CHAPTER 6

Algebraic and Combinatorial Circuits

Algebraic circuits combine operations drawn from an algebraic system. In this chapter we develop algebraic and combinatorial circuits for a variety of generally non-Boolean problems, including multiplication and inversion of matrices, convolution, the discrete Fourier transform, and sorting networks. These problems are used primarily to illustrate concepts developed in later chapters, so that this chapter may be used for reference when studying those chapters.

For each of the problems examined here the natural algorithms are straight-line and the graphs are directed and acyclic; that is, they are circuits. Not only are straight-line algorithms the ones typically used for these problems, but in some cases they are the best possible.

The quality of the circuits developed here is measured by circuit size, the number of circuit operations, and circuit depth, the length of the longest path between input and output vertices. Circuit size is a measure of the work necessary to execute the corresponding straight-line program. Circuit depth is a measure of the minimal time needed for a problem on a parallel machine.

For some problems, such as matrix inversion, we give serial (large-depth) as well as parallel (small-depth) circuits. The parallel circuits generally require considerably more circuit elements than the corresponding serial circuits.

6.1 Straight-Line Programs

Straight-line programs (SLP) are defined in Section 2.2. Each SLP step is an input, computation, or output step. The notation $(s \text{ READ } x)$ indicates that the sth step is an input step on which the value x is read. The notation $(s \text{ OUTPUT } i)$ indicates that the result of the ith step is to be provided as output. Finally, the notation $(s \text{ OP } i \ \ldots \ k)$ indicates that the sth step computes the value of the operator OP on the results generated at steps $i, \ldots, k$. We require that $s > i, \ldots, k$ so that the result produced at step s depends only on the results produced at earlier steps. In this chapter we consider SLPs in which the inputs and operators have values over a set $\mathcal{A}$ that is generally not binary. Thus, the circuits considered here are generally not logic circuits. The basis Ω for an SLP is the set of operators it uses. A **circuit** is the graph of a straight-line program. By its nature this graph is directed and acyclic.

An example of a straight-line program that computes the fast Fourier transform (FFT) on four inputs is given below. (The FFT is introduced in Section 6.7.3.) Here the function $f_{+,\,\alpha}(a, b) = a + b\alpha$ where α is a power of a constant ω that is a principal nth root of unity of a commutative ring $\mathcal{R}$. (See Section 6.7.1.) The arguments a and b are variables with values in $\mathcal{R}$.

$$
\begin{array}{llll}
(1 & \text{READ} & a_0) & \\
(2 & \text{READ} & a_2) & \\
(3 & \text{READ} & a_1) & \\
(4 & \text{READ} & a_3) & \\
(5 & f_{+,\,\omega^0} & 1 & 2) \\
(6 & f_{+,\,\omega^2} & 1 & 2)
\end{array}
\qquad
\begin{array}{llll}
(7 & f_{+,\,\omega^0} & 3 & 4) \\
(8 & f_{+,\,\omega^2} & 3 & 4) \\
(9 & f_{+,\,\omega^0} & 5 & 7) \\
(10 & f_{+,\,\omega^1} & 6 & 8) \\
(11 & f_{+,\,\omega^2} & 5 & 7) \\
(12 & f_{+,\,\omega^3} & 6 & 8)
\end{array}
$$

The graph of the above SLP is the familiar FFT **butterfly graph** shown in Fig. 6.1. Assignment statements are associated with vertices of in-degree zero and operator statements are associated with other vertices. We attach the name of the operator or variable associated with each step to the corresponding vertex in the graph. We often suppress the unique indices of vertices, although they are retained in Fig. 6.1.

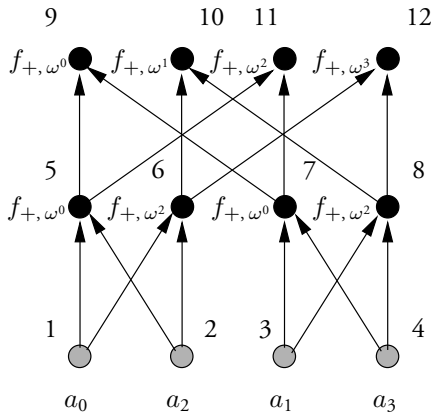

Figure 6.1 The FFT butterfly graph on four inputs.

The function g_s is associated with the sth step. The identity function with value v is associated with the assignment statement $(r \text{ READ } v)$. Associated with the computation step $(s \text{ OP } i \ \ldots \ k)$ is the function $g_s = \text{OP}(g_i, \ldots, g_k)$, where $g_i, \ldots, g_k$ are the functions computed at the steps on which the sth step depends. If a straight-line program has n inputs and m outputs, it computes a function $f : \mathcal{A}^n \mapsto \mathcal{A}^m$. If s_1, s_2, …, s_m are the output steps, then $f = (g_{s_1}, g_{s_2}, \ldots, g_{s_m})$. The function computed by a circuit is the function computed by the corresponding straight-line program.

In the example above, $g_{11} = f_{+,\,\omega^2}(g_5, g_7) = g_5 + g_7\omega^2$, where $g_5 = f_{+,\,\omega^0}(g_1, g_2) = a_0 + a_2\omega^0 = a_0 + a_2$ and $g_7 = f_{+,\,\omega^0}(g_3, g_4) = a_1 + a_3\omega^0 = a_1 + a_3$. Thus,

$$g_{11} = a_0 + a_1\omega^2 + a_2 + a_3\omega^2$$

which is the value of the polynomial $p(x)$ at $x = \omega^2$ when $\omega^4 = 1$:

$$p(x) = a_0 + a_1x + a_2x^2 + a_3x^3$$

The **size** of a circuit is the number of operator statements it contains. Its **depth** is the length of (number of edges on) the longest path from an input to an output vertex. The **basis** Ω is the set of operators used in the circuit. The size and depth of the smallest and shallowest circuits for a function f over the basis Ω are denoted $C_\Omega(f)$ and $D_\Omega(f)$, respectively. In this chapter we derive upper bounds on the size and depth of circuits.

6.2 Mathematical Preliminaries

In this section we introduce rings, fields and matrices, concepts widely used in this chapter.

6.2.1 Rings and Fields

Rings and fields are algebraic systems that consists of a set with two special elements, 0 and 1, and two operations called addition and multiplication that obey a small set of rules.

DEFINITION 6.2.1 *A* **ring** $\mathcal{R}$ *is a five-tuple* $(R, +, *, \mathbf{0}, \mathbf{1})$*, where* R *is* **closed** *under* **addition** $+$ *and* **multiplication** $*$ *(that is,* $+ : R^2 \mapsto R$ *and* $* : R^2 \mapsto R$*) and* $+$ *and* $*$ *are associative (for all* $a, b, c \in R$*,* $a + (b + c) = (a + b) + c$ *and* $a * (b * c) = (a * b) * c$*). Also,* $\mathbf{0}, \mathbf{1} \in R$*, where* $\mathbf{0}$ *is the* **identity under addition** *(for all* $a \in R$*,* $a + \mathbf{0} = \mathbf{0} + a = a$*) and* $\mathbf{1}$ *is the* **identity under multiplication** *(for all* $a \in R$*,* $a * \mathbf{1} = \mathbf{1} * a = a$*). In addition,* $\mathbf{0}$ *is an* **annihilator under multiplication** *(for all* $a \in R$*,* $a * \mathbf{0} = \mathbf{0} * a = \mathbf{0}$*). Every element of* R *has an* **additive inverse** *(for all* $a \in R$*, there exists an element* $-a$ *such that* $(-a) + a = a + (-a) = \mathbf{0}$*). Finally, addition is* **commutative** *(for all* $a, b \in R$*,* $a + b = b + a$*) and multiplication* **distributes over** *addition (for all* $a, b, c \in R$*,* $a * (b + c) = (a * b) + (a * c)$ *and* $(b + c) * a = (b * a) + (c * a)$*). A ring is* **commutative** *if multiplication is commutative (for all* $a, b \in R$*,* $a * b = b * a$*). A* **field** *is a commutative ring in which each element other than* $\mathbf{0}$ *has a* **multiplicative inverse** *(for all* $a \in R$*,* $a \neq \mathbf{0}$*, there exists an element* a^{-1} *such that* $a * a^{-1} = \mathbf{1}$*).*

Let $\mathbb{Z}$ be the set of positive and non-negative integers and let $+$ and $*$ denote integer addition and multiplication. Then $(\mathbb{Z}, +, *, 0, 1)$ is a commutative ring. (See Problem 6.1.) Similarly, the system $(\{0, 1\}, +, *, 0, 1)$, where $+$ is addition modulo 2 (for all $a, b \in \{0, 1\}$, $a + b$ is the remainder after division by 2 or the EXCLUSIVE OR operation) and $*$ is the AND

operation, is a commutative ring, as the reader can show. A third commutative ring is the integers modulo p together with the operations of addition and multiplication modulo p. (See Problem 6.2.) The ring of matrices introduced in the next section is not commutative. Some important commutative rings are introduced in Section 6.7.1.

6.2.2 Matrices

A **matrix over a set** R is a rectangular array of elements drawn from R consisting of some number m of rows and some number n of columns. Rows are indexed by integers from the set $\{1, 2, 3, \ldots, m\}$ and columns are indexed by integers from the set $\{1, 2, 3, \ldots, n\}$. The entry in the ith row and jth column of A is denoted $a_{i,j}$, as suggested in the following example:

$$A = [a_{i,j}] = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

Thus, $a_{2,3} = 7$ and $a_{3,1} = 9$.

The **transpose** of a matrix A, denoted A^T, is the matrix obtained from A by exchanging rows and columns, as shown below for the matrix A above:

$$A^T = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}$$

Clearly, the transpose of the transpose of a matrix A, $(A^T)^T$, is the matrix A.

A **column n-vector $\boldsymbol{x}$** is a matrix containing one column and n rows, for example:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \\ \vdots \\ 8 \end{bmatrix}$$

A **row m-vector $\boldsymbol{y}$** is a matrix containing one row and m columns, for example:

$$\boldsymbol{y} = [y_1, y_2, \ldots, y_m] = [1, 5, \ldots, 9]$$

The transpose of a row vector is a column vector and vice versa.

A **square matrix** is an $n \times n$ matrix for some integer n. The **main diagonal** of an $n \times n$ square matrix A is the set of elements $\{a_{1,1}, a_{2,2}, \ldots, a_{n-1,n-1}, a_{n,n}\}$. The diagonal below (above) the main diagonal is the elements $\{a_{2,1}, a_{3,2}, \ldots, a_{n,n-1}\}$ ($\{a_{1,2}, a_{2,3}, \ldots, a_{n-1,n}\}$). The $n \times n$ **identity matrix**, I_n, is a square $n \times n$ matrix with value 1 on the main diagonal and 0 elsewhere. The $n \times n$ **zero matrix**, 0_n, has value 0 in each position. A matrix is **upper** (**lower**) **triangular** if all elements below (above) the main diagonal are 0. A square matrix A is **symmetric** if $A = A^T$, that is, $a_{i,j} = a_{j,i}$ for all $1 \leq i, j \leq n$.

The **scalar product** of a scalar $c \in R$ and an $n \times m$ matrix A over R, denoted cA, has value $ca_{i,j}$ in row i and column j.

The **matrix-vector product** between an $m \times n$ matrix A and a column n-vector $\boldsymbol{x}$ is the column m-vector $\boldsymbol{b}$ below:

$$\boldsymbol{b} = A\boldsymbol{x} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{n-1,2} & \dots & a_{m-1,n} \\ a_{m,1} & a_{n,2} & \dots & a_{m,n} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}$$

$$= \begin{bmatrix} a_{1,1} * x_1 & + & a_{1,2} * x_2 & + & \cdots & + & a_{1,n} * x_n \\ a_{2,1} * x_1 & + & a_{2,2} * x_2 & + & \cdots & + & a_{2,n} * x_n \\ \vdots & & \vdots & & \ddots & & \vdots \\ a_{m-1,1} * x_1 & + & a_{m-1,2} * x_2 & + & \cdots & + & a_{m-1,n} * x_n \\ a_{m,1} * x_1 & + & a_{m,2} * x_2 & + & \cdots & + & a_{m,n} * x_n \end{bmatrix}$$

Thus, b_j is defined as follows for $1 \leq j \leq n$:

$$b_j = a_{i,1} * x_1 + a_{i,2} * x_2 + \cdots + a_{i,m} * x_m$$

The matrix-vector product between a row m-vector $\boldsymbol{x}$ and an $m \times n$ matrix A is the row n-vector $\boldsymbol{b}$ below:

$$\boldsymbol{b} = [b_i] = \boldsymbol{x}A$$

where for $1 \leq i \leq n$ b_i satisfies

$$b_i = x_1 * a_{1,i} + x_2 * a_{2,i} + \cdots + x_m * a_{m,i}$$

The special case of a matrix-vector product between a row n-vector, $\boldsymbol{x}$, and a column n vector, $\boldsymbol{y}$, denoted $\boldsymbol{x} \cdot \boldsymbol{y}$ and defined below, is called the **inner product** of the two vectors:

$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} x_i * y_i$$

If the entries of the $n \times n$ matrix A and the column n-vectors $\boldsymbol{x}$ and $\boldsymbol{b}$ shown below are drawn from a ring $\mathcal{R}$ and A and $\boldsymbol{b}$ are given, then the following matrix equation defines a **linear system** of n equations in the n unknowns $\boldsymbol{x}$:

$$A\boldsymbol{x} = \boldsymbol{b}$$

An example of a linear system of four equations in four unknowns is

$$\begin{array}{rcrcrcrcl} 1 * x_1 & + & 2 * x_2 & + & 3 * x_3 & + & 4 * x_4 & = & 17 \\ 5 * x_1 & + & 6 * x_2 & + & 7 * x_3 & + & 8 * x_4 & = & 18 \\ 9 * x_1 & + & 10 * x_2 & + & 11 * x_3 & + & 12 * x_4 & = & 19 \\ 13 * x_1 & + & 14 * x_2 & + & 15 * x_3 & + & 16 * x_4 & = & 20 \end{array}$$

It can be expressed as follows:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 17 \\ 18 \\ 19 \\ 20 \end{bmatrix}$$

Solving a linear system, when it is possible, consists of finding values for $\boldsymbol{x}$ given values for A and $\boldsymbol{b}$. (See Section 6.6.)

Consider the set of $m \times n$ matrices whose entries are drawn from a ring $\mathcal{R}$. The **matrix addition function** $f_{A+B}^{(m,n)} : \mathcal{R}^{2mn} \mapsto \mathcal{R}^{mn}$ on two $m \times n$ matrices $A = [a_{i,j}]$ and $B = [b_{i,j}]$ generates a matrix $C = f_{A+B}^{(m,n)}(A, B) = A +_{m,n} B = [c_{i,j}]$, where $+_{m,n}$ is the infix matrix addition operator and $c_{i,j}$ is defined as

$$c_{i,j} = a_{i,j} + b_{i,j}$$

The straight-line program based on this equation uses one instance of the ring addition operator $+$ for each entry in C. It follows that over the basis $\{+\}$, $C_+(f_{A+B}^{(m,n)}) = mn$ and $D_+(f_{A+B}^{(m,n)}) = 1$. Two special cases of matrix addition are the addition of square matrices ($m = n$), denoted $+_n$, and the addition of row or column vectors that are either $1 \times n$ or $m \times 1$ matrices.

The **matrix multiplication function** $f_{A\times B}^{(n)} : \mathcal{R}^{(m+p)n} \mapsto \mathcal{R}^{mp}$ multiplies an $m \times n$ matrix $A = [a_{i,j}]$ by an $n \times p$ matrix $B = [b_{i,j}]$ to produce the $m \times p$ matrix $C = f_{A\times B}^{(n)}(A, B) = A \times_n B = [c_{i,j}]$, where

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} * b_{k,j} \tag{6.1}$$

and $\times_n$ is the infix matrix multiplication operator. The subscript on $\times_n$ is usually dropped when the dimensions of the matrices are understood. The **standard matrix multiplication algorithm** for multiplying an $m \times n$ matrix A by an $n \times p$ matrix B forms mp **inner products** of the kind shown in equation (6.1). Thus, it uses mnp instances of the ring multiplication operator and $m(n-1)p$ instances of the ring addition operator.

A fast algorithm for matrix multiplication is given in Section 6.3.1. It is now straightforward to show the following result. (See Problem 6.4.)

