EEL 4837Programming for Electrical Engineers II

Ivan Ruchkin

Assistant Professor

Department of Electrical and Computer Engineering University of Florida at Gainesville

> iruchkin@ece.ufl.edu http://ivan.ece.ufl.edu

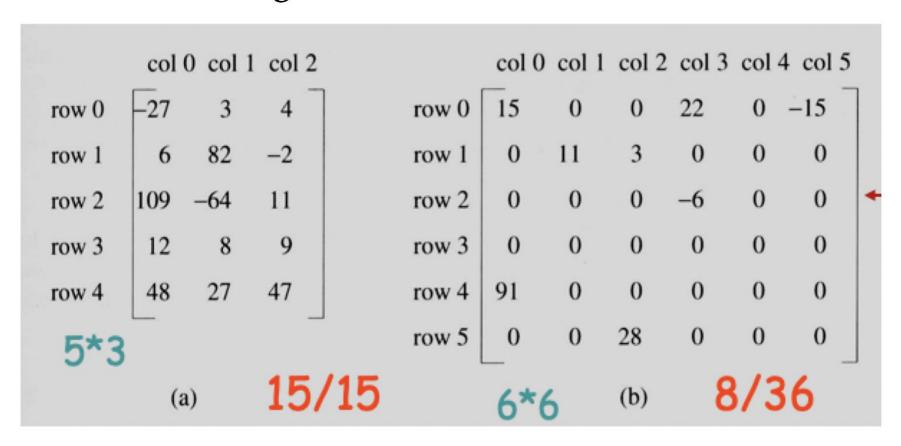
Matrices

Readings:

- Cormen 28
- Weiss 1.7 (using STL)

Matrices & Sparse Matrices

In mathematics, a matrix contains m rows and n columns of elements. We write $m \times n$ to designate a matrix with m rows and n columns

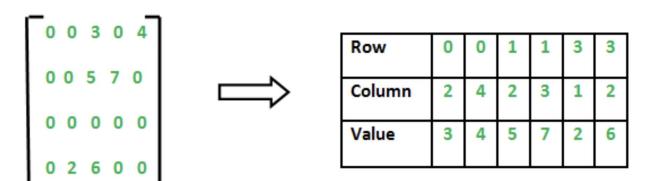


A Separate Sparse Matrix Representation

The standard representation of a matrix is a two dimensional array defined as

```
a[MAX_ROWS][MAX_COLS]
```

- We can locate quickly any element by writing *a*[*i*][*j*]
- Sparse matrix wastes space
 - We must consider alternate forms of representation.
 - Our representation of sparse matrices should store only nonzero elements.
 - Each element is characterized by <row, col, value>.



```
int m[2][3] =
{
    {2, 7, 1},
    {5, 6, 0}
};
```

// print 7
cout << m[0][1];
// get 2nd row
m[1];</pre>

Matrix Algorithms

- 1. Given a matrix A, write a program to compute the transpose of A.
- 2. Given two m x n matrices A and B,write a program to compute the sum C= A + B
- 3. Given an m x n matrix A and an n x p matrix B, write a program to compute their product $C = A \times B$
- 4. Given an n x n square matrix A, write a program to compute determinant of A.
- 5. Given an n x n square matrix A (that is not singular), write a program to compute the inverse A⁻¹

```
for (int i = 0; i < R1; i++)
   for (int j = 0; j < C1; j++) {
     transpose[j][i] = a[i][j];
for (int i = 0; i < R1; i++) {
  for (int j = 0; j < C1; j++) {
    sum[i][j] = a[i][j] + b[i][j];
for (int i = 0; i < R1; i++) {
  for (int j = 0; j < C2; j++) {
    prod[i][j] = 0;
    for (int k = 0; k < R2; k++) {
      prod[i][j] += a[i][k] * b[k][j];
```

Exercises: Sparse Matrix Algorithms

- 1. Given a matrix A, write a program to compute the transpose of A.
- 2. Given two m x n matrices A and B, write a program to compute the sum C = A + B
- 3. Given an m x n matrix A and an n x p matrix B, write a program to compute their product $C = A \times B$

Implement the above when A and B are sparse matrices.

- 4. Given a matrix A in regular representation, write a program to create a sparse matrix representation of A.
- 5. Given a matrix A in sparse matrix representation, write a program to generate a regular representation of A.

Exercises: Sparse Matrix Algorithms

1. Given a matrix A, write a program to compute the transpose of A.

```
      Matrix 1: (4x4)

      Row
      Column
      Value

      1
      2
      10

      1
      4
      12

      3
      3
      5

      4
      1
      15

      4
      2
      12
```

- We traverse through the matrix element by element
- At each time we replace the pair <row, column> by the pair <column, row>
 - This will mean that the resulting entries will not be sorted
- Then sort the entries using the pair <row, column> as key

Exercises: Sparse Matrix Algorithms

2. Given two m x n matrices A and B, write a program to compute the sum C = A + B

| Matrix 1: (4x4) | | | | |
|-----------------|--------|-------|--|--|
| Row | Column | Value | | |
| 1 | 2 | 10 | | |
| 1 | 4 | 12 | | |
| 3 | 3 | 5 | | |
| 4 | 1 | 15 | | |
| 4 | 2 | 12 | | |
| | | | | |

| Matrix 2: (4X4) | | | | |
|-----------------|--------|-------|--|--|
| Row | Column | Value | | |
| 1 | 3 | 8 | | |
| 2 | 4 | 23 | | |
| 3 | 3 | 9 | | |
| 4 | 1 | 20 | | |
| 4 | 2 | 25 | | |
| | | | | |

| Resu | ılt of A | Addition: | (4x4) |
|------|----------|-----------|-------|
| Row | Column | Value | |
| 1 | 2 | 10 | |
| 1 | 3 | 8 | |
| 1 | 4 | 12 | |
| 2 | 4 | 23 | |
| 3 | 3 | 14 | |
| 4 | 1 | 35 | |
| 4 | 2 | 37 | |

