Appendix A

Math Review

This book does not expect that the reader has any more mathematical background than college algebra. That is, you should already be comfortable with algebraic expressions, polynomials, exponents and logarithms, functions, etc. For some of you, much time has passed since you learned this material. To ensure that you can handle discrete structures, reading this appendix is recommended. Note that some college algebra topics are covered in the main body of this book; that is, you won't find all of them here.

A.1 Fractions

Fractions do not appear often in discrete structures, but often enough that reviewing – and practicing – the basics is a good idea. Students lose many points by performing manual manipulations of fractions carelessly.

Definition 71: Common Fraction

A *common fraction*[1] is the quotient of two integers x (the *numerator*) and y (the *denominator*), written x/y or $\frac{x}{y}$, where the denominator is not zero.
(LaTeX: `\frac{x}{y}` produces $\frac{x}{y}$.)

---

common fraction

    In the form x/y, the / is commonly known as a *forward slash*, but may also be known as a *solidus*, *virgule*, or even *separatrix*,[2] depending on the context

---

[1]Common fractions are also known as *vulgar fractions*. No kidding!
[2]Still not kidding. Who says appendices have to be boring?

in which it is used. The little horizontal line in $\frac{x}{y}$ is a *vinculum*.[3]

<div style="border:1px solid green">

Table 21: Common Fraction Equalities

$\frac{x}{z} + \frac{y}{z} = \frac{x+y}{z}$	$\frac{x}{w} + \frac{y}{z} = \frac{xz+wy}{wz}$
$\frac{x}{z} - \frac{y}{z} = \frac{x-y}{z}$	$\frac{x}{w} - \frac{y}{z} = \frac{xz-wy}{wz}$
$\left(\frac{x}{z}\right)\left(\frac{y}{z}\right) = \frac{xy}{z^2}$	$\left(\frac{x}{w}\right)\left(\frac{y}{z}\right) = \frac{xy}{wz}$
$\frac{\frac{x}{z}}{\frac{y}{z}} = \left[\left(\frac{x}{z}\right)\left(\frac{z}{y}\right) =\right] \frac{x}{y}$	$\frac{\frac{x}{w}}{\frac{y}{z}} = \frac{xz}{wy}$

(a) Equal Denominators   (b) Unequal Denominators

</div>

Table 21 shows the basics of fraction addition, subtraction, multiplication, and division.

Example 197 demonstrates several fraction operations, including removing common factors to reduce the result to lowest terms.

<div style="border:1px solid brown">

Example 197:

Problem: Evaluate $\frac{\frac{1}{6}+\frac{3}{8}}{\frac{5}{12}} - \frac{4}{5}$ and, if necessary, reduce the result.

Solution: $\frac{\frac{1}{6}+\frac{3}{8}}{\frac{5}{12}} - \frac{4}{5} = \frac{\frac{8}{48}+\frac{18}{48}}{\frac{5}{12}} - \frac{4}{5} = \frac{\frac{26}{48}}{\frac{5}{12}} - \frac{4}{5} = \frac{\frac{26}{48}}{\frac{20}{48}} - \frac{4}{5} = \left(\frac{26}{48}\right)\left(\frac{48}{20}\right) - \frac{4}{5}$

$= \frac{26}{20} - \frac{4}{5} = \frac{26}{20} - \frac{16}{20} = \frac{10}{20} = \frac{1}{2}$

</div>

A.2   Rational Numbers

Fractions consisting of integers lead us directly to numbers that either can, or cannot, be expressed as common fractions.

---

[3]'Vinculum' was also used as the name of a processing unit in a Borg vessel in *Star Trek: Voyager*.

> **Definition 72: Rational Number**
>
> A value that can be expressed as the ratio of two integers is a *rational number*.

rational number

Notice that the word "ratio" appears in both the definition and within the word "rational." This makes the definition easy to remember.

How can we determine whether or not decimal values, such as 3.375, are rational? If the fractional part (the part to the right of the decimal point) is a terminating sequence of digits (that is, if the fractional part does not continue indefinitely), then the entire value can be expressed as a fraction.

> **Example 198:**
>
> *Problem*: Demonstrate that the decimal value 3.375 can be expressed as a rational number.
>
> *Solution*: 3.375 is a representation of the sum of two fractions: $\frac{3}{1}$ and $\frac{375}{1000}$. $\frac{3}{1} + \frac{375}{1000} = \frac{3000}{1000} + \frac{375}{1000} = \frac{3375}{1000}$. Thus, 3.375 is a rational number, because we can express it exactly by the (unreduced) fraction $\frac{3375}{1000}$.

Decimal values with repeating fractional parts (such as $17.4444\ldots$ or $0.87262626\ldots$)[4] are also rational numbers, as Example 199 illustrates.

> **Example 199:**
>
> *Problem*: Demonstrate that $0.87262626\ldots$ can be represented as the ratio of two integers.
>
> *Solution*: Let $x = 0.87262626\ldots$. We can multiply both sides by 100, which leaves the fractional part consisting of only the repeating digits ($100x = 87.262626\ldots$). Multiplying by 10,000 leaves the fractional part with the same repeating pattern: $10000x = 8726.262626\ldots$. In both cases, the fractional part is representing the same value. By sub-

---

[4] To highlight the repeating portions of such numbers, mathematicians use an overbar (LaTeX: \overline{}) to indicate the repeating part. For example, $0.87262626\ldots$ can be represented as $0.87\overline{26}$. This 'overbar' can also be called a vinculum.

tracting the left and right sides of the $100x$ equality from those of the $10000x$ equality, we learn that $9900x = 8639$, or $x = \frac{8639}{9900}$. That is, $0.87262626\ldots = \frac{8639}{9900}$.

You may be surprised to read that 3.375 (and any so-called 'terminating' decimal value) is actually a repeating decimal! This is correct because we can append infinitely many zeros to it $(3.37500000\ldots)$ without changing its value; thus, it is technically a repeating decimal. As a practical matter, writing such values with "000..." at the end is unnecessary if we accept that, notationally, 3.375 represents $3.375000\ldots$.

irrational

What about values that cannot be expressed as the ratio of two integers? They are called *irrational* numbers. Examples include the values π $(3.1415926\ldots)$, e $(2.71828\ldots)$, and $\sqrt{2}$ $(1.4142\ldots)$. If a decimal value's fractional part is terminating or repeating, the value is rational. Otherwise, the value is irrational.

A.3   Set Basics

Set notation and set operators are used throughout discrete structures. Students frequently know the basics before studying discrete structures, which is why we cover them in this appendix. More advanced set topics are covered in the main part of this book.

set

Definition 73: Set

A *set* is an unordered collection of unique objects.

We show the elements of a set by listing them, comma-separated, inside of 'curly brackets' (LaTeX: \{ and \}), a.k.a. 'braces.'[5] We'll use upper-case letters to label sets for later reference.

Example 200:

A rainbow's colors are red, orange, yellow, green, blue, indigo, and violet.[6] The set of a rainbow's colors can be written as:

---

[5] If you can get over the association with the appliances of teenage dental torture.

> R = {red, orange, yellow, green, blue, indigo, violet}.
>
> Because order doesn't matter to sets, there are $7! = 5,040$ acceptable ways to list the elements of set R.

As the definition says, the elements of a set must be unique. That is, an element may appear in a specific set at most once.

Example 201:

The set of letters in the word 'letter': S = {l, e, t, r}

Note that, if you need to have a set that allows duplicates, the concept has already been invented. Such sets are known as *multisets* or *bags*. *multiset, bag*

A.3.1   Set Notation

Much notation is associated with sets. Here are the basics:

- To indicate that an element is a member of a set, use $\in$ (LATEX: `\in`). Referring to Example 201, e $\in$ S. To negate this, use $\notin$ (LATEX: `\not\in`), as in f $\notin$ S.

- A set with no members, known as the *empty set*, is represented by a   *empty set*
set of curly brackets with nothing in-between ({ }) or by the symbols $\emptyset$ (LATEX: `\emptyset`) or $\varnothing$ (LATEX: `amssymb`'s `\varnothing`).[7]

- The elements of some sets are difficult (sometimes impossible!) to enumerate. To define such sets, we use *set builder* notation. The format: *set builder*
{variable(s) | description of set membership}. (LATEX: `\mid` produces the vertical line.)

- The entire collection of set elements available in a given situation is known as the *universal set* , which we will represent with the symbol $\mathcal{U}$ *universal set*
(LATEX: `\mathcal{U}`).

---

[6]The first letters of which, in order, form a name famous to schoolchildren if not to history: Roy G. Biv. But we're talking sets here, not sequences!

