Algorithms in Systems Engineering
IE170

Lecture 25

Dr. Ted Ralphs

References for Today's Lecture

- Required reading
  - CLRS Chapter 28

Vectors and Matrices

- Vectors and matrices are constructs that arise naturally in many applications.
- Operating on vectors and matrices requires numerical algorithms.
- An $m \times n$ *matrix* is an array of mn real numbers:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

- A is said to have n *columns* and m *rows*.
- An n*-dimensional column vector* is a matrix with one column.
- An n*-dimensional row vector* is a matrix with one row .
- By default, a *vector* will be considered a column vector.
- The set of all n*-dimensional vectors* will be denoted $\mathbb{R}^n$.
- The set of all $m \times n$ *matrices* will be denoted $\mathbb{R}^{m \times n}$.

Matrices

- The *transpose* of a matrix A is

$$A^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$$

- If $x, y \in \mathbb{R}^n$, then $x^T y = \sum_{i=1}^n x_i y_i$.
- This is called the *inner product* of x and y.
- If $A \in \mathbb{R}^{m \times n}$, then A_j is the j^{th} column, and a_j is the j^{th} row.
- If $A \in \mathbb{R}^{m \times k}, B \in \mathbb{R}^{k \times n}$, then $[AB]_{ij} = a_i^T B_j$.
- A *square matrix* is one for which the number of columns equals the number of rows.

Storing Vectors and Matrices

- The *density* of a matrix is the percentage of entries that are nonzero.
- *Dense* vectors can simply be stored in an array.
- *Dense* matrices can be stored in a 2-dimensional array.
- Matrices that arise in practice, however, are typically *sparse*.
- Sparse matrices can be stored using a strategy similar adjacency lists using three vectors:
  - The first vector stores the locations of the nonzero entries in each column.
  - The second vector stores the actual values corresponding to each one of those locations.
  - The third vector stores the location in the first two matrices of the entries corresponding to each column.

Multiplying Matrices

- In many numerical algorithms, we are faced with the problem of multiplying two matrices together.
- Multiplication of square matrices can be accomplished either using the obvious iterative algorithm or a recursive divide and conquer algorithm.
- Algorithm for multiplying 2×2 matrices

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

- To calculate the answer, we have:

$$\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dg \\ u &= cf + dh \end{aligned}$$

Generalizing

- The algorithm for 2×2 matrices can be generalized.
- Simply divide each matrix into $\frac{n}{2} \times \frac{n}{2}$ submatrices.

$$\begin{bmatrix} R & S \\ T & U \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

- To calculate the answer, we have:

$$\begin{aligned} R &= AE + BG \\ S &= AF + BH \\ T &= CE + DG \\ U &= CF + DH \end{aligned}$$

Running Time for Matrix Multiplication

- The obvious iterative algorithm involves $\Theta(n^3)$ multiplications and $\Theta(n^3)$ additions.
- In the recursive method, each call has two steps:
  - 8 recursive calls to multiply square matrices of size $n/2$.
  - A recombine step involving 4 matrix additions.
- The running time is hence the solution to the recursion $T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$.
- The solution to this by the Master Theorem is $T(n) \in \Theta(n^3)$, so it is no better than the iterative method.
- Surprisingly, there is a recursive method discovered by Strassen that involves multiplying only 7 pairs of matrices of size $n/2$.
- This results in a running time $T(n) \in \Theta(n^{\lg 7}) = \Theta(n^{2.81})$.