	*1*	*1*	*0*	*1*
1	*0*	*0*	*1*	*0*
1	*0*	*1*	*0*	*1*
1	*1*	*0*	*0*	*1*
1	*1*	*1*	*1*	*1*

We can use the half adder to calculate the full adder as shown in the circuit below.



Discussion

Recall from *Applications of Number Theory* the algorithm for adding two integers a and b written in base 2:

Let $a = (a_{n-1}a_{n-2}\cdots a_1a_0)_2$ and $b = (b_{n-1}b_{n-2}\cdots b_1b_0)_2$, and suppose the sum of a and b is $s = (s_ns_{n-1}s_{n-2}\cdots s_1s_0)_2$. Note that if the binary expansions for a and b do not have the same number of digits we add zeros to the left of the smaller one to make them the same length.

Since $s = a + b$ we have

$a_0 + b_0 = 2c_0 + s_0$

$a_1 + b_1 + c_0 = 2c_1 + s_1$

$a_2 + b_2 + c_1 = 2c_2 + s_2$

$\vdots$

$a_{n-1} + b_{n-1} + c_{n-2} = 2c_{n-1} + s_{n-1}$

$s_{n-1} = c_{n-1}$.

At the first stage of the addition we only add the first bits of a and b and get out the values for c_0 and s_0. The half adder gives us this operation. In fact, the half adder gives us a "first stage" of each of the following additions as well. However, we must go further than adding the bits of a and b to get the carries and sums on subsequent stages because we must also consider the carry from the previous addition. The full adder gives us the carry and sum when we input the appropriate bits for a and b and the previous carry.

EXERCISE 4.6.1. *Explain each stage of the algorithm used to find the sum of the binary numbers* 1011 *and* 110 *by giving each step including which adder is used and its input and output.*

5. Minimizing Circuits

5.1. Minimizing Circuits. A circuit is minimized if it is a sum-of-products that uses the least number of products of literals and each product contains the least number of literals possible to produce the desired output.

The laws of Boolean algebra can often be used to simplify a complicated Boolean expression, particularly the sum-of-products expressions that we have used to represent Boolean functions. Any such simplification also leads to a simplification of combinatorial circuits.

5.2. Example 5.2.1.

EXAMPLE 5.2.1. $F(x, y, z) = xyz + xy\overline{z} + x\overline{y}\,\overline{z} + \overline{x}\,\overline{y}\,\overline{z}$

Mimimized Expression: $xy + \overline{y}\,\overline{z}$.

Discussion

We can simplify the expression in Example 5.2.1 as follows.

$(xyz + xy\overline{z}) + (x\overline{y}\,\overline{z} + \overline{x}\,\overline{y}\,\overline{z}) = xy(z + \overline{z}) + \overline{y}\,\overline{z}(x + \overline{x})$

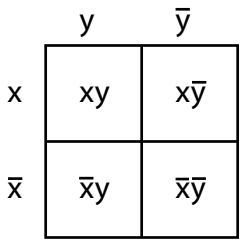
$= xy(1) + \overline{y}\,\overline{z}(1) = xy + \overline{y}\,\overline{z}$

There are various methods that do not require "ad-hoc" manipulation of variables for simplifying expressions and we cover two of the basic methods here: Karnaugh Maps and Quine McCluskey.

5.3. Karnaugh Maps.

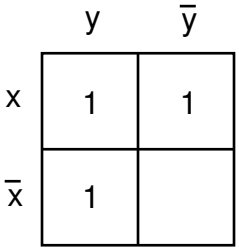
(1) For a given number of variables, n, form a table containing 1's for all possible minterms arranged so the minterms that are adjacent (left to right or above and below) differ by only a singe literal.
(2) Circle blocks of adjacent 1's. Start with the largest possible block with *number of rows and columns powers of 2.* Continue circling the largest block possible so that every 1 is within at least one block.
(3) The simplified form is the sum of the products represented by each block.

5.4. Two Variables. The Table below shows how the Karnaugh map represents the minterms for two variables.

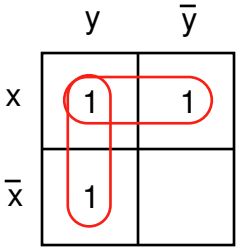


EXAMPLE 5.4.1 (Example 1: Two Variables). *Simplify* $xy + x\overline{y} + \overline{x}y$.

Solution. *First create a table with 1's corresponding to each minterm.*



Next circle blocks of size 2×2, 1×2, *and/or blocks of size* 1×1



Last, write the sum of the terms represented by the circled blocks.

$$x + y$$

Discussion

When using the Karnaugh maps to simplify functions of two variable we try to find blocks of 1's. If all 4 squares have ones, we have a 2×2 block we circle. Otherwise look for 1×2 blocks of ones. Circle any block of this size possible. If there is a 1 not covered, see if it is in a 1×2 block. If it is circle the block, if not circle the one. Continue in this manner until all the ones are circled.

