Lecture 31 Csanky's Algorithm

In 1976, Csanky gave a parallel algorithm to invert matrices [26]. This was one of the very first NC algorithms. It set the stage for a large body of research in parallel linear algebra that culminated with Mulmuley's 1986 result that the rank of a matrix over an arbitrary field can be computed in NC [82].

In this lecture we will develop Csanky's algorithm. Along the way, we give some NC algorithms for problems of independent interest, including the calculation of the characteristic polynomial and determinant of a matrix and the solution of linear recurrences. First we recall some basic NC algorithms:

Inner product The inner product of two vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ can be computed in $O(\log n)$ parallel arithmetic steps by n processors. First, produce in parallel the products $a_i b_i$, $1 \leq i \leq n$; then add the products in a treelike fashion.

Matrix multiplication If A is an $m \times n$ matrix and B is an $n \times p$ matrix, their product AB can be computed by $O(mpn)$ processors in $O(\log n)$ time. AB has mp entries, each obtained as the inner product of a row of A and a column of B.

Powers of A The powers $A^1, A^2, \ldots, A^n$ of an $n \times n$ matrix A can be obtained as the products of prefixes of the n-component sequence $(A, A, \ldots, A)$. This can be accomplished in $O(\log^2 n)$ time by $O(n^4)$ processors arranged in a parallel prefix circuit of width n in which the associative operation is $n \times n$ matrix multiplication.

31.1 Inversion of Lower Triangular Matrices

Given an $n \times n$ lower triangular matrix A, break it up into submatrices

$$A = \left[\begin{array}{c|c} B & 0 \\ \hline C & D \end{array}\right]$$

where B is $\lfloor \frac{n}{2} \rfloor \times \lfloor \frac{n}{2} \rfloor$, C is $\lceil \frac{n}{2} \rceil \times \lfloor \frac{n}{2} \rfloor$, and D is $\lceil \frac{n}{2} \rceil \times \lceil \frac{n}{2} \rceil$. Recursively compute B^{-1} and D^{-1} in parallel. Then

$$A^{-1} = \left[\begin{array}{c|c} B^{-1} & 0 \\ \hline -D^{-1}CB^{-1} & D^{-1} \end{array}\right] .$$

The parallel computation time of this algorithm satisfies the relation

$$T(n) = T(\frac{n}{2}) + 2M(\frac{n}{2})$$

where $T(\frac{n}{2})$ is the time needed to invert B and D in parallel and $2M(\frac{n}{2})$ is the time needed to form the matrix product $-D^{-1}CB^{-1}$. With $O(n^3)$ processors, we have $M(n) = O(\log n)$, whence $T(n) = O(\log^2 n)$.

31.2 Solution of Linear Recurrences

It may seem surprising that the n^{th} term of a linear recurrence such as the Fibonacci sequence $F_0 = 1$, $F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$ should be computable without first computing the first $n-1$ terms. In fact, the n^{th} term of any linear recurrence can be computed in parallel polylog time.

A general linear recurrence is a system of the form

$$\begin{aligned} x_1 &= c_1 \\ x_2 &= a_{21}x_1 + c_2 \\ x_3 &= a_{31}x_1 + a_{32}x_2 + c_3 \\ &\vdots \\ x_n &= a_{n1}x_1 + \cdots + a_{n,n-1}x_{n-1} + c_n \end{aligned}$$

where the a_{ij} and c_i are given, and we wish to solve for the x_i. For example, the Fibonacci sequence is given by the system $c_1 = c_2 = 1$ and $c_i = 0$ for $i \geq 3$, $a_{i,i-1} = a_{i,i-2} = 1$ for $i \geq 3$, and all other $a_{ij} = 0$.

Let $a_{ij} = 0$ for $j \geq i$, let A be the $n \times n$ matrix (a_{ij}), let x be the vector (x_i), and let c be the vector (c_i). The system above is then equivalent to the matrix-vector equation

$$Ax + c = x ,$$

or equivalently,

$$c = (I - A)x .$$

The matrix $I - A$ is lower triangular with 1's on the diagonal, and thus can be inverted in NC by the method described in the previous section. This allows us to solve for x:

$$x = (I - A)^{-1}c .$$

31.3 The Characteristic Polynomial of a Matrix

We give a linear recurrence for the coefficients of the characteristic polynomial of a given matrix A, which can then be solved by the method of the previous section. This linear recurrence was known to Sir Isaac Newton.

