

The continuum hypothesis was stated by Cantor in 1877. He labored unsuccessfully to prove it, becoming extremely dismayed that he could not. By 1900, settling the continuum hypothesis was considered to be among the most important unsolved problems in mathematics. It was the first problem posed by David Hilbert in his famous 1900 list of open problems in mathematics.

The continuum hypothesis is still an open question and remains an area for active research. However, it has been shown that it can be neither proved nor disproved under the standard set theory axioms in modern mathematics, the Zermelo-Fraenkel axioms. The Zermelo-Fraenkel axioms were formulated to avoid the paradoxes of naive set theory, such as Russell's paradox, but there is much controversy whether they should be replaced by some other set of axioms for set theory.

Exercises

- Determine whether each of these sets is finite, countably infinite, or uncountable. For those that are countably infinite, exhibit a one-to-one correspondence between the set of positive integers and that set.
 - the negative integers
 - the even integers
 - the integers less than 100
 - the real numbers between 0 and $\frac{1}{2}$
 - the positive integers less than 1,000,000,000
 - the integers that are multiples of 7
- Determine whether each of these sets is finite, countably infinite, or uncountable. For those that are countably infinite, exhibit a one-to-one correspondence between the set of positive integers and that set.
 - the integers greater than 10
 - the odd negative integers
 - the integers with absolute value less than 1,000,000
 - the real numbers between 0 and 2
 - the set $A \times \mathbf{Z}^+$ where $A = \{2, 3\}$
 - the integers that are multiples of 10
- Determine whether each of these sets is countable or uncountable. For those that are countably infinite, exhibit a one-to-one correspondence between the set of positive integers and that set.
 - all bit strings not containing the bit 0
 - all positive rational numbers that cannot be written with denominators less than 4
 - the real numbers not containing 0 in their decimal representation
 - the real numbers containing only a finite number of 1s in their decimal representation
- Determine whether each of these sets is countable or uncountable. For those that are countably infinite, exhibit a one-to-one correspondence between the set of positive integers and that set.
 - integers not divisible by 3
 - integers divisible by 5 but not by 7
 - the real numbers with decimal representations consisting of all 1s
 - the real numbers with decimal representations of all 1s or 9s
- Show that a finite group of guests arriving at Hilbert's fully occupied Grand Hotel can be given rooms without evicting any current guest.
- Suppose that Hilbert's Grand Hotel is fully occupied, but the hotel closes all the even numbered rooms for maintenance. Show that all guests can remain in the hotel.
- Suppose that Hilbert's Grand Hotel is fully occupied on the day the hotel expands to a second building which also contains a countably infinite number of rooms. Show that the current guests can be spread out to fill every room of the two buildings of the hotel.
- Show that a countably infinite number of guests arriving at Hilbert's fully occupied Grand Hotel can be given rooms without evicting any current guest.
- *9. Suppose that a countably infinite number of buses, each containing a countably infinite number of guests, arrive at Hilbert's fully occupied Grand Hotel. Show that all the arriving guests can be accommodated without evicting any current guest.
- Give an example of two uncountable sets A and B such that $A - B$ is
 - finite.
 - countably infinite.
 - uncountable.
- Give an example of two uncountable sets A and B such that $A \cap B$ is
 - finite.
 - countably infinite.
 - uncountable.
- Show that if A and B are sets and $A \subset B$ then $|A| \leq |B|$.
- Explain why the set A is countable if and only if $|A| \leq |\mathbf{Z}^+|$.
- Show that if A and B are sets with the same cardinality, then $|A| \leq |B|$ and $|B| \leq |A|$.
- Show that if A and B are sets, A is uncountable, and $A \subseteq B$, then B is uncountable.
- Show that a subset of a countable set is also countable.
- If A is an uncountable set and B is a countable set, must $A - B$ be uncountable?

18. Show that if A and B are sets $|A| = |B|$, then $|\mathcal{P}(A)| = |\mathcal{P}(B)|$.
19. Show that if A, B, C , and D are sets with $|A| = |B|$ and $|C| = |D|$, then $|A \times C| = |B \times D|$.
20. Show that if $|A| = |B|$ and $|B| = |C|$, then $|A| = |C|$.
21. Show that if A, B , and C are sets such that $|A| \leq |B|$ and $|B| \leq |C|$, then $|A| \leq |C|$.
22. Suppose that A is a countable set. Show that the set B is also countable if there is an onto function f from A to B .
23. Show that if A is an infinite set, then it contains a countably infinite subset.
24. Show that there is no infinite set A such that $|A| < |\mathbf{Z}^+| = \aleph_0$.
25. Prove that if it is possible to label each element of an infinite set S with a finite string of keyboard characters, from a finite list characters, where no two elements of S have the same label, then S is a countably infinite set.
26. Use Exercise 25 to provide a proof different from that in the text that the set of rational numbers is countable. [Hint: Show that you can express a rational number as a string of digits with a slash and possibly a minus sign.]
- *27. Show that the union of a countable number of countable sets is countable.
28. Show that the set $\mathbf{Z}^+ \times \mathbf{Z}^+$ is countable.
- *29. Show that the set of all finite bit strings is countable.
- *30. Show that the set of real numbers that are solutions of quadratic equations $ax^2 + bx + c = 0$, where a, b , and c are integers, is countable.
- *31. Show that $\mathbf{Z}^+ \times \mathbf{Z}^+$ is countable by showing that the polynomial function $f : \mathbf{Z}^+ \times \mathbf{Z}^+ \rightarrow \mathbf{Z}^+$ with $f(m, n) = (m + n - 2)(m + n - 1)/2 + m$ is one-to-one and onto.
- *32. Show that when you substitute $(3n + 1)^2$ for each occurrence of n and $(3m + 1)^2$ for each occurrence of m in the right-hand side of the formula for the function $f(m, n)$ in Exercise 31, you obtain a one-to-one polynomial function $\mathbf{Z} \times \mathbf{Z} \rightarrow \mathbf{Z}$. It is an open question whether there is a one-to-one polynomial function $\mathbf{Q} \times \mathbf{Q} \rightarrow \mathbf{Q}$.
33. Use the Schröder-Bernstein theorem to show that $(0, 1)$ and $[0, 1]$ have the same cardinality.
34. Show that $(0, 1)$ and \mathbf{R} have the same cardinality. [Hint: Use the Schröder-Bernstein theorem.]
35. Show that there is no one-to-one correspondence from the set of positive integers to the power set of the set of positive integers. [Hint: Assume that there is such a one-to-one correspondence. Represent a subset of the set of positive integers as an infinite bit string with i th bit 1 if i belongs to the subset and 0 otherwise. Suppose that you can list these infinite strings in a sequence indexed by the positive integers. Construct a new bit string with its i th bit equal to the complement of the i th bit of the i th string in the list. Show that this new bit string cannot appear in the list.]
- *36. Show that there is a one-to-one correspondence from the set of subsets of the positive integers to the set real numbers between 0 and 1. Use this result and Exercises 34 and 35 to conclude that $\aleph_0 < |\mathcal{P}(\mathbf{Z}^+)| = |\mathbf{R}|$. [Hint: Look at the first part of the hint for Exercise 35.]
- *37. Show that the set of all computer programs in a particular programming language is countable. [Hint: A computer program written in a programming language can be thought of as a string of symbols from a finite alphabet.]
- *38. Show that the set of functions from the positive integers to the set $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is uncountable. [Hint: First set up a one-to-one correspondence between the set of real numbers between 0 and 1 and a subset of these functions. Do this by associating to the real number $0.d_1d_2\dots d_n\dots$ the function f with $f(n) = d_n$.]
- *39. We say that a function is **computable** if there is a computer program that finds the values of this function. Use Exercises 37 and 38 to show that there are functions that are not computable.
- *40. Show that if S is a set, then there does not exist an onto function f from S to $\mathcal{P}(S)$, the power set of S . Conclude that $|S| < |\mathcal{P}(S)|$. This result is known as **Cantor's theorem**. [Hint: Suppose such a function f existed. Let $T = \{s \in S \mid s \notin f(s)\}$ and show that no element s can exist for which $f(s) = T$.]

2.6 Matrices

Introduction

Matrices are used throughout discrete mathematics to express relationships between elements in sets. In subsequent chapters we will use matrices in a wide variety of models. For instance, matrices will be used in models of communications networks and transportation systems. Many algorithms will be developed that use these matrix models. This section reviews matrix arithmetic that will be used in these algorithms.