Now, suppose the blocks representing xy and $x\overline{y}$ are circled. The product that corresponds to this block is x since $xy + x\overline{y} = x(y + \overline{y}) = x$. A method of determining the product corresponding to a block is to look for the literal(s) that are the same in every entry in that block.

5.5. Three Variables. The Table below shows how the Karnaugh map represents the minterms for three variables.

	yz	yz̄	ȳz̄	ȳz
x	xyz	xyz̄	xȳz̄	xȳz
x̄	x̄yz	x̄yz̄	x̄ȳz̄	x̄ȳz

The Karnaugh maps may be set up with the variables in different arrangements than the one in the above chart, but they *must* be set up so that adjacent squares differ by only one literal. Notice the entries in the left column differ from the ones in the right column by a single variable. We consider the top left adjacent to the top right and the bottom left adjacent to the bottom right. If you imagine rolling the chart and taping the red edges you have a tube which best represents this Karnaugh map.

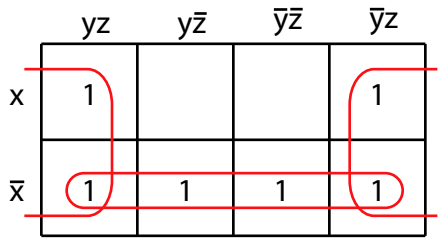
EXAMPLE 5.5.1. *Example 2: Three Variables*

Simplify $xyz + x\overline{y}z + \overline{x}yz + \overline{x}y\overline{z} + \overline{x}\,\overline{y}\,\overline{z} + \overline{x}\,\overline{y}z$.

Solution*: First create a table with 1's corresponding to each minterm.*

	yz	yz̄	ȳz̄	ȳz
x	1			1
x̄	1	1	1	1

Next circle blocks of size 2×4, 1×4, 2×2, 1×2 *and/or blocks of size* 1×1



Last, write the sum of the terms represented by the circled blocks.

$$z + \overline{x}$$

Discussion

To simplify we check as follows

(1) If all the squares have ones the function simplifies to 1.
(2) Check for blocks of size 2×2, 4×1, 2×1, or 1×1. Circle the largest block you can find.
(3) If there is a 1 not circled, find the largest possible block containing it and if possible try to use a block that contains other 1's that are not circled.
(4) continue until all the 1's have been circled.
(5) Write down the sum of the products represented by the circled blocks.

5.6. Four Variables. The Table below shows how the Karnaugh map represents the minterms for four variables.

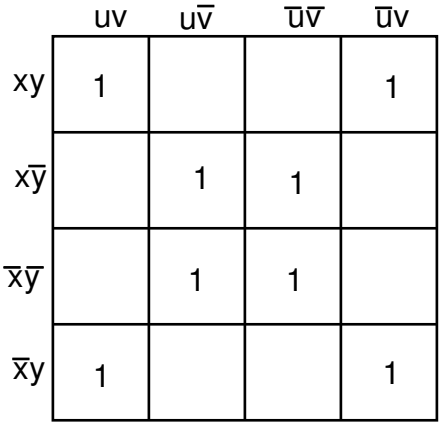
	uv	uv̄	ūv̄	ūv
xy	xyuv	xyuv̄	xyūv̄	xyūv
xȳ	xȳuv	xȳuv̄	xȳūv̄	xȳūv
x̄ȳ	x̄ȳuv	x̄ȳuv̄	x̄ȳūv̄	x̄ȳūv
x̄y	x̄yuv	x̄yuv̄	x̄yūv̄	x̄yūv

As with the map with three variables, we consider the squares on the left adjacent to the corresponding ones on the right. In addition we wish to consider the top

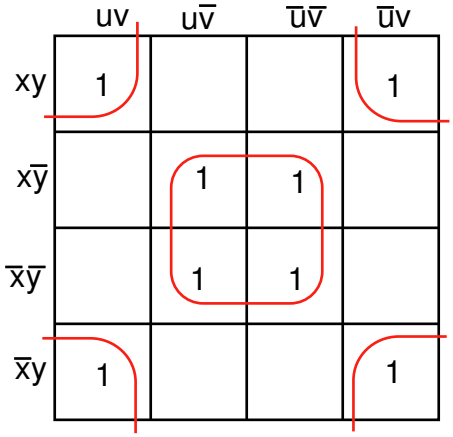
row entries adjacent to the corresponding entries in the bottom row. This may be visualized by rolling the chart so we tape the red edges together to create a tube. Now curve the tube around so the blue edges may be taped together to create a donut shape (also called a torus).