THEOREM 6.2.1 *Let $M_{n\times n}$ be the set of $n \times n$ matrices over a commutative ring $\mathcal{R}$. The system $\mathcal{M}_{n\times n} = (M_{n\times n}, +_n, \times_n, 0_n, I_n)$, where $+_n$ and $\times_n$ are the matrix addition and multiplication operators and 0_n and I_n are the $n \times n$ zero and identity matrices, is a ring.*

The ring of matrices $\mathcal{M}_{n\times n}$ is not a commutative ring because matrix multiplication is not commutative. For example, the following two matrices do not commute, that is, $AB \neq BA$:

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

A **linear combination** of a subset of the rows of an $n \times m$ matrix A is a sum of scalar products of the rows in this subset. A linear combination is **non-zero** if the sum of the scalar

product is not the zero vector. A set of rows of a matrix A over a field $\mathcal{R}$ is **linearly independent** if all linear combinations are non-zero except when each scalar is zero.

The **rank of an** $n \times m$ **matrix** A over a field $\mathcal{R}$, $f^{(n)}_{\text{rank}} : \mathcal{R}^{n^2} \mapsto \mathbb{N}$, is the maximum number of linearly independent rows of A. It is also the maximum number of linearly independent columns of A. (See Problem 6.5.) We write $\text{rank}(A) = f^{(n)}_{\text{rank}}(A)$. An $n \times n$ matrix A is **non-singular** if $\text{rank}(A) = n$.

If an $n \times n$ matrix A over a field $\mathcal{R}$ is non-singular, it has an **inverse** A^{-1} that is an $n \times n$ matrix with the following properties:

$$AA^{-1} = A^{-1}A = I_n$$

where I_n is the $n \times n$ identity matrix. That is, there is a (partial) inverse function $f^{(n)}_{\text{inv}} : \mathcal{R}^{n^2} \mapsto \mathcal{R}^{n^2}$ that is defined for non-singular square matrices A such that $f^{(n)}_{\text{inv}}(A) = A^{-1}$. $f^{(n)}_{\text{inv}}$ is partial because it is not defined for singular matrices. Below we exhibit a matrix and its inverse over a field $\mathcal{R}$.

$$\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

Algorithms for matrix inversion are given in Section 6.5.

We now show that the inverse $(AB)^{-1}$ of the product AB of two invertible matrices, A and B, over a field $\mathcal{R}$ is the product of their inverses in reverse order.

LEMMA 6.2.1 *Let A and B be invertible square matrices over a field $\mathcal{R}$. Then the following relationship holds:*

$$(AB)^{-1} = B^{-1}A^{-1}$$

Proof To show that $(AB)^{-1} = B^{-1}A^{-1}$, we multiply AB either on the left or right by $B^{-1}A^{-1}$ to produce the identity matrix:

$$\begin{array}{ccccccccc} AB(AB)^{-1} & = & ABB^{-1}A^{-1} & = & A(BB^{-1})A^{-1} & = & AA^{-1} & = & I \\ (AB)^{-1}AB & = & B^{-1}A^{-1}AB & = & B^{-1}(A^{-1}A)B & = & B^{-1}B & = & I \end{array}$$

■

The transpose of the product of an $m \times n$ matrix A and an $n \times p$ matrix B over a ring $\mathcal{R}$ is the product of their transposes in reverse order:

$$(A\,B)^T = B^T\,A^T$$

(See Problem 6.6.) In particular, the following identity holds for an $m \times n$ matrix A and a column n-vector $\boldsymbol{x}$:

$$\boldsymbol{x}^T A^T = (A\boldsymbol{x})^T$$

A **block matrix** is a matrix in which each entry is a matrix with fixed dimensions. For example, when n is even it may be convenient to view an $n \times n$ matrix as a 2×2 matrix whose four entries are $(n/2) \times (n/2)$ matrices.

Two special types of matrix that are frequently encountered are the Toeplitz and circulant matrices. An $n \times n$ **Toeplitz matrix** T has the property that its (i, j) entry $t_{i,j} = a_r$ for

$j = i - n + 1 + r$ and $0 \leq r \leq 2n - 2$. A generic Toeplitz matrix T is shown below:

$$T = \begin{bmatrix} a_{n-1} & a_n & a_{n+1} & \cdots & a_{2n-2} \\ a_{n-2} & a_{n-1} & a_n & \cdots & a_{2n-3} \\ a_{n-3} & a_{n-2} & a_{n-1} & \cdots & a_{2n-4} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_0 & a_1 & a_2 & \cdots & a_{n-1} \end{bmatrix}$$

An $n \times n$ **circulant matrix** C has the property that the entries on the kth row are a right cyclic shift by $k - 1$ places of the entries on the first row, as suggested below.

$$C = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ a_{n-1} & a_0 & a_1 & \cdots & a_{n-2} \\ a_{n-2} & a_{n-1} & a_0 & \cdots & a_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_0 \end{bmatrix}$$

The circulant is a type of Toeplitz matrix. Thus the function defined by the product of a Toeplitz matrix and a vector contains as a subfunction the function defined by the product of a circulant matrix and a vector. Consequently, any algorithm to multiply a vector by a Toeplitz matrix can be used to multiply a circulant by a vector.

As stated in Section 2.11, a **permutation** $\pi : \mathcal{R}^n \mapsto \mathcal{R}^n$ of an n-tuple $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ over the set $\mathcal{R}$ is a rearrangement $\pi(\boldsymbol{x}) = (x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$ of the components of $\boldsymbol{x}$. A $n \times n$ **permutation matrix** P has entries from the set $\{0, 1\}$ (here 0 and 1 are the identities under addition and multiplication for a ring $\mathcal{R}$) with the property that each row and column of P has exactly one instance of 1. (See the example below.) Let A be an $n \times n$ matrix. Then AP contains the columns of A in a permuted order determined by P. A similar statement applies to PA. Shown below is a permutation matrix P and the result of multiplying it on the right by a matrix A on the left. In this case P interchanges the first two columns of A.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 3 & 4 \\ 6 & 5 & 7 & 8 \\ 10 & 9 & 11 & 12 \\ 14 & 13 & 15 & 16 \end{bmatrix}$$

6.3 Matrix Multiplication

Matrix multiplication is defined in Section 6.2. The **standard matrix multiplication algorithm** computes the matrix product using the formula for $c_{i,j}$ given in (6.1). It performs nmp multiplications and $n(m-1)p$ additions. As shown in Section 6.3.1, however, matrices can be multiplied with many fewer operations.

Boolean matrix multiplication is matrix multiplication for matrices over $\mathcal{B}$ when $+$ denotes OR and $*$ denotes AND. Another example is matrix multiplication over the set of integers

modulo a prime p, a set that forms a finite field under addition and multiplication modulo p. (See Problem 6.3.)

In the next section we describe Strassen's algorithm, a straight-line program realizable by a logarithmic-depth circuit of size $O(n^{2.807})$. This is not the final word on matrix multiplication, however. Winograd and Coppersmith [81] have improved the bound to $O(n^{2.38})$. Despite this progress, the smallest asymptotic bound on matrix multiplication remains unknown.

Since later in this chapter we design algorithms that make use of matrix multiplication, it behooves us to make the following definition concerning the number of ring operations to multiply two $n \times n$ matrices over a ring $\mathcal{R}$.

DEFINITION 6.3.1 *Let $K \geq 1$. Then $M_{\text{matrix}}(n, K)$ is the size of the smallest circuit of depth $K \log_2 n$ over a commutative ring $\mathcal{R}$ for the multiplication of two $n \times n$ matrices.*

The following assumptions on the rate of growth of $M_{\text{matrix}}(n, K)$ with n make subsequent analysis easier. They are satisfied by Strassen's algorithm.

ASSUMPTION 6.3.1 *We assume that for all c satisfying $0 \leq c \leq 1$ and $n \geq 1$,*

$$M_{\text{matrix}}(cn, K) \leq c^2 M_{\text{matrix}}(n, K)$$

ASSUMPTION 6.3.2 *We assume there exists an integer $n_0 > 0$ such that, for $n \geq n_0$,*

$$2n^2 \leq M_{\text{matrix}}(n, K)$$

6.3.1 Strassen's Algorithm

Strassen [318] has developed a fast algorithm for multiplying two square matrices over a commutative ring $\mathcal{R}$. This algorithm makes use of the additive inverse of ring elements to reduce the total number of operations performed.

Let n be even. Given two $n \times n$ matrices, A and B, we write them and their product C as 2×2 matrices whose components are $(n/2) \times (n/2)$ matrices:

$$C = \begin{bmatrix} u & v \\ w & x \end{bmatrix} = A \times B = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

Using the standard algorithm, we can form C with eight multiplications and four additions of $(n/2) \times (n/2)$ matrices. Strassen's algorithm exchanges one of these multiplications for 10 such additions. Since one multiplication of two $(n/2) \times (n/2)$ matrices is much more costly than an addition of two such matrices, a large reduction in the number of operations is obtained. We now derive Strassen's algorithm.

Let D be the the 4×4 matrix shown below whose entries are $(n/2) \times (n/2)$ matrices. (Thus, D is a $2n \times 2n$ matrix.)

$$D = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}$$

The entries u, v, w, and x of the product $A \times B$ can also be produced by the following matrix-vector product:

$$\begin{bmatrix} u \\ w \\ v \\ x \end{bmatrix} = D \times \begin{bmatrix} e \\ g \\ f \\ h \end{bmatrix}$$

We now write D as a sum of seven matrices as shown in Fig. 6.2; that is,

$$D = A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7$$

Let $P_1, P_2, \ldots, P_7$ be the products of the $(n/2) \times (n/2)$ matrices

$$\begin{aligned} P_1 &= (a+d) \times (e+h) & P_5 &= (a+b) \times h \\ P_2 &= (c+d) \times e & P_6 &= (-a+c) \times (e+f) \\ P_3 &= a \times (f-h) & P_7 &= (b-d) \times (g+h) \\ P_4 &= d \times (-e+g) \end{aligned}$$

$$A_1 = \begin{bmatrix} a+d & 0 & 0 & a+d \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ a+d & 0 & 0 & a+d \end{bmatrix} \qquad A_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ c+d & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -(c+d) & 0 & 0 & 0 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & a & -a \\ 0 & 0 & a & -a \end{bmatrix} \qquad A_4 = \begin{bmatrix} -d & d & 0 & 0 \\ -d & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} 0 & 0 & 0 & -(a+b) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a+b \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad A_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -a+c & 0 & -a+c & 0 \end{bmatrix}$$

$$A_7 = \begin{bmatrix} 0 & b-d & 0 & b-d \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 6.2 The decomposition of the 4×4 matrix D as the sum of seven 4×4 matrices.

Then the product of the vector $[e, g, f, h]^T$ with D is the following sum of seven column vectors.

$$\begin{bmatrix} u \\ w \\ v \\ x \end{bmatrix} = \begin{bmatrix} P_1 \\ 0 \\ 0 \\ P_1 \end{bmatrix} + \begin{bmatrix} 0 \\ P_2 \\ 0 \\ -P_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ P_3 \\ P_3 \end{bmatrix} + \begin{bmatrix} P_4 \\ P_4 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -P_5 \\ 0 \\ P_5 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ P_6 \end{bmatrix} + \begin{bmatrix} P_7 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

It follows that u, v, w, and x are given by the following equations:

$$\begin{aligned} u &= P_1 + P_4 - P_5 + P_7 & v &= P_3 + P_5 \\ w &= P_2 + P_4 & x &= P_1 - P_2 + P_3 + P_6 \end{aligned}$$

Associativity and commutativity under addition and distributivity of multiplication over addition are used to obtain this result. In particular, commutativity of the ring multiplication operator is not assumed. This is important because it allows this algorithm to be used when the entries in the original 2×2 matrices are themselves matrices, since matrix multiplication is not commutative.

Thus, an algorithm exists to form the product of two $n \times n$ matrices with seven multiplications of $(n/2) \times (n/2)$ matrices and 18 additions or subtractions of such matrices. Let $n = 2^k$ and $M(k)$ be the number of operations over the ring $\mathcal{R}$ used by this algorithm to multiply $n \times n$ matrices. Then, $M(k)$ satisfies

$$M(k) = 7M(k-1) + 18\left(2^{k-1}\right)^2 = 7M(k-1) + (18)4^{k-1}$$

If the standard algorithm is used to multiply 2×2 matrices, $M(1) = 12$ and $M(k)$ satisfies the following recurrence:

$$M(k) = (36/7)7^k - (18/3)4^k$$

The depth (number of operations on the longest path), $D(k)$, of this straight-line algorithm for the product of two $n \times n$ matrices when $n = 2^k$ satisfies the following bound:

$$D(k) = D(k-1) + 3$$

because one level of addition or subtraction is used before products are formed and one or two levels are used after they are formed. Since $D(1) = 2$ if the standard algorithm is used to multiply 2×2 matrices, $D(k) = 3k - 1 = 3\log n - 1$.

These size and depth bounds can be improved to those in the following theorem by using the standard matrix multiplication algorithm on small matrices. (See Problem 6.8.)

THEOREM 6.3.1 *The matrix multiplication function for $n \times n$ matrices over a commutative ring $\mathcal{R}$, $f^{(n)}_{A\times B}$, has circuit size and depth satisfying the following bounds over the basis Ω containing addition, multiplication, and additive inverse over $\mathcal{R}$:*

$$\begin{aligned} C_\Omega\left(f^{(n)}_{A\times B}\right) &\le 4.77n^{\log_2 7} \\ D_\Omega\left(f^{(n)}_{A\times B}\right) &= O(\log n) \end{aligned}$$

We emphasize again that subtraction plays a central role in Strassen's algorithm. Without it we show in Section 10.4 that the standard algorithm is nearly best possible.

Strassen's algorithm is practical for sufficiently large matrices, say with $n \geq 64$. It can also be used to multiply Boolean matrices even though the addition operator (OR) and the multiplication operator (AND) over the set $\mathcal{B}$ do not constitute a ring. (See Problem 6.9.)

6.4 Transitive Closure

The edges of a directed graph $G = (V, E)$, $n = |V|$, specify paths of length 1 between pairs of vertices. (See Fig. 6.3.) This information is captured by the Boolean $n \times n$ **adjacency matrix** $A = [a_{i,j}]$, $1 \leq i, j \leq n$, where $a_{i,j}$ is 1 if there is an edge from vertex i to vertex j in E and 0 otherwise. (The adjacency matrix for the graph in Fig. 6.3 is given after Lemma 6.4.1.) Our goal is to compute a matrix A^* whose i, j entry $a^*_{i,j}$ has value 1 if there is a path of length 0 or more between vertices i and j and value 0 otherwise. A^* is called the **transitive closure** of the matrix A. The transitive closure function $f^{(n)}_{A^*} : \mathcal{B}^{n^2} \mapsto \mathcal{B}^{n^2}$ maps an arbitrary $n \times n$ Boolean matrix A onto its $n \times n$ transitive closure matrix; that is, $f^{(n)}_{A^*}(A) = A^*$. In this section we add and multiply Boolean matrices over the set $\mathcal{B}$ using OR as the element addition operation and AND as the element multiplication operation. (Note that $(\mathcal{B}, \vee, \wedge, 0, 1)$ is not a ring; it satisfies all the rules for a ring except for the condition that each element of $\mathcal{B}$ have an (additive) inverse under $\vee$.)

To compute A^* we use the following facts: a) the entry in the rth row and sth column of the Boolean matrix product $A^2 = A \times A$ is 1 if there is a path containing two edges from vertex r to vertex s and 0 otherwise (which follows from the definition of Boolean matrix multiplication given in Section 6.3), and b) the entry in the rth row and sth column of $A^k = A^{k-1} \times A$ is 1 if there is a path containing k edges from vertex r to vertex s and 0 otherwise, as the reader is asked to show. (See Problem 6.11.)

LEMMA 6.4.1 *Let A be the Boolean adjacency matrix for a directed graph and let A^k be the kth power of A. Then the following identity holds for $k \geq 1$, where $+$ denotes the addition (OR) of Boolean matrices:*

$$(I + A)^k = I + A + \cdots + A^k \tag{6.2}$$

Proof The proof is by induction. The base case is $k = 1$, for which the identity holds. Assume that it holds for $k \leq K-1$. We show that it holds for $k = K$. Since $(I+A)^{K-1} =$

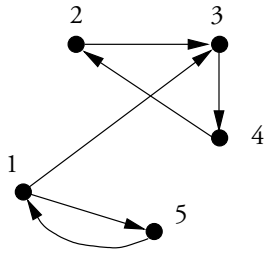

Figure 6.3 A graph that illustrates transitive closure.