[7]`amssymb` is an optional package for LATEX that supplies additional math symbols

---

Example 202:

e ∈ S	Show that e is a member of set S
f ∉ S	Show that f is not an element of S
{ t | t is a positive multiple of 3}	$\{3, 6, 9, 12, \ldots\}$
{ α | α is a UNICODE symbol}	There are 110,181 in UNICODE 6.1!
$\mathcal{U} = \{0, 1, 2, 3\}$	All possible Base 4 digits

A.3.2   Fundamental Set Operators

We will cover four set operators here. Others will introduced in the body of this book as they are needed.

- Union (∪, LaTeX: `\cup`): A binary operator that combines the elements from the two operand sets into a single result set. If an element appears in both operand sets, it will appear only once in the result set.

- Intersection (∩, LaTeX : `\cap`): Like union, intersection is a binary operator. Unlike union, the result set contains the elements that the operand sets have in common.

- Difference (−, LaTeX : `-`): In S − T, the result set contains those elements of S that are **not** also in T. Alternatively, think of the result set temporarily containing all elements of S. Remove from the result set all items also found in the set $S \cap T$. The remaining elements comprise S−T. Some people use the term *relative complement* instead of *difference*, and use \ (LaTeX: `\backslash`) instead of − as the symbol.

relative complement

- Complement ($\overline{set}$, LaTeX : `\overline{set}`): In a given setting, we know that the elements of a set are limited by the universal set. The complement of a set S, $\overline{\text{S}}$, is the set of all elements of the universal set that are not members of S. That is, $\overline{\text{S}} = \mathcal{U} - \text{S}$.

- Cardinality (\| *set* \|, LaTeX: `$\mid set \mid$`): The cardinality of a set is the quantity of items in the set.

Example 203:

Let $\mathcal{U} = \{0, 1, 3, 4, 5, 7, 8, 9\}, \mathrm{D} = \{1, 3, 5, 7, 9\}, \mathrm{E} = \{4, 8\}$, and $\mathrm{F} = \{1, 4, 7, 9\}$.

    (a) $\mathrm{E} \cup \mathrm{F} = \{1, 4, 7, 8, 9\}$ (f) $\mathrm{F} - \mathrm{D} \quad = \{4\}$

    (b) $\mathrm{D} \cap \mathrm{F} = \{1, 7, 9\}$ (g) $\mathrm{E} - \mathrm{D} \quad = \{4, 8\} \,(= \mathrm{E})$

    (c) $\overline{\mathrm{F}} \quad\quad = \{0, 3, 5, 8\}$ (h) $\mathrm{E} - \mathrm{E} \quad = \emptyset$

    (d) $\overline{\emptyset} \quad\quad = \mathcal{U}$ (i) $\mid \mathrm{E} \mid \quad\quad = 2$

    (e) $\mathrm{F} \cap \overline{\mathrm{D}} = \{4\}$ (j) $\mid \mathrm{E} - \mathrm{E} \mid = 0$

The fact that $\mathrm{F} - \mathrm{D} = \mathrm{F} \cap \overline{\mathrm{D}}$ in Example 203 is not merely a coincidence, it is also true for all pairs of sets from the same universal set. That is, for any sets A and B whose elements are drawn from the same set $\mathcal{U}$, $\mathrm{A} - \mathrm{B} = \mathrm{A} \cap \overline{\mathrm{B}}$. Operators, such as set difference and complement, that can be defined in terms of primitive operators are sometimes called *derived operators*.   *derived operator*

Set operators are subject to rules of *precedence* and *associativity* as are   *precedence* any other operators. You are likely familiar with these concepts from the   *associativity* programming language(s) you know. For example, in Java multiplication has precdence over addition, telling us that, after the evaluation of the statement `a = 4 + 5 * 6;`, a receives the value 34 instead of 54. While you can find precedence and associativity suggestions for set operators, no canonical rules exist.[8] Rather than adding to the confusion, we will simply use parentheses when necessary to order the operators in our set expressions.

A.3.3 Set Visualization with Venn Diagrams

To make the set operators easier to understand, *Venn Diagrams* are often used.   *venn diagram* As Figure A.1 suggests, they are found on the internet nearly as frequently as they are in discrete structures books.

The enclosing rectangle represents the universal set. Frequently, the label $\mathcal{U}$ is placed in the upper-right corner as a reminder. Each set of interest within the universe is represented with a labeled circle or other (usually curved) 'blob.' These circles are positioned to overlap in a way that produces a region for every possible relationship between the sets. Figure A.2 shows common Venn diagram representations for two, three, and four sets.

---

[8]Some applications avoid the issue entirely. The database language SQL handles precedence by considering all set operators to have the same precedence.
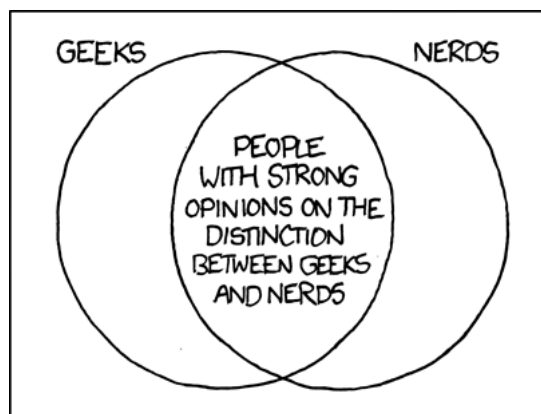
Figure A.1: Geeks and Nerds Venn Diagram (http://xkcd.com/747/). Credit: Randall Munroe, xkcd.com.
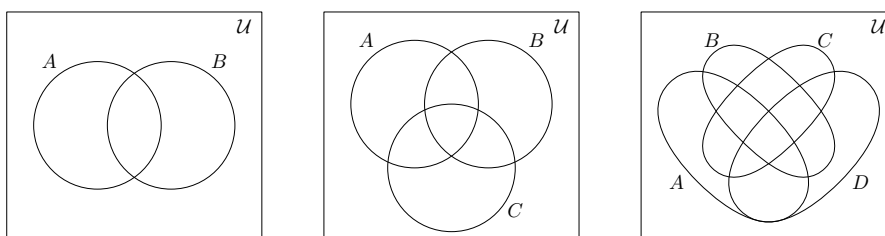

Figure A.2: Standard 2-set, 3-set, and 4-set Venn Diagrams

By shading or otherwise highlighting the appropriate regions, visualizations of the results of set operations can be created. Figure A.3 shows Venn diagram representations of the results of some of the expressions in Example 203. While the diagrams with the bold colors are eye-catching, the versions with the cross-hatching are easier to create by hand by students taking timed quizzes or exams.

A.3.4   Notations for Sets of Numbers

A reasonably well-accepted collection of labels exist that mathematicians use to represent collections of numbers. Here are the labels used in this book:
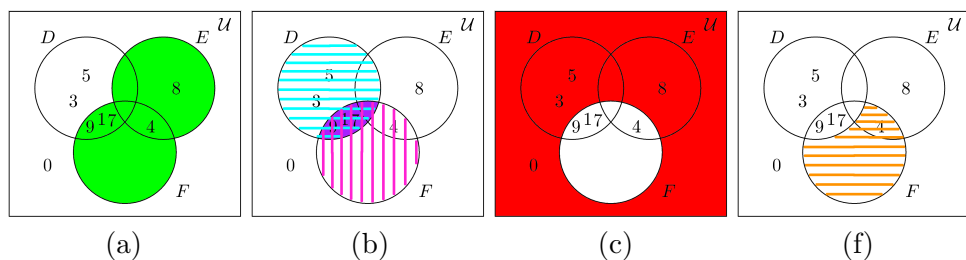
Figure A.3: Venn diagrams of (a) $E \cup F$, (b) $D \cap F$, (c) $\overline{F}$, and (f) $F - D$. Labels correspond to those of Example 203.