DEFINITION 10

Let \mathbf{A} be a square zero–one matrix and let r be a positive integer. The r th *Boolean power* of \mathbf{A} is the Boolean product of r factors of \mathbf{A} . The r th Boolean product of \mathbf{A} is denoted by $\mathbf{A}^{[r]}$. Hence

$$\mathbf{A}^{[r]} = \underbrace{\mathbf{A} \odot \mathbf{A} \odot \mathbf{A} \odot \cdots \odot \mathbf{A}}_{r \text{ times}}.$$

(This is well defined because the Boolean product of matrices is associative.) We also define $\mathbf{A}^{[0]}$ to be \mathbf{I}_n .

EXAMPLE 9 Let $\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$. Find $\mathbf{A}^{[n]}$ for all positive integers n .

Solution: We find that

$$\mathbf{A}^{[2]} = \mathbf{A} \odot \mathbf{A} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

We also find that

$$\mathbf{A}^{[3]} = \mathbf{A}^{[2]} \odot \mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{A}^{[4]} = \mathbf{A}^{[3]} \odot \mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Additional computation shows that

$$\mathbf{A}^{[5]} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

The reader can now see that $\mathbf{A}^{[n]} = \mathbf{A}^{[5]}$ for all positive integers n with $n \geq 5$. ◀

Exercises

1. Let $\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 3 \\ 2 & 0 & 4 & 6 \\ 1 & 1 & 3 & 7 \end{bmatrix}$.

- What size is \mathbf{A} ?
- What is the third column of \mathbf{A} ?
- What is the second row of \mathbf{A} ?
- What is the element of \mathbf{A} in the (3, 2)th position?
- What is \mathbf{A}^t ?

2. Find $\mathbf{A} + \mathbf{B}$, where

a) $\mathbf{A} = \begin{bmatrix} 1 & 0 & 4 \\ -1 & 2 & 2 \\ 0 & -2 & -3 \end{bmatrix},$

$\mathbf{B} = \begin{bmatrix} -1 & 3 & 5 \\ 2 & 2 & -3 \\ 2 & -3 & 0 \end{bmatrix}.$

b) $\mathbf{A} = \begin{bmatrix} -1 & 0 & 5 & 6 \\ -4 & -3 & 5 & -2 \end{bmatrix},$

$\mathbf{B} = \begin{bmatrix} -3 & 9 & -3 & 4 \\ 0 & -2 & -1 & 2 \end{bmatrix}.$

3. Find \mathbf{AB} if

a) $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & 4 \\ 1 & 3 \end{bmatrix}.$

b) $\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 2 & 3 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 3 & -2 & -1 \\ 1 & 0 & 2 \end{bmatrix}.$

c) $\mathbf{A} = \begin{bmatrix} 4 & -3 \\ 3 & -1 \\ 0 & -2 \\ -1 & 5 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} -1 & 3 & 2 & -2 \\ 0 & -1 & 4 & -3 \end{bmatrix}.$

4. Find the product
- \mathbf{AB}
- , where

a) $\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & -1 & -1 \\ -1 & 1 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & 1 & -1 \\ 1 & -1 & 0 \\ -1 & 0 & 1 \end{bmatrix}.$

b) $\mathbf{A} = \begin{bmatrix} 1 & -3 & 0 \\ 1 & 2 & 2 \\ 2 & 1 & -1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & -1 & 2 & 3 \\ -1 & 0 & 3 & -1 \\ -3 & -2 & 0 & 2 \end{bmatrix}.$

c) $\mathbf{A} = \begin{bmatrix} 0 & -1 \\ 7 & 2 \\ -4 & -3 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 4 & -1 & 2 & 3 & 0 \\ -2 & 0 & 3 & 4 & 1 \end{bmatrix}.$

5. Find a matrix
- \mathbf{A}
- such that

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix} \mathbf{A} = \begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix}.$$

[Hint: Finding \mathbf{A} requires that you solve systems of linear equations.]

6. Find a matrix
- \mathbf{A}
- such that

$$\begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 1 \\ 4 & 0 & 3 \end{bmatrix} \mathbf{A} = \begin{bmatrix} 7 & 1 & 3 \\ 1 & 0 & 3 \\ -1 & -3 & 7 \end{bmatrix}.$$

7. Let
- \mathbf{A}
- be an
- $m \times n$
- matrix and let
- $\mathbf{0}$
- be the
- $m \times n$
- matrix that has all entries equal to zero. Show that
- $\mathbf{A} = \mathbf{0} + \mathbf{A} = \mathbf{A} + \mathbf{0}$
- .

8. Show that matrix addition is commutative; that is, show that if
- \mathbf{A}
- and
- \mathbf{B}
- are both
- $m \times n$
- matrices, then
- $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$
- .

9. Show that matrix addition is associative; that is, show that if
- \mathbf{A}
- ,
- \mathbf{B}
- , and
- \mathbf{C}
- are all
- $m \times n$
- matrices, then
- $\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$
- .

10. Let
- \mathbf{A}
- be a
- 3×4
- matrix,
- \mathbf{B}
- be a
- 4×5
- matrix, and
- \mathbf{C}
- be a
- 4×4
- matrix. Determine which of the following products are defined and find the size of those that are defined.

- a)
- \mathbf{AB}
- b)
- \mathbf{BA}
- c)
- \mathbf{AC}
-
- d)
- \mathbf{CA}
- e)
- \mathbf{BC}
- f)
- \mathbf{CB}

11. What do we know about the sizes of the matrices
- \mathbf{A}
- and
- \mathbf{B}
- if both of the products
- \mathbf{AB}
- and
- \mathbf{BA}
- are defined?

12. In this exercise we show that matrix multiplication is distributive over matrix addition.

- a) Suppose that
- \mathbf{A}
- and
- \mathbf{B}
- are
- $m \times k$
- matrices and that
- \mathbf{C}
- is a
- $k \times n$
- matrix. Show that
- $(\mathbf{A} + \mathbf{B})\mathbf{C} = \mathbf{AC} + \mathbf{BC}$
- .
-
- b) Suppose that
- \mathbf{C}
- is an
- $m \times k$
- matrix and that
- \mathbf{A}
- and
- \mathbf{B}
- are
- $k \times n$
- matrices. Show that
- $\mathbf{C}(\mathbf{A} + \mathbf{B}) = \mathbf{CA} + \mathbf{CB}$
- .

13. In this exercise we show that matrix multiplication is associative. Suppose that
- \mathbf{A}
- is an
- $m \times p$
- matrix,
- \mathbf{B}
- is a
- $p \times k$
- matrix, and
- \mathbf{C}
- is a
- $k \times n$
- matrix. Show that
- $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$
- .

14. The
- $n \times n$
- matrix
- $\mathbf{A} = [a_{ij}]$
- is called a
- diagonal matrix**
- if
- $a_{ij} = 0$
- when
- $i \neq j$
- . Show that the product of two
- $n \times n$
- diagonal matrices is again a diagonal matrix. Give a simple rule for determining this product.

15. Let

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

Find a formula for \mathbf{A}^n , whenever n is a positive integer.

16. Show that
- $(\mathbf{A}^t)^t = \mathbf{A}$
- .

17. Let
- \mathbf{A}
- and
- \mathbf{B}
- be two
- $n \times n$
- matrices. Show that

- a)
- $(\mathbf{A} + \mathbf{B})^t = \mathbf{A}^t + \mathbf{B}^t$
- .
-
- b)
- $(\mathbf{AB})^t = \mathbf{B}^t \mathbf{A}^t$
- .

If \mathbf{A} and \mathbf{B} are $n \times n$ matrices with $\mathbf{AB} = \mathbf{BA} = \mathbf{I}_n$, then \mathbf{B} is called the **inverse** of \mathbf{A} (this terminology is appropriate because such a matrix \mathbf{B} is unique) and \mathbf{A} is said to be **invertible**. The notation $\mathbf{B} = \mathbf{A}^{-1}$ denotes that \mathbf{B} is the inverse of \mathbf{A} .

18. Show that

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

is the inverse of

$$\begin{bmatrix} 7 & -8 & 5 \\ -4 & 5 & -3 \\ 1 & -1 & 1 \end{bmatrix}.$$

19. Let
- \mathbf{A}
- be the
- 2×2
- matrix

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

Show that if $ad - bc \neq 0$, then

$$\mathbf{A}^{-1} = \begin{bmatrix} \frac{d}{ad - bc} & \frac{-b}{ad - bc} \\ \frac{-c}{ad - bc} & \frac{a}{ad - bc} \end{bmatrix}.$$

20. Let

$$\mathbf{A} = \begin{bmatrix} -1 & 2 \\ 1 & 3 \end{bmatrix}.$$

- a) Find
- \mathbf{A}^{-1}
- . [Hint: Use Exercise 19.]