$I + A + \cdots + A^{K-1}$, multiply both sides by $I + A$:

$$\begin{aligned}(I+A)^K &= (I+A) \times (I+A)^{K-1} \\ &= (I+A) \times (I + A + \cdots + A^{K-1}) \\ &= I + (A+A) + \cdots + (A^{K-1} + A^{K-1}) + A^K\end{aligned}$$

However, since A^j is a Boolean matrix, $A^j + A^j = A^j$ for all j and the result follows. ■

The adjacency matrix A of the graph in Fig. 6.3 is given below along with its powers up to the fifth power. Note that every non-zero entry appearing in A^5 appears in at least one of the other matrices. The reason for this fact is explained in the proof of Lemma 6.4.2.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad A^2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \quad A^3 = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$A^4 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad A^5 = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

LEMMA 6.4.2 *If there is a path between pairs of vertices in the directed graph $G = (V, E)$, $n = |V|$, there is a path of length at most $n - 1$.*

Proof We suppose that the shortest path between vertices i and j in V has length $k \geq n$. Such a path has $k + 1$ vertices. Because $k + 1 \geq n + 1$, some vertex is repeated more than once. (This is an example of the pigeonhole principle.) Consider the subpath defined by the edges between the first and last instance of this repeated vertex. Since it constitutes a loop, it can be removed to produce a shorter path between vertices i and j. This contradicts the hypothesis that the shortest path has length n or more. Thus, the shortest path has length at most $n - 1$. ■

Because the shortest path has length at most $n-1$, any non-zero entries in A^k, $k \geq n$, are also found in one of the matrices A^j, $j \leq n - 1$. Since the identity matrix I is the adjacency matrix for the graph that has paths of length zero between two vertices, the transitive closure, which includes such paths, is equal to:

$$A^* = I + A + A^2 + A^3 + \cdots + A^{n-1} = (I + A)^{n-1}$$

It also follows that $A^* = (I + A)^k$ for all $k \geq n - 1$, which leads to the following result.

THEOREM 6.4.1 *Over the basis $\Omega = \{\text{AND, OR}\}$ the transitive closure function, $f^{(n)}_{A^*}$, has circuit size and depth satisfying the following bounds (that is, a circuit of this size and depth can be*

constructed with AND *and* OR *gates for it):*

$$C_\Omega\left(f_{A^*}^{(n)}\right) \leq M_{\text{matrix}}(cn, K)\lceil \log_2 n \rceil$$
$$D_\Omega\left(f_{A^*}^{(n)}\right) \leq K(\log n)\lceil \log_2 n \rceil$$

Proof Let $k = 2^p$ be the smallest power of 2 such that $k \geq n-1$. Then, $p = \lceil \log_2(n-1) \rceil$. Since $A^* = (I + A)^k$, it can be computed with a circuit that squares the matrix $I + A$ p times. Each squaring can be done with a circuit for the standard matrix multiplication algorithm described in (6.1) using $M_{\text{matrix}}(cn, K) = O(n^3)$ operations and depth $\lceil \log_2 2n \rceil$. The desired result follows. ■

The above statement says that the transitive closure function on $n \times n$ matrices has circuit size and depth at most a factor $O(\log n)$ times that of matrix multiplication. We now show that Boolean matrix multiplication is a subfunction of the transitive closure function, which implies that the former has a circuit size and depth no larger than the latter. We subsequently show that the size bound can be improved to a constant multiple of the size bound for matrix multiplication. Thus the transitive closure and Boolean matrix multiplication functions have comparable size.

THEOREM 6.4.2 *The $n \times n$ matrix multiplication function $f_{A\times B}^{(n)} : \mathcal{R}^{2n^2} \mapsto \mathcal{R}^{n^2}$ for Boolean matrices is a subfunction of the transitive closure function $f_{A^*}^{(3n)} : \mathcal{R}^{18n^2} \mapsto \mathcal{R}^{9n^2}$.*

Proof Observe that the following relationship holds for $n \times n$ matrices A and B, since the third and higher powers of the $3n \times 3n$ matrix on the left are 0.

$$\begin{bmatrix} 0 & A & 0 \\ 0 & 0 & B \\ 0 & 0 & 0 \end{bmatrix}^* = \begin{bmatrix} I & A & AB \\ 0 & I & B \\ 0 & A & I \end{bmatrix}$$

It follows that the product AB of $n \times n$ matrices is a subfunction of the transitive closure function on a $3n \times 3n$ matrix. ■

COROLLARY 6.4.1 *It follows that*

$$C_\Omega\left(f_{A\times B}^{(n)}\right) \leq C_\Omega\left(f_{A^*}^{(3n)}\right)$$
$$D_\Omega\left(f_{A\times B}^{(n)}\right) \leq D_\Omega\left(f_{A^*}^{(3n)}\right)$$

over the basis $\Omega = \{\text{AND, OR}\}$.

Not only can a Boolean matrix multiplication algorithm be devised from one for transitive closure, but the reverse is also true, as we show. Let n be a power of 2 and divide an $n \times n$ matrix A into four $(n/2) \times (n/2)$ matrices:

$$A = \begin{bmatrix} U & V \\ W & X \end{bmatrix} \tag{6.3}$$

Compute X^* recursively and use it to form $Y = U + VX^*W$ by performing two multiplications of $(n/2) \times (n/2)$ matrices and one addition of such matrices. Recursively form Y^* and then assemble the matrix B shown below with four further multiplications and one addition of $(n/2) \times (n/2)$ matrices.

$$B = \begin{bmatrix} Y^* & Y^*VX^* \\ X^*WY^* & X^* + X^*WY^*VX^* \end{bmatrix} \tag{6.4}$$

We now show that $B = A^*$.

THEOREM 6.4.3 *Under Assumptions 6.3.1 and 6.3.2, a circuit of size $O(M_{\text{matrix}}(n, K))$ and depth $O(n)$ exists to form the transitive closure of $n \times n$ matrices.*

Proof We assume that n is a power of 2 and use the representation for the matrix A given in (6.3). If n is not a power of 2, we augment the matrix A by embedding it in a larger matrix in which all the new entries, are 0 except for the new diagonal entries, which are 1. Given that $4M(n) \leq M(2n)$, the bound applies.

We begin by showing that $B = A^*$. Let $F \subset V$ and $S \subset V$ be the first and second sets of $n/2$ vertices, respectively, corresponding to the first and second halves of the rows and columns of the matrix A. Then, $F \cup S = V$ and $F \cap S = \emptyset$. Observe that X^* is the adjacency matrix for those paths originating on and terminating with vertices in F and visiting no other vertices. Similarly, $Y = U + VX^*W$ is the adjacency matrix for those paths consisting of an edge from a vertex in F to a vertex in F or paths of length more than 1 consisting of an edge from vertices in F to vertices in S, a path of length 0 or more within vertices in S, and an edge from vertices in S to vertices in F. It follows that Y^* is the adjacency matrix for all paths between vertices in F that may visit any vertices in V. A similar line of reasoning demonstrates that the other entries of A^* are correct.

The size of a circuit realizing this algorithm, $T(n)$, satisfies

$$T(n) = 2T(n/2) + 6M_{\text{matrix}}(n/2, K) + 2(n/2)^2$$

because the above algorithm (see Fig. 6.4) uses two circuits for transitive closure on $(n/2) \times (n/2)$ matrices, six circuits for multiplying, and two for adding two such matrices.

Because we assume that $n^2 \leq M_{\text{matrix}}(n, K)$, it follows that $T(n) \leq 2T(n/2) + 8M_{\text{matrix}}(n/2, K)$. Let $T(m) \leq cM_{\text{matrix}}(cm, K)$ for $m \leq n/2$ be the inductive hypothesis. Then we have the inequalities

$$T(n) \leq (2c + 8)M_{\text{matrix}}(n/2, K) \leq (c/2 + 2)M_{\text{matrix}}(n, K)$$

which follow from $M_{\text{matrix}}(n/2, K) \leq M_{\text{matrix}}(n, K)/4$ (see Assumption 6.3.2). Because $(c/2 + 2) \leq c$ for $c \geq 4$, for $c = 4$ we have the desired bound on circuit size.

The depth $D(n)$ of the above circuit satisfies $D(n) = 2D(n/2) + 6K \log_2 n$, from which we conclude that $D(n) = O(n)$. ■

A **semiring** $(S, +, \cdot, 0, 1)$ is a set S, two operations $+$ and $\cdot$ and elements $0, 1 \in S$ with the following properties:

a) S is closed under $+$ and $\cdot$;

b) $+$ and $\cdot$ are associative;

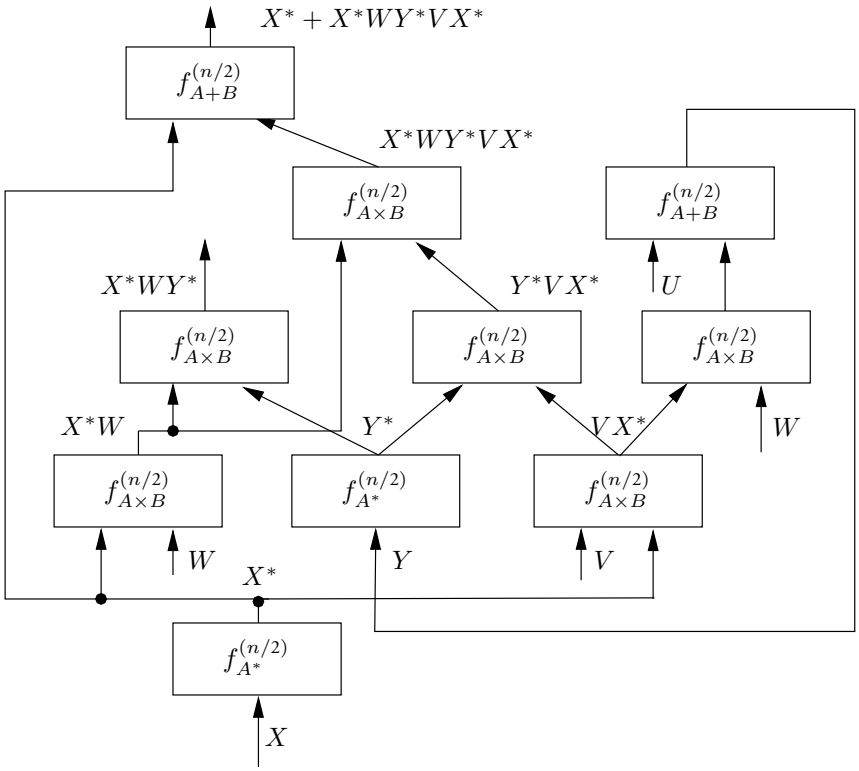

Figure 6.4 A circuit for the transitive closure of a Boolean matrix based on the construction of equation (6.4).

c) for all $a \in S$, $a + 0 = 0 + a = a$;

d) for all $a \in S$, $a \cdot 1 = 1 \cdot a = a$;

e) $+$ is commutative and **idempotent**; i.e. $a + a = a$;

f) $\cdot$ distributes over $+$; i.e. for all $a, b, c \in S$, $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(b + c) \cdot a = b \cdot a + c \cdot a$.

The above definitions and results generalize to matrices over semirings. To show this, it suffices to observe that the properties used to derive these results are just these properties. (See Problem 6.12.)

6.5 Matrix Inversion

The inverse of a non-singular $n \times n$ matrix M defined over a field $\mathcal{R}$ is another matrix M^{-1} whose product with M is the $n \times n$ identity matrix I; that is,

$$MM^{-1} = M^{-1}M = I$$

Given a linear system of n equations in the column vector $\boldsymbol{x}$ of n unknowns defined by the non-singular $n \times n$ coefficient matrix M and the vector $\boldsymbol{b}$, namely,

$$M\boldsymbol{x} = \boldsymbol{b} \tag{6.5}$$

the solution $\boldsymbol{x}$ can be obtained through a matrix-vector multiplication with M^{-1}:

$$\boldsymbol{x} = M^{-1}\boldsymbol{b}$$

In this section we present two algorithms for matrix inversion. Such algorithms compute the (partial) matrix inverse function $f^{(n)}_{A^{-1}} : \mathcal{R}^{n^2} \mapsto \mathcal{R}^{n^2}$ that maps non-singular $n \times n$ matrices over a field $\mathcal{R}$ onto their inverses. The first result, Theorem 6.5.4, demonstrates that $C_\Omega\left(f^{(n)}_{A^{-1}}\right) = \Theta(M_{\text{matrix}}(n, K))$ with a circuit whose depth is more than linear in n. The second, Theorem 6.5.6, demonstrates that $D_\Omega\left(f^{(n)}_{A^{-1}}\right) = O(\log^2 n)$ with a circuit whose size is $O(nM_{\text{matrix}}(n, K))$.

Before describing the two matrix inversion algorithms, we present a result demonstrating that matrix multiplication of $n \times n$ matrices is no harder than inverting a $3n \times 3n$ matrix; the function defining the former task is a subfunction of the function defining the latter task.

LEMMA 6.5.1 *The matrix inverse function $f^{(3n)}_{A^{-1}}$ contains as a subfunction the function $f^{(n)}_{A\times B} : \mathcal{R}^{2n^2} \mapsto \mathcal{R}^{n^2}$ that maps two matrices over $\mathcal{R}$ to their product.*

Proof The proof follows by writing a $3n \times 3n$ matrix as a 3×3 matrix of $n \times n$ matrices and then specializing the entries to be the identity matrix I, the zero matrix 0, or matrices A and B:

$$\begin{bmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{bmatrix}^{-1} = \begin{bmatrix} I & -A & AB \\ 0 & I & -B \\ 0 & 0 & I \end{bmatrix}$$

This identity is established by showing that the product of these two matrices is the identity matrix. ■

6.5.1 Symmetric Positive Definite Matrices

Our first algorithm to invert a non-singular $n \times n$ matrix M has a circuit size linear in $M_{\text{matrix}}(n, K)$, which, in light of Lemma 6.5.1, is optimal to within a constant multiplicative factor. This algorithm makes use of symmetric positive definite matrices, the Schur complement, and LDL$^{\text{T}}$ factorization, terms defined below. This algorithm has depth $O(n \log^2 n)$.

The second algorithm, Csanky's algorithm, has circuit depth $O(\log^2 n)$, which is smaller, but circuit size $O(nM_{\text{matrix}}(n, K))$, which is larger. Symmetric positive definite matrices are defined below.

DEFINITION 6.5.1 *A matrix M is* **positive definite** *if for all non-zero vectors $\boldsymbol{x}$ the following condition holds:*

$$\boldsymbol{x}^T M\boldsymbol{x} = \sum_{1\le i,j\le n} x_i m_{i,j} x_j > 0 \tag{6.6}$$

A matrix is **symmetric positive definite** *(SPD) if it is both symmetric and positive definite.*

We now show that an algorithm to invert SPD matrices can be used to invert arbitrary non-singular matrices by adding a circuit to multiply matrices.

LEMMA 6.5.2 *If M is a non-singular $n \times n$ matrix, then the matrix $P = M^T M$ is symmetric positive definite. M can be inverted by inverting P and then multiplying P^{-1} by M^T. Let $f^{(n)}_{\text{SPD\_inverse}} : \mathcal{R}^{n^2} \mapsto \mathcal{R}^{n^2}$ be the inverse function for $n \times n$ SPD matrices over the field $\mathcal{R}$. Then the size and depth of $f^{(n)}_{A^{-1}}$ over $\mathcal{R}$ satisfy the following bounds:*

$$C\left(f^{(n)}_{A^{-1}}\right) \leq C\left(f^{(n)}_{\text{SPD\_inverse}}\right) + M_{\text{matrix}}(n, K)$$
$$D\left(f^{(n)}_{A^{-1}}\right) \leq D\left(f^{(n)}_{\text{SPD\_inverse}}\right) + O(\log n)$$

Proof To show that P is symmetric we note that $\left(M^T\ M\right)^T = M^T M$. To show that it is positive definite, we observe that

$$\begin{aligned} \boldsymbol{x}^T P \boldsymbol{x} &= \boldsymbol{x}^T M^T M \boldsymbol{x} \\ &= (M\boldsymbol{x})^T M\boldsymbol{x} \\ &= \sum_{i=1}^{n} \left(\sum_{j=1}^{n} m_{i,j} x_j \right)^2 \end{aligned}$$

which is positive unless the product $M\boldsymbol{x}$ is identically zero for the non-zero vector $\boldsymbol{x}$. But this cannot be true if M is non-singular. Thus, P is symmetric and positive definite.

To invert M, invert P to produce $M^{-1}\left(M^T\right)^{-1}$. If we multiply this product on the right by M^T, the result is the inverse M^{-1}. ■

6.5.2 Schur Factorization

We now describe **Schur factorization**. Represent an $n \times n$ matrix M as the 2×2 matrix

$$M = \begin{bmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{bmatrix} \tag{6.7}$$

where $M_{1,1}$, $M_{1,2}$, $M_{2,1}$, and $M_{2,2}$ are $k \times k$, $k \times n-k$, $n-k \times k$, and $n-k \times n-k$ matrices, $1 \leq k \leq n-1$. Let $M_{1,1}$ be invertible. Then by straightforward algebraic manipulation M can be factored as

$$M = \begin{bmatrix} I & 0 \\ M_{2,1}M_{1,1}^{-1} & I \end{bmatrix} \begin{bmatrix} M_{1,1} & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & M_{1,1}^{-1}M_{1,2} \\ 0 & I \end{bmatrix} \tag{6.8}$$

Here I and O denote identity and zero matrices (all entries are zero) of a size that conforms to the size of other submatrices of those matrices in which they are found. This is the Schur factorization. Also,

$$S = M_{2,2} - M_{2,1}M_{1,1}^{-1}M_{1,2}$$

is the **Schur complement** of M. To show that M has this factorization, it suffices to carry out the product of the above three matrices.

