CS110: Matrix operations in C

Lecture 10

V. Kamakoti

29th January 2008

Arrays

- Matrix is
  - A two dimensional array
  - Array of one dimensional arrays
- Stored in memory
  - Row-wise (Row Major order)
  - Column-wise (Column Major order)

Storage of matrix

- A - 2X3 matrix has elements
  - A[0][0],A[0][1],A[0][2];
  - A[1][0],A[1][1],A[1][2];
  - A[2][0],A[2][1],A[2][2];
- Assume it is an integer matrix and A[0][0] is stored at 1000

Row-major order

- Store the first row, then, the second, then, the third and so on…
- A - 2X3 matrix has elements
  - A[0][0],A[0][1],A[0][2]; stored at 1000-1011
  - A[1][0],A[1][1],A[1][2]; stored 1012 - 1023
  - A[2][0],A[2][1],A[2][2]; stored 1024 - 1035

Column-major order

- Store the first column, then, the second column, then, the third and so on..
- A - 2X3 matrix has elements
  - A[0][0],A[1][0],A[2][0]; stored at 1000-1011
  - A[0][1],A[1][1],A[2][1]; stored 1012 - 1023
  - A[0][2],A[1][2],A[2][2]; stored 1024 - 1035
- The C compiler assumes Row major order.

Lab - 3

- Write a program that reads in the entries of a 3 by 3 matrix, and prints it out in the form of a matrix. The entries could be floating point entries too.

Solution

```
#include<stdio.h>
main() {
      int i,j, a[3][3];
      for (i = 0; i <= 2; i++) {
        for (j = 0; j <=2; j++) {
          printf(“Enter a[%d][%d] \n”,i,j);
          scanf(“%d”,&a[i][j]);
        }
      }
}
```

What Happens

```
for (i = 0; i <= 2; i++) {
      for (j = 0; j <=2; j++)
      {
         printf("Enter a[%d][%d] \n",i,j);
         scanf("%d",&a[i][j]);
      }
}
```

For every i, 0 <= i <= 2, j repeats thrice taking values 0,1 and 2. Therefore when i = 0 and j = 0, a[0][0] shall be read in, when i = 0 and j = 1, a[0][1] shall be read in and so on ….

To Print

```
for (i = 0; i <= 2; i++) {
      for (j = 0; j <=2; j++)
      {
         printf("%d ",a[i][j]);
      }
      printf("\n");
}
```

Print all elements of a given row on the same line with a space in between. After one row is printed go to the next line.

You have to

- Repeat above for Floating point numbers
- The effort would be to
  - Format the output properly
  - Let p be a floating point number
  - printf(“%16f”,..) prints a floating point with 16 spaces - if less then blanks are introduced.
  - Use the tab escape sequence “\t” to align.
  - This would be true for integers too.

Next in Lab - 3

- Write a program that reads in orders of two matrices and decides whether two such matrices can be multiplied. Print out the decision.

```
main() {
  int row1,col1,row2,col2;
  printf("Enter number of rows and columns in
    Matrix 1 \n");
  scanf("%d %d",&row1,&col1);
  //Similarly for Matrix 2
 if (col1 != row2) { printf("Not possible\n"); exit(0);
}  // To use exit() we need to include <stdlib.h>
```

Next in Lab - 3

- Write a program that reads in two arbitrary matrices and multiplies them. Your output should be the two matrices and the resulting product matrix.

Matrix Multiplication : Outline

main() {

1. declare all variables required
2. read in Matrix A
3. read in Matrix B
4. check if A and B are compatible to be multiplied
5. initialize Matrix C to have zeroes to begin with
6. multiply A and B to give C
7. print Matrix C

}

Of course, all the steps above have to follow C language's syntax rules

Using Matrix Operations : Read a Matrix

```
main(){
   int a[11][11], b[11][11], c[11][11]; / *max size 10 by 10 */
   int i,j,k;
    int aRows, aCols, bRows, bCols, cRows, cCols;


   scanf("%d%d", &aRows, &aCols);
   for(int i = 1; i <= aRows; i++)
       for(int j = 1; j <= aCols; j++)
           scanf("%d", &a[i][j]);

 /*continued on next slide */
```