- b) Find
- \mathbf{A}^3
- .

- c) Find
- $(\mathbf{A}^{-1})^3$
- .

- d) Use your answers to (b) and (c) to show that
- $(\mathbf{A}^{-1})^3$
- is the inverse of
- \mathbf{A}^3
- .

21. Let
- \mathbf{A}
- be an invertible matrix. Show that
- $(\mathbf{A}^n)^{-1} = (\mathbf{A}^{-1})^n$
- whenever
- n
- is a positive integer.

22. Let
- \mathbf{A}
- be a matrix. Show that the matrix
- \mathbf{AA}^t
- is symmetric. [Hint: Show that this matrix equals its transpose with the help of Exercise 17b.]

23. Suppose that
- \mathbf{A}
- is an
- $n \times n$
- matrix where
- n
- is a positive integer. Show that
- $\mathbf{A} + \mathbf{A}^t$
- is symmetric.

24. a) Show that the system of simultaneous linear equations

$$\begin{aligned}a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\&\vdots \\a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n.\end{aligned}$$

in the variables x_1, x_2, \dots, x_n can be expressed as $\mathbf{AX} = \mathbf{B}$, where $\mathbf{A} = [a_{ij}]$, \mathbf{X} is an $n \times 1$ matrix with x_i the entry in its i th row, and \mathbf{B} is an $n \times 1$ matrix with b_i the entry in its i th row.

- b) Show that if the matrix $\mathbf{A} = [a_{ij}]$ is invertible (as defined in the preamble to Exercise 18), then the solution of the system in part (a) can be found using the equation $\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$.

25. Use Exercises 18 and 24 to solve the system

$$\begin{aligned}7x_1 - 8x_2 + 5x_3 &= 5 \\-4x_1 + 5x_2 - 3x_3 &= -3 \\x_1 - x_2 + x_3 &= 0\end{aligned}$$

26. Let

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Find

- a) $\mathbf{A} \vee \mathbf{B}$. b) $\mathbf{A} \wedge \mathbf{B}$. c) $\mathbf{A} \odot \mathbf{B}$.

27. Let

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

Find

- a) $\mathbf{A} \vee \mathbf{B}$. b) $\mathbf{A} \wedge \mathbf{B}$. c) $\mathbf{A} \odot \mathbf{B}$.

28. Find the Boolean product of \mathbf{A} and \mathbf{B} , where

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

29. Let

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

Find

- a) $\mathbf{A}^{[2]}$. b) $\mathbf{A}^{[3]}$.
c) $\mathbf{A} \vee \mathbf{A}^{[2]} \vee \mathbf{A}^{[3]}$.

30. Let \mathbf{A} be a zero-one matrix. Show that

- a) $\mathbf{A} \vee \mathbf{A} = \mathbf{A}$. b) $\mathbf{A} \wedge \mathbf{A} = \mathbf{A}$.

31. In this exercise we show that the meet and join operations are commutative. Let \mathbf{A} and \mathbf{B} be $m \times n$ zero-one matrices. Show that

- a) $\mathbf{A} \vee \mathbf{B} = \mathbf{B} \vee \mathbf{A}$. b) $\mathbf{B} \wedge \mathbf{A} = \mathbf{A} \wedge \mathbf{B}$.

32. In this exercise we show that the meet and join operations are associative. Let \mathbf{A} , \mathbf{B} , and \mathbf{C} be $m \times n$ zero-one matrices. Show that

- a) $(\mathbf{A} \vee \mathbf{B}) \vee \mathbf{C} = \mathbf{A} \vee (\mathbf{B} \vee \mathbf{C})$.
b) $(\mathbf{A} \wedge \mathbf{B}) \wedge \mathbf{C} = \mathbf{A} \wedge (\mathbf{B} \wedge \mathbf{C})$.

33. We will establish distributive laws of the meet over the join operation in this exercise. Let \mathbf{A} , \mathbf{B} , and \mathbf{C} be $m \times n$ zero-one matrices. Show that

- a) $\mathbf{A} \vee (\mathbf{B} \wedge \mathbf{C}) = (\mathbf{A} \vee \mathbf{B}) \wedge (\mathbf{A} \vee \mathbf{C})$.
b) $\mathbf{A} \wedge (\mathbf{B} \vee \mathbf{C}) = (\mathbf{A} \wedge \mathbf{B}) \vee (\mathbf{A} \wedge \mathbf{C})$.

34. Let \mathbf{A} be an $n \times n$ zero-one matrix. Let \mathbf{I} be the $n \times n$ identity matrix. Show that $\mathbf{A} \odot \mathbf{I} = \mathbf{I} \odot \mathbf{A} = \mathbf{A}$.

35. In this exercise we will show that the Boolean product of zero-one matrices is associative. Assume that \mathbf{A} is an $m \times p$ zero-one matrix, \mathbf{B} is a $p \times k$ zero-one matrix, and \mathbf{C} is a $k \times n$ zero-one matrix. Show that $\mathbf{A} \odot (\mathbf{B} \odot \mathbf{C}) = (\mathbf{A} \odot \mathbf{B}) \odot \mathbf{C}$.

Key Terms and Results

TERMS

set: a collection of distinct objects

axiom: a basic assumption of a theory

paradox: a logical inconsistency

element, member of a set: an object in a set

roster method: a method that describes a set by listing its elements

set builder notation: the notation that describes a set by stating a property an element must have to be a member

\emptyset (**empty set, null set**): the set with no members

universal set: the set containing all objects under consideration

Venn diagram: a graphical representation of a set or sets

$S = T$ (**set equality**): S and T have the same elements

$S \subseteq T$ (**S is a subset of T**): every element of S is also an element of T

$S \subset T$ (**S is a proper subset of T**): S is a subset of T and $S \neq T$

finite set: a set with n elements, where n is a nonnegative integer

infinite set: a set that is not finite

$|S|$ (**the cardinality of S**): the number of elements in S

$P(S)$ (**the power set of S**): the set of all subsets of S

$\mathbf{A} \cup \mathbf{B}$ (**the union of \mathbf{A} and \mathbf{B}**): the set containing those elements that are in at least one of \mathbf{A} and \mathbf{B}

$\mathbf{A} \cap \mathbf{B}$ (**the intersection of \mathbf{A} and \mathbf{B}**): the set containing those elements that are in both \mathbf{A} and \mathbf{B} .

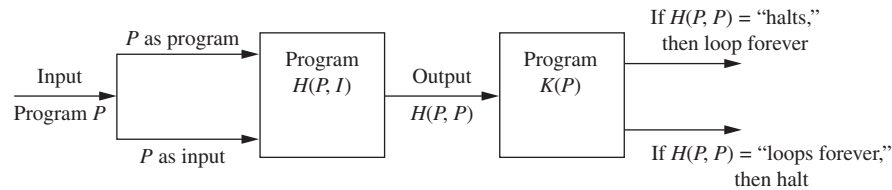


FIGURE 2 Showing that the Halting Problem is Unsolvable.

is “halt,” then by the definition of K we see that $K(K)$ loops forever, in violation of what H tells us. In both cases, we have a contradiction.

Thus, H cannot always give the correct answers. Consequently, there is no procedure that solves the halting problem. ◀