The first and last matrix in this product are invertible. If S is also invertible, the middle matrix is invertible, as is the matrix M itself. The inverse of M, M^{-1}, is given by the product

$$M^{-1} = \begin{bmatrix} I & -M_{1,1}^{-1}M_{1,2} \\ 0 & I \end{bmatrix} \begin{bmatrix} M_{1,1}^{-1} & 0 \\ 0 & S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -M_{2,1}M_{1,1}^{-1} & I \end{bmatrix} \tag{6.9}$$

This follows from three observations: a) the inverse of a product is the product of the inverses in reverse order (see Lemma 6.2.1), b) the inverse of a 2×2 upper (lower) triangular matrix is the matrix with the off-diagonal term negated, and c) the inverse of a 2×2 diagonal matrix is a diagonal matrix in which the ith diagonal element is the multiplicative inverse of the ith diagonal element of the original matrix. (See Problem 6.13 for the latter two results.)

The following fact is useful in inverting SPD matrices.

LEMMA 6.5.3 *If M is an $n \times n$ SPD matrix, its Schur complement is also SPD.*

Proof Represent M as shown in (6.7). In (6.6) let $\boldsymbol{x} = \boldsymbol{u} \cdot \boldsymbol{v}$; that is, let $\boldsymbol{x}$ be the concatenation of the two column vectors. Then

$$\begin{aligned} \boldsymbol{x}^T M \boldsymbol{x} &= \left[\boldsymbol{u}^T, \boldsymbol{v}^T\right] \begin{bmatrix} M_{1,1}\boldsymbol{u} + M_{1,2}\boldsymbol{v} \\ M_{2,1}\boldsymbol{u} + M_{2,2}\boldsymbol{v} \end{bmatrix} \\ &= \boldsymbol{u}^T M_{1,1}\boldsymbol{u} + \boldsymbol{u}^T M_{1,2}\boldsymbol{v} + \boldsymbol{v}^T M_{2,1}\boldsymbol{u} + \boldsymbol{v}^T M_{2,2}\boldsymbol{v} \end{aligned}$$

If we say that

$$\boldsymbol{u} = -M_{1,1}^{-1}M_{1,2}\,\boldsymbol{v}$$

and use the fact that $M_{1,2}^T = M_{2,1}$ and $\left(M_{1,1}^{-1}\right)^T = \left(M_{1,1}^T\right)^{-1} = M_{1,1}^{-1}$, it is straightforward to show that S is symmetric and

$$\boldsymbol{x}^T M \boldsymbol{x} = \boldsymbol{v}^T S \boldsymbol{v}$$

where S is the Schur complement of M. Thus, if M is SPD, so is its Schur complement. ■

6.5.3 Inversion of Triangular Matrices

Let T be $n \times n$ lower triangular and non-singular. Without loss of generality, assume that $n = 2^r$. (T can be extended to a $2^r \times 2^r$ matrix by placing it on the diagonal of a $2^r \times 2^r$ matrix along with a $2^r - n \times 2^r - n$ identity matrix.) Represent T as a 2×2 matrix of $n/2 \times n/2$ matrices:

$$T = \begin{bmatrix} T_{1,1} & 0 \\ T_{2,1} & T_{2,2} \end{bmatrix}$$

The inverse of T, which is lower triangular, is given below, as can be verified directly:

$$T^{-1} = \begin{bmatrix} T_{1,1}^{-1} & 0 \\ -T_{2,2}^{-1}T_{2,1}T_{1,1}^{-1} & T_{2,2}^{-1} \end{bmatrix}$$

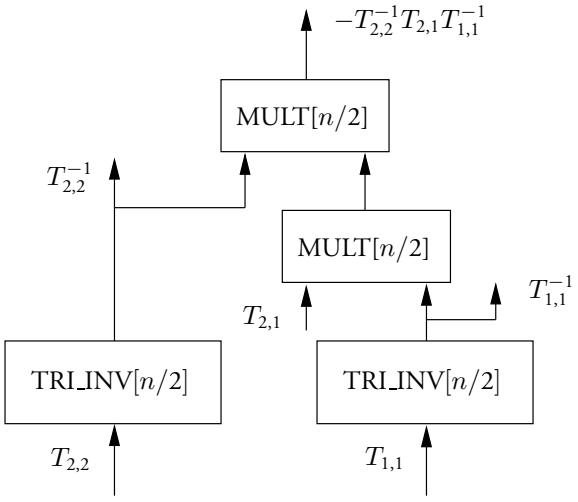

Figure 6.5 A recursive circuit TRI_INV[n] for the inversion of a triangular matrix.

This representation for the inverse of T defines the recursive algorithm TRI_INV[n] in Fig. 6.5. When $n = 1$ this algorithm requires one operation; on an $n \times n$ matrix it requires two calls to TRI_INV[$n/2$] and two matrix multiplications. Let $f_{\text{tri\_inv}}^{(n)} : \mathcal{R}^{(n^2+n)/2} \mapsto \mathcal{R}^{(n^2+n)/2}$ be the function corresponding to the inversion of an $n \times n$ lower triangular matrix. The algorithm TRI_INV[n] provides the following bounds on the size and depth of the smallest circuit to compute $f_{\text{tri\_inv}}^{(n)}$.

THEOREM 6.5.1 *Let n be a power of 2. Then the matrix inversion function $f_{\text{tri\_inv}}^{(n)}$ for $n \times n$ lower triangular matrices satisfies the following bounds:*

$$C\left(f_{\text{tri\_inv}}^{(n)}\right) \leq M_{\text{matrix}}(n, K)$$
$$D\left(f_{\text{tri\_inv}}^{(n)}\right) = O(\log^2 n)$$

Proof From Fig. 6.5 it is clear that the following circuit size and depth bounds hold if the matrix multiplication algorithm has circuit size $M_{\text{matrix}}(n, K)$ and depth $K \log_2 n$:

$$C\left(f_{\text{tri\_inv}}^{(n)}\right) \leq 2C\left(f_{\text{tri\_inv}}^{(n/2)}\right) + 2M_{\text{matrix}}(n/2, K)$$
$$D\left(f_{\text{tri\_inv}}^{(n)}\right) \leq D\left(f_{\text{tri\_inv}}^{(n/2)}\right) + 2K \log n$$

The solution to the first inequality follows by induction from the fact that $M_{\text{matrix}}(1, K) = 1$ and the assumption that $4M_{\text{matrix}}(n/2, K) \leq M_{\text{matrix}}(n, K)$. The second inequality follows from the observation that $d > 0$ can be chosen so that $d \log^2(n/2) + c \log n \leq d \log^2 n$ for any $c > 0$ for n sufficiently large. ■

6.5.4 LDL$^{\mathrm{T}}$ Factorization of SPD Matrices

Now that we know that the Schur complement S of M is SPD when M is SPD, we can show that every SPD matrix M has a factorization as the product LDL^T of a unit lower triangular matrix L (each of its diagonal entries is the multiplicative unit of the field $\mathcal{R}$), a diagonal matrix D, and the transpose of L.

THEOREM 6.5.2 *Every $n \times n$ SPD matrix M has a factorization as the product $M = LDL^T$, where L is a unit lower triangular matrix and D is a diagonal matrix.*

Proof The proof is by induction on n. For $n = 1$ the result is obvious because we can write $[m_{1,1}] = [1][m_{1,1}][1]$. Assume that it holds for $n \leq N - 1$. We show that it holds for $n = N$.

Form the Schur factorization of the $N \times N$ matrix M. Since the $k \times k$ submatrix $M_{1,1}$ of M as well as the $n - k \times n - k$ submatrix S of M are SPD, by the inductive hypothesis they can be factored in the same fashion. Let

$$M_{1,1} = L_1 D_1 L_1^T, S = L_2 D_2 L_2^T$$

Then the middle matrix on the right-hand side of equation (6.8) can be represented as

$$\begin{bmatrix} M_{1,1} & 0 \\ 0 & S \end{bmatrix} = \begin{bmatrix} L_1 & 0 \\ 0 & L_2 \end{bmatrix} \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \begin{bmatrix} L_1^T & 0 \\ 0 & L_2^T \end{bmatrix}$$

Substituting the above product for the middle matrix in (6.8) and multiplying the two left and two right matrices gives the following representation for M:

$$M = \begin{bmatrix} L_1 & 0 \\ M_{2,1} M_{1,1}^{-1} L_1 & L_2 \end{bmatrix} \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \begin{bmatrix} L_1^T & L_1^T M_{1,1}^{-1} M_{1,2} \\ 0 & L_2^T \end{bmatrix} \tag{6.10}$$

Since M is symmetric, $M_{1,1}$ is symmetric, $M_{1,2} = M_{2,1}^T$, and

$$L_1^T M_{1,1}^{-1} M_{1,2} = L_1^T (M_{1,1}^{-1})^T M_{2,1}^T = (M_{2,1} M_{1,1}^{-1} L_1)^T$$

Thus, it suffices to compute L_1, D_1, L_2, D_2, and $M_{2,1} M_{1,1}^{-1} L_1$. ■

When $n = 2^r$ and $k = n/2$, the proof of Theorem 6.5.2 describes a recursive procedure, LDL$^{\mathrm{T}}[n]$, defined on $n \times n$ SPD matrices that produces their LDL$^{\mathrm{T}}$ factorization. Figure 6.6 captures the steps involved. They are also described below.

- The LDL$^{\mathrm{T}}$ factorization of the $n/2 \times n/2$ matrix $M_{1,1}$ is computed using the procedure LDL$^{\mathrm{T}}[n/2]$ to produce the $n/2 \times n/2$ triangular and diagonal matrices L_1 and D_1, respectively.

- The product $M_{2,1} M_{1,1}^{-1} L_1 = M_{2,1} \left(L_1^{-1}\right)^T D_1^{-1}$ which may be computed by inverting the lower triangular matrix L_1 with the operation TRI_INV$[n/2]$, computing the product $M_{2,1} \left(L_1^{-1}\right)^T$ using MULT$[n/2]$, and multiplying the result with D_1^{-1} using a procedure SCALE$[n/2]$ that inverts D_1 and multiplies it by a square matrix.

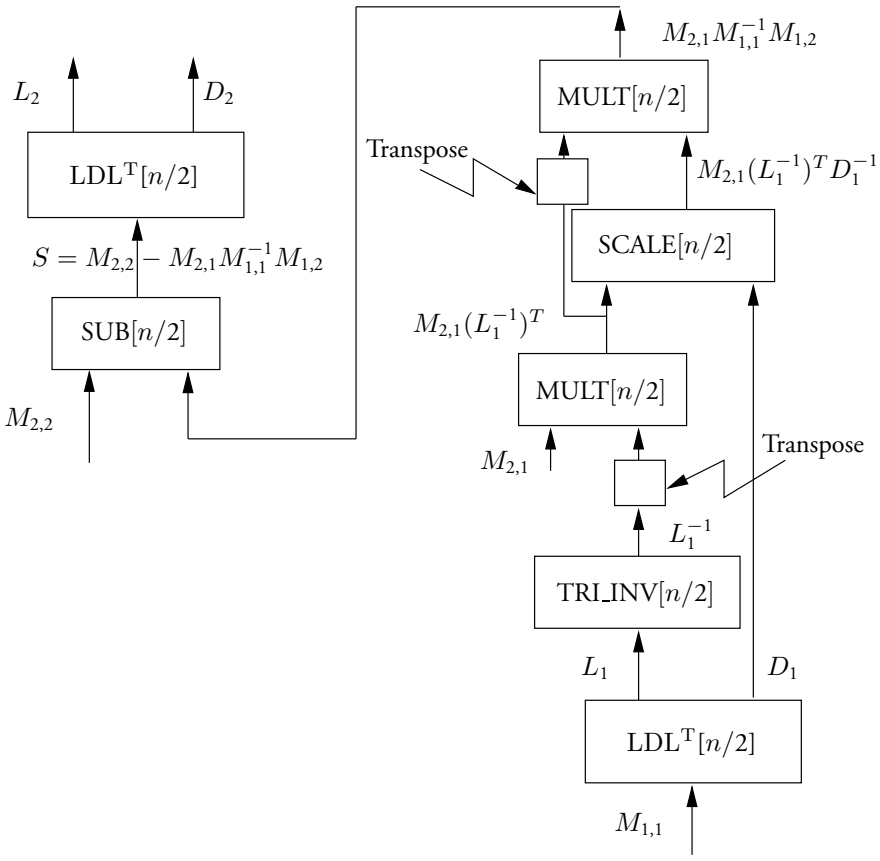

Figure 6.6 An algebraic circuit to produce the LDL$^{\mathrm{T}}$ factorization of an SPD matrix.

- $S = M_{2,2} - M_{2,1}M_{1,1}^{-1}M_{1,2}$ can be formed by multiplying $M_{2,1}\left(L_1^{-1}\right)^T D_1^{-1}$ by the transpose of $M_{2,1}\left(L_1^{-1}\right)^T$ using MULT[$n/2$] and subtracting the result from $M_{2,2}$ by the subtraction operator SUB[$n/2$].
- The LDL$^{\mathrm{T}}$ factorization of the $n/2 \times n/2$ matrix S is computed using the procedure LDL$^{\mathrm{T}}$[$n/2$] to produce the $n/2 \times n/2$ triangular and diagonal matrices L_2 and D_2, respectively.

Let's now determine the size and depth of circuits to implement the algorithm for LDL$^{\mathrm{T}}$[n]. Let $f_{\mathrm{LDL^T}}^{(n)} : \mathcal{R}^{n^2} \mapsto \mathcal{R}^{(n^2+n)/2}$ be the function defined by the LDL$^{\mathrm{T}}$ factorization of an $n \times n$ SPD matrix, $f_{\mathrm{tri\_inv}}^{(n)} : \mathcal{R}^{(n^2+n)/2} \mapsto \mathcal{R}^{(n^2+n)/2}$ be the inversion of an $n \times n$ lower triangular matrix, $f_{\mathrm{scale}}^{(n)} : \mathcal{R}^{n^2+n} \mapsto \mathcal{R}^{n^2}$ be the computation of $N(D^{-1})$ for an $n \times n$ matrix N and a diagonal matrix D, $f_{\mathrm{mult}}^{(n)} : \mathcal{R}^{2n^2} \mapsto \mathcal{R}^{n^2}$ be the multiplication of two $n \times n$ matrices, and $f_{\mathrm{sub}}^{(n)} : \mathcal{R}^{2n^2} \mapsto \mathcal{R}^{n^2}$ the subtraction of two $n \times n$ matrices. Since a transposition can be done

without any operators, the size and depth of the circuit for $\mathrm{LDL}^{\mathrm{T}}[n]$ constructed above satisfy the following inequalities:

$$C\left(f^{(n)}_{\mathrm{LDL}^{\mathrm{T}}}\right) \le C\left(f^{(n/2)}_{\mathrm{tri\_inv}}\right) + C\left(f^{(n/2)}_{\mathrm{scale}}\right) + 2C\left(f^{(n/2)}_{\mathrm{mult}}\right) + C\left(f^{(n/2)}_{\mathrm{sub}}\right) + 2C\left(f^{(n/2)}_{\mathrm{LDL}^{\mathrm{T}}}\right)$$
$$D\left(f^{(n)}_{\mathrm{LDL}^{\mathrm{T}}}\right) \le D\left(f^{(n/2)}_{\mathrm{tri\_inv}}\right) + D\left(f^{(n/2)}_{\mathrm{scale}}\right) + 2D\left(f^{(n/2)}_{\mathrm{mult}}\right) + D\left(f^{(n/2)}_{\mathrm{sub}}\right) + 2D\left(f^{(n/2)}_{\mathrm{LDL}^{\mathrm{T}}}\right)$$

The size and depth of a circuit for $f^{(n)}_{\mathrm{tri\_inv}}$ are $M_{\mathrm{matrix}}(n,K)$ and $O(\log^2 n)$, as shown in Theorem 6.5.1. The circuits for $f^{(n)}_{\mathrm{scale}}$ and $f^{(n)}_{\mathrm{sub}}$ have size n^2 and depth 1; the former multiplies the elements of the jth column of N by the multiplicative inverse of jth diagonal element of D_1 for $1 \le j \le n$, while the latter subtracts corresponding elements from the two input matrices.