The characteristic polynomial of a matrix A is defined to be

$$\begin{aligned} \det(xI - A) &= x^n - s_1 x^{n-1} + s_2 x^{n-2} - \cdots \pm s_n \\ &= \prod_{i=1}^{n} (x - \lambda_i) \end{aligned}$$

where x is an indeterminate, $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A (multiplicities counted), and $\det B$ is the determinant of B. The coefficient s_1 is called the *trace* of A and is denoted $\operatorname{tr} A$. It is both the sum of the eigenvalues and the sum of the diagonal elements of A:

$$\begin{aligned} s_1 &= \operatorname{tr} A \\ &= \sum_{i=1}^{n} \lambda_i \\ &= \sum_{i=1}^{n} a_{ii} , \end{aligned}$$

so it can be easily computed in NC. It can also be shown that λ_i^m is an eigenvalue of A^m of the same multiplicity as λ_i of A, therefore

$$\operatorname{tr} A^m = \sum_{i=1}^{n} \lambda_i^m .$$

The constant coefficient s_n is the determinant of A and is the product of the eigenvalues:

$$\begin{aligned} s_n &= \det A \\ &= \prod_{i=1}^{n} \lambda_i . \end{aligned}$$

The intermediate coefficients are called the *elementary symmetric polynomials in* $\lambda_1, \ldots, \lambda_n$ and are given by

$$s_k \quad = \sum_{1 \le i_1 < \cdots < i_k \le n} \lambda_{i_1} \lambda_{i_2} \cdots \lambda_{i_k} \ ;$$

in other words, the sum of all products of k-element submultisets of the multiset of eigenvalues of A.

Define

$$f_k^m \quad = \sum_{\substack{1 \le i_1 < \cdots < i_k \le n \\ j \notin \{i_1, \ldots, i_k\}}} \lambda_{i_1} \lambda_{i_2} \cdots \lambda_{i_k} \lambda_j^m \ .$$

At the extremes,

$$\begin{aligned} f_k^0 &= (n-k) s_k \\ f_0^m &= \mathrm{tr}\ A^m \ . \end{aligned}$$

Then

$$\begin{aligned} & s_k \cdot \mathrm{tr}\ A^m \\ &= \Big(\sum_{1 \le i_1 < \cdots < i_k \le n} \lambda_{i_1} \lambda_{i_2} \cdots \lambda_{i_k} \Big) \cdot \Big(\sum_{j=1}^{n} \lambda_j^m \Big) \\ &= \sum_{\substack{1 \le i_1 < \cdots < i_k \le n \\ j \notin \{i_1, \ldots, i_k\}}} \lambda_{i_1} \lambda_{i_2} \cdots \lambda_{i_k} \lambda_j^m + \sum_{\substack{1 \le i_1 < \cdots < i_k \le n \\ j \in \{i_1, \ldots, i_k\}}} \lambda_{i_1} \lambda_{i_2} \cdots \lambda_{i_k} \lambda_j^m \\ &= f_k^m + f_{k-1}^{m+1} \ . \end{aligned}$$

It follows that

$$\begin{aligned} & s_k \cdot \mathrm{tr}\ A^0 - s_{k-1} \cdot \mathrm{tr}\ A^1 + s_{k-2} \cdot \mathrm{tr}\ A^2 - \cdots \pm s_1 \cdot \mathrm{tr}\ A^{k-1} \mp \mathrm{tr}\ A^k \\ &= (f_k^0 + fk - 1^1) - (f_{k-1}^1 + f_{k-2}^2) + \cdots \pm (f_1^{k-1} + f_0^k) \mp f_0^k \\ &= f_k^0 \\ &= (n-k) s_k \ . \end{aligned}$$

This gives a recurrence for s_k in terms of $s_1, \ldots, s_{k-1}$:

$$s_k \quad = \quad \frac{1}{k}(s_{k-1} \cdot \mathrm{tr}\ A - s_{k-2} \cdot \mathrm{tr}\ A^2 + \cdots \pm \mathrm{tr}\ A^k) \ . \tag{39}$$

The tr A^m can be computed in NC by computing the powers of A using parallel prefix and summing the diagonal elements. The recurrence (39) can then be solved using the method of the previous section.

31.4 Inversion of Arbitrary Nonsingular Matrices

We use the *Cayley-Hamilton Theorem*, which says that every matrix satisfies its characteristic equation:

$$A^n - s_1 A^{n-1} + s_2 A^{n-2} - \cdots \mp s_{n-1} A \pm s_n I \quad = \quad 0 \; .$$

Multiplying by A^{-1} and rearranging terms, we get

$$A^{-1} \quad = \quad \frac{1}{s_n}(s_{n-1} I - s_{n-2} A + \cdots \pm s_1 A^{n-2} \mp A^{n-1}) \; . \tag{40}$$

The coefficients s_k of the characteristic polynomial and powers of A are computed by the method of the previous section. The matrix polynomial (40) can be computed in time $O(\log n)$ using $O(n^3)$ processors. The complete algorithm to compute A^{-1} from A runs in $O(\log^2 n)$ parallel arithmetic steps on $O(n^4)$ processors.