Exercises

1. List all the steps used by Algorithm 1 to find the maximum of the list 1, 8, 12, 9, 11, 2, 14, 5, 10, 4.
2. Determine which characteristics of an algorithm described in the text (after Algorithm 1) the following procedures have and which they lack.
 - a) **procedure** *double*(n : positive integer)
 while $n > 0$
 $n := 2n$
 - b) **procedure** *divide*(n : positive integer)
 while $n \geq 0$
 $m := 1/n$
 $n := n - 1$
 - c) **procedure** *sum*(n : positive integer)
 $sum := 0$
 while $i < 10$
 $sum := sum + i$
 - d) **procedure** *choose*(a, b : integers)
 $x := \text{either } a \text{ or } b$
3. Devise an algorithm that finds the sum of all the integers in a list.
4. Describe an algorithm that takes as input a list of n integers and produces as output the largest difference obtained by subtracting an integer in the list from the one following it.
5. Describe an algorithm that takes as input a list of n integers in nondecreasing order and produces the list of all values that occur more than once. (Recall that a list of integers is **nondecreasing** if each integer in the list is at least as large as the previous integer in the list.)
6. Describe an algorithm that takes as input a list of n integers and finds the number of negative integers in the list.
7. Describe an algorithm that takes as input a list of n integers and finds the location of the last even integer in the list or returns 0 if there are no even integers in the list.
8. Describe an algorithm that takes as input a list of n distinct integers and finds the location of the largest even integer in the list or returns 0 if there are no even integers in the list.
9. A **palindrome** is a string that reads the same forward and backward. Describe an algorithm for determining whether a string of n characters is a palindrome.
10. Devise an algorithm to compute x^n , where x is a real number and n is an integer. [Hint: First give a procedure for computing x^n when n is nonnegative by successive multiplication by x , starting with 1. Then extend this procedure, and use the fact that $x^{-n} = 1/x^n$ to compute x^n when n is negative.]
11. Describe an algorithm that interchanges the values of the variables x and y , using only assignments. What is the minimum number of assignment statements needed to do this?
12. Describe an algorithm that uses only assignment statements that replaces the triple (x, y, z) with (y, z, x) . What is the minimum number of assignment statements needed?
13. List all the steps used to search for 9 in the sequence 1, 3, 4, 5, 6, 8, 9, 11 using
 - a) a linear search.
 - b) a binary search.
14. List all the steps used to search for 7 in the sequence given in Exercise 13 for both a linear search and a binary search.
15. Describe an algorithm that inserts an integer x in the appropriate position into the list a_1, a_2, \dots, a_n of integers that are in increasing order.
16. Describe an algorithm for finding the smallest integer in a finite sequence of natural numbers.
17. Describe an algorithm that locates the first occurrence of the largest element in a finite list of integers, where the integers in the list are not necessarily distinct.
18. Describe an algorithm that locates the last occurrence of the smallest element in a finite list of integers, where the integers in the list are not necessarily distinct.

19. Describe an algorithm that produces the maximum, median, mean, and minimum of a set of three integers. (The **median** of a set of integers is the middle element in the list when these integers are listed in order of increasing size. The **mean** of a set of integers is the sum of the integers divided by the number of integers in the set.)
 20. Describe an algorithm for finding both the largest and the smallest integers in a finite sequence of integers.
 21. Describe an algorithm that puts the first three terms of a sequence of integers of arbitrary length in increasing order.
 22. Describe an algorithm to find the longest word in an English sentence (where a sentence is a sequence of symbols, either a letter or a blank, which can then be broken into alternating words and blanks).
 23. Describe an algorithm that determines whether a function from a finite set of integers to another finite set of integers is onto.
 24. Describe an algorithm that determines whether a function from a finite set to another finite set is one-to-one.
 25. Describe an algorithm that will count the number of 1s in a bit string by examining each bit of the string to determine whether it is a 1 bit.
 26. Change Algorithm 3 so that the binary search procedure compares x to a_m at each stage of the algorithm, with the algorithm terminating if $x = a_m$. What advantage does this version of the algorithm have?
 27. The **ternary search algorithm** locates an element in a list of increasing integers by successively splitting the list into three sublists of equal (or as close to equal as possible) size, and restricting the search to the appropriate piece. Specify the steps of this algorithm.
 28. Specify the steps of an algorithm that locates an element in a list of increasing integers by successively splitting the list into four sublists of equal (or as close to equal as possible) size, and restricting the search to the appropriate piece.
- In a list of elements the same element may appear several times. A **mode** of such a list is an element that occurs at least as often as each of the other elements; a list has more than one mode when more than one element appears the maximum number of times.
29. Devise an algorithm that finds a mode in a list of nondecreasing integers. (Recall that a list of integers is nondecreasing if each term is at least as large as the preceding term.)
 30. Devise an algorithm that finds all modes. (Recall that a list of integers is nondecreasing if each term of the list is at least as large as the preceding term.)
 31. Devise an algorithm that finds the first term of a sequence of integers that equals some previous term in the sequence.
 32. Devise an algorithm that finds all terms of a finite sequence of integers that are greater than the sum of all previous terms of the sequence.


33. Devise an algorithm that finds the first term of a sequence of positive integers that is less than the immediately preceding term of the sequence.
34. Use the bubble sort to sort 6, 2, 3, 1, 5, 4, showing the lists obtained at each step.
35. Use the bubble sort to sort 3, 1, 5, 7, 4, showing the lists obtained at each step.
36. Use the bubble sort to sort d, f, k, m, a, b , showing the lists obtained at each step.
- *37. Adapt the bubble sort algorithm so that it stops when no interchanges are required. Express this more efficient version of the algorithm in pseudocode.
38. Use the insertion sort to sort the list in Exercise 34, showing the lists obtained at each step.
39. Use the insertion sort to sort the list in Exercise 35, showing the lists obtained at each step.
40. Use the insertion sort to sort the list in Exercise 36, showing the lists obtained at each step.

The **selection sort** begins by finding the least element in the list. This element is moved to the front. Then the least element among the remaining elements is found and put into the second position. This procedure is repeated until the entire list has been sorted.

41. Sort these lists using the selection sort.

- | | |
|------------------|------------------|
| a) 3, 5, 4, 1, 2 | b) 5, 4, 3, 2, 1 |
| c) 1, 2, 3, 4, 5 | |

42. Write the selection sort algorithm in pseudocode.

 43. Describe an algorithm based on the linear search for determining the correct position in which to insert a new element in an already sorted list.

44. Describe an algorithm based on the binary search for determining the correct position in which to insert a new element in an already sorted list.

45. How many comparisons does the insertion sort use to sort the list 1, 2, ..., n ?
46. How many comparisons does the insertion sort use to sort the list $n, n-1, \dots, 2, 1$?

The **binary insertion sort** is a variation of the insertion sort that uses a binary search technique (see Exercise 44) rather than a linear search technique to insert the i th element in the correct place among the previously sorted elements.

47. Show all the steps used by the binary insertion sort to sort the list 3, 2, 4, 5, 1, 6.
48. Compare the number of comparisons used by the insertion sort and the binary insertion sort to sort the list 7, 4, 3, 8, 1, 5, 4, 2.

- *49. Express the binary insertion sort in pseudocode.

50. a) Devise a variation of the insertion sort that uses a linear search technique that inserts the j th element in the correct place by first comparing it with the $(j-1)$ st element, then the $(j-2)$ th element if necessary, and so on.
- b) Use your algorithm to sort 3, 2, 4, 5, 1, 6.
- c) Answer Exercise 45 using this algorithm.
- d) Answer Exercise 46 using this algorithm.

51. When a list of elements is in close to the correct order, would it be better to use an insertion sort or its variation described in Exercise 50?
52. Use the greedy algorithm to make change using quarters, dimes, nickels, and pennies for
 - a) 87 cents. b) 49 cents.
 - c) 99 cents. d) 33 cents.
53. Use the greedy algorithm to make change using quarters, dimes, nickels, and pennies for
 - a) 51 cents. b) 69 cents.
 - c) 76 cents. d) 60 cents.
54. Use the greedy algorithm to make change using quarters, dimes, and pennies (but no nickels) for each of the amounts given in Exercise 52. For which of these amounts does the greedy algorithm use the fewest coins of these denominations possible?
55. Use the greedy algorithm to make change using quarters, dimes, and pennies (but no nickels) for each of the amounts given in Exercise 53. For which of these amounts does the greedy algorithm use the fewest coins of these denominations possible?
56. Show that if there were a coin worth 12 cents, the greedy algorithm using quarters, 12-cent coins, dimes, nickels, and pennies would not always produce change using the fewest coins possible.
57. Use Algorithm 7 to schedule the largest number of talks in a lecture hall from a proposed set of talks, if the starting and ending times of the talks are 9:00 A.M. and 9:45 A.M.; 9:30 A.M. and 10:00 A.M.; 9:50 A.M. and 10:15 A.M.; 10:00 A.M. and 10:30 A.M.; 10:10 A.M. and 10:25 A.M.; 10:30 A.M. and 10:55 A.M.; 10:15 A.M. and 10:45 A.M.; 10:30 A.M. and 11:00 A.M.; 10:45 A.M. and 11:30 A.M.; 10:55 A.M. and 11:25 A.M.; 11:00 A.M. and 11:15 A.M.
58. Show that a greedy algorithm that schedules talks in a lecture hall, as described in Example 7, by selecting at each step the talk that overlaps the fewest other talks, does not always produce an optimal schedule.
- *59. a) Devise a greedy algorithm that determines the fewest lecture halls needed to accommodate n talks given the starting and ending time for each talk.
b) Prove that your algorithm is optimal.

Suppose we have s men m_1, m_2, \dots, m_s and s women w_1, w_2, \dots, w_s . We wish to match each person with a member



of the opposite gender. Furthermore, suppose that each person ranks, in order of preference, with no ties, the people of the opposite gender. We say that a matching of people of opposite genders to form couples is **stable** if we cannot find a man m and a woman w who are not assigned to each other such that m prefers w over his assigned partner and w prefers m to her assigned partner.

