$$\det \begin{bmatrix} 2 & 1 & 3 \\ 4 & -2 & 10 \\ 5 & -3 & 13 \end{bmatrix}$$

$$2 * \det \begin{bmatrix} -2 & 10 \\ -3 & 13 \end{bmatrix} + \quad -1 * \det \begin{bmatrix} 4 & 10 \\ 5 & 13 \end{bmatrix} + \quad 3 * \det \begin{bmatrix} 4 & -2 \\ 5 & -3 \end{bmatrix} = 0$$

INPUT OUTPUT

13.4 Determinants and Permanents

Input description: An $n \times n$ matrix M.

Problem description: What is the determinant $|M|$ or permanent $perm(M)$ of the matrix m?

Discussion: Determinants of matrices provide a clean and useful abstraction in linear algebra that can be used to solve a variety of problems:

- Testing whether a matrix is *singular*, meaning that the matrix does not have an inverse. A matrix M is singular iff $|M| = 0$.
- Testing whether a set of d points lies on a plane in fewer than d dimensions. If so, the system of equations they define is singular, so $|M| = 0$.
- Testing whether a point lies to the left or right of a line or plane. This problem reduces to testing whether the sign of a determinant is positive or negative, as discussed in Section 17.1 (page 564).
- Computing the area or volume of a triangle, tetrahedron, or other simplicial complex. These quantities are a function of the magnitude of the determinant, as discussed in Section 17.1 (page 564).

The determinant of a matrix M is defined as a sum over all $n!$ possible permutations π_i of the n columns of M:

$$|M| = \sum_{i=1}^{n!} (-1)^{sign(\pi_i)} \prod_{j=1}^{n} M[j, \pi_j]$$

where $sign(\pi_i)$ denotes the number of pairs of elements out of order (called *inversions*) in permutation π_i.

A direct implementation of this definition yields an $O(n!)$ algorithm, as does the cofactor expansion method I learned in high school. Better algorithms to evaluate determinants are based on LU-decomposition, discussed in Section 13.1 (page 395). The determinant of M is simply the product of the diagonal elements of the LU-decomposition of M, which can be found in $O(n^3)$ time.

A closely related function called the *permanent* arises in combinatorial problems. For example, the permanent of the adjacency matrix of a graph G counts the number of perfect matchings in G. The permanent of a matrix M is defined by

$$perm(M) = \sum_{i=1}^{n!} \prod_{j=1}^{n} M[j, \pi_j]$$

differing from the determinant only in that all products are positive.

Surprisingly, it is NP-hard to compute the permanent, even though the determinant can easily be computed in $O(n^3)$ time. The fundamental difference is that $det(AB) = det(A) \times det(B)$, while $perm(AB) \neq perm(A) \times perm(B)$. There are permanent algorithms running in $O(n^2 2^n)$ time that prove to be considerably faster than the $O(n!)$ definition. Thus, finding the permanent of a 20×20 matrix is not out of the realm of possibility.

Implementations: The linear algebra package LINPACK contains a variety of Fortran routines for computing determinants, optimized for different data types and matrix structures. It can be obtained from Netlib, as discussed in Section 19.1.5 (page 659).

JScience provides an extensive linear algebra package (including determinants) as part of its comprehensive scientific computing library. *JAMA* is another matrix package written in Java. Links to both and many related libraries are available from *http://math.nist.gov/javanumerics/*.

Nijenhuis and Wilf [NW78] provide an efficient Fortran routine to compute the permanent of a matrix. See Section 19.1.10 (page 661). Cash [Cas95] provides a C routine to compute the permanent, motivated by the Kekulé structure count of computational chemistry.

Two different codes for approximating the permanent are provided by Barvinok. The first, based on [BS07], provides codes for approximating the permanent and a Hafnian of a matrix, as well as the number of spanning forests in a graph. See *http://www.math.lsa.umich.edu/∼barvinok/manual.html*. The second, based on [SB01], can provide estimates of the permanent of 200×200 matrices in seconds. See *http://www.math.lsa.umich.edu/∼barvinok/code.html*.

Notes: Cramer's rule reduces the problems of matrix inversion and solving linear systems to that of computing determinants. However, algorithms based on LU-determination are faster. See [BM53] for an exposition on Cramer's rule.

Determinants can be computed in $o(n^3)$ time using fast matrix multiplication, as shown in [AHU83]. Section 13.3 (page 401) discusses such algorithms. A fast algorithm for computing the sign of the determinant—an important problem for performing robust geometric computations—is due to Clarkson [Cla92].

The problem of computing the permanent was shown to be #P-complete by Valiant [Val79], where #P is the class of problems solvable on a "counting" machine in polynomial time. A counting machine returns the number of distinct solutions to a problem. Counting the number of Hamiltonian cycles in a graph is a #P-complete problem that is trivially NP-hard (and presumably harder), since any count greater than zero proves that the graph is Hamiltonian. Counting problems can be #P-complete even if the corresponding decision problem can be solved in polynomial time, as shown by the permanent and perfect matchings.

Minc [Min78] is the primary reference on permanents. A variant of an $O(n^2 2^n)$-time algorithm due to Ryser for computing the permanent is presented in [NW78].

Recently, probabilistic algorithms have been developed for estimating the permanent, culminating in a fully-polynomial randomized approximation scheme that provides an arbitrary close approximation in time that depends polynomially upon the input matrix and the desired error [JSV01].

Related Problems: Solving linear systems (see page 395), matching (see page 498), geometric primitives (see page 564).