Let $C_{\mathrm{SPD}}(n) = C\left(f^{(n)}_{\mathrm{LDL}^{\mathrm{T}}}\right)$ and $D_{\mathrm{SPD}}(n) = D\left(f^{(n)}_{\mathrm{LDL}^{\mathrm{T}}}\right)$. Since $M_{\mathrm{matrix}}(n/2,K) \le (1/4)M_{\mathrm{matrix}}(n,K)$ is assumed (see Assumption 6.3.1), and $2m^2 \le M_{\mathrm{matrix}}(m,K)$ (see Assumption 6.3.2), the above inequalities become

$$\begin{aligned} C_{\mathrm{SPD}}(n) &\le M_{\mathrm{matrix}}(n/2,K) + (n/2)^2 + 2M_{\mathrm{matrix}}(n/2,K) + (n/2)^2 + 2C_{\mathrm{SPD}}(n/2) \\ &\le 2C_{\mathrm{SPD}}(n/2) + M_{\mathrm{matrix}}(n,K) \qquad (6.11) \\ D_{\mathrm{SPD}}(n) &\le O(\log^2(n/2)) + 1 + 2O(\log(n/2)) + 1 + 2D_{\mathrm{SPD}}(n/2) \\ &\le 2D_{\mathrm{SPD}}(n/2) + K\log_2^2 n \ \text{ for some } K > 0 \qquad (6.12) \end{aligned}$$

As a consequence, we have the following results.

THEOREM 6.5.3 *Let n be a power of two. Then there exists a circuit to compute the* $\mathrm{LDL}^{\mathrm{T}}$ *factorization of an $n \times n$ matrix whose size and depth satisfy*

$$C\left(f^{(n)}_{\mathrm{LDL}^{\mathrm{T}}}\right) \le 2M_{\mathrm{matrix}}(n,K)$$
$$D\left(f^{(n)}_{\mathrm{LDL}^{\mathrm{T}}}\right) \le O\left(n\log^2 n\right)$$

Proof From (6.11) we have that

$$C_{\mathrm{SPD}}(n) \le \sum_{j=0}^{\log n} 2^j M_{\mathrm{matrix}}(n/2^j,K)$$

By Assumption 6.3.2, $M_{\mathrm{matrix}}(n/2,K) \le (1/4)M_{\mathrm{matrix}}(n,K)$. It follows by induction that $M_{\mathrm{matrix}}(n/2^j,K) \le (1/4)^j M_{\mathrm{matrix}}(n,K)$, which bounds the above sum by a geometric series whose sum is at most $2M_{\mathrm{matrix}}(n,K)$. The bound on $D\left(f^{(n)}_{\mathrm{LDL}^{\mathrm{T}}}\right)$ follows from the observation that $(2c)(n/2)\log^2(n/2) + c\log^2 n \le cn\log^2 n$ for $n \ge 2$ and $c > 0$. ■

This result combined with earlier observations provides a matrix inversion algorithm for arbitrary non-singular matrices.

THEOREM 6.5.4 *The matrix inverse function $f_{A^{-1}}^{(n)}$ for arbitrary non-singular $n \times n$ matrices over an arbitrary field $\mathcal{R}$ can be computed by an algebraic circuit whose size and depth satisfy the following bounds:*

$$C\left(f_{A^{-1}}^{(n)}\right) = \Theta(M_{\text{matrix}}(n, K))$$
$$D\left(f_{A^{-1}}^{(n)}\right) = O(n \log^2 n)$$

Proof To invert a non-singular $n \times n$ matrix M that is not SPD, form the product $P = M^T M$ (which is SPD) with one instance of MULT[n] and then invert it. Then multiply P^{-1} by M^T on the right with a second instance of MULT[n]. To invert P, compute its LDL$^{\text{T}}$ factorization and invert it by forming $\left(L^T\right)^{-1} D^{-1} L^{-1}$. Inverting LDL^T requires one application of TRI_INV[n], one application of SCALE[n], and one application of MULT[n], in addition to the steps used to form the factorization. Thus, three applications of MULT[n] are used in addition to the factorization steps. The following bounds hold:

$$C\left(f_{A^{-1}}^{(n)}\right) \leq 4M_{\text{matrix}}(n, K) + n^2 \leq 4.5M_{\text{matrix}}(n)$$
$$D\left(f_{A^{-1}}^{(n)}\right) = O\left(n \log^2 n\right) + O(\log n) = O\left(n \log^2 n\right)$$

The lower bound on $C\left(f_{A^{-1}}^{(n)}\right)$ follows from Lemma 6.5.1. ■

6.5.5 Fast Matrix Inversion*

In this section we present a depth-$O(\log^2 n)$ circuit for the inversion of $n \times n$ matrices known as **Csanky's algorithm**, which is based on the method of Leverrier. Since this algorithm uses a number of well-known matrix functions and properties that space precludes explaining in detail, advanced knowledge of matrices and polynomials is required for this section.

The determinant of an $n \times n$ matrix A, $\det(A)$, is defined below in terms of the set of all permutations π of the integers $\{1, 2, \ldots, n\}$. Here the **sign** of π, denoted $\sigma(\pi)$, is the number of swaps of pairs of integers needed to realize π from the identity permutation.

$$\det(A) = \sum_{\pi} (-1)^{\sigma(\pi)} \prod_{i=1}^{n} a_{i,\pi(i)}$$

Here $\prod_{i=1}^{n} a_{i,\pi(i)}$ is the product $a_{1,\pi(1)} \cdots a_{n,\pi(n)}$. The **characteristic polynomial** of a matrix A, namely, $\phi_A(x)$ in the variable x, is the determinant of $xI - A$, where I is the $n \times n$ identity matrix:

$$\begin{aligned}\phi_A(x) &= \det(xI - A) \\ &= x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_0\end{aligned}$$

If x is set to zero, this equation implies that $c_0 = \det(-A)$. Also, it can be shown that $\phi_A(A) = 0$, a fact known as the **Cayley-Hamilton theorem**: A matrix satisfies its own characteristic polynomial. This implies that

$$A\left(A^{n-1} + c_{n-1}A^{n-2} + c_{n-2}A^{n-3} + \cdots + c_1\right) = -c_0 I$$

Thus, when $c_0 \neq 0$ the inverse of A can be computed from

$$A^{-1} = \frac{-1}{c_0}\left(A^{n-1} + c_{n-1}A^{n-2} + c_{n-2}A^{n-3} + \cdots + c_1\right)$$

Once the characteristic polynomial of A has been computed, its inverse can be computed by forming the $n-1$ successive powers of A, namely, $A, A^2, A^3, \ldots, A^{n-1}$, multiplying them by the coefficients of $\phi_A(x)$, and adding the products together. These powers of A can be computed using a prefix circuit having $O(n)$ instances of the associative matrix multiplication operator and depth $O(\log n)$ measured in the number of instances of this operator. We have defined $M_{\text{matrix}}(n, K)$ to be the size of the smallest $n \times n$ matrix multiplication circuit with depth $K \log n$ (Definition 6.3.1). Thus, the successive powers of A can be computed by a circuit of size $O(nM_{\text{matrix}}(n, K))$ and depth $O(\log^2 n)$. The size bound can be improved to $O(\sqrt{n}M_{\text{matrix}}(n, K))$. (See Problem 6.15.)

To complete the derivation of the Csanky algorithm we must produce the coefficients of the characteristic polynomial of A. For this we invoke **Leverrier's theorem**. This theorem uses the notion of the **trace of a matrix** A, that is, the sum of the elements on its main diagonal, denoted $tr(A)$.

THEOREM 6.5.5 (Leverrier) *The coefficients of the characteristic polynomial of the $n \times n$ matrix A satisfy the following identity, where $s_r = tr(A^r)$ for $1 \leq r \leq n$:*

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ s_1 & 2 & 0 & \cdots & 0 \\ s_2 & s_1 & 3 & & 0 \\ \vdots & & & \ddots & \vdots \\ s_{n-1} & \cdots & s_2 & s_1 & n \end{bmatrix} \begin{bmatrix} c_{n-1} \\ c_{n-2} \\ c_{n-3} \\ \vdots \\ c_0 \end{bmatrix} = - \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{bmatrix} \tag{6.13}$$

Proof The degree-n characteristic polynomial $\phi_A(x)$ of A can be factored over a field of characteristic zero. If $\lambda_1, \lambda_2, \ldots, \lambda_n$ are its roots, we write

$$\phi_A(x) = \prod_{i=1}^{n} (x - \lambda_i)$$

From expanding this expression, it is clear that the coefficient c_{n-1} of x^{n-1} is $-\sum_{j=1}^{n} \lambda_j$. Similarly, expanding $\det(xI - A)$, c_{n-1} is the negative sum of the diagonal elements of A, that is, $c_{n-1} = -tr(A)$. It follows that $tr(A) = \sum_{j=1}^{n} \lambda_j$.

The λ_j's are called the **eigenvalues** of A, that is, values such that there exists an n-vector $\boldsymbol{u}$ (an **eigenvector**) such that $A\boldsymbol{u} = \lambda_j \boldsymbol{u}$. It follows that $A^r \boldsymbol{u} = \lambda_j^r \boldsymbol{u}$. It can be shown that $\lambda_1^r, \ldots, \lambda_n^r$ are precisely the eigenvalues of A^r, so $\Pi_{j=1}^{n}(x - \lambda_j^r)$ is the characteristic polynomial of A^r. Since $s_r = tr(A^r)$, $s_r = \sum_{j=1}^{n} \lambda_j^r$.

Let $s_0 = 1$ and $s_k = 0$ for $k < 0$. Then, to complete the proof of (6.13), we must show that the following identity holds for $1 \leq i \leq n$:

$$s_{i-1}c_{n-1} + s_{i-2}c_{n-2} + \cdots + s_1 c_{n-i+1} + ic_{n-i} = -s_i$$

Moving s_i to the left-hand side, substituting for the traces, and using the definition of the characteristic polynomial yield

$$ic_{n-i} + \sum_{j=1}^{n} \frac{\phi_A(\lambda_j) - \left(\lambda_j^{n-i} c_{n-i} + \lambda_j^{n-i-1} c_{n-i-1} + \cdots + \lambda_j c_1 + c_0\right)}{\lambda_j^{n-i}} = 0$$

Since $\phi_A(\lambda_j) = 0$, when we substitute l for $n - i$ it suffices to show the following for $0 \leq l \leq n - 1$:

$$(n-l)c_l = \sum_{j=1}^{n} \sum_{k=0}^{l} \frac{c_k}{\lambda_j^l} \tag{6.14}$$

This identity can be shown by induction using as the base case $l = 0$ and the following facts about the derivatives of the characteristic polynomial of A, which are easy to establish:

$$c_0 = (-1)^n \prod_{j=1}^{n} \lambda_j$$

$$c_k = \left. \frac{d^k \phi_A(x)}{dx^k} \right|_{x=0} = (-1)^k c_0 \sum_{\substack{j_1 \cdots \sum_{j_k} \\ j_r \neq j_s}} \prod_{t=1}^{k} \frac{1}{\lambda_{j_t}}$$

The reader is asked to show that (6.14) follows from these identities. (See Problem 6.17.) ■

Csanky's algorithm computes the traces of powers, namely the s_r's, and then inverts the lower triangular matrix given above, thereby solving for the coefficients of the characteristic polynomial. The coefficients are then used with a prefix computation, as mentioned earlier, to compute the inverse. Each of the n s_r's can be computed in $O(n)$ steps once the powers of A have been formed by the prefix computation described above. The lower triangular matrix is non-singular and can be inverted by a circuit with $M_{\text{matrix}}(n, K)$ operations and depth $O(\log^2 n)$, as shown in Theorem 6.5.1. The following theorem summarizes these results.

THEOREM 6.5.6 *The matrix inverse function for non-singular $n \times n$ matrices over a field of characteristic zero, $f_{A^{-1}}^{(n)}$, has an algebraic circuit whose size and depth satisfy the following bounds:*

$$C\left(f_{A^{-1}}^{(n)}\right) = O(nM_{\text{matrix}}(n, K))$$
$$C\left(f_{A^{-1}}^{(n)}\right) = O(\log^2 n)$$

The size bound can be improved to $O(\sqrt{n} M_{\text{matrix}}(n, K))$, as suggested in Problems 6.15 and 6.16.

6.6 Solving Linear Systems

A general linear system with $n \times n$ coefficient matrix M, n-vector $\boldsymbol{x}$ of unknowns and n-vector $\boldsymbol{b}$ is defined in (6.5) and repeated below:

$$M\boldsymbol{x} = \boldsymbol{b}$$

This system can be solved for $\boldsymbol{x}$ in terms of M and $\boldsymbol{b}$ using the following steps when M is not SPD. If it is SPD, the first step is unnecessary and can be eliminated.

a) Premultiply both sides by the transpose of M to produce the following linear system in which the coefficient matrix $M^T M$ is SPD:

$$M^T M\boldsymbol{x} = M^T \boldsymbol{b} = \boldsymbol{b}^*$$

b) Compute the LDL$^{\mathrm{T}}$ decomposition of M^TM.

c) Solve the system (6.15) by solving three subsequent systems:

$$LDL^T\boldsymbol{x} = \boldsymbol{b}^* \tag{6.15}$$
$$L\boldsymbol{u} = \boldsymbol{b}^* \tag{6.16}$$
$$D\boldsymbol{v} = \boldsymbol{u} \tag{6.17}$$
$$L^T\boldsymbol{x} = \boldsymbol{v} \tag{6.18}$$

Clearly, $L\boldsymbol{u} = LD\boldsymbol{v} = LDL^T\boldsymbol{x} = \boldsymbol{b}^*$.

The vector $\boldsymbol{b}^*$ is formed by a matrix-vector multiplication that can be done with n^2 multiplications and $n(n-1)$ additions, for a total of $2n^2 - n$ operations.

Since L is unit lower triangular, the system (6.16) is solved by **forward elimination**. The value of u_1 is b_1^*. The value of u_2 is $b_1^* - l_{2,1}u_1$, obtained by eliminating u_1 from the second equation. Similarly, on the jth step, the values of $u_1, u_2, \ldots, u_{j-1}$ are known and their weighted values can be subtracted from b_j^* to provide the value of u_j; that is,

$$u_j = b_j^* - l_{j,1}u_1 - l_{j,2}u_2 - \cdots - l_{j,j-1}u_{j-1}$$

for $1 \leq j \leq n$. Here $n(n-1)/2$ products are formed and $n(n-1)/2$ subtractions taken for a total of $n(n-1)$ operations.

Since D is diagonal, the system (6.17) is solved for $\boldsymbol{v}$ by multiplying u_j by the multiplicative inverse of $d_{j,j}$; that is,

$$v_j = u_j d_{j,j}^{-1}$$

for $1 \leq j \leq n$. This is called **normalization**. Here n divisions are performed.

Finally, the system (6.18) is solved for $\boldsymbol{x}$ by **backward substitution**, which is forward elimination applied to the elements of $\boldsymbol{x}$ in reverse order.

THEOREM 6.6.1 *Let $f^{(n)}_{\mathrm{SPD\_solve}} : R^{n^2+n} \mapsto R^n$ be the (partial) function that computes the solution to a linear system of equations defined by an $n \times n$ symmetric positive definite coefficient matrix M. Then*

$$C(f^{(n)}_{\mathrm{SPD\_solve}}) \leq C(f^{(n)}_{LDL^T}) + O(n^2)$$
$$D(f^{(n)}_{\mathrm{SPD\_solve}}) \leq C(f^{(n)}_{LDL^T}) + O(n)$$

If M is not SPD but is non-singular, an additional $O(M_{\mathrm{matrix}}(n,K))$ circuit elements and depth $O(\log n)$ suffice to compute it.

6.7 Convolution and the FFT Algorithm

The discrete Fourier transform (DFT) and convolution are widely used techniques with important applications in signal processing and computer science.

In this section we introduce the DFT, describe the fast Fourier transform algorithm, and derive the convolution theorem. The naive DFT algorithm on sequences of length n uses $O(n^2)$ operations; the fast Fourier transform algorithm uses only $O(n \log n)$ operations, a saving of a factor of at least 100 for $n \geq 1,000$. The convolution theorem provides a way to use the DFT to convolve two sequences in $O(n \log n)$ steps, many fewer than the naive algorithm for convolution, which uses $O(n^2)$ steps.

6.7.1 Commutative Rings*

Since the DFT is defined over commutative rings having an nth root of unity, we digress briefly to discuss such rings. (Commutative rings are defined in Section 6.2.)