Table 22: Sets of Numbers

Symbol	Name	Set Representation
$\mathbb{Z}$	All Integers	$\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$\mathbb{Z}^+, \mathbb{N}^+$	Positive Integers	$\{1, 2, 3, \ldots\}$
$\mathbb{Z}^*, \mathbb{N}_0$	Non-negative Integers	$\{0, 1, 2, 3, \ldots\}$
$\mathbb{Z}^-$	Negative Integers	$\{\ldots, -3, -2, -1\}$
$\mathbb{Z}^{\text{even}}$	Even Integers	$\{\ldots, -4, -2, 0, 2, 4, \ldots\}$
$\mathbb{Z}^{\text{odd}}$	Odd Integers	$\{\ldots, -3, -1, 1, 3, \ldots\}$
$\mathbb{Q}$	Rational Numbers	$\{\frac{a}{b} \mid a, b \in \mathbb{Z} \text{ and } b \neq 0\}$
$\overline{\mathbb{Q}}$	Irrational Numbers	$\{i \mid i \notin Q \text{ and } i \notin \text{Imaginary}\}$
$\mathbb{R}$	Real Numbers	$\{\mathbb{Q} \cup \overline{\mathbb{Q}}\}$

Some explanations are in order. $\mathbb{Z}$ (LaTeX: `\mathbb{Z}`) is used for integers because the German word for "integers" is "zahlen," and many influential mathematicians were German. (That, and several sets of numbers have names that being with "I" in English, specifically "integer," "irrational," and "imaginary.") $\mathbb{Z}^{\text{even}}$ and $\mathbb{Z}^{\text{odd}}$ are non-standard notations, but accepted symbols for these sets do not exist and $\mathbb{Z}^{\text{even}}$ and $\mathbb{Z}^{\text{odd}}$ have the advantange of being easily understood. Finally, $\overline{\mathbb{Q}}$ is as accepted a symbol for the irrational numbers as there is. We have just seen that the overline represents the complement of a set, thus $\overline{\mathbb{Q}}$ represents "not rational," or "irrational."

The relationships between most of these can be shown in a non-too-messy Venn diagram; see Figure A.4.

You may be wondering about the omission of natural numbers, a set represented by the symbol $\mathbb{N}$. The problem: What are the natural numbers, exactly? Most mathematicians now consider zero to be 'natural,' but others
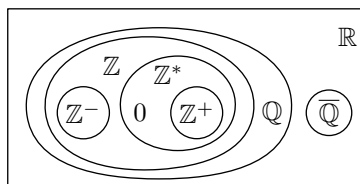
Figure A.4: A Visualization of Number Set Relationships

prefer the traditional interpretation, where a symbol isn't needed until there is something to count. To avoid confusion, we will ignore $\mathbb{N}$ and will use $\mathbb{Z}^*$ when we want natural numbers including zero and $\mathbb{Z}^+$ when zero is excluded.

A.4   Associativity, Friends and a Distant Cousin

We know that algebraic expressions consist mostly of operands (values), operators (unary as well as binary), and grouping symbols (such as parentheses). Set and logical expressions, both of which are common in discrete structures, have these components as well.

To correctly evaluate an expression, we need to know the ways in which its components can be rearranged without changing the eventual evaluation of the expression. The properties that govern these rearrangements include associativity, commutativity, and distributivity. We include transitivity in this section, too, even though it covers a rather different area than do the others.[9]

A.4.1   Associativity

The associative property covers groupings across binary operators (such as subtraction in arithmetic and intersection in sets).

associativity

> **Definition 74: Associativity**
>
> Assume that $\triangle$ is a binary operator that can be applied to values a, b, and c. The operator $\triangle$ is *associative* if $(a\triangle b)\triangle c = a\triangle(b\triangle c)$ for all acceptable values a, b, and c.

---

[9]It also ends in "tivity"; for us, that's reason enough to put it here!

The symbol $\triangle$ in Definition 74 (and the geometric shapes used in the upcoming definitions in this subsection) represents any acceptable binary operator, not a specific one. More formally, associativity is known as the *associative property* and the expression is known as the *associative law.*

Not all operators are associative, as Example 204 shows.

Example 204:

Addition over integers is an associative binary operator: $(9 + 13) + 5 = 9 + (13 + 5)$

Union over sets is also associative: $G \cup (H \cup I) = (G \cup H) \cup I$

Subtraction over integers is **not** associative: $(2 - 3) - 4 \neq 2 - (3 - 4)$.

Be aware that you cannot select just any three values to show that an operator is associative. Consider $(0-0)-0 = 0-(0-0)$. This is certainly true, but that relationship is not generally true. Thus, subtraction over integers is not associative.

A.4.2   Commutativity

Commutativity covers positioning of operands around binary operators.

Definition 75: Commutativity   *commutativity*

Assume that $\bowtie$ is a binary operator that can be applied to values d and e. The operator $\bowtie$ is *commutative* if $d \bowtie e = e \bowtie d$ for all acceptable values d and e.

Example 205:

Multiplication over integers is commutative: $8 * 5 = 5 * 8$

Intersection over sets is, too: $J \cap K = K \cap J$

Set difference is **not** commutative: $L - M \neq M - L$

A.4.3   Distributivity

Unlike associativity and commutativity, distributivity requires two distinct operators.

distributivity

> **Definition 76: Distributivity**
>
> Assume that $\square$ and $\diamond$ are binary operators that can be applied to values x, y, and z. $\square$ is *distributive* over $\diamond$ if both of these conditions hold:
>   (a) $x \square (y \diamond z) = (x \square y) \diamond (x \square z)$ ($\square$ is *left distributive* over $\diamond$), **and**
>   (b) $(y \diamond z) \square x = (y \square x) \diamond (z \square x)$ ($\square$ is *right distributive* over $\diamond$),
> for all acceptable values x, y and z.

Notice that the difference between (a) and (b) in Definition 76 is due to the commutativity of the operator represented by $\square$. So long as $\square$ is commutative over the values, there is no behavioral distinction between left and right distributivity.

> **Example 206:**
>
> With integers, multiplication is distributive over addition, because, for example, $4 * (7 + 9) = (4 * 7) + (4 * 9)$ and $(7 + 9) * 4 = (7 * 4) + (9 * 4)$ (either way, $64 = 64$).
> However, addition is not distributive over multiplication: $3 + (5 * 6) \neq (3 + 5) * (3 + 6)$ ($33 \neq 72$).

A.4.4   Transitivity

As mentioned at the top of this section, transitivity does not fit well with the other properties. It is still important in discrete structures, and even though it should be known to you already, to be complete we should mention it in this appendix somewhere. This section is as good a place as any to cover it.

Definition 77: Transitivity *transitivity*

A binary relationship denoted by a relational operator $\odot$ is *transitive* if, whenever $a \odot b$ and $b \odot c$ are both true, then $a \odot c$ is also true.

Binary *relational operators* are those that compare two values. For instance, in mathematics '$<$' is used to indicate that a number n is less than another number m, as in $n < m$. If so, the operator evaluates to true; otherwise, false. As it happens, '$<$' is transitive, as demonstrated by Example 207 and detailed later on in Table 25.

relational operator

Example 207:

(a) $-9 < -3$ and $-3 < 5$. By transitivity of $<$, we know that $-9 < 5$.

(b) $(3 + 5) = 8$ and $8 = 2^3$. Because $=$ is transitive, we know that $(3 + 5) = 2^3$.

Example 208:

Consider the game Rock-Paper-Scissors.[10] Paper beats rock and rock beats scissors. If the relationship "beats" were transitive, then paper would beat scissors, but of course it does not. Thus, the "beats" operator is not transitive in the context of Rock, Paper, Scissors.

A.5 Properties of Inequalitites

Many people are fuzzy on what an "inequality" is, which makes the concept a good place to start. An *inequality* is a comparison between two values that can be true when the values are not equal. Thus, the relational operators $\neq$, $<$, and $>$ are all unquestionably inequalities. The operators $\leq$ and $\geq$ are also considered to be inequalities, even though $4 \leq 4$ is true (that is, even though

inequality

---

[10]See http://en.wikipedia.org/wiki/Rock-paper-scissors for the rules.

these operators can also be true in an equality situation, we consider them to be inequalities because they can be true when the operands differ, as in $4 \leq 5$).

Tables 23, 24, and 25 detail several properties of inequalities that you should know.

Table 23: Inequality Properties of Addition and Subtraction

For real numbers a, b, and c:

	If	Then
(a)	$a < b$	$a + c < b + c$
(b)	$a \leq b$	$a + c \leq b + c$
(c)	$a > b$	$a + c > b + c$
(d)	$a \geq b$	$a + c \geq b + c$
(e)	$a \neq b$	$a + c \neq b + c$

Note: In all of these, addition can be replaced by subtraction.

Table 24: Inequality Properties of Multiplication and Division

For real numbers a, b, and c, when $c \neq 0$ and ...