EXAMPLE 5.6.1 (Example 3: Four Variables). *Simplify* $xyuv + xy\overline{u}v + x\overline{y}u\overline{v} + x\overline{y}\,\overline{u}\,\overline{z} + \overline{y}\,\overline{y}u\overline{v} + \overline{y}\,\overline{y}\,\overline{u}\,\overline{z} + \overline{x}yuv + \overline{x}y\overline{u}v$.

Solution. *First create a table with 1's corresponding to each minterm.*



Next circle blocks of size 4×4, 2×4, 1×4, 2×2, 1×2 *and/or blocks of size* 1×1



Last, write the sum of the terms represented by the circled blocks.

$$yv + \overline{y}\,\overline{v}$$

Discussion

This time we check for blocks of size 4×4, 2×4, 2×2, 4×1, 2×1, or 1×1.

The Karnaugh maps begin to lose their ease of use when we move to more variables. It is still possible to use Karnaugh maps for four variables, but the adjacent squares are more difficult to visualize.

EXERCISE 5.6.1. *Draw Karnaugh maps for five variables and identify which blocks are adjacent.*

5.7. Quine-McCluskey Method. The Quine-McCluskey method is a method used for simplifying the conjunctive normal form of a Boolean expression. This method is easier to program than the Karnaugh Maps.

EXAMPLE 5.7.1. *Simplify the expression*

$$xyuv + xy\overline{u}v + x\overline{y}u\overline{v} + x\overline{y}\,\overline{u}\,\overline{v} + \overline{x}\,\overline{y}u\overline{v} + \overline{x}\,\overline{y}\,\overline{u}\,\overline{v} + \overline{x}yuv$$

Solution.

(1) Represent each minterm by a binary string. Use 1 for the variable and 0 for the complement:
(2) Make a table listing the minterms, the binary strings, and the number of 1's in the stings. Enumerate the list.

	Minterm	String	no. of ones
1	$xyuv$	1111	4
2	$xy\overline{u}v$	1101	3
3	$\overline{x}yuv$	0111	3
4	$x\overline{y}u\overline{v}$	1010	2
5	$\overline{x}\,\overline{y}u\overline{v}$	0010	1
6	$x\overline{y}\,\overline{u}\,\overline{v}$	1000	1
7	$\overline{x}\,\overline{y}\,\overline{u}\,\overline{v}$	0000	0

(3) Find the strings that differ by only one bit. Replace the different bit with a dash and remove the variable from the product that corresponds to that bit.
(4) Create a new table by listing the combined product, the new product, and the binary string.

	Product	String
$(1,2)$	xyv	$11-1$
$(1,3)$	yuv	-111
$(4,5)$	$\overline{y}u\overline{v}$	-010
$(4,6)$	$x\overline{y}\,\overline{v}$	$10-0$
$(5,7)$	$\overline{x}\,\overline{y}\,\overline{v}$	$00-0$
$(6,7)$	$\overline{y}\,\overline{u}\,\overline{v}$	-000

(5) Use the previous table to continue to reduce the products by finding strings that differ by one bit.

	Product	String
$((4,5),(6,7))$	$\overline{y}\,\overline{v}$	$-0-0$

(6) Since there are no bit strings in the last table that differ by a single bit we are ready to find the simplified expression.
(7) From the last table we have the string $\overline{y}\,\overline{v}$. This simplified the minterms numbered 4 through 7. From the table before we look for strings that simplify the minterms numbered 1 through 3. We use xyv and yuv to "cover" the minterms 1, 2, and 3. The simplified form is

$$\overline{y}\,\overline{v} + xyv + yuv$$

Discussion

Notice that a string with a certain number of 1's will have exactly one more or one less 1 if one of the bits are changed. Thus when we are looking for bit strings that differ by only one bit we only need to look at the bits that have exactly one more or one less 1. This is the only reason we add a column with the number of 1's in it. We could also add this column in the later tables.

Combining $(4,5)$ and $(6,7)$ gives us the same string as we get by combining $(4,6)$ and $(5,7)$, so we only use one of these combinations. Any time the numbers are the same we will obtain the same string.

We continue creating tables until we get to a table where none of the bit strings in that table differ by a single bit (including the dashes). Once we have all the tables created we need to "cover" all the minterms. In this example, we start with the last table and use the strings given in that table to cover the minterms numbered 4, 5, 6, 7. Since this was the only product in the last table we now move to the second to the last table and look one or more products that cover the remaining minterms, 1, 2, and 3. Since there is a product that covers 1 and 2 we will use it. Now we have to cover 3. Any product that covers 3 will do, but there is only one choice in this table so we use it. If there had not been a product in the second to the last table that covered 3, we would move up to the previous table.

It is possible to an expression to have more than one minimal expression. It is also possible that there are several choices of strings from a given table that could be used to cover the minterms. We chose the combinations that use the fewest possible strings to cover the most minterms.