- We traverse through the two matrices element by element
- At each time we copy the smaller <row, column> entry to the new array
- If the same entry exists in both arrays, we put the sum of the values
- When one matrix is done, we copy the remaining entries of the other

Solving Linear Equations with Matrices

- We view a linear equation as the matrix form AX = b where A is n x n matrix and X and b are n x 1 matrices
- Here we will consider square full-rank/invertible matrices
- We can derive X by simply computing the product A⁻¹ b, but that will be expensive

Linear Equation system

$$2x_2 - 3x_3 = 4
-2x_1 + x_2 + 2x_3 = -6
2x_1 + x_3 = 0$$

Augmented Matrix

$$\left(\begin{array}{ccc|c}
0 & 2 & -3 & 4 \\
-2 & 1 & 2 & -6 \\
2 & 0 & 1 & 0
\end{array}\right)$$

- 1. Transposing equations doesn't change their solution.
- 2. Scaling an equation doesn't change its solution.
- 3. If a set of numbers satisfies two equations, then it also satisfies the equation which is one plus a scalar multiple of the other.
- 1. Transposing (switching) rows in an augmented matrix does not change the solution.
- 2. Scaling any row in an augmented matrix does not change the solution.
- 3. Adding to any row in an augmented matrix any multiple of any other row in the matrix does not change the solution.

Gaussian Elimination Step 1: Echelon Matrix

Transpose the first and third equations:

$$\begin{pmatrix} 0 & 2 & -3 & 4 \\ -2 & 1 & 2 & -6 \\ 2 & 0 & 1 & 0 \end{pmatrix} \longleftrightarrow \begin{pmatrix} 2 & 0 & 1 & 0 \\ -2 & 1 & 2 & -6 \\ 0 & 2 & -3 & 4 \end{pmatrix}$$

Now we can add the first row to the second and get another zero in that column.

$$\begin{pmatrix} 2 & 0 & 1 & 0 \\ -2 & 1 & 2 & -6 \\ 0 & 2 & -3 & 4 \end{pmatrix} \xrightarrow{\longleftarrow} \begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 1 & 3 & -6 \\ 0 & 2 & -3 & 4 \end{pmatrix}$$

We add (-2) times the second row to the third row.

$$\begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 1 & 3 & -6 \\ 0 & 2 & -3 & 4 \end{pmatrix} \xrightarrow{-2} \rightsquigarrow \begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 1 & 3 & -6 \\ 0 & 0 & -9 & 16 \end{pmatrix}$$

The matrix is now in "row echelon" form and we can solve it using back substitution

Gaussian Elimination Step 2: back substitution

Starting with the last matrix above, we scale the last row by $-\frac{1}{9}$:

$$\begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 1 & 3 & -6 \\ 0 & 0 & -9 & 16 \end{pmatrix} \quad | \quad -\frac{1}{9} \quad \rightsquigarrow \begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 1 & 3 & -6 \\ 0 & 0 & 1 & -\frac{16}{9} \end{pmatrix}$$

Now we can zero out the third column above that bottom entry, by adding (-3) times the third row to the second row, then adding (-1) times the third row to the first row.

$$\begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 1 & 3 & -6 \\ 0 & 0 & 1 & -\frac{16}{9} \end{pmatrix} \stackrel{\longleftarrow}{\smile}_{-3}^{+} \xrightarrow{-1} \rightsquigarrow \begin{pmatrix} 2 & 0 & 0 & \frac{16}{9} \\ 0 & 1 & 0 & -\frac{6}{9} \\ 0 & 0 & 1 & -\frac{16}{9} \end{pmatrix}$$

The top row can be scaled by $\frac{1}{2}$, and we finally have

$$\begin{pmatrix} 2 & 0 & 0 & \frac{16}{9} \\ 0 & 1 & 0 & -\frac{6}{9} \\ 0 & 0 & 1 & -\frac{16}{9} \end{pmatrix} \quad | \frac{1}{2} \quad \leadsto \begin{pmatrix} 1 & 0 & 0 & \frac{8}{9} \\ 0 & 1 & 0 & -\frac{6}{9} \\ 0 & 0 & 1 & -\frac{16}{9} \end{pmatrix}$$

The matrix is now in "reduced row echelon" form

We can also implement back substitution implicitly without actually creating a full reduced row echelon matrix

Example: Back Substitution Implementation

```
double x[N]; // An array to store solution
// Start calculating from last equation up to the first
for (int i = N-1; i >= 0; i--) {
   // Start with the RHS of the equation
   x[i] = mat[i][N];
   // Initialize column j to i+1 since matrix is upper triangular
   for (int j=i+1; j<N; j++)
       /* Subtract all the lhs values except the coefficient of the variable
          whose value is being calculated */
       x[i] -= mat[i][j]*x[j];
   // Divide the RHS by the coefficient of the unknown being calculated
   x[i] = x[i]/mat[i][i];
                                             What is the time complexity?
```