60. Suppose we have three men m_1, m_2 , and m_3 and three women w_1, w_2 , and w_3 . Furthermore, suppose that the preference rankings of the men for the three women, from highest to lowest, are $m_1: w_3, w_1, w_2$; $m_2: w_1, w_2, w_3$; $m_3: w_2, w_3, w_1$; and the preference rankings of the women for the three men, from highest to lowest, are $w_1: m_1, m_2, m_3$; $w_2: m_2, m_1, m_3$; $w_3: m_3, m_2, m_1$. For each of the six possible matchings of men and women to form three couples, determine whether this matching is stable.

The **deferred acceptance algorithm**, also known as the **Gale-Shapley algorithm**, can be used to construct a stable matching of men and women. In this algorithm, members of one gender are the **suitors** and members of the other gender the **suitees**. The algorithm uses a sequence of rounds; in each round every suitor whose proposal was rejected in the previous round proposes to his or her highest ranking suitee who has not already rejected a proposal from this suitor. A suitee rejects all proposals except that from the suitor that this suitee ranks highest among all the suitors who have proposed to this suitee in this round or previous rounds. The proposal of this highest ranking suitor remains pending and is rejected in a later round if a more appealing suitor proposes in that round. The series of rounds ends when every suitor has exactly one pending proposal. All pending proposals are then accepted.

61. Write the deferred acceptance algorithm in pseudocode.
62. Show that the deferred acceptance algorithm terminates.
- *63. Show that the deferred acceptance always terminates with a stable assignment.
64. Show that the problem of determining whether a program with a given input ever prints the digit 1 is unsolvable.
65. Show that the following problem is solvable. Given two programs with their inputs and the knowledge that exactly one of them halts, determine which halts.
66. Show that the problem of deciding whether a specific program with a specific input halts is solvable.

3.2 The Growth of Functions

Introduction


In Section 3.1 we discussed the concept of an algorithm. We introduced algorithms that solve a variety of problems, including searching for an element in a list and sorting a list. In Section 3.3 we will study the number of operations used by these algorithms. In particular, we will estimate the number of comparisons used by the linear and binary search algorithms to find an element in a sequence of n elements. We will also estimate the number of comparisons used by the

One useful fact is that the leading term of a polynomial determines its order. For example, if $f(x) = 3x^5 + x^4 + 17x^3 + 2$, then $f(x)$ is of order x^5 . This is stated in Theorem 4, whose proof is left as Exercise 50.

THEOREM 4

Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$, where a_0, a_1, \dots, a_n are real numbers with $a_n \neq 0$. Then $f(x)$ is of order x^n .

EXAMPLE 13

The polynomials $3x^8 + 10x^7 + 221x^2 + 1444$, $x^{19} - 18x^4 - 10,112$, and $-x^{99} + 40,001x^{98} + 100,003x$ are of orders x^8 , x^{19} , and x^{99} , respectively. 

Unfortunately, as Knuth observed, big- O notation is often used by careless writers and speakers as if it had the same meaning as big-Theta notation. Keep this in mind when you see big- O notation used. The recent trend has been to use big-Theta notation whenever both upper and lower bounds on the size of a function are needed.

Exercises

In Exercises 1–14, to establish a big- O relationship, find witnesses C and k such that $|f(x)| \leq C|g(x)|$ whenever $x > k$.

1. Determine whether each of these functions is $O(x)$.

a) $f(x) = 10$	b) $f(x) = 3x + 7$
c) $f(x) = x^2 + x + 1$	d) $f(x) = 5 \log x$
e) $f(x) = \lfloor x \rfloor$	f) $f(x) = \lceil x/2 \rceil$

2. Determine whether each of these functions is $O(x^2)$.

a) $f(x) = 17x + 11$	b) $f(x) = x^2 + 1000$
c) $f(x) = x \log x$	d) $f(x) = x^4/2$
e) $f(x) = 2^x$	f) $f(x) = \lfloor x \rfloor \cdot \lceil x \rceil$

3. Use the definition of “ $f(x)$ is $O(g(x))$ ” to show that $x^4 + 9x^3 + 4x + 7$ is $O(x^4)$.

4. Use the definition of “ $f(x)$ is $O(g(x))$ ” to show that $2^x + 17$ is $O(3^x)$.

5. Show that $(x^2 + 1)/(x + 1)$ is $O(x)$.

6. Show that $(x^3 + 2x)/(2x + 1)$ is $O(x^2)$.

7. Find the least integer n such that $f(x)$ is $O(x^n)$ for each of these functions.

a) $f(x) = 2x^3 + x^2 \log x$
b) $f(x) = 3x^3 + (\log x)^4$
c) $f(x) = (x^4 + x^2 + 1)/(x^3 + 1)$
d) $f(x) = (x^4 + 5 \log x)/(x^4 + 1)$

8. Find the least integer n such that $f(x)$ is $O(x^n)$ for each of these functions.

a) $f(x) = 2x^2 + x^3 \log x$
b) $f(x) = 3x^5 + (\log x)^4$
c) $f(x) = (x^4 + x^2 + 1)/(x^4 + 1)$
d) $f(x) = (x^3 + 5 \log x)/(x^4 + 1)$

9. Show that $x^2 + 4x + 17$ is $O(x^3)$ but that x^3 is not $O(x^2 + 4x + 17)$.

10. Show that x^3 is $O(x^4)$ but that x^4 is not $O(x^3)$.

11. Show that $3x^4 + 1$ is $O(x^4/2)$ and $x^4/2$ is $O(3x^4 + 1)$.

12. Show that $x \log x$ is $O(x^2)$ but that x^2 is not $O(x \log x)$.

13. Show that 2^n is $O(3^n)$ but that 3^n is not $O(2^n)$. (Note that this is a special case of Exercise 60.)

14. Determine whether x^3 is $O(g(x))$ for each of these functions $g(x)$.

a) $g(x) = x^2$	b) $g(x) = x^3$
c) $g(x) = x^2 + x^3$	d) $g(x) = x^2 + x^4$
e) $g(x) = 3^x$	f) $g(x) = x^3/2$

15. Explain what it means for a function to be $O(1)$.

16. Show that if $f(x)$ is $O(x)$, then $f(x)$ is $O(x^2)$.

17. Suppose that $f(x)$, $g(x)$, and $h(x)$ are functions such that $f(x)$ is $O(g(x))$ and $g(x)$ is $O(h(x))$. Show that $f(x)$ is $O(h(x))$.

18. Let k be a positive integer. Show that $1^k + 2^k + \cdots + n^k$ is $O(n^{k+1})$.

19. Determine whether each of the functions 2^{n+1} and 2^{2n} is $O(2^n)$.

20. Determine whether each of the functions $\log(n + 1)$ and $\log(n^2 + 1)$ is $O(\log n)$.

21. Arrange the functions \sqrt{n} , $1000 \log n$, $n \log n$, $2n!$, 2^n , 3^n , and $n^2/1,000,000$ in a list so that each function is big- O of the next function.

22. Arrange the function $(1.5)^n$, n^{100} , $(\log n)^3$, $\sqrt{n} \log n$, 10^n , $(n!)^2$, and $n^{99} + n^{98}$ in a list so that each function is big- O of the next function.

23. Suppose that you have two different algorithms for solving a problem. To solve a problem of size n , the first algorithm uses exactly $n(\log n)$ operations and the second algorithm uses exactly $n^{3/2}$ operations. As n grows, which algorithm uses fewer operations?

24. Suppose that you have two different algorithms for solving a problem. To solve a problem of size n , the first algorithm uses exactly $n^2 2^n$ operations and the second algorithm uses exactly $n!$ operations. As n grows, which algorithm uses fewer operations?