. . . when c is positive:				. . . when c is negative:		
	If	Then			If	Then
(a)	$a < b$	$ac < bc$		(f)	$a < b$	$ac > bc$
(b)	$a \leq b$	$ac \leq bc$		(g)	$a \leq b$	$ac \geq bc$
(c)	$a > b$	$ac > bc$		(h)	$a > b$	$ac < bc$
(d)	$a \geq b$	$ac \geq bc$		(i)	$a \geq b$	$ac \leq bc$
(e)	$a \neq b$	$ac \neq bc$		(j)	$a \neq b$	$ac \neq bc$

Note: In all of these, multiplication can be replaced by division.

Table 25: Inequality Properties of Transitivity

For real numbers a, b, and c:

	If	Then		If	Then
(a)	$a < b$ and $b < c$	$a < c$	(e)	$a > b$ and $b > c$	$a > c$
(b)	$a \leq b$ and $b \leq c$	$a \leq c$	(f)	$a \geq b$ and $b \geq c$	$a \geq c$
(c)	$a < b$ and $b \leq c$	$a < c$	(g)	$a > b$ and $b \geq c$	$a > c$
(d)	$a \leq b$ and $b < c$	$a < c$	(h)	$a \geq b$ and $b > c$	$a > c$

Transitivity, in the context of relations, is covered in Chapter 8.

Example 209:

Let $a = 5$, $b = 5$, and $c = 8$. $a \leq b$ is true (5 is less than **or** equal to 5). $b < c$ is also true ($5 < 8$). Because we know $a \leq b$ and $b < c$, by Table 25(d) we may conclude that $a < c$ ($5 < 8$). Note that although we can also say that $5 \leq 8$ is true, stating that $5 < 8$ is more accurate, because it correctly shows the impossibility of $a = c$ being true when $a \leq b$ and $b < c$.

Example 210:

Problem: Demonstrate that $2m > p$ when $6n \geq p$ and $m > 3n$, where all variables are integers.

Solution: By Table 24(c), we can multiply both sides of $m > 3n$ by 2 to produce $2m > 6n$. Table 25(g) allows us to conclude that $2m > p$ because $2m > 6n$ and $6n \geq p$.

A.6   Summation and Product Notations

Expressing lengthy sums (such as $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8$) requires a lot of writing, or some sort of abbreviation technique (perhaps $1 + 2 + \ldots + 8$). Such an abbreviation could be misinterpreted. In this case, is $1 + 2 + \ldots + 8$ representing $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8$ or $1 + 2 + 4 + 8$ (the sum of the first four positive powers of 2)? To help avoid confusion, mathematicians use summation and product notations.

A.6.1   Summation Notation

summation notation   *Summation notation* provides an unambiguous way to express the idea that a long sequence of terms that follow a computational pattern should be added together. Here is an example:

$$\sum_{i=0}^{3} 2^i \qquad (\text{ \LaTeX: } \texttt{\textbackslash sum\textbackslash limits\_\{i=0\}\^{}\{3\} 2\^{}\{i\}})^{[11]}$$

This example expresses the sum $1 + 2 + 4 + 8$. Here's how: The variable i takes on integer values, starting with 0 and ending with 3. For each of those four integers, we replace the i in 2^i with the integer and add together the results. Thus:

> **Example 211:**
>
> The expression $\sum_{i=0}^{3} 2^i$ expands to $2^0 + 2^1 + 2^2 + 2^3$, which is equal to $1 + 2 + 4 + 8 = 15$.

Each part of $\sum_{i=0}^{3} 2^i$ has a name:

- Σ (\LaTeX: \Sigma, for a stand-alone version) is the Greek letter (capital) sigma. (The lower case sigma, σ (\LaTeX: \sigma), is used elsewhere in mathematics, but not to represent summation.)

index of summation   - i is the *index of summation*, or just *index* for short.

---

[11]The \limits command tells \LaTeX to format the limits above and below the sigma. Discarding it produces a more compact version $\left(\sum_{i=0}^{3} 2^i\right)$, useful for in-line expressions.

- 0 is the *lower limit*.

- 3 is the *upper limit*.

- 2^i (LaTeX: `2^{i}`) is the *sequence*.

A.6.2   Product Notation

Closely related to summation notation is *product notation*, which is identical

in appearance except for a different Greek letter, Π (LaTeX: `\Pi`), the capital version of the symbol π (LaTeX: `\pi`) that represents the ratio of a circle's circumference to its diameter.

Here's the same example as above using product instead of summation notation:

$$\prod_{i=0}^{3} 2^i \qquad (\text{LaTeX: } \texttt{\textbackslash prod\textbackslash limits\_\{i=0\}^\{3\} 2\^{}\{i\}})$$

And, of course, the terms from the sequence are multiplied instead of added.

Example 212:

Changing Σ to Π in Example 211: $\prod_{i=0}^{3} 2^i = 2^0 \cdot 2^1 \cdot 2^2 \cdot 2^3 = 1 \cdot 2 \cdot 4 \cdot 8 = 64$. (LaTeX: `\cdot` produces the multiplication dot.)

A.6.3   Interpreting Sequences within Sum and Product Notation

Whether using summation or product notation, be careful with the formatting of the sequence, as Example 213 shows.

Example 213:

The summation expression $\sum_{i=1}^{2} \frac{4}{i} + i$ is interpreted as $\left(\sum_{i=1}^{2} \frac{4}{i} \right) + i$, rather than as $\sum_{i=1}^{2} (\frac{4}{i} + i)$. In most cases, the evaluation of the two versions is likely to be different. Here, $\left(\sum_{i=1}^{2} \frac{4}{i} \right) + i = \left(\frac{4}{1} + \frac{4}{2} \right) + i = 6 + i$ (the trailing

---

> i is out of the scope of the summation, and so remains untouched), while
> $\sum_{i=1}^{2}(\frac{4}{i} + i) = \left(\frac{4}{1} + 1\right) + \left(\frac{4}{2} + 2\right) = 5 + 4 = 9.$

A.6.4   Nesting Summation and Product Notations

Can the sequence of a summation include another summation, or even a product? Certainly!

The key to evaluating these nested notations is to work from the inside to the outside. That is, expand the inner-most summation or product first, then deal with the next inner-most, etc. Example 214 demonstrates.

Example 214:

Problem: Evaluate $\sum_{a=1}^{5} \sum_{b=0}^{2} (2a + b)$.

Solution: Here, $\sum_{b=0}^{2}(2a + b)$ is the inner-most summation. Don't worry about the a in the sequence; it will be dealt with when we address the outer summation.

$$\sum_{b=0}^{2}(2a + b) = (2a + 0) + (2a + 1) + (2a + 2) = 6a + 3$$

We have simplified the original nested summation to the single summation $\sum_{a=1}^{5}(6a + 3)$, which we already know how to evaluate:

$$\sum_{a=1}^{5}(6a + 3) = (6{\cdot}1 + 3) + (6{\cdot}2 + 3) + (6{\cdot}3 + 3) + (6{\cdot}4 + 3) + (6{\cdot}5 + 3)$$
$$= 9 + 15 + 21 + 27 + 33$$
$$= 105$$

Thus, $\sum_{a=1}^{5} \sum_{b=0}^{2}(2a + b) = 105.$

An observation on Example 214: We can simplify $\sum_{a=1}^{5}(6a+3)$ a bit before starting its evaluation. Doing so will save us some operations. Notice that $6a + 3 = 3(2a + 1)$. Because we are doing a summation, we can multiply the result of the summation by three instead of multiplying each sequence term by three. (You should recognize this as an application of distributivity; see

section A.4. That is, $\sum_{a=1}^{5}(6a+3) = \sum_{a=1}^{5} 3(2a+1) = 3\left(\sum_{a=1}^{5}(2a+1)\right)$.) You can simplify the summation even further by turning the "+1" into a "+5" and by pulling out the "2", but while these steps will simplify the summation, it makes the expression as a whole a bit of a mess. Try it and see for yourself!

To conclude this subsection, Example 215 examines the connection between nested summations and nested FOR loops in programming languages.

Example 215:

Problem: Write nested pseudocode FOR loops that computes $\sum\limits_{a=1}^{5}\sum\limits_{b=0}^{2}(2a+b)$.