DEFINITION 6.7.1 *A commutative ring $\mathcal{R} = (R, +, *, 0, 1)$ has a* **principal nth root of unity** *ω if $\omega \in R$ satisfies the following conditions:*

$$\omega^n = 1 \tag{6.19}$$

$$\sum_{k=0}^{n-1} \omega^{lk} = 0 \text{ for each } 1 \leq l \leq n-1 \tag{6.20}$$

*The elements $\omega^0, \omega^1, \omega^2, \ldots, \omega^{n-1}$ are the n***th roots of unity** *and the elements ω^0, $\omega^{-1}, \omega^{-2}, \ldots, \omega^{-(n-1)}$ are the n***th inverse roots of unity**. *(Note that $\omega^{-j} = \omega^{n-j}$ is the multiplicative inverse of ω^j since $\omega^j \omega^{n-j} = \omega^n = 1$.)*

Two commutative rings that have principal nth roots of unity are the complex numbers and the ring $\mathbb{Z}_m$ of integers modulo $m = 2^{tn/2} + 1$ when $t \geq 2$ and $n = 2^q$, as we show. The reader is asked to show that $\mathbb{Z}_m$ has a principal nth root of unity, as stated below. (See Problem 6.24.)

LEMMA 6.7.1 *Let $\mathbb{Z}_m$ be the ring of integers modulo m when $m = 2^{tn/2} + 1$, $t \geq 2$ and $n = 2^q$. Then $\omega = 2^t$ is a principal nth root of unity.*

An example of the ring $\mathbb{Z}_m$ is given by $t = 2$, $n = 4$, and $m = 2^4 + 1 = 17$. In this ring $\omega = 4$ is a principal fourth root of unity. This is true because $\omega^n = 4^4 = 16 \cdot 16 = (16+1)(16-1)+1 = 1 \bmod (16+1)$ and $\sum_{j=0}^{n-1} \omega^{pj} = ((4^p)^n - 1)/(4^p - 1) \bmod (17)$ $= ((4^n)^p - 1)/(4^p - 1) \bmod (17) = (1^p - 1)/(4^p - 1) \bmod (17) = 0 \bmod (17)$.

LEMMA 6.7.2 *$e^{2\pi i/n} = \cos(2\pi/n) + i \sin(2\pi/n)$ is a principal nth root of unity over the complex numbers where $i = \sqrt{-1}$ is the "imaginary unit."*

Proof The first condition is satisfied because $(e^{2\pi i/n})^n = e^{2\pi i} = 1$. Also, $\sum_{k=0}^{n-1} \omega^{lk} = (\omega^{ln} - 1)/(\omega^l - 1) = 0$ if $1 \leq l \leq n-1$ for $\omega = e^{2\pi i/n}$. ■

6.7.2 The Discrete Fourier Transform

The discrete Fourier transform has many applications. In Section 6.7.4 we see that it can be used to compute the convolution of two sequences efficiently, which is the same as computing the coefficients of the product of two polynomials. The discrete Fourier transform can also be used to construct a fast algorithm (circuit) for the multiplication of two binary integers [302]. It is widely used in processing analog data such as speech and music.

The n-point **discrete Fourier transform** $F_n : R^n \mapsto R^n$ maps n-tuples $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1})$ over R to n-tuples $\boldsymbol{f} = (f_0, f_1, \ldots, f_{n-1})$ over R; that is, $F_n(\boldsymbol{a}) = \boldsymbol{f}$. The components of $\boldsymbol{f}$ are defined as the values of the following polynomial $p(x)$ at the nth roots of unity:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \tag{6.21}$$

Then f_r, the rth component of $F_n(\boldsymbol{a})$, is defined as

$$f_r = p(\omega^r) = \sum_{k=0}^{n-1} a_k \omega^{rk} \tag{6.22}$$

This computation is equivalent to the following matrix-vector multiplication:

$$F_n(\boldsymbol{a}) = [w^{ij}] \times \boldsymbol{a} \tag{6.23}$$

where $[\omega^{ij}]$ is the $n \times n$ **Vandermonde matrix** whose i, j entry is ω^{ij}, $0 \leq i, j \leq n-1$, and $\boldsymbol{a}$ is treated as a column vector.

The n-point **inverse discrete Fourier transform** $F_n^{-1} : R^n \mapsto R^n$ is defined as the values of the following polynomial $q(x)$ at the inverse nth roots of unity:

$$q(x) = (f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1} x^{n-1})/n \tag{6.24}$$

That is, the inverse DFT maps an n-tuple $\boldsymbol{f}$ to an n-tuple $\boldsymbol{g}$, namely, $F_n^{-1}(\boldsymbol{f}) = \boldsymbol{g}$, where g_s is defined as follows:

$$g_s = q(\omega^{-s}) = \frac{1}{n} \sum_{l=0}^{n-1} f_l \omega^{-ls} \tag{6.25}$$

This computation is equivalent to the following matrix-vector multiplication:

$$F_n^{-1}(\boldsymbol{f}) = \left[\frac{1}{n}\omega^{-ij}\right] \times \boldsymbol{f}$$

Because of the following lemma it is legitimate to call F_n^{-1} the **inverse** of F_n.

LEMMA 6.7.3 *For all $\boldsymbol{a} \in R^n$, $\boldsymbol{a} = F_n^{-1}(F_n(\boldsymbol{a}))$.*

Proof Let $\boldsymbol{f} = F_n(\boldsymbol{a})$ and $\boldsymbol{g} = F_n^{-1}(\boldsymbol{f})$. Then g_s satisfies the following:

$$\begin{aligned} g_s &= \frac{1}{n} \sum_{l=0}^{n-1} f_l \omega^{-ls} = \frac{1}{n} \sum_{l=0}^{n-1} \sum_{k=0}^{n-1} a_k \omega^{(k-s)l} \\ &= \sum_{k=0}^{n-1} a_k \frac{1}{n} \sum_{l=0}^{n-1} \omega^{(k-s)l} \\ &= a_s \end{aligned}$$

The second equation results from a change in the order of summation. The last follows from the definition of nth roots of unity. It follows that the matrix $[\omega^{-ij}/n]$ is the inverse of $[\omega^{ij}]$. ■

The computation of the n-point DFT and its inverse using the naive algorithms suggested by their definitions requires $O(n^2)$ steps. Below we show that a fast DFT algorithm exists for which only $O(n \log n)$ steps suffice.

6.7.3 Fast Fourier Transform

The **fast Fourier transform algorithm** is a consequence of the following observation: when n is even, the polynomial $p(x)$ in equation (6.21) can be decomposed as

$$\begin{aligned} p(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \\ &= (a_0 + a_2 x^2 + \cdots + a_{n-2} x^{n-2}) \\ &\quad + x\,(a_1 + a_3 x^2 + \cdots + a_{n-1} x^{n-2}) \\ &= p_e(x^2) + x p_o(x^2) \end{aligned} \tag{6.26}$$

Here $p_e(y)$ and $p_o(y)$ are polynomials of degree $(n/2) - 1$.

Let n be a power of 2, that is, $n = 2^d$. As stated above, the n-point DFT of $\boldsymbol{a}$ is obtained by evaluating $p(x)$ at the nth roots of unity. Because of the decomposition of $p(x)$, it suffices to evaluate $p_e(y)$ and $p_o(y)$ at $y = (\omega^0)^2, (\omega^1)^2, (\omega^2)^2, \ldots, (\omega^{n-1})^2 = (\omega^2)^0, (\omega^2)^1, (\omega^2)^2, \ldots, (\omega^2)^{n-1}$ and combine their values with one multiplication and one addition for each of the n roots of unity. However, because ω^2 is a $(n/2)$th principal root of unity (see Problem 6.25), $(\omega^2)^{(n/2)+r} = (\omega^2)^r$ and the n powers of ω^2 collapse to $n/2$ distinct powers of ω^2, namely, the $(n/2)$th roots of unity. Thus, $p(x)$ at the nth roots of unity can be evaluated by evaluating $p_e(y)$ and $p_o(y)$ at the $(n/2)$th roots of unity and combining their values with one addition and multiplication for each of the nth roots of unity. In other words, the n-point DFT of $\boldsymbol{a}$ can be done by performing the $(n/2)$-point DFT of its even and odd subsequences and combining the results with $O(n)$ additional steps. This is the fast Fourier transform (FFT) algorithm.

We denote by $F^{(d)}$ the directed acyclic graph associated with the straight-line program resulting from this realization of the FFT on $n = 2^d$ inputs. A circuit for the 16-point FFT algorithm inputs, $F^{(4)}$, is shown in Fig. 6.7. It is computed from the eight-point FFT on the even and odd components of $\boldsymbol{a}$, as shown in the boxed regions. These components are permuted because each of these smaller FFTs is computed recursively in turn. (The index of

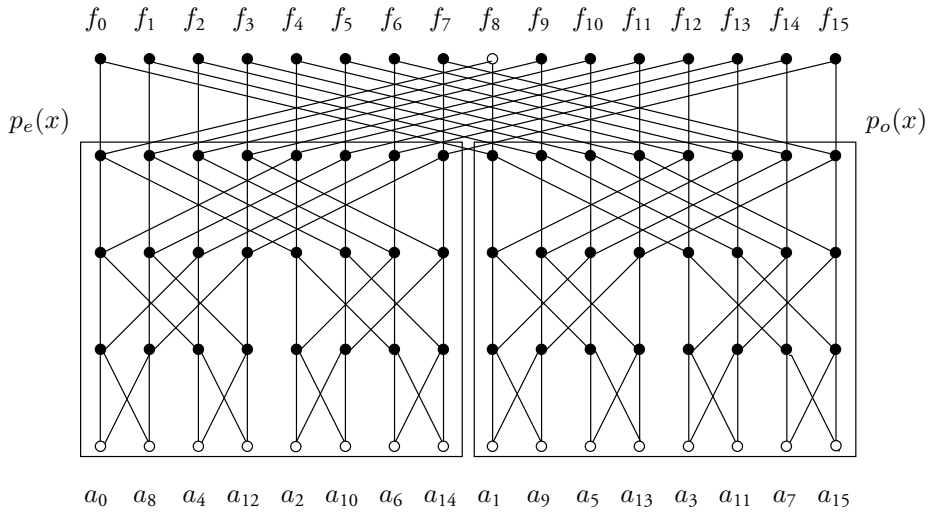

Figure 6.7 A circuit $F^{(4)}$ for the FFT algorithm on 16 inputs.

the ith input vertex from the left is obtained by writing the integer i as a binary number, reversing the bits, and converting the resulting binary number to an integer. This is called the **bit-reverse permutation** of the binary representation of the integer. For example, the third input from the left has index 3, which is (011) in binary. Reversed, the binary number is (110), which represents 12.) Inputs are associated with the open vertices at the bottom of the graph. Each vertex except for input vertices is associated with an addition and a multiplication. For example, the white vertex at the top of the graph computes $f_8 = p_e((\omega^8)^2) + \omega^8 p_o((\omega^8)^2)$, where $(\omega^8)^2 = \omega^{16} = \omega$.

Let $C(F^{(d)})$ and $D(F^{(d)})$ be the size and depth of circuits for the 2^d-point FFT algorithm for integer $d \geq 1$. The construction given above leads to the following recurrences for these two measures:

$$C\left(F^{(d)}\right) \leq 2C\left(F^{(d-1)}\right) + 2^{d+1}$$
$$D\left(F^{(d)}\right) \leq D\left(F^{(d-1)}\right) + 2$$

Also, examination of the base case of $n = 2$ demonstrates that $C\left(F^{(1)}\right) = 3$ and $D\left(F^{(1)}\right) = 2$, from which we have the following theorem.

THEOREM 6.7.1 *Let $n = 2^d$. The circuit for the n-point FFT algorithm over a commutative ring $\mathcal{R}$ has the following circuit size and depth bounds:*

$$C\left(F^{(d)}\right) \leq 2n \log n$$
$$D\left(F^{(d)}\right) \leq 2 \log n$$

The FFT graph is used in later chapters to illustrate tradeoffs between space and time, space and the number of I/O operations, and area and time for computation with VLSI machines. For each of these applications we decompose the FFT graph into sub-FFT graphs. One such decomposition is shown in Fig. 6.7. A more general decomposition is shown in Fig. 6.8 and described below.

LEMMA 6.7.4 *The 2^d-point FFT graph $F^{(d)}$ can be decomposed into 2^e 2^{d-e}-point* bottom *FFT graphs, $\{F_{b,j}^{(d-e)} \mid 1 \leq j \leq 2^e\}$, and 2^{d-e} 2^e-point* top *FFT graphs, $\{F_{t,j}^{(e)} \mid 1 \leq j \leq 2^{d-e}\}$. The ith input of $F_{t,j}^{(e)}$ is the jth output of $F_{b,i}^{(d-e)}$.*

In Fig. 6.8 the vertices and edges have been grouped together as recognizable FFT graphs and surrounded by shaded boxes. The edges between boxes are not edges of the FFT graph but instead are used to identify vertices that are simultaneously outputs of bottom FFT subgraphs and inputs to top FFT subgraphs.

COROLLARY 6.7.1 *$F^{(d)}$ can be decomposed into $\lfloor d/e \rfloor$ stages each containing 2^{d-e} copies of $F^{(e)}$ and one stage containing 2^{d-k} copies of $F^{(k)}$, $k = d - \lfloor d/e \rfloor e$. ($F(0)$ is a single vertex.) The output vertices of one stage are the input vertices to the next.*

Proof From Lemma 6.7.4, each of the 2^e bottom FFT subgraphs $F^{(d-e)}$ can be further decomposed into 2^{d-2e} top FFT subgraphs $F^{(e)}$ and 2^e bottom FFT subgraphs $F^{(d-2e)}$. By repeating this process t times, $t \leq d/e$, $F^{(d)}$ can be decomposed into t stages each

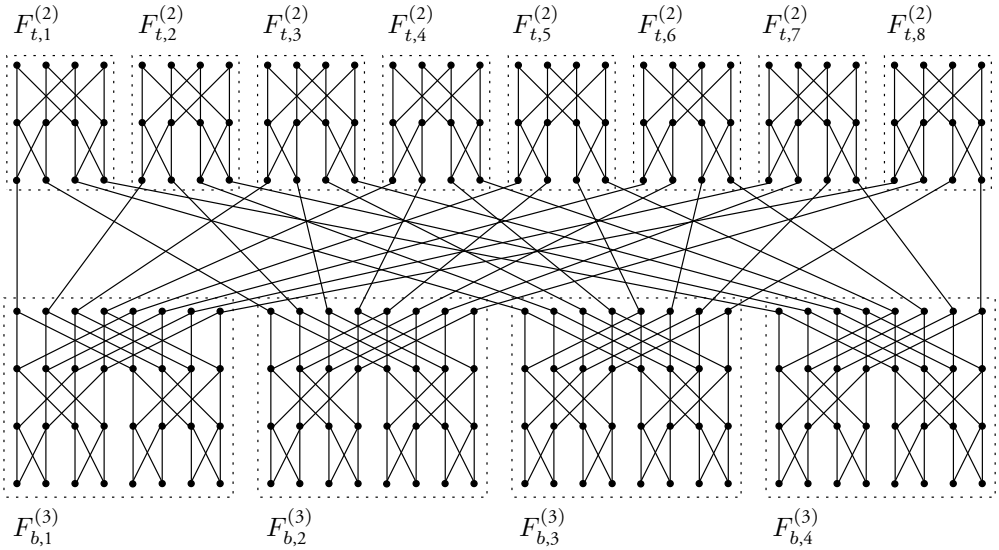

Figure 6.8 Decomposition of the 32-point FFT graph $F^{(5)}$ into four copies of $F^{(3)}$ and 8 copies of $F^{(2)}$. The edges between bottom and top sub-FFT graphs do not exist in the FFT graph. They are used here to identify common vertices and highlight the communication needed among sub-FFT graphs.

containing 2^{d-e} copies of $F^{(e)}$ and one stage containing 2^{d-te} copies of $F^{(d-te)}$. The result follows by setting $t = \lfloor d/e \rfloor$. ■

6.7.4 Convolution Theorem

The **convolution function** $f^{(n,m)}_{\text{conv}} : R^{n+m} \mapsto R^{n+m-1}$ over a commutative ring $\mathcal{R}$ maps an n-tuple $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1})$ and an m-tuple $\boldsymbol{b} = (b_0, b_1, \ldots, b_{m-1})$ onto an $(n+m-1)$-tuple $\boldsymbol{c}$, denoted $\boldsymbol{c} = \boldsymbol{a} \otimes \boldsymbol{b}$, where c_j is defined as follows:

$$c_j = \sum_{r+s=j} a_r * b_s \text{ for } 0 \leq j \leq n+m-2$$

Here $\sum$ and $*$ are addition and multiplication over the ring $\mathcal{R}$. The direct computation of the convolution function using the above formula takes $O(nm)$ steps. The convolution theorem given below and the fast Fourier transform algorithm described above allow the convolution function to be computed in $O(n \log n)$ steps when $n = m$.

Associate with $\boldsymbol{a}$ and $\boldsymbol{b}$ the following polynomials in the variable x:

$$a(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$
$$b(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_{n-1} x^{n-1}$$

Then the coefficient of the term x^j in the product polynomial $c(x) = a(x)b(x)$ is clearly the term c_j in the convolution $\boldsymbol{c} = \boldsymbol{a} \otimes \boldsymbol{b}$.