25. Give as good a big- O estimate as possible for each of these functions.
 a) $(n^2 + 8)(n + 1)$ b) $(n \log n + n^2)(n^3 + 2)$
 c) $(n! + 2^n)(n^3 + \log(n^2 + 1))$
26. Give a big- O estimate for each of these functions. For the function g in your estimate $f(x)$ is $O(g(x))$, use a simple function g of smallest order.
 a) $(n^3 + n^2 \log n)(\log n + 1) + (17 \log n + 19)(n^3 + 2)$
 b) $(2^n + n^2)(n^3 + 3^n)$
 c) $(n^n + n2^n + 5^n)(n! + 5^n)$
27. Give a big- O estimate for each of these functions. For the function g in your estimate that $f(x)$ is $O(g(x))$, use a simple function g of the smallest order.
 a) $n \log(n^2 + 1) + n^2 \log n$
 b) $(n \log n + 1)^2 + (\log n + 1)(n^2 + 1)$
 c) $n^{2^n} + n^{n^2}$
28. For each function in Exercise 1, determine whether that function is $\Omega(x)$ and whether it is $\Theta(x)$.
29. For each function in Exercise 2, determine whether that function is $\Omega(x^2)$ and whether it is $\Theta(x^2)$.
30. Show that each of these pairs of functions are of the same order.
 a) $3x + 7, x$
 b) $2x^2 + x - 7, x^2$
 c) $\lfloor x + 1/2 \rfloor, x$
 d) $\log(x^2 + 1), \log_2 x$
 e) $\log_{10} x, \log_2 x$
31. Show that $f(x)$ is $\Theta(g(x))$ if and only if $f(x)$ is $O(g(x))$ and $g(x)$ is $O(f(x))$.
32. Show that if $f(x)$ and $g(x)$ are functions from the set of real numbers to the set of real numbers, then $f(x)$ is $O(g(x))$ if and only if $g(x)$ is $\Omega(f(x))$.
33. Show that if $f(x)$ and $g(x)$ are functions from the set of real numbers to the set of real numbers, then $f(x)$ is $\Theta(g(x))$ if and only if there are positive constants k, C_1 , and C_2 such that $C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|$ whenever $x > k$.
34. a) Show that $3x^2 + x + 1$ is $\Theta(3x^2)$ by directly finding the constants k, C_1 , and C_2 in Exercise 33.
 b) Express the relationship in part (a) using a picture showing the functions $3x^2 + x + 1$, $C_1 \cdot 3x^2$, and $C_2 \cdot 3x^2$, and the constant k on the x -axis, where C_1, C_2 , and k are the constants you found in part (a) to show that $3x^2 + x + 1$ is $\Theta(3x^2)$.
35. Express the relationship $f(x)$ is $\Theta(g(x))$ using a picture. Show the graphs of the functions $f(x)$, $C_1|g(x)|$, and $C_2|g(x)|$, as well as the constant k on the x -axis.
36. Explain what it means for a function to be $\Omega(1)$.
37. Explain what it means for a function to be $\Theta(1)$.
38. Give a big- O estimate of the product of the first n odd positive integers.
39. Show that if f and g are real-valued functions such that $f(x)$ is $O(g(x))$, then for every positive integer n , $f^n(x)$ is $O(g^n(x))$. [Note that $f^n(x) = f(x)^n$.]
40. Show that for all real numbers a and b with $a > 1$ and $b > 1$, if $f(x)$ is $O(\log_b x)$, then $f(x)$ is $O(\log_a x)$.
41. Suppose that $f(x)$ is $O(g(x))$ where f and g are increasing and unbounded functions. Show that $\log |f(x)|$ is $O(\log |g(x)|)$.
42. Suppose that $f(x)$ is $O(g(x))$. Does it follow that $2^{f(x)}$ is $O(2^{g(x)})$?
43. Let $f_1(x)$ and $f_2(x)$ be functions from the set of real numbers to the set of positive real numbers. Show that if $f_1(x)$ and $f_2(x)$ are both $\Theta(g(x))$, where $g(x)$ is a function from the set of real numbers to the set of positive real numbers, then $f_1(x) + f_2(x)$ is $\Theta(g(x))$. Is this still true if $f_1(x)$ and $f_2(x)$ can take negative values?
44. Suppose that $f(x)$, $g(x)$, and $h(x)$ are functions such that $f(x)$ is $\Theta(g(x))$ and $g(x)$ is $\Theta(h(x))$. Show that $f(x)$ is $\Theta(h(x))$.
45. If $f_1(x)$ and $f_2(x)$ are functions from the set of positive integers to the set of positive real numbers and $f_1(x)$ and $f_2(x)$ are both $\Theta(g(x))$, is $(f_1 - f_2)(x)$ also $\Theta(g(x))$? Either prove that it is or give a counterexample.
46. Show that if $f_1(x)$ and $f_2(x)$ are functions from the set of positive integers to the set of real numbers and $f_1(x)$ is $\Theta(g_1(x))$ and $f_2(x)$ is $\Theta(g_2(x))$, then $(f_1 f_2)(x)$ is $\Theta((g_1 g_2)(x))$.
47. Find functions f and g from the set of positive integers to the set of real numbers such that $f(n)$ is not $O(g(n))$ and $g(n)$ is not $O(f(n))$.
48. Express the relationship $f(x)$ is $\Omega(g(x))$ using a picture. Show the graphs of the functions $f(x)$ and $Cg(x)$, as well as the constant k on the real axis.
49. Show that if $f_1(x)$ is $\Theta(g_1(x))$, $f_2(x)$ is $\Theta(g_2(x))$, and $f_2(x) \neq 0$ and $g_2(x) \neq 0$ for all real numbers $x > 0$, then $(f_1/f_2)(x)$ is $\Theta((g_1/g_2)(x))$.
50. Show that if $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$, where a_0, a_1, \dots, a_{n-1} , and a_n are real numbers and $a_n \neq 0$, then $f(x)$ is $\Theta(x^n)$.
- Big- O , big- Θ , and big- Ω notation can be extended to functions in more than one variable. For example, the statement $f(x, y)$ is $O(g(x, y))$ means that there exist constants C, k_1 , and k_2 such that $|f(x, y)| \leq C|g(x, y)|$ whenever $x > k_1$ and $y > k_2$.
51. Define the statement $f(x, y)$ is $\Theta(g(x, y))$.
52. Define the statement $f(x, y)$ is $\Omega(g(x, y))$.
53. Show that $(x^2 + xy + x \log y)^3$ is $O(x^6 y^3)$.
54. Show that $x^5 y^3 + x^4 y^4 + x^3 y^5$ is $\Omega(x^3 y^3)$.
55. Show that $\lfloor xy \rfloor$ is $O(xy)$.
56. Show that $\lceil xy \rceil$ is $\Omega(xy)$.
57. (Requires calculus) Show that if $c > d > 0$, then n^d is $O(n^c)$, but n^c is not $O(n^d)$.
58. (Requires calculus) Show that if $b > 1$ and c and d are positive, then $(\log_b n)^c$ is $O(n^d)$, but n^d is not $O((\log_b n)^c)$.

59. (Requires calculus) Show that if d is positive and $b > 1$, then n^d is $O(b^n)$ but b^n is not $O(n^d)$.
60. (Requires calculus) Show that if $c > b > 1$, then b^n is $O(c^n)$ but c^n is not $O(b^n)$.

The following problems deal with another type of asymptotic notation, called **little- o** notation. Because little- o notation is based on the concept of limits, a knowledge of calculus is needed for these problems. We say that $f(x)$ is $o(g(x))$ [read $f(x)$ is “little-oh” of $g(x)$], when

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0.$$

61. (Requires calculus) Show that
- x^2 is $o(x^3)$.
 - $x \log x$ is $o(x^2)$.
 - x^2 is $o(2^x)$.
 - $x^2 + x + 1$ is not $o(x^2)$.
62. (Requires calculus)
- Show that if $f(x)$ and $g(x)$ are functions such that $f(x)$ is $o(g(x))$ and c is a constant, then $cf(x)$ is $o(g(x))$, where $(cf)(x) = cf(x)$.
 - Show that if $f_1(x)$, $f_2(x)$, and $g(x)$ are functions such that $f_1(x)$ is $o(g(x))$ and $f_2(x)$ is $o(g(x))$, then $(f_1 + f_2)(x)$ is $o(g(x))$, where $(f_1 + f_2)(x) = f_1(x) + f_2(x)$.
63. (Requires calculus) Represent pictorially that $x \log x$ is $o(x^2)$ by graphing $x \log x$, x^2 , and $x \log x/x^2$. Explain how this picture shows that $x \log x$ is $o(x^2)$.
64. (Requires calculus) Express the relationship $f(x)$ is $o(g(x))$ using a picture. Show the graphs of $f(x)$, $g(x)$, and $f(x)/g(x)$.
- *65. (Requires calculus) Suppose that $f(x)$ is $o(g(x))$. Does it follow that $2^{f(x)}$ is $o(2^{g(x)})$?
- *66. (Requires calculus) Suppose that $f(x)$ is $o(g(x))$. Does it follow that $\log |f(x)|$ is $o(\log |g(x)|)$?
67. (Requires calculus) The two parts of this exercise describe the relationship between little- o and big- O notation.
- Show that if $f(x)$ and $g(x)$ are functions such that $f(x)$ is $o(g(x))$, then $f(x)$ is $O(g(x))$.
 - Show that if $f(x)$ and $g(x)$ are functions such that $f(x)$ is $O(g(x))$, then it does not necessarily follow that $f(x)$ is $o(g(x))$.
68. (Requires calculus) Show that if $f(x)$ is a polynomial of degree n and $g(x)$ is a polynomial of degree m where $m > n$, then $f(x)$ is $o(g(x))$.