Solution: Each Σ represents an iteration operation over the index of summation, starting with the lower limit and stopping after the upper limit is considered. The parallels between index-controlled loops (such as FOR loops) and summation notation are numerous, making FOR loops a natural stand-in for summation in a program. Apart from the summation-to-FOR-loop translation, we only need a variable to hold the sum being computed.

```
sum <- 0
for a from 1 through 5
    for b from 0 through 2
        sum <- sum + 2a + b
output sum
```

In Example 214, we considered the summations essentially independently, working inside-out. Computers do not treat the instructions representing nested FOR loops that way, but they could (especially if certain code optimizations are applied). We could manually apply the same inside-out process to our code example in Example 215 to eliminate the inner loop:

```
sum <- 0;
for a from 1 through 5
    sum <- sum + 6a + 3
output sum
```

$$\begin{array}{r r l} & 6 & \leftarrow \text{quotient (q)} \\ \text{divisor (s)} \longrightarrow \quad 4\,\big)\,\overline{27} & & \leftarrow \text{dividend (n)} \\ & \underline{24} & \\ & 3 & \leftarrow \text{remainder (r)} \end{array}$$

Figure A.5: Long Division with Component Names

A.7   Integer Division

Remember learning long division of integers in grade school? All of the numbers were integers, and most of the parts had hard-to-remember names. Figure A.5 reminds us of the parts and their names.

To save space, we can summarize the division in Figure A.5 using the representation $27/4 = 6\,\text{r}\,3$, where the 'r' means 'remainder.'[12] In discrete structures, the results (the quotient and the remainder) are both of use, but the remainder is of particular interest.

A.7.1   Modulo and Integer Division Operators

modulo

integer division

divides

The *modulo* operator, represented by %, the percent sign (LATEX: \%), is a binary operator that takes the dividend and divisor and produces the remainder of the integer division.[13] The *integer division* operator, represented by \, the back slash (LATEX: \backslash), takes the same operands but produces the quotient. Finally, the *divides* operator, represented by | (LATEX: \mid or |) also accepts the same operands, but in reverse order and returns a boolean result: True when the division results in no remainder, false otherwise. The reversing of the operands is important to remember; that is, 16 % 2 = 0 corresponds to 2 | 16, and we would read it aloud as "2 divides 16."

> **Example 216:**
>
> Some parallel examples of the modulo, integer division, and divides operators:[14]

---

[12] If you want to be old-school, you can use the ÷ symbol to represent division (LATEX: \div). It's called an *obelus* and is not, to our knowledge, a part of a Borg ship. Yet.

[13] Sometimes people refer to the modulo operator as the 'modulus' operator. 'Modulus' can mean several things, but none of them is an alternate name for modulo.

(a)	$27\%4 = 3$		(f)	$27 \setminus 4 = 6$		(k)	$4 \mid 27 =$ False	
(b)	$4\%4 = 0$		(g)	$4 \setminus 4 = 1$		(l)	$4 \mid 4 =$ True	
(c)	$4\%27 = 4$		(h)	$4 \setminus 27 = 0$		(m)	$27 \mid 4 =$ False	
(d)	$0\%5 = 0$		(i)	$0 \setminus 5 = 0$		(n)	$5 \mid 0 =$ True	
(e)	$3\%0 =$ Undef.		(j)	$3 \setminus 0 =$ Undef.		(o)	$0 \mid 3 =$ Undef.	

If any of the modulo examples from Example 216 are confusing, (c) is probably the one. Think of it this way: In $27\%4$, we ask ourselves how many instances of the divisor (4) we can subtract from the dividend (27) before what's left (the remainder) is less than the divisor. To evaluate $4\%27$, we ask the same question. Because $4 < 27$, we cannot subtract any instances of 27 from 4 without the result being less than 27, and so the remainder is the original dividend: 4.

A word of caution: When writing "/", "\", and "|" by hand, be careful to indicate the slope (or the lack thereof!) clearly, so that the reader (often the person grading your answer) understands which operator you are using. The division symbols are yet another reason to use a scientific document processing system such as LATEX to format your work whenever possible.

A.7.2 Congruences

Two integers are congruent if they produce the same remainder when divided by a given value. The following definition has the details.

> **Definition 78: Congruent**
>
> If $b, r \in \mathbb{Z}$ and $m \in \mathbb{Z}^+$, then b and r are *congruent* modulo m (written $b \equiv r \pmod{m}$) if and only if $b \% m = r \% m$ (or, if and only if $m \mid (b-r)$). (LATEX: \equiv produces '≡')

congruent

Some terminology to go with Definition 78: b is the *base*, r is the *remainder* or *residue*, and m is the *modulus*.[15]

---

[14]Why no examples with negative dividends or divisors? Because they really complicate the story, that's why! There's more than one way to handle negatives with modulo, and negatives don't often appear with modulo in computer science applications. And before you ask about floating point values: This is a *discrete* structures book! (That's the answer whenever we don't feel like dealing with real numbers, for any reason. Yes, it's very handy.)

[15] And here is one of those uses of 'modulus' now!

> **Example 217:**
>
> *Problem*: Are 35 and 108 congruent modulo 17?
>
> *Solution*: As Definition 78 states, $35 \equiv 108 \,(\mathrm{mod}\ 17))$ if $17 \,|\, (35 - 108))$. $35 - 108 = -73$, and $17 \,|\, -73$ is false. (73 is not a multiple of 17, so 17 cannot divide -73 evenly.) Thus, 35 and 108 are not congruent modulo 17 (which can be written: $35 \not\equiv 108 \,(\mathrm{mod}\ 17))$. (LaTeX: `\not\equiv`)

If the -73 from Example 217 bothers you, you can switch the base and the residue: $35 \equiv 108 \,(\mathrm{mod}\ 17)$ is equivalent to $108 \equiv 35 \,(\mathrm{mod}\ 17)$.

Congruences are useful in many advanced areas of computer science, including hash functions and cryptography. Cryptography is hard, though; telling time is comparatively easy, and congruences apply to time, too, as Example 218 shows.

> **Example 218:**
>
> *Problem*: It is now 7:50. What time will it be in 155 minutes?
>
> *Solution*: As we well know, there are 60 minutes in an hour. An integer division $(155/60 = 2\,\mathrm{r}\,35)$ reveals that 155 minutes is the same as 2 hours and 35 minutes. As a congruence, $155 \equiv 35 \,(\mathrm{mod}\ 60)$. Adding 2:35 to 7:50 by adding hours to hours and minutes to minutes gives a time of 9:85, which still needs some work because $85 \geq 60$. Congruences to the rescue: $85 \equiv 25 \,(\mathrm{mod}\ 60)$. Replacing the 85 in 9:85 with the residue (25) and incrementing the hours by one produces the answer: In 155 minutes, it will be 10:25.

A.8   Evens and Odds[16]

When we introduce proof techniques in Chapters 4 and 5, we will start with simple examples that frequently involve basic properties of integers. Two of those properties are the ideas of 'even' and 'odd' numbers.

---

[16]You're thinking of skipping this section, aren't you? It's only a page long! Read it!

> **Definition 79: Even**
>
> An integer n is *even* if and only if there exists an integer k such that $n = 2k$.

even

There are other equally valid ways of defining even. For example, n is even if and only if $2 \mid n$. Another: n is even if and only if $n \% 2 = 0$. A third: n is even if and only if $n \equiv 0 \,(\text{mod } 2)$. Why, then, did we choose the one that needs a second varible to be defined? Because, for many of our proof examples, that form is the most useful.

Please note: Regardless of the way the concept of evenness is expressed, the integer zero is an even number. Zero is a special case when we are dividing by integers, and zero is a special case when we consider positive and negative values, but zero is not a special case with evens and odds: Zero is even.

With 'even' defined, we can move on to 'odd':

> **Definition 80: Odd**
>
> An integer n is *odd* if and only if there exists an integer k such that $n = 2k + 1$.

odd

All of the other ways we gave to define 'even' can be adjusted to define 'odd': n is odd if and only if $2 \nmid n$ (LaTeX: `\nmid`), or if and only if $n \% 2 = 1$, or if and only if $n \equiv 1 \,(\text{mod } 2)$. We can even adjust the 'k' definition by adding (or subtracting) any odd integer from $2k$. For example, $n = 2k - 1001$ is just as valid as $n = 2k + 1$. The advantage of the "+1" form is that applying functions, such as squaring, to $2k + 1$ is usually much easier than applying them to an expression such as $2k - 1001$.

Finally, notice that we can partition (see Section 6.3.5) all integers into one or the other of these two subsets. That is, no integer is both even and odd, and no integer (not even zero!) is neither even nor odd.

A.9 Logarithms and Exponents

In the days before reliable calculating devices, logarithms were used to simplify otherwise tedious-to-evaluate expressions. We use them today in computer science primarily for algorithm analysis. For example, consider the binary

search algorithm. The amount of work a computer performs in carrying out that algorithm is described in terms of a logarithm of the quantity of data being searched.

