IIT Bombay

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: More Matrix Applications

Quick Recap

- We have seen the use of matrices for
  - Representing a system of N simultaneous linear equations
  - Using Gaussian elimination to solve the system

Overview

- In this session, we will see two more applications of matrices
  - Matrix multiplication
  - Magic Squares

Matrix Multiplication

IIT Bombay

A[3][4] x B[4][2] = C[3][2]

-	-	-	-		-	-	-	-
-	-	-	-		-	-	-	-
-	-	-	-		-	-	-	-
					-	-		

Calculating one element of the resultant matrix

IIT Bombay

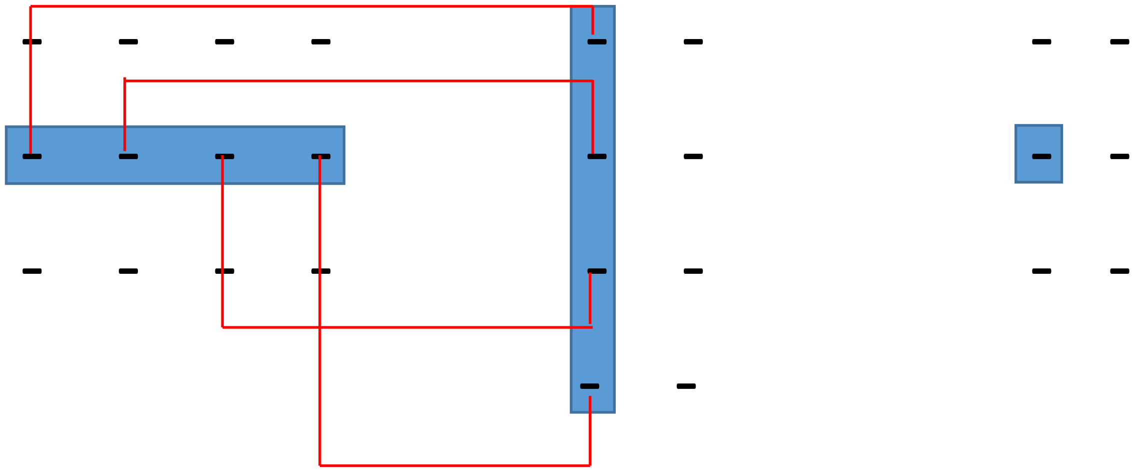
A[3][4] x B[4][2] = C[3][2]

-	-	-	-	-	-	-	-	
-	-	-	-	-	-		-	C[1][0]
-	-	-	-	-	-	-	-	
				-	-			

Calculating one element of the resultant matrix

IIT Bombay

A[3][4] x B[4][2] = C[3][2]



C[1][0] = A[1][0] x B[0][0] + A[1][1] x B[1][0] + A[1][2] x B[2][0] + A[1][3] x B[3][0]

Matrix Multiplication

IIT Bombay

Given: m x n matrix A; n x p matrix B,

The matrix product C = AB will be m × p matrix.

$$C[i][j] = \sum_{k=0}^{n-1} A[i][k] * B[k][j]$$

A segment of the program: matmult.cpp

IIT Bombay

```
// Read matrices A, and B
  for (i=0; i<m; i=i+1) {
    for (j=0; j<p; j=j+1) {
      C[i][j]=0;
      for (k=0; k<n; k=k+1) {
        C[i][j]=C[i][j]  +  A[i][k]*B[k][j];
      }
    }
  }
// Output result matrix C
```

Magic Squares

A magic square is an n x n square matrix containing unique positive integers, where the sum of elements of every row and every column is same

8	1	6
3	5	7
4	9	2

Magic Squares

IIT Bombay

- Additionally, sum of elements on each of the main diagonals is also the same as above

8	1	6	
3	5	7	→ 15
4	9	2	
			→ 15

- Known to Chinese (Lo Shu square)
- 3 x 3 magic squares were known to Indians since Vedic times

Determine if the given matrix is a normal magic square

IIT Bombay

- We note the following:
  - Any n x n matrix will have n rows, n columns and two diagonals

- To test a given matrix for being a magic square
  - First find out what should be the sum
    [sum for normal magic square = n * (n*n +1) /2]
  - Then, calculate sums of rows and columns
  - Calculate the sums for two diagonals

Determine if the given matrix is a normal magic square

IIT Bombay

- If any of these sums is not as desired
  - The given matrix is not magic square

- We will need arrays of size n for these sums
  - Calculate these sums
  - Now check if each of these sums is equal to the required sum

Program: magic.cpp

IIT Bombay

```
int main(){
   int square[20][20], N, i, j, sum;
   int  rsum[20], csum[20], d1sum =0, d2sum =0;

  /*  Read N*/
  cout << “ Give value of N ” << endl;
  cin >> N;
```

magic.cpp …

IIT Bombay

```
cout  << Give elements of the matrix" << endl;
for (i=0; i<N; i++) {
    for (j=0; j< N; j++) {
        cin >> square[i][j];
    }
}
for  (i=0; i < N; i++){
    rsum[i] = 0;
    jsum[i] = 0;
}
```

magic.cpp …

IIT Bombay

```
/* calculate the sum for this being N x N magic square */
 sum = N * (N*N +1) /2;

 /* find the row sums  and check against required sum*/
 for (i=0; i< N; i++){
       for (j=0; j < N; j++){
            rsum[i] += square[i][j];
       }
       if (rsum[i] != sum) {
          cout << " Not a magic square" << endl;   return 1;
       }
}
```

magic.cpp ...

IIT Bombay

```
/* find the column sums  and check against required sum*/
 for (j=0; j< N; j++){
       for (i=0; i < N; i++){
            csum[j] += square[i][j];
       }
       if (csum[j] != sum) {
          cout << " Not a magic square" << endl;   return 1;
       }
 }
```

magic.cpp ...

```
/* calculate the sums of two diagonals */
   for (i=0; i< N; i ++) d1sum += square[i][i];
   for (j=0; j < N; j ++) d2sum += square[j][N-j-1];
```

magic.cpp …

IIT Bombay

```
/* Now check if these diagonal sums are correct or not */
  if (d1sum != sum) {
        cout << " Not a magic square" << endl;   return 1;
  }
  if (d2sum != sum) {
        cout << " Not a magic square" << endl;   return 1;
  }
  /* If we reach this point, then the square is a magic square */

  cout << "Given matrix is a magic square" << endl;
  return 0;
```

Summary

IIT Bombay

- In this session,
  - We studied matrix multiplication, and wrote a C++ program
  - We discussed normal magic squares, and wrote a C++ program to test if a given square matrix represents a magic square or not

- These programs are available in the files
  - matmult.cpp
  - magic.cpp
- Download, compile, and run these with your sample data