69. (Requires calculus) Show that if $f_1(x)$ is $O(g(x))$ and $f_2(x)$ is $o(g(x))$, then $f_1(x) + f_2(x)$ is $O(g(x))$.

70. (Requires calculus) Let H_n be the n th **harmonic number**

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}.$$

Show that H_n is $O(\log n)$. [Hint: First establish the inequality

$$\sum_{j=2}^n \frac{1}{j} < \int_1^n \frac{1}{x} dx$$

by showing that the sum of the areas of the rectangles of height $1/j$ with base from $j-1$ to j , for $j = 2, 3, \dots, n$, is less than the area under the curve $y = 1/x$ from 2 to n .]

- *71. Show that $n \log n$ is $O(\log n!)$.

72. Determine whether $\log n!$ is $\Theta(n \log n)$. Justify your answer.

- *73. Show that $\log n!$ is greater than $(n \log n)/4$ for $n > 4$. [Hint: Begin with the inequality $n! > n(n-1)(n-2) \cdots \lceil n/2 \rceil$.]

Let $f(x)$ and $g(x)$ be functions from the set of real numbers to the set of real numbers. We say that the functions f and g are **asymptotic** and write $f(x) \sim g(x)$ if $\lim_{x \rightarrow \infty} f(x)/g(x) = 1$.

74. (Requires calculus) For each of these pairs of functions, determine whether f and g are asymptotic.

- $f(x) = x^2 + 3x + 7$, $g(x) = x^2 + 10$
- $f(x) = x^2 \log x$, $g(x) = x^3$
- $f(x) = x^4 + \log(3x^8 + 7)$,
 $g(x) = (x^2 + 17x + 3)^2$
- $f(x) = (x^3 + x^2 + x + 1)^4$,
 $g(x) = (x^4 + x^3 + x^2 + x + 1)^3$.

75. (Requires calculus) For each of these pairs of functions, determine whether f and g are asymptotic.

- $f(x) = \log(x^2 + 1)$, $g(x) = \log x$
- $f(x) = 2^{x+3}$, $g(x) = 2^{x+7}$
- $f(x) = 2^{2^x}$, $g(x) = 2^{x^2}$
- $f(x) = 2^{x^2+x+1}$, $g(x) = 2^{x^2+2x}$

3.3 Complexity of Algorithms

Introduction

When does an algorithm provide a satisfactory solution to a problem? First, it must always produce the correct answer. How this can be demonstrated will be discussed in Chapter 5. Second, it should be efficient. The efficiency of algorithms will be discussed in this section.

How can the efficiency of an algorithm be analyzed? One measure of efficiency is the time used by a computer to solve a problem using the algorithm, when input values are of a specified

offer little help in overcoming the complexity of algorithms of exponential or factorial time complexity. Because of the increased speed of computation, increases in computer memory, and the use of algorithms that take advantage of parallel processing, many problems that were considered impossible to solve five years ago are now routinely solved, and certainly five years from now this statement will still be true. This is even true when the algorithms used are intractable.

Exercises

1. Give a big- O estimate for the number of operations (where an operation is an addition or a multiplication) used in this segment of an algorithm.

```
t := 0
for i := 1 to 3
  for j := 1 to 4
    t := t + ij
```

2. Give a big- O estimate for the number additions used in this segment of an algorithm.

```
t := 0
for i := 1 to n
  for j := 1 to n
    t := t + i + j
```

3. Give a big- O estimate for the number of operations, where an operation is a comparison or a multiplication, used in this segment of an algorithm (ignoring comparisons used to test the conditions in the **for** loops, where a_1, a_2, \dots, a_n are positive real numbers).

```
m := 0
for i := 1 to n
  for j := i + 1 to n
    m := max( $a_i a_j$ , m)
```

4. Give a big- O estimate for the number of operations, where an operation is an addition or a multiplication, used in this segment of an algorithm (ignoring comparisons used to test the conditions in the **while** loop).

```
i := 1
t := 0
while i ≤ n
  t := t + i
  i := 2i
```

5. How many comparisons are used by the algorithm given in Exercise 16 of Section 3.1 to find the smallest natural number in a sequence of n natural numbers?
6. a) Use pseudocode to describe the algorithm that puts the first four terms of a list of real numbers of arbitrary length in increasing order using the insertion sort.
b) Show that this algorithm has time complexity $O(1)$ in terms of the number of comparisons used.
7. Suppose that an element is known to be among the first four elements in a list of 32 elements. Would a linear search or a binary search locate this element more rapidly?
8. Given a real number x and a positive integer k , determine the number of multiplications used to find x^{2^k} starting

with x and successively squaring (to find x^2, x^4 , and so on). Is this a more efficient way to find x^{2^k} than by multiplying x by itself the appropriate number of times?

9. Give a big- O estimate for the number of comparisons used by the algorithm that determines the number of 1s in a bit string by examining each bit of the string to determine whether it is a 1 bit (see Exercise 25 of Section 3.1).

- *10. a) Show that this algorithm determines the number of 1 bits in the bit string S :

```
procedure bit count( $S$ : bit string)
count := 0
while  $S \neq 0$ 
  count := count + 1
   $S := S \wedge (S - 1)$ 
return count {count is the number of 1s in  $S$ }
```

Here $S - 1$ is the bit string obtained by changing the rightmost 1 bit of S to a 0 and all the 0 bits to the right of this to 1s. [Recall that $S \wedge (S - 1)$ is the bitwise AND of S and $S - 1$.]

- b) How many bitwise AND operations are needed to find the number of 1 bits in a string S using the algorithm in part (a)?

11. a) Suppose we have n subsets S_1, S_2, \dots, S_n of the set $\{1, 2, \dots, n\}$. Express a brute-force algorithm that determines whether there is a disjoint pair of these subsets. [Hint: The algorithm should loop through the subsets; for each subset S_i , it should then loop through all other subsets; and for each of these other subsets S_j , it should loop through all elements k in S_i to determine whether k also belongs to S_j .]

- b) Give a big- O estimate for the number of times the algorithm needs to determine whether an integer is in one of the subsets.

12. Consider the following algorithm, which takes as input a sequence of n integers a_1, a_2, \dots, a_n and produces as output a matrix $\mathbf{M} = \{m_{ij}\}$ where m_{ij} is the minimum term in the sequence of integers a_i, a_{i+1}, \dots, a_j for $j \geq i$ and $m_{ij} = 0$ otherwise.

```
initialize  $\mathbf{M}$  so that  $m_{ij} = a_i$  if  $j \geq i$  and  $m_{ij} = 0$  otherwise
for  $i := 1$  to  $n$ 
  for  $j := i + 1$  to  $n$ 
    for  $k := i + 1$  to  $j$ 
       $m_{ij} := \min(m_{ij}, a_k)$ 
return  $\mathbf{M} = \{m_{ij}\}$  { $m_{ij}$  is the minimum term of  $a_i, a_{i+1}, \dots, a_j$ }
```

- a) Show that this algorithm uses $O(n^3)$ comparisons to compute the matrix **M**.
- b) Show that this algorithm uses $\Omega(n^3)$ comparisons to compute the matrix **M**. Using this fact and part (a), conclude that the algorithm uses $\Theta(n^3)$ comparisons. [Hint: Only consider the cases where $i \leq n/4$ and $j \geq 3n/4$ in the two outer loops in the algorithm.]

13. The conventional algorithm for evaluating a polynomial $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ at $x = c$ can be expressed in pseudocode by

```

procedure polynomial( $c, a_0, a_1, \dots, a_n$ : real numbers)
    power := 1
    y :=  $a_0$ 
    for  $i := 1$  to  $n$ 
        power := power *  $c$ 
        y := y +  $a_i$  * power
    return y { $y = a_n c^n + a_{n-1} c^{n-1} + \dots + a_1 c + a_0$ }

```

where the final value of y is the value of the polynomial at $x = c$.