   The name "logarithm" frightens nearly as many students as does the word "proof". You may find them less imposing if you know that they are just exponents written in a different way. When working with integers, we know that a sequence of products of the same value can be more tersely expressed using an exponent. For example, $3 \cdot 3 \cdot 3 \cdot 3 = 3^4 = 81$. In the expression $3^4 = 81$, 3 is the *base*, 81 is the *power*, and 4 is the *logarithm* (or *exponent*). Definition 81 gives another way to structure exponential expressions.

blp relationship

Definition 81: BLP Relationship

Assume that b, l, and p are postive real numbers. The Base–Logarithm–Power ('BLP')[17] relationship states:

$$b^l = p \ \text{ if and only if } \ \log_b p = l \qquad (\ \text{LaTeX: } \verb|\log_2|)$$

where b is the base, l is the logarithm, and p is the power.

Example 219:

Problem: Evaluate: $\log_4 64$.

Solution: Let's name the result of the evaluation of this expression x. That is, $\log_4 64 = x$. By BLP (Definition 81), we can rewrite $\log_4 64 = x$ as $4^x = 64$. Because $64 = 4^3$, $x = 3$, and so $\log_4 64 = 3$.

---

[17] The right-hand side of the relationship ($\log_b p = l$) is sometimes referred to as the *logarithmic function*, but if this entire biconditional relationship has a formal name, no reference I consulted uses it. Such a key idea deserves some sort of name, so we introduce Base–Logarithm–Power, or 'BLP' for short (and suggest the pronounciation "blip").

Example 220:

Problem: Evaluate: $\log_x x$ when $x > 0$.

Solution: At first glance, you might think that the problem is too vaguely specified for an evaluation to be performed. However, by applying BLP, we can transform $\log_x x = y$ into $x^y = x$. As x is just a short-cut way of writing x^1, we see that $y = 1$, and so $\log_x x = 1$. This is a useful identity to remember.

There are many exponent and logarithm identities that are good to learn. Table 26 summarizes some of the laws that we think are worth knowing.

Table 26: Laws of Exponents and Logarithms

Laws of Exponents		Laws of Logarithms	
(a)	$w^{x+y} = w^x w^y$	(f)	$\log_b(x^y) = y \log_b x$
(b)	$(w^x)^y = w^{xy}$	(g)	$\log_b(xy) = \log_b x + \log_b y$
(c)	$v^x w^x = (vw)^x$	(h)	$\log_b(\frac{x}{y}) = \log_b x - \log_b y$
(d)	$\frac{w^x}{w^y} = w^{x-y}$	(i)	$b^{\log_b x} = x$
(e)	$\frac{v^x}{w^x} = (\frac{v}{w})^x$	(j)	$\log_a x = \frac{\log_b x}{\log_b a}$

Example 221:

Problem: Evaluate $\dfrac{\log_5 4 + \log_5 8}{\frac{1}{\log_2 5}}$.

Solution: We start by making the log bases the same, and simplify from there.

$$\frac{\log_5 4 + \log_5 8}{\frac{1}{\log_2 5}} = \frac{\log_5 4 + \log_5 8}{\left(\frac{\log_5 5}{\log_5 2}\right)} \qquad \text{Table 26, Line j, with } b = 5$$

$$= \frac{\log_5 4 + \log_5 8}{\log_5 2} \qquad \text{Result of Ex. 220, and } 1/(1/y) = y$$

$$= \frac{\log_5 32}{\log_5 2} \qquad \text{Table 26, Line g}$$

$$= \log_2 32 \qquad \text{Table 26, Line j}$$

$$= \log_2 2^5 \qquad \text{It's good to know powers of 2!}$$

$$= 5 \log_2 2 \qquad \text{Table 26, Line f}$$

$$= 5 \qquad \text{Example 220 again}$$

To help put logarithic functions in context, Figure A.6 shows a plot of three functions: $f(n) = 1$ (a constant function), $f(n) = \log_2 n$ (a logarithmic function commonly used in computer science), and $f(n) = n$ (a linear function), with added axis lines for $n = 1$ and $n = 2$. There are several items of interest in this plot. First, observe how slowly $\log_2 n$ grows; it is much closer to $f(n) = 1$ than to $f(n) = n$ as n becomes larger. Second, observe that $\log_2 n = 1$ when $n = 2$ (not a surprise, now that we know $\log_b b = 1$). Third, $\log_2 1 = 0$ (also not a surprise, as $b^0 = 1$ implies that $\log_b 1 = 0$). Fourth, when $n < 1$, $\log_2 n < 0$. In computer science, n usually represents the size of a problem in terms of the size of the data structure (that is, the quantity of data items stored). Thus, values of n below 1 aren't often of interest.

A.10   Working with Quadratic Equations

We have occasion to factor quadratic equations in this book. Not many occasions, but we do have them. One reason to factor them is to find their roots.

quadratic

> **Definition 82: Quadratic**
>
> A *quadratic* is an polynomial expression in one variable of the form $ax^2 + bx + c$, where a, b, and c are real constants with $a \neq 0$.

quadratic equation

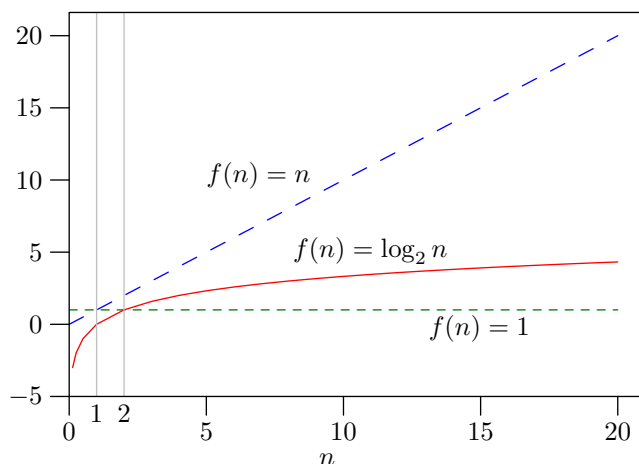Setting a quadratic equal to a value creates a *quadratic equation*.

Figure A.6: A linear-axes plot of $f(n) = 1$, $f(n) = \log_2 n$, and $f(n) = n$.

A.10.1 Factoring Quadratics

First, two definitions.

Definition 83: Perfect Square   *perfect square*

An integer n is a *perfect square* if the product of some integer m with itself equals n. Alternatively, n is a perfect square if its square root is an integer.

Definition 84: Quadratic Formula   *quadratic formula*

Given a quadratic equation $ax^2 + bx + c = 0$, the value(s) of x that make this equation true (known as the *root(s)* of the equation) are given by the evaluation of the *quadratic formula*: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. The expression within the square root, $b^2 - 4ac$, is the *discriminant*.

There are several techniques used to factor quadratic polynomials. As this is a discrete structures book, we will limit ourselves to quadratics that can be

discriminant

factored using only integers. How do you know? If a, b, and c are integers and the evaluation of $b^2 - 4ac$ (known as the *discriminant*; see Definition 84) is a perfect square, then the roots will be integers.

Because we are sticking with integers, factoring quadratics will be comparatively easy to accomplish. Here are three techniques:

(1) *'Guess and Check'.* Despite the inelegant name, this method does have a formal origin: Vieta's formulas.[18] Easiest to use when $a = 1$, in 'guess and check' we look for values y and z such that $x^2 + bx + c = (x + y)(x + z) = x^2 + (y + z)x + yz$. Note that $y + z = b$ and $y \cdot z = c$. To apply this method, we 'guess' a pair of factors of c, and 'check' that they satisfy those two equations.

(2) *The AC Method.* This method, which is a generalization of 'guess and check', is likely to be slower when applied by hand, but you may find it to be easier to use when $a \neq 1$. The idea comes from this equation: $ax^2 + bx + c = \frac{a^2 x^2 + abx + ac}{a} = \frac{(ax + d)(ax + e)}{a}$, where d and e are factors of the product $a \cdot c$ such that $d \cdot e = a \cdot c$, and their sum equals b. As with 'guess and check', the hard part is finding the factors d and e; after that, just substitute the values and simplify. The denominator a will divide away when the discriminant is a perfect square.

(3) *Apply the quadratic formula.* The factors of $ax^2 + bx + c$ are a, $x - \frac{-b + \sqrt{b^2 - 4ac}}{2a}$, and $x - \frac{-b - \sqrt{b^2 - 4ac}}{2a}$. That is,

$$ax^2 + bx + c = a\left(x - \frac{-b + \sqrt{b^2 - 4ac}}{2a}\right)\left(x - \frac{-b - \sqrt{b^2 - 4ac}}{2a}\right).$$

When the discriminant is a perfect square, the square roots will evaluate to integers. This method is as general as they come, but it does rely on your memory of the quadratic formula and your ability to perform a fair amount of math by hand without making any mistakes.