Read the other Matrix; Initialize the product

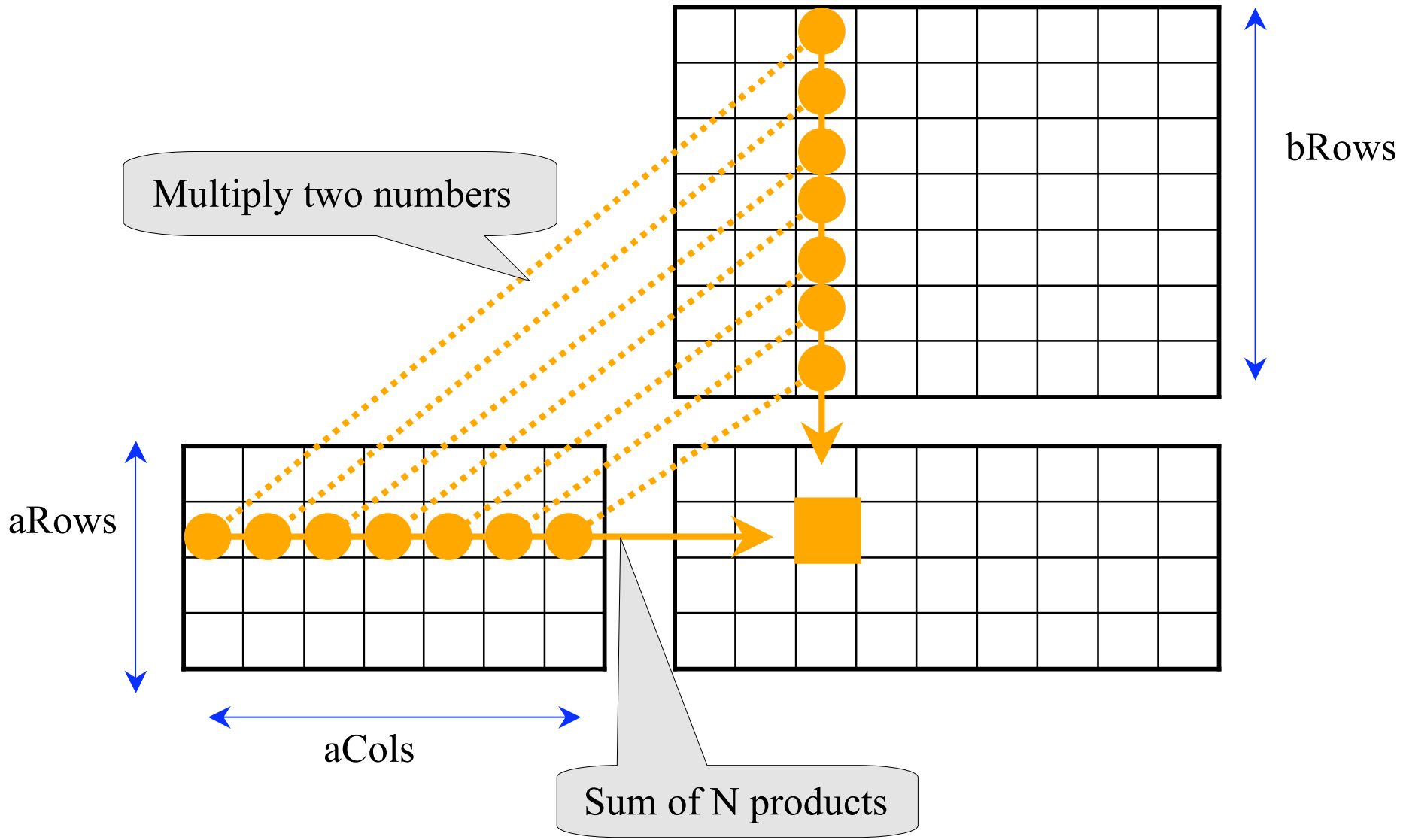
```
scanf("%d%d", &bRows, &bCols);
 for(i = 1; i <= bRows; i++)
     for(j = 1; j <= bCols; j++)
         scanf("%d", &b[i][j]);


/* initialize entries in Matrix c to 0 */
for(i = 1; i <= aRows; i++)