- a) Evaluate $3x^2 + x + 1$ at $x = 2$ by working through each step of the algorithm showing the values assigned at each assignment step.
- b) Exactly how many multiplications and additions are used to evaluate a polynomial of degree n at $x = c$? (Do not count additions used to increment the loop variable.)
14. There is a more efficient algorithm (in terms of the number of multiplications and additions used) for evaluating polynomials than the conventional algorithm described in the previous exercise. It is called **Horner's method**. This pseudocode shows how to use this method to find the value of $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ at $x = c$.

```

procedure Horner( $c, a_0, a_1, a_2, \dots, a_n$ : real numbers)
    y :=  $a_n$ 
    for  $i := 1$  to  $n$ 
        y := y *  $c$  +  $a_{n-i}$ 
    return y { $y = a_n c^n + a_{n-1} c^{n-1} + \dots + a_1 c + a_0$ }

```

- a) Evaluate $3x^2 + x + 1$ at $x = 2$ by working through each step of the algorithm showing the values assigned at each assignment step.
- b) Exactly how many multiplications and additions are used by this algorithm to evaluate a polynomial of degree n at $x = c$? (Do not count additions used to increment the loop variable.)
15. What is the largest n for which one can solve within one second a problem using an algorithm that requires $f(n)$ bit operations, where each bit operation is carried out in 10^{-9} seconds, with these functions $f(n)$?
- a) $\log n$ b) n c) $n \log n$
d) n^2 e) 2^n f) $n!$
16. What is the largest n for which one can solve within a day using an algorithm that requires $f(n)$ bit operations, where each bit operation is carried out in 10^{-11} seconds, with these functions $f(n)$?

- a) $\log n$ b) $1000n$ c) n^2
d) $1000n^2$ e) n^3 f) 2^n
g) 2^{2n} h) 2^{2^n}

17. What is the largest n for which one can solve within a minute using an algorithm that requires $f(n)$ bit operations, where each bit operation is carried out in 10^{-12} seconds, with these functions $f(n)$?
- a) $\log \log n$ b) $\log n$ c) $(\log n)^2$
d) $1000000n$ e) n^2 f) 2^n
g) 2^{n^2}
18. How much time does an algorithm take to solve a problem of size n if this algorithm uses $2n^2 + 2^n$ operations, each requiring 10^{-9} seconds, with these values of n ?
- a) 10 b) 20 c) 50 d) 100
19. How much time does an algorithm using 2^{50} operations need if each operation takes these amounts of time?
- a) 10^{-6} s b) 10^{-9} s c) 10^{-12} s
20. What is the effect in the time required to solve a problem when you double the size of the input from n to $2n$, assuming that the number of milliseconds the algorithm uses to solve the problem with input size n is each of these function? [Express your answer in the simplest form possible, either as a ratio or a difference. Your answer may be a function of n or a constant.]
- a) $\log \log n$ b) $\log n$ c) $100n$
d) $n \log n$ e) n^2 f) n^3
g) 2^n
21. What is the effect in the time required to solve a problem when you increase the size of the input from n to $n + 1$, assuming that the number of milliseconds the algorithm uses to solve the problem with input size n is each of these function? [Express your answer in the simplest form possible, either as a ratio or a difference. Your answer may be a function of n or a constant.]
- a) $\log n$ b) $100n$ c) n^2
d) n^3 e) 2^n f) 2^{n^2}
g) $n!$
22. Determine the least number of comparisons, or best-case performance,
- a) required to find the maximum of a sequence of n integers, using Algorithm 1 of Section 3.1.
b) used to locate an element in a list of n terms with a linear search.
c) used to locate an element in a list of n terms using a binary search.
23. Analyze the average-case performance of the linear search algorithm, if exactly half the time the element x is not in the list and if x is in the list it is equally likely to be in any position.
24. An algorithm is called **optimal** for the solution of a problem with respect to a specified operation if there is no algorithm for solving this problem using fewer operations.

- a) Show that Algorithm 1 in Section 3.1 is an optimal algorithm with respect to the number of comparisons of integers. [Note: Comparisons used for bookkeeping in the loop are not of concern here.]
 - b) Is the linear search algorithm optimal with respect to the number of comparisons of integers (not including comparisons used for bookkeeping in the loop)?
25. Describe the worst-case time complexity, measured in terms of comparisons, of the ternary search algorithm described in Exercise 27 of Section 3.1.
 26. Describe the worst-case time complexity, measured in terms of comparisons, of the search algorithm described in Exercise 28 of Section 3.1.
 27. Analyze the worst-case time complexity of the algorithm you devised in Exercise 29 of Section 3.1 for locating a mode in a list of nondecreasing integers.
 28. Analyze the worst-case time complexity of the algorithm you devised in Exercise 30 of Section 3.1 for locating all modes in a list of nondecreasing integers.
 29. Analyze the worst-case time complexity of the algorithm you devised in Exercise 31 of Section 3.1 for finding the first term of a sequence of integers equal to some previous term.
 30. Analyze the worst-case time complexity of the algorithm you devised in Exercise 32 of Section 3.1 for finding all terms of a sequence that are greater than the sum of all previous terms.
 31. Analyze the worst-case time complexity of the algorithm you devised in Exercise 33 of Section 3.1 for finding the first term of a sequence less than the immediately preceding term.
 32. Determine the worst-case complexity in terms of comparisons of the algorithm from Exercise 5 in Section 3.1 for determining all values that occur more than once in a sorted list of integers.
 33. Determine the worst-case complexity in terms of comparisons of the algorithm from Exercise 9 in Section 3.1 for determining whether a string of n characters is a palindrome.
 34. How many comparisons does the selection sort (see preamble to Exercise 41 in Section 3.1) use to sort n items? Use your answer to give a big- O estimate of the complexity of the selection sort in terms of number of comparisons for the selection sort.
 35. Find a big- O estimate for the worst-case complexity in terms of number of comparisons used and the number of terms swapped by the binary insertion sort described in the preamble to Exercise 47 in Section 3.1.
 36. Show that the greedy algorithm for making change for n cents using quarters, dimes, nickels, and pennies has $O(n)$ complexity measured in terms of comparisons needed.
 37. Find the complexity of a brute-force algorithm for scheduling the talks by examining all possible subsets of the talks. [Hint: Use the fact that a set with n elements has 2^n subsets.]
 38. Find the complexity of the greedy algorithm for scheduling the most talks by adding at each step the talk with the earliest end time compatible with those already scheduled (Algorithm 7 in Section 3.1). Assume that the talks are not already sorted by earliest end time and assume that the worst-case time complexity of sorting is $O(n \log n)$.
 39. Describe how the number of comparisons used in the worst case changes when these algorithms are used to search for an element of a list when the size of the list doubles from n to $2n$, where n is a positive integer.
 - a) linear search
 - b) binary search
 40. Describe how the number of comparisons used in the worst case changes when the size of the list to be sorted doubles from n to $2n$, where n is a positive integer when these sorting algorithms are used.
 - a) bubble sort
 - b) insertion sort
 - c) selection sort (described in the preamble to Exercise 41 in Section 3.1)
 - d) binary insertion sort (described in the preamble to Exercise 47 in Section 3.1)

An $n \times n$ matrix is called **upper triangular** if $a_{ij} = 0$ whenever $i > j$.

41. From the definition of the matrix product, describe an algorithm in English for computing the product of two upper triangular matrices that ignores those products in the computation that are automatically equal to zero.
42. Give a pseudocode description of the algorithm in Exercise 41 for multiplying two upper triangular matrices.
43. How many multiplications of entries are used by the algorithm found in Exercise 41 for multiplying two $n \times n$ upper triangular matrices?

In Exercises 44–45 assume that the number of multiplications of entries used to multiply a $p \times q$ matrix and a $q \times r$ matrix is pqr .

44. What is the best order to form the product \mathbf{ABC} if \mathbf{A} , \mathbf{B} , and \mathbf{C} are matrices with dimensions 3×9 , 9×4 , and 4×2 , respectively?
45. What is the best order to form the product \mathbf{ABCD} if \mathbf{A} , \mathbf{B} , \mathbf{C} , and \mathbf{D} are matrices with dimensions 30×10 , 10×40 , 40×50 , and 50×30 , respectively?

*46. In this exercise we deal with the problem of **string matching**.

- a) Explain how to use a brute-force algorithm to find the first occurrence of a given string of m characters, called the **target**, in a string of n characters, where $m \leq n$, called the **text**. [Hint: Think in terms of finding a match for the first character of the target and checking successive characters for a match, and if they do not all match, moving the start location one character to the right.]
- b) Express your algorithm in pseudocode.
- c) Give a big- O estimate for the worst-case time complexity of the brute-force algorithm you described.

Exercises 37 and 38 deal with the problem of scheduling the most talks possible given the start and end times of n talks.

37. Find the complexity of a brute-force algorithm for scheduling the talks by examining all possible subsets of the talks. [Hint: Use the fact that a set with n elements has 2^n subsets.]