Example 222:

Problem: Factor $x^2 - x - 6$ using 'guess and check'.

Solution: To match the additions in the form $ax^2 + bx + c$, we need

---

[18] wikipedia.org/wiki/Vieta's_formulas

b and c to be negative; specifically, $a = 1$, $b = -1$, and $c = -6$. In 'guess and check', we need to find factors of -6. Ignoring the negation for a moment, the factors of 6 are $1, 2, 3$, and 6. We need a pair of these four factors whose product is 6. The only two pairs that work are 1 & 6 and 2 & 3. To get $yz = -6$ from these factors, there are four possibilities for y and z: -1 & 6, 1 & -6, -2 & 3, and 2 & -3. To satisfy the other equation ($y + z = -1$), only the pair 2 & -3 works. Thus, $x^2 - x - 6 = (x+2)(x-3)$.

Notice that the pairings of the factors are the smallest with the largest, the second-smallest with the second-largest, etc. You do not need to consider all possible pairings of the factors, because the products of the other pairings won't equal c.

Example 223:

Problem: Factor $6x^2 + 13x - 5$ using the AC method.

Solution: Here, $a = 6$, $b = 13$, and $c = -5$. In the AC method, we want to find factors of ac named d and e such that $\frac{(ax+d)(ax+e)}{a}$, subject to $de = ac$ and $d+e = b$. $ac = -30$, and 30's factors are $1, 2, 3, 5, 6, 10, 15$, and 30. Possible d & e pairs are 1 & 30, 2 & 15, 3 & 10, and 5 & 6, for a total of 8 pairs when negatives are added (remember, we need the product to be -30). Only the pair -2 & 15 sum to b. Substituting and simplifying:

$$\frac{(ax+d)(ax+e)}{a} = \frac{(6x-2)(6x+15)}{6} \quad \text{Substitution}$$
$$= \frac{(2)(3x-1)(3)(2x+5)}{6} \quad \text{Factor the binomials}$$
$$= (3x-1)(2x+5) \quad \text{Divide away the 6's}$$

Thus, $6x^2 + 13x - 5 = (3x-1)(2x+5)$.

Example 224:

Problem: Factoring $6x^2 + 13x - 5$ using the AC method was a pain, with all of those factors of 30!

Solution: You can always use the quadratic formula:

$$ax^2 + bx + c = a\left(x - \frac{-b+\sqrt{b^2-4ac}}{2a}\right)\left(x - \frac{-b-\sqrt{b^2-4ac}}{2a}\right)$$

$$= 6\left(x - \frac{-13+\sqrt{13^2-(4\cdot6\cdot-5)}}{2\cdot6}\right)\left(x - \frac{-13-\sqrt{13^2-(4\cdot6\cdot-5)}}{2\cdot6}\right)$$

$$= 6\left(x - \frac{-13+\sqrt{289}}{12}\right)\left(x - \frac{-13-\sqrt{289}}{12}\right)$$

$$= 6\left(x - \frac{1}{3}\right)\left(x - \frac{-5}{2}\right)$$

$$= 6\left(\frac{3x-1}{3}\right)\left(\frac{2x+5}{2}\right)$$

$$= (3x - 1)(2x + 5)$$

After trading the pain of factors of 30 for the tag-team pains of square roots and fractions, once again $6x^2 + 13x - 5 = (3x - 1)(2x + 5)$.

A.10.2   Roots of Quadratic Equations

roots

As mentioned in Definition 84, the value(s) for x that cause $ax^2 + bx + c = 0$ to be true are known as the *root(s)* of the quadratic equation.

Having refreshed our memory of how to factor quadratics, finding the root(s) of (a.k.a. solving) a quadratic equation is easy. Each quadratic factors

binomial

to two *binomials* (expressions that are the sum of two terms; e.g., $ax + d$). If either evaluates to zero, the evalation of the entire quadratic must be zero. It follows that the roots of the quadratic equation $ax^2+bx+c = (hx+i)(jx+k) = 0$ are simply $r_1 = -\frac{i}{h}$ and $r_2 = -\frac{k}{j}$. This shows that a quadratic equation has at most two unique roots, a situation that will occur when the discriminant is positive. A single root ($r_1 = r_2$) exists when the discriminant evaluates to zero, and no (real) roots exist otherwise.

Example 225:

Problem: Solve (that is, determine the roots of) $6x^2 + 13x - 5$.

Solution: In Examples 223 and 224, we factored this quadratic and learned that $6x^2+13x-5 = (3x-1)(2x+5)$. To make $(3x-1)(2x+5) = 0$, we need $3x - 1 = 0$ or $2x + 5 = 0$. That is, either $x = \frac{1}{3}$ or $x = -\frac{5}{2}$. Thus, the roots of $6x^2 + 13x - 5 = 0$ are $r_1 = \frac{1}{3}$ and $r_2 = -\frac{5}{2}$.[19]

A.11 Positional Number Systems

Number systems are not core discrete mathematics topics. However, values drawn from them are often used as the subject of counting problems in discrete math, and as you will see such problems in this book, the basics of such systems are worth providing.

Fundamentally, a *number system* is any representation used to express quantities. Using pebbles to represent heads of cabbage is a number system, although not one that scales well to corporate farming. The *decimal* system we use daily is also an example of a number system, but more specifically a *positional number system*. Positional systems are useful because they have symbols that individually represent small quantities, but those same symbols can also represent much larger quantities if the symbols are grouped and ordered.

number system

decimal

Definition 85: Positional System

positional system

A *positional (number) system* represents quantities using a combination of symbols (known as *digits* or *glyphs*) and relative locations of those symbols to represent quantities abstractly. Specifically, a positional system of base b employs a set of b digits. A sequence of these digits $d_n d_{n-1} \ldots d_1 d_0$ represents an integer value of base b equivalent to the decimal quantity $d_n \cdot b^n + d_{n-1} \cdot b^{n-1} + \cdots + d_1 \cdot b^1 + d_0 \cdot b^0 = \sum\limits_{i=0}^{n} d_i b^i$.

By changing the base, we can create an infinite number of positional number systems. Happily, only a few of those are commonly encoutered in computer science.

A.11.1 Decimal (Base 10)

Decimal is a good system with which to start, because we are already quite familiar with it. We all learned, a long time ago, that there are ten digits available in decimal: '0','1','2','3','4','5','6','7','8', and '9'. As Definition 85 says, the base of the system matches the number of available digits. Thus, we know decimal's base is ten. Each digit position in a decimal number represents

---

[19]Yeah, OK, this example's roots aren't integers. But they are rational!

a power of ten, and the digit in that position tells us how many of that power we include in the representation of the value.

Example 226:

681 is decimal is a representation of the sum of 600, 80, and 1. That is, $681 = 6 \cdot 10^2 + 8 \cdot 10^1 + 1 \cdot 10^0$. Beats the heck out of carrying around 681 pebbles!

A.11.2   Binary (Base 2)

Modern computers store information in primary storage as sequences of electrical charges – 'power is present' and 'no power is present'. Using this two-valued system as the basis for storing data naturally led to the use of the *binary* *binary* positional number system.

As the name 'binary' suggests, the base is two. As the necessary quantity of digits is less than the quantity used by decimal, rather than create two new gylphs, binary simply borrows the first two glyphs from decimal: '0' and '1'. *bit* (A single binary digit is called a *bit* [20].) To distinguish binary values from decimal values, we usually append a subscript of '2' at the end of a binary value, as in '1011011_2'. (You can append a '10' to the end of decimal values if you wish, but in practice un-subscripted values are assumed to be base 10.)

Because binary, like decimal, is a positional system, we can determine the quantity being represented using the summation provided in Definition 85, as Example 227 demonstrates.

Example 227:

Problem: Determine the decimal quantity represented by the binary value 1011011_2.

Solution: Each position of a binary value represents a power of 2, starting with 2^0 on the right and increasing the exponent by one for each position we move to the left. Writing the position values beneath the digits is sometimes helpful:

---

[20] The name *bit* is just a concatenation of the letters 'bi' from 'binary' and 't' from 'digit'.