Convolution is used in signal processing and integer multiplication. In signal processing, convolution describes the results of passing a signal through a linear filter. In binary integer

multiplication the polynomials $a(2)$ and $b(2)$ represent binary numbers; convolution is related to the computation of their product.

The **convolution theorem** is one of the most important applications of the DFT. It demonstrates that convolution, which appears to require $O(n^2)$ operations when $n = m$, can in fact be computed by a circuit with $O(n)$ operations plus a small multiple of the number needed to compute the DFT and its inverse.

THEOREM 6.7.2 *Let $\mathcal{R} = (R, +, *, \mathbf{0}, \mathbf{1})$ be a commutative ring and let $\boldsymbol{a}, \boldsymbol{b} \in R^n$. Let $F_{2n} : R^{2n} \mapsto R^{2n}$ and $F_{2n}^{-1} : R^{2n} \mapsto R^{2n}$ be the $2n$-point DFT and its inverse over R. Let $F_{2n}(\boldsymbol{a}) \times F_{2n}(\boldsymbol{b})$ denote the $2n$-tuple obtained from the term-by-term product of the components of $F_{2n}(\boldsymbol{a})$ and $F_{2n}(\boldsymbol{b})$. Then, the convolution $\boldsymbol{a} \otimes \boldsymbol{b}$ satisfies the following identity:*

$$\boldsymbol{a} \otimes \boldsymbol{b} = F_{2n}^{-1}(F_{2n}(\boldsymbol{a}) \times F_{2n}(\boldsymbol{b}))$$

Proof The n-point DFT $F_n : R^n \mapsto R^n$ transforms the n-tuple of coefficients $\boldsymbol{a}$ of the polynomial $p(x)$ of degree $n-1$ into the n-tuple $\boldsymbol{f} = F_n(\boldsymbol{a})$. In fact, the rth component of $\boldsymbol{f}$, f_r, is the value of the polynomial $p(x)$ at the rth of the n roots of unity, namely $f_r = p(\omega^r)$. The n-point inverse DFT $F_n^{-1} : R^n \mapsto R^n$ inverts the process through a similar computation. If $q(x)$ is the polynomial of degree $n-1$ whose lth coefficient is f_l/n, then the sth component of the inverse DFT on $\boldsymbol{f}$, namely $F_n^{-1}(\boldsymbol{f})$, is $a_s = q(\omega^{-s})$.

As stated above, to compute the convolution of n-tuples $\boldsymbol{a}$ and $\boldsymbol{b}$ it suffices to compute the coefficients of the product polynomial $c(x) = a(x)b(x)$. Since the product $c(x)$ is of degree $2n-2$, we can treat it as a polynomial of degree $2n-1$ and take the $2n$-point DFT, F_{2n}, of it and its inverse, F_{2n}^{-1}, of the result. This seemingly futile process leads to an efficient algorithm for convolution. Since the DFT is obtained by evaluating a polynomial

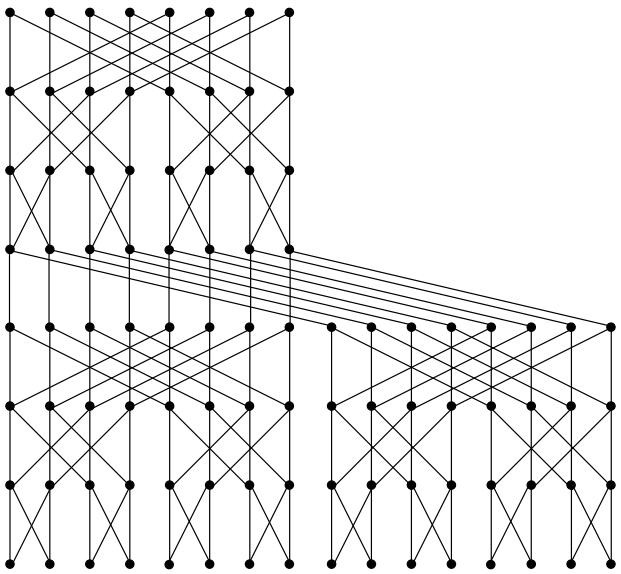

Figure 6.9 The DAG associated with the straight-line program resulting from the application of the FFT to the convolution theorem with sequences of length 8.

at the n roots of unity, the DFT of $c(x)$ can be done at the $2n$ roots of unity by evaluating $a(x)$ and $b(x)$ at the $2n$ roots of unity (that is, computing the DFTs of their coefficients as if they had degree $2n-1$), multiplying their values together, and taking the $2n$-point inverse DFT, that is, performing the computation stated in the theorem. ■

The combination of the convolution theorem and the algorithm for the FFT provides a fast straight-line program for convolution, as stated below. The directed acyclic graph for this straight-line program is shown in Fig. 6.9 on page 269.

THEOREM 6.7.3 *Let $n = 2^d$. The convolution function $f_{\text{conv}}^{(n,n)} : R^{2n} \mapsto R^{2(n-1)}$ over a commutative ring $\mathcal{R}$ can be computed by a straight-line program over $\mathcal{R}$ with size and depth satisfying the following bounds:*

$$C\left(f_{\text{conv}}^{(n,n)}\right) \leq 12n \log 2n$$
$$D\left(f_{\text{conv}}^{(n,n)}\right) \leq 4 \log 2n$$

6.8 Merging and Sorting Networks

The **sorting problem** is to put into ascending or descending order a collection of items that are drawn from a totally ordered set. A set is **totally ordered** if for every two distinct elements of the set one is larger than the other. The **merging problem** is to merge two sorted lists into one sorted list. Sorting and merging algorithms can be either straight-line or non-straight-line. An example of a non-straight-line merging algorithm is the following:

> Create a new sorted list from two sorted lists by removing the smaller item from the two lists and appending it to the new list until one list is empty, at which point append the non-empty list to the end of the new list.

The binary sorting function $f_{\text{sort}}^{(n)} : \mathcal{B}^n \mapsto \mathcal{B}^n$ described in Section 2.11 sorts a Boolean n-tuple into descending order. The combinational circuit given there is an example of a straight-line sorting network, a network realized by a straight-line program. When the set of elements to be sorted is not Boolean, sorting networks can become quite a bit more complicated, as we see below.

In this section we describe **sorting networks**, circuits constructed from comparator operators that take n elements drawn from a finite totally ordered set $\mathcal{A}$ and put them into sorted order. A **comparator function** $\otimes : \mathcal{A}^2 \mapsto \mathcal{A}^2$ with arguments a and b returns their maximum and minimum; that is, $\otimes(a, b) = (\max(a, b), \min(a, b))$.

It is convenient to show a comparator operator as a vertical edge between two lines carrying values, as in Fig. 6.10(a). The values on the two lines to the right of the edge are the values to its left in sorted order, the smaller being on the upper line. A sorting network is an example of a **comparator network**, a circuit in which the only operator is a comparator. Input values appear on the left and output values appear on the right in sorted order.

Shown in Fig. 6.10(b) is an **insertion-sorting network** on five inputs that inserts an element into a previously sorted sublist. Two inputs are sorted at the wavefront labeled A. Between wavefronts A and B a new item is inserted that is compared against the previously sorted sublist and inserted into its proper position. The same occurs between wavefronts B and C and after

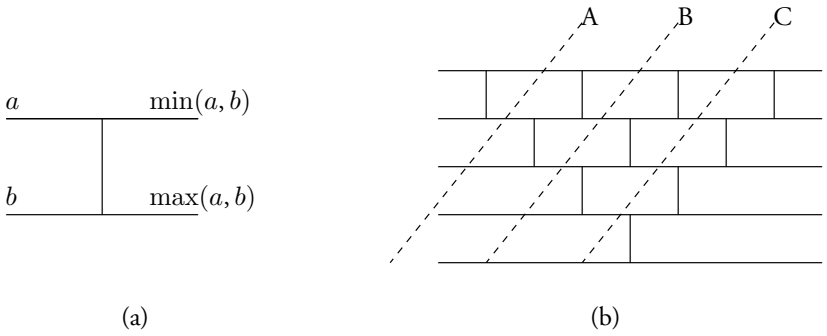

Figure 6.10 (a) A comparison operator, and (b) an insertion-sorting network.

wavefront C. An insertion-sorting network can be realized with one comparator for the first two inputs and $k-1$ more for the kth input, $3 \leq k \leq n$. Let $C_{\text{insert}}(n)$ and $D_{\text{insert}}(n)$ denote the size and depth of an insertion-sorting network on n elements. Then $C(2) = 1$ and $D(2) = 1$, and

$$C_{\text{insert}}(n) \leq C_{\text{insert}}(n-1) + n - 1 = n(n-1)/2$$
$$D_{\text{insert}}(n) \leq \max(D_{\text{insert}}(n-1) + 1, n-1) = n - 1$$

The depth bound follows because there is a path of length $n-1$ through the chain of comparators added at the last wavefront and every path through the sorting network is extended by one comparator with the addition of the new wavefront. A simple proof by induction establishes these results.

6.8.1 Sorting Via Bitonic Merging

We now describe **Batcher's bitonic merging network** $BM(m)$, which is the basis for a sorting network. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ be ordered sequences of length m. That is, $x_j \leq x_{j+1}$ and $y_j \leq y_{j+1}$. As suggested in Fig. 6.11, the even-indexed components of $\boldsymbol{x}$ are merged with the odd-indexed components of $\boldsymbol{y}$, as are the odd-indexed components of $\boldsymbol{x}$ and the even-indexed components of $\boldsymbol{y}$. Each of the four lists that are merged are themselves sorted. The two lists are interleaved and the kth and $(k+1)$st elements, k even, are compared and swapped if necessary. To prove correctness of this circuit, we use the zero-one principle which is stated below for sorting networks but applied later to merging networks.

THEOREM 6.8.1 (Zero-one principle) *If a comparator network for inputs over a set $\mathcal{A}$ correctly sorts all binary inputs, it correctly sorts all inputs.*

Proof The proof is by contradiction. Suppose the network correctly sorts all 0-1 sequences but fails to sort the input sequence $(a_1, a_2, \ldots, a_n)$. Then there are inputs a_i and a_j such that $a_i < a_j$ but the network puts a_j before a_i.

Since a sorting network contains only comparators, if we replace each entry a_r in an input sequence $(a_1, a_2, \ldots, a_n)$ with a new entry $h(a_r)$, where $h(a)$ is monotonically non-decreasing in a ($h(a)$ is non-decreasing as a increases), each comparison of entries a_r and a_s is replaced by a comparison of entries $h(a_r)$ and $h(a_s)$. Since $a_r < a_s$ only

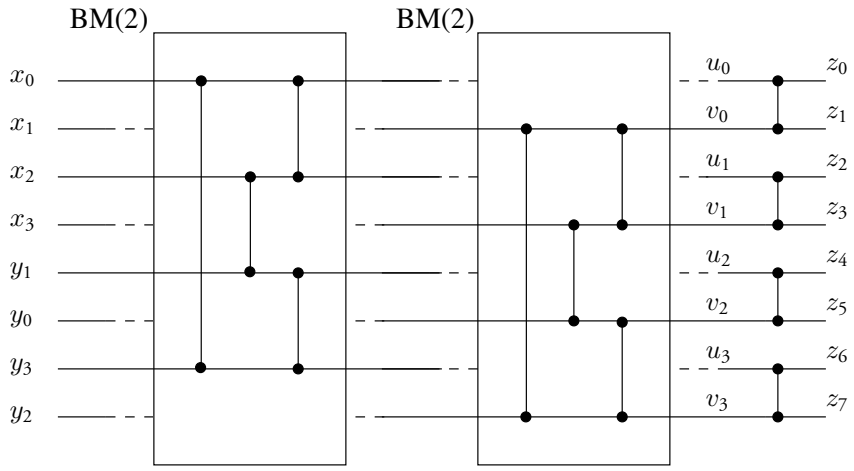

Figure 6.11 A recursive construction of the bitonic merging network $BM(4)$. The even-indexed elements of one sorted sequence are merged with the odd-indexed elements of the other, the resulting sequences interleaved, and the even- and succeeding odd-indexed elements compared. The inputs of one sequence are permuted to demonstrate that $BM(4)$ uses two copies of $BM(2)$.

if $h(a_r) \leq h(a_s)$, the set of comparisons made by the sorting network will be exactly the same on $(a_1, a_2, \ldots, a_n)$ as on $(h(a_1), h(a_2), \ldots, h(a_n))$. Thus, the original output $(b_1, b_2, \ldots, b_n)$ will be replaced by the output sequence $(h(b_1), h(b_2), \ldots, h(b_n))$.

Since it is presumed that the comparator network puts a_i and a_j in the incorrect order, let $h(x)$ be the following monotone function:

$$h(x) = \begin{cases} 0 & \text{if } x \leq a_i \\ 1 & \text{if } x > a_i \end{cases}$$

Then the input and output sequences to the comparator network are binary. However, the output sequence is not sorted (a_j appears before a_i but $h(a_j) = 1$ and $h(a_i) = 0$), contradicting the hypothesis of the theorem. It follows that all sequences over $\mathcal{A}$ must be sorted correctly. ■

We now show that Batcher's bitonic merging circuit correctly merges two sorted lists. If a correct m-sorter exists, then a correct $2m$-sorter can be constructed by combining two m-sorters with a correct $2m$-input bitonic merging circuit. It follows that a correct $2m$-input bitonic merging circuit exists if and only if the resulting sorting network is correct. This is the core idea in a proof by induction of correctness of the $2m$-input bitonic merging circuit. The basis for induction is the fact that individual comparators correctly sort sequences of two elements.

Suppose that $\boldsymbol{x}$ and $\boldsymbol{y}$ are sorted $0-1$ sequences of length m. Let $\boldsymbol{x}$ have k 0's and $m-k$ 1's, and let $\boldsymbol{y}$ have l 0's and $m-l$ 1's. Then the leftmost merging network of Fig. 6.11 selects exactly $\lceil k/2 \rceil$ 0's from $\boldsymbol{x}$ and $\lfloor l/2 \rfloor$ 0's from $\boldsymbol{y}$ to produce the sequence $\boldsymbol{u}$ consisting of $a = \lceil k/2 \rceil + \lfloor l/2 \rfloor$ 0's followed by 1's. Similarly, the rightmost merging network produces

the sequence $\boldsymbol{v}$ consisting of $b = \lfloor k/2 \rfloor + \lceil l/2 \rceil$ 0's followed by 1's. Since $\lceil x \rceil - \lfloor x \rfloor$ is 0 or 1, it follows that either $a = b$, $a = b - 1$, or $a = b + 1$. Thus, when $\boldsymbol{u}$ and $\boldsymbol{v}$ are interleaved to produce the sequence $\boldsymbol{z}$ it contains a sequence of $a + b$ 0's followed by 1's when $a = b$ or $a = b + 1$, or $2a$ 0's followed by 1 0 followed by 1's when $a = b - 1$, as suggested below:

$$\boldsymbol{z} = \overbrace{0, 0, \ldots, 0,}^{2a} 1, 0, 1, \ldots, 1$$

Thus, if for each $0 \leq k \leq m - 1$ the outputs in positions $2k$ and $2k + 1$ are compared and swapped, if necessary, the output will be properly sorted.

The graph of $BM(4)$ of Fig. 6.11 illustrates that $BM(4)$ is constructed of two copies of $BM(2)$. In addition, it demonstrates that the operations of each of the two $BM(2)$ subnetworks can be performed in parallel. Another important observation is that this graph is isomorphic to an FFT graph when the comparators are replaced by two-input butterfly graphs, as shown in Fig. 6.12.

THEOREM 6.8.2 *Batcher's $2n$-input bitonic merging circuit $BM(n)$ for merging two sorted n-sequences, $n = 2^k$, has the following size and depth bounds over the basis Ω of comparators:*

$$C_\Omega(BM(n)) \leq n(\log n + 1)$$
$$D_\Omega(BM(n)) \leq \log n + 1$$

Proof Let $C(k)$ and $D(k)$ be the size and depth of $BM(n)$. Then $C(0) = 1$, $D(0) = 1$, $C(k) = 2C(k-1) + 2^k$, and $D(k) = D(k-1) + 1$. It follows that $C(k) = (k+1)2^k$ and $D(k) = k + 1$. (See Problem 6.29.) ■

This leads us to the recursive construction of a Batcher's bitonic sorting network $BS(n)$ for sequences of length n, $n = 2^k$. It merges the output of two copies of $BS(n/2)$ using a copy of Batcher's n-input bitonic merging circuit $BM(n/2)$. The proof of the following theorem is left as an exercise. (See Problem 6.28.)