$\dfrac{1}{64}$	$\dfrac{0}{32}$	$\dfrac{1}{16}$	$\dfrac{1}{8}$	$\dfrac{0}{4}$	$\dfrac{1}{2}$	$\dfrac{1}{1}$
(2^6)	(2^5)	(2^4)	(2^3)	(2^2)	(2^1)	(2^0)

From this, we can see how to form the decimal equivalent: We need a 64 $(1 \cdot 2^6)$, we don't include a 32 $(0 \cdot 2^5)$, etc. Summing the powers of two that we keep $(64 + 16 + 8 + 2 + 1)$ gives our answer: 91_{10} represents the same value as does 1011011_2.

When converting from binary to decimal, we employed the summation approach, which has a foundation of multiplication by 2. To convert from decimal to binary, it stands to reason that division can be used.

To convert from decimal to binary, we divide the decimal value by 2 and record the remainder. We next divide the quotient of the previous division by 2, and again save the remainder. This process of repeated divisions continues until the quotient becomes zero. The sequence of remainders, ordered from most-recently to least-recently discovered, are the digits of the binary representation.

Example 228:

Problem: Determine the binary representation of the decimal value 103.

Solution: As we are converting to binary, we perform divisions by 2. Here, we 'stack' the divisions to save space:

```
              0   R   1
        2 )   1   R   1
        2 )   3   R   0
        2 )   6   R   0
      2 )  1  2   R   1
      2 )  2  5   R   1
      2 )  5  1   R   1
    2 )  1  0  3
```

With the stacking, we read the remainders top-down to get the answer: $103_{10} = 1100111_2$.

Actually, this same process works to convert decimal to any other base with one small change: Take the description above and replace each occurrence of '2' by 'base b'. Algorithm 1 details the process. Note that a stack is used to hold the remainders and produce them in the proper order. If you do not know how a stack operates, you can replace it with an array. Store the remainders starting at index 0, and output them in reverse order.

Algorithm 26: Decimal to Base b Conversion

```
1  dividend <- decimal value
2  divisor <- base b
3  initialize stack
4  loop
5      quotient <- dividend \ divisor
6      remainder <- dividend % divisor
7      push remainder onto stack
8      dividend <- quotient
9      if dividend is 0, leave loop
10 end loop
11 pop stack content to output
```

If you need to do a base conversion manually, the hardest part is remembering the order in which to write the remainders. Remembering what the stack does for us in Algorithm 1 is one good way to get it right. Another is to look at Example 228 and think about continuing the division process longer than is necessary. Doing so will create a bunch of zeros. Only the left side of the sequence of remainders can have zeros added to it without changing the result. That is, $1_2 = 0001_2 \neq 1000_2$.

The really annoying part of the division algorithm is the need to do a lot of divisions. There is an alternative approach that works well for small decimal-to-binary conversions and can be done in your head a bit more easily than several long divisions.

Example 229:

Problem: Determine the binary representation of the decimal value 103 again ... this time with only the powers of your mind!

Solution: What we need is the correct collection of powers of 2 that sum to 103. Start by thinking of the largest power of 2 that is less than or equal to 103: 64. Write down a '1', and compute $103 - 64$: 39. Now consider all of the powers of 2 from the next smaller power of 2 (32) down to and including 0, in decreasing order. For each one, if we need it to reach the remaining amount, write down a '1' and subtract it to get the new remaining amount (write down a '0' otherwise). Stop when the remaining amount is zero. In this example, we need 32 to get to 39; we write down a '1' and compute the difference $39 - 32 = 7$. We don't need 16 or 8 (write down two '0's). At this point, it is likely obvious that we need all of the rest (4, 2, and 1) to reach 7, and so we end with three more '1's. Our final collection is: 1100111.

A.11.3   Octal (Base 8)

Binary may be what computers like to use, but most human beings would rather not have to work with long strings of zeros and ones. A compromise positional number system is *octal* (base 8). This is a compromise because the *octal* representations are shorter (easier for people to remember) but also easy to convert back and forth to binary. As with binary, we can use a subset of the decimal digits for octal representations, specifically the digits 0 through 7. As octal is also a positional number system, each digit position is worth a power of 8. A difference from binary is that we have the ability to have several of any given power of 8 included in our sum; that is, 42_8 means that we have 4 8s plus 2 1s. In binary, we could only accept a power of two or ignore it.

   The decimal-to-octal and octal-to-decimal conversions can be done as we did them for binary (repeated multiplications or divisions by 8), or we can do them in two steps: decimal-to-binary-to-octal and octal-to-binary-to-decimal.

Example 230:

Problem: Convert 2065_8 to decimal.

Solution: We will do this in two ways: Through binary, and directly.

Option #1: Octal-to-binary-to-decimal. We start with an observation: A three-digit binary number has possible decimal values ranging from

0 (000) through 7 (111), exactly the range of the octal digits. A consequence is that we can convert octal to binary by simply expanding each octal digit to its three-digit binary representation. In this example, $2065_8 \rightarrow (010)(000)(110)(101) \rightarrow 10000110101_2$ (after dropping the extraneous leading 0). Having seen Example 227, we know what to do next: $1024 + 32 + 16 + 4 + 1 = 1077_{10}$.

Option #2: Octal-to-decimal. To accomplish this, we just apply the sum from Definition 85: $2 \cdot 8^3 + 6 \cdot 8^1 + 5 \cdot 8^0 = 1024 + 48 + 5 = 1077$.

Example 231:

Problem: Determine the octal representation of the decimal value 103.

Solution: We will also do this example in two ways.

Option #1: Decimal-to-binary-to-octal. We have already accomplished the first half of this in Example 228: $103_{10} = 1100111_2$. To convert from binary to octal, we can construct groups of three binary digits and convert them to octal digits. Note that *the grouping by threes must start on the right and move left*! That is, $1100111_2 \rightarrow (1)(100)(111) \rightarrow 147_8$.

Option #2: Decimal-to-octal. We can apply repeated divisions, as used in Example 228:

```
                    0   R   1
            8   )   1   R   4
        8   )   1   2   R   7
    8   )   1   0   3
```

Thus, $103_{10} = 147_8$ (and also verifies the result of Option #1).

A.11.4   Hexadecimal (Base 16)

hexadecimal

We motivated octal by presenting it as a compromise between binary and decimal. *Hexadecimal* ('hex' for short) can be thought of in the same way. The name 'hexadecimal' suggests its base: Together, 'hex' for six and 'decimal' for ten give 16 as the base, which, like base eight for octal, is a power of two.

But we have a new problem: A base 16 system needs 16 digits, and we only have the ten from decimal to work with. The solution is to use the first six letters of the alphabet ('A' through 'F') to represent the values 10 through 15. Table 27 will be a handy reference until you become more familiar with the system.

Table 27: Binary, Octal, Decimal, and Hex: 0 through 15_{10}

Binary	Octal	Decimal	Hex	Binary	Octal	Decimal	Hex
0000	0	0	0	1000	10	8	8
0001	1	1	1	1001	11	9	9
0010	2	2	2	1010	12	10	A
0011	3	3	3	1011	13	11	B
0100	4	4	4	1100	14	12	C
0101	5	5	5	1101	15	13	D
0110	6	6	6	1110	16	14	E
0111	7	7	7	1111	17	15	F

Hexadecimal is more commonly used than is octal for two main reasons. First, because it has a base larger than 10, hex representations of values often require fewer digits than do the octal or the decimal representations. Second, a hex digit is representable with four binary digits, which is exactly half of a byte.[21] Thus, we can represent the content of a byte with no more than two hex digits instead of up to three decimal or octal digits.

If you understand conversions between decimal, binary, and octal, you understand them for hexadecimal; just remember to use base 16 instead of base 8. The conversions between octal and hex are easily performed via binary.

Example 232:

Problem: Express 1352_8 in hexadecimal.

Solution: First, we convert the octal digits into three-digit binary values to get the binary equivalent: $1352_8 \to (001)(011)(101)(010) \to 1011101010_2$ To convert from binary to hex, we can group the bits by fours (as we did

---

[21] Half a byte is called a *nybble*. Byte ... tiny bite ... nybble; get it? 'Nibble' is an acceptable alternate spelling.

by threes for octal). Again, we must group from right to left: $1011101010_2 \rightarrow$ $(10)(1110)(1010) \rightarrow 2EA_{16}$.

A.11.5   Other Bases

As we hope is becoming clear from the preceding examinations of binary, octal, and hexadecimal, any integer value greater than one can be used as the base in a positional number system. Definition 85 can be used to convert any of them to decimal, and the division technique of Example 228 can convert from decimal to them.

Example 233:

Problem: Determine the decimal value of 413_5.

Solution: By Definition 85: $4 \cdot 5^2 + 1 \cdot 5^1 + 3 \cdot 5^0 = 100 + 5 + 3 = 108_{10}$.