THEOREM 6.8.3 *Batcher's n-input bitonic sorting circuit $BS(n)$ for $n = 2^k$ has the following size and depth bounds over the basis Ω of comparators:*

$$C_\Omega(BS(n)) = \frac{n}{4}[\log^2 n + \log n]$$

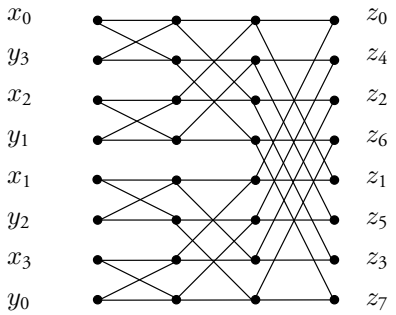

Figure 6.12 The graph resulting from the replacement of comparators in Fig. 6.11 with two-input butterfly graphs and the permutation of inputs. All edges are directed from left to right.

$$D_\Omega(BS(n)) = \frac{1}{2}\log n(\log n - 1)$$

6.8.2 Fast Sorting Networks

Ajtai, Komlós, and Szemerédi [14] have shown the existence of a sorting network (known as the **AKS sorting network**) on n inputs whose circuit size and depth are $O(n \log n)$ and $O(\log n)$, respectively. The question had been open for many years whether such a sorting network existed. Prior to [14] it was thought that sorting networks required $\Omega(\log^2 n)$ depth.

Problems

MATHEMATICAL PRELIMINARIES

6.1 Show that $(\mathbb{Z}, +, *, 0, 1)$ is a commutative ring, where $+$ and $*$ denote integer addition and multiplication and 0 and 1 denote the first two integers.

6.2 Let $\mathbb{Z}_p$ be the **set of integers modulo** p, $p > 0$, under addition and multiplication modulo p with additive identity 0 and multiplicative identity 1. Show that $\mathbb{Z}_p$ is a ring.

6.3 A **field** $\mathcal{F}$ is a commutative ring in which each element other than **0** has a multiplicative inverse. Show that $(\mathbb{Z}_p, +, *, 0, 1)$ is a field when p is a prime.

MATRICES

6.4 Let $M_{n\times n}$ be the set of $n \times n$ matrices over a ring $\mathcal{R}$. Show that $(M_{n\times n}, +_n, \times_n, 0_n, I_n)$ is a ring, where $+_n$ and $\times_n$ are the matrix addition and multiplication operators and 0_n and I_n are the $n \times n$ zero and identity matrices.

6.5 Show that the maximum number of linearly independent rows and of linearly independent columns of an $n \times m$ matrix A over a field are the same.

Hint: Use the fact that permuting the rows and/or columns of A and adding a scalar product of one row (column) of A to any other row (column) does not change its rank. Use row and column permutations as well as additions of scalar products to rows and/or columns of A to transform A into a matrix that contains the largest possible identity matrix in its upper left-hand corner. This is called **Gaussian elimination**.

6.6 Show that $(A\,B)^T = B^T\,A^T$ for all $m \times n$ matrices A and $n \times p$ matrices B over a commutative ring $\mathcal{R}$.

MATRIX MULTIPLICATION

6.7 The standard matrix-vector multiplication algorithm for a general $n \times n$ matrix requires $O(n^2)$ operations. Show that at most $O(n^{\log_2 3})$ operations are needed when the matrix is Toeplitz.

Hint: Assume that n is a power of two and treat the matrix as a 2×2 matrix of $n/2 \times n/2$ matrices. Also note that only $2n - 1$ values determine all the entries in a Toeplitz matrix. Thus, the difference between two $n \times n$ Toeplitz matrices does not require n^2 operations.

6.8 Generalize Strassen's matrix multiplication algorithm to matrices that are $m \times m$ for $m = p2^k$, p and k both integers. Derive bounds on the size and depth of a circuit realizing this version of the algorithm.

For arbitrary n, show how $n \times n$ matrices can be embedded into $m \times m$ matrices, $m = p2^k$, so that this new version of the algorithm can be used. Show that upper bounds of $4.77n^{\log_2 7}$ and $O(\log n)$ on the size and depth of this algorithm can be obtained.

6.9 Show that Strassen's matrix multiplication algorithm can be used to multiply square Boolean matrices by replacing OR by addition modulo $n + 1$. Derive a bound on the size and depth of a circuit to realize this algorithm.

6.10 Show that, when one of two $n \times n$ Boolean matrices A and B is fixed and known in advance, A and B can be multiplied by a circuit with $O(n^3/\log n)$ operations and depth $O(\log n)$ to produce the product $C = AB$ using the information provided below.

a) Multiplication of A and B is equivalent to n multiplications of A with an $n \times 1$ vector $\boldsymbol{x}$, a column of B.

b) Since A is a $0 - 1$ matrix, the product $A\boldsymbol{x}$ consists of sums of variables in $\boldsymbol{x}$.

c) The product $A\boldsymbol{x}$ can be further decomposed into the sum $A_1\boldsymbol{x}_1 + A_2\boldsymbol{x}_2 + \cdots + A_k\boldsymbol{x}_k$ where $k = \lceil n/\lceil \log n \rceil \rceil$, A_j is the $n \times \lceil \log n \rceil$ submatrix consisting of columns $(j-1)\lceil \log n \rceil + 1$ through $j\lceil \log n \rceil$ of A, and $\boldsymbol{x}_j$ is the jth set of $\lceil \log n \rceil$ rows (variables) in $\boldsymbol{x}$.

d) There are at most n distinct sums of $\lceil \log n \rceil$ variables each of which can be formed in at most $2n$ addition steps, thereby saving a factor of $\lceil \log n \rceil$.

TRANSITIVE CLOSURE

6.11 Let $A = [a_{i,j}]$, $1 \leq i, j \leq n$, be a Boolean matrix that is the adjacency matrix of a directed graph $G = (V, E)$ on $n = |V|$ vertices. Give a proof by induction that the entry in the rth row and sth column of $A^k = A^{k-1} \times A$ is 1 if there is a path containing k edges from vertex r to vertex s and 0 otherwise.

6.12 Consider a directed graph $G = (V, E)$ in which each edge carries a label drawn from a semiring. Let the entry in the ith row and jth column of the adjacency matrix of G contain the label of the edge between vertices i and j if there is such an edge and the empty set otherwise. Assume that the labels of edges in G are drawn from a semiring. Show that Theorems 6.4.1, 6.4.2, and 6.4.3 hold for such labeled graphs.

MATRIX INVERSION

6.13 Show that over fields the following properties hold for matrix inversion:

a) The inverse of a 2×2 upper (lower) triangular matrix is the matrix with the off-diagonal term negated.

b) The inverse of a 2×2 diagonal matrix is a diagonal matrix in which the ith diagonal element is the multiplicative inverse of the ith diagonal element of the original matrix.

6.14 Show that a lower triangular Toeplitz matrix T can be inverted by a circuit of size $O(n \log n)$ and depth $O(\log^2 n)$.

Hint: Assume that $n = 2^k$, write T as a 2×2 matrix of $n/2 \times n/2$ matrices, and devise a recursive algorithm to invert T.

6.15 Exhibit a circuit to compute the characteristic polynomial $\phi_A(x)$ of an $n \times n$ matrix A over a field $\mathcal{R}$ that has $O(\max(n^3, \sqrt{n}M_{\text{matrix}}(n)))$ field operations and depth $O(\log^2 n)$.

Hint: Consider the case $n = k^2$. Represent the integer i, $0 \leq i \leq n-1$, by the unique pair of integers (r, s), $0 \leq r, s \leq k-1$, where $i = rk + s$. Represent the coefficient c_{i+1}, $0 \leq i \leq n-2$, of $\phi_A(x)$ by $c_{r,s}$. Then we can write $\phi_A(x)$ as follows:

$$\phi_A(x) = \sum_{r=0}^{k-1} A^{rk} \left(\sum_{s=0}^{k-1} c_{r,s} A^s \right)$$

Show that it suffices to perform $k^2 n^2 = n^3$ scalar multiplications and $k(k-1)n^2 \leq n^3$ additions to form the inner sums, k multiplications of $n \times n$ matrices, and kn^2 scalar additions to combine these products. In addition, $A^2, A^3, \ldots, A^{k-1}$and $A^k, A^{2k}, \ldots, A^{(k-1)k}$ must be computed.

6.16 Show that the traces of powers, s_r, $1 \leq r \leq n$, for an $n \times n$ matrix A over a field can be computed with $O(\sqrt{n}M_{\text{matrix}}(n))$ operations.

Hint: By definition $s_r = \sum_{j=1}^{n} a_{j,j}^{(r)}$, where $a_{j,j}^{(r)}$ is the jth diagonal term of the matrix A^r. Let n be a square. Represent r uniquely by a pair (a, b), where $1 \leq a, b \leq \sqrt{n} - 1$ and $r = a\sqrt{n} + b$. Then $A^r = A^{a\sqrt{n}} A^b$. Thus, $a_{j,j}^{(r)}$ can be computed as the product of the jth row of $A^{a\sqrt{n}}$ with the jth column of A^b. Then, for each j, $1 \leq j \leq n$, form the $\sqrt{n} \times n$ matrix R_j whose ath row is the jth row of $A^{a\sqrt{n}}$, $0 \leq a \leq \sqrt{n} - 1$. Also form the $n \times \sqrt{n}$ matrix C_j whose bth column is the jth column of A^b, $1 \leq b \leq \sqrt{n} - 1$. Show that the product $R_j C_j$ contains each of the terms $a_{j,j}^{(r)}$ for all values of r, $0 \leq r \leq n-1$ and that the products $R_j C_j$, $1 \leq j \leq n$, can be computed efficiently.

6.17 Show that (6.14) holds by applying the properties of the coefficients of the characteristic polynomial of an $n \times n$ matrix stated in (6.15).

Hint: Use proof by induction on l to establish (6.14).

CONVOLUTION

6.18 Consider the convolution $f_{\text{conv}}^{(n,m)} : R^{n+m} \mapsto R^{n+m-2}$ of an n-tuple $\boldsymbol{a}$ with an m-tuple $\boldsymbol{b}$ when $n \ll m$. Develop a circuit for this problem whose size is $O(m \log n)$ that uses the convolution theorem multiple times.

Hint: Represent the m-tuple $\boldsymbol{b}$ as sequence of $\lceil m/n \rceil$ n-tuples.

6.19 The **wrapped convolution** $f_{\text{wrapped}}^{(n)} : \mathcal{R}^{2n} \mapsto \mathcal{R}^n$ maps n-tuples $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\boldsymbol{b} = (b_0, b_1, \ldots, b_{n-1})$, denoted $\boldsymbol{a} \star \boldsymbol{b}$, to the n-tuple $\boldsymbol{c}$ the jth component of which, c_j, is defined as follows:

$$c_j = \sum_{r+s \,=\, j \bmod n} a_r * b_s$$

Show that the wrapped convolution on n-tuples contains the standard convolution on $\lfloor (n+1)/2 \rfloor$-tuples as a subfunction and vice versa.

Hint: In both halves of the problem, it helps to characterize the standard and wrapped convolutions as matrix-vector products. It is straightforward to show that the wrapped convolution contains the standard convolution as a subfunction. To show the other result, observe that the matrix characterizing the standard convolution contains a Toeplitz matrix as a submatrix. Consider, for example, the standard convolution of two six-tuples. The matrix associated with the wrapped convolution contains a special type of Toeplitz matrix.

6.20 Show that the standard convolution function $f^{(n,n)}_{\text{conv}} : R^{2n} \mapsto R^{2n-2}$ is a subfunction of the integer multiplication function, $f^{(n)}_{\text{mult}} : \mathcal{B}^{2n\lceil \log n \rceil} \mapsto \mathcal{B}^{2n\lceil \log n \rceil}$ of Section 2.9 when R is the ring of integers modulo 2.

Hint: Represent the two sequences to be convolved as binary numbers that have been padded with zeros so that at most one bit in a sequence appears among $\lceil \log n \rceil$ positions.

DISCRETE FOURIER TRANSFORM

6.21 Let $n = 2^k$. Use proof by induction to show that for all elements a of a commutative ring $\mathcal{R}$ the following identity holds, where $\prod$ is the product operation:

$$\sum_{j=0}^{n-1} a^j = \prod_{j=0}^{k-1} (1 + a^{2^j})$$

6.22 Let $n = 2^k$ and let $\mathcal{R}$ be a commutative ring. For $\omega \in \mathcal{R}$, $\omega \neq 0$, let $m = \omega^{n/2} + 1$. Show that for $1 \leq p < n$

$$\sum_{j=0}^{n-1} \omega^{pj} = 0 \bmod m$$

Hint: Represent p as the product of the largest power of 2 with an odd integer and apply the result of Problem 6.21.

6.23 Let n and ω be positive powers of two. Let $m = \omega^{n/2} + 1$. Show that in the ring $\mathbb{Z}_m$ of integers modulo m the integer n has a multiplicative inverse and that ω is a principal nth root of unity.

6.24 Let n be even. Use the results of Problems 6.21, 6.22, and 6.23 to show that $\mathbb{Z}_m$, the set of integers modulo m, $m = 2^{tn/2} + 1$ for any positive integer $t \geq 2$, is a commutative ring in which $\omega = 2^t$ is a principal nth root of unity.

6.25 Let ω be a principal nth root of unity of the commutative ring $\mathcal{R} = (R, +, *, 0, 1)$. Show that ω^2 is a principal $(n/2)$th root of unit.

6.26 A **circulant** is an $n \times n$ matrix in which the rth row is the rth cyclic shift of the first row, $2 \leq r \leq n$. When n is a prime, show that computing the DFT of a vector of length n is equivalent to multiplying by an $(n-1) \times (n-1)$ circulant.

6.27 Show that the multiplication of circulant matrix with a vector can be done by a circuit of size $O(n \log n)$ and depth $O(\log n)$.

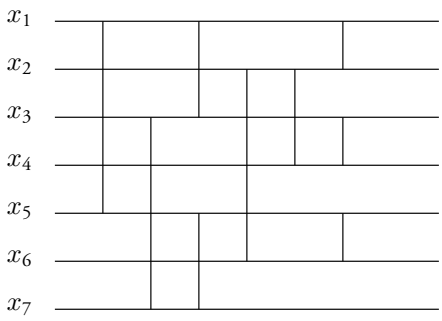

Figure 6.13 A bitonic sorter on seven inputs.

MERGING AND SORTING

6.28 Prove Theorem 6.8.3.

6.29 Show that the recurrences given below and stated in the proof of Theorem 6.8.2 have the solutions shown, where $C(0) = 1$ and $D(0) = 1$:

$$C(k) = 2C(k-1) + 2^k = (k+1)2^k$$
$$D(k) = D(k-1) + 1 = k + 1$$

6.30 A sequence $(x_1, x_2, \ldots, x_n)$ is **bitonic** if there is an integer $0 \leq k \leq n$ such that $x_1 > \ldots > x_k \leq \ldots \leq x_n$.

a) Show that a bitonic sorting network can be constructed as follows: i) sort $(x_1, x_3, x_5, \ldots)$ and $(x_2, x_4, x_6, \ldots)$ in bitonic sorters whose lines are interleaved, ii) compare and interchange the outputs in pairs, beginning with the least significant pairs. (See Fig. 6.13.)

b) Show that two ordered lists can be merged with a bitonic sorter and that an n-sorter can be constructed from bitonic sorters.

c) Determine the number of comparators in a 2^k-sorter based on merging with bitonic sorters.

Chapter Notes

The bulk of this chapter concerns matrix computations, a topic with a long history. Many books have been written on this subject to which the interested reader may refer. (See [25], [44], [104], [197], and [361].)

Among the more important recent results in this area are the matrix multiplication algorithm of Strassen [318]. Many other improvements have been made on this work, among the most significant of which is the demonstration by Coppersmith and Winograd [81] that two $n \times n$ matrices can be multiplied with $O(n^{2.376})$ ring operations.

The relationships between transitive closure and matrix multiplication embodied in Theorems 6.4.2 and 6.4.3 as well as the generalization of these results to closed semirings are taken from the book by Aho, Hopcroft, and Ullman [10].

Winograd [363] demonstrated that matrix multiplication is no harder than matrix inversion, whereas Aho, Hopcroft, and Ullman [10] demonstrated the converse.

Csanky's algorithm for matrix inversion is reported in [82]. Leverrier's method for computing the characteristic function of a matrix is described in [97].

Although the FFT algorithm became well known through the work of Cooley and Tukey [80], the idea actually begins with Gauss in 1805! (See Heideman, Johnson, and Burrus [129].)

The zero-one principle for the study of comparator networks is due to Knuth [169]. Oddly enough, Batcher's odd-even merging network is due to Batcher [29].

Borodin and Munro [56] is a good early source for arithmetic complexity, the size and depth of arithmetic circuits for problems related to matrices and polynomials. More recent work on the parallel evaluation of arithmetic circuits is surveyed by JáJá [147, Chapter 8] and von zur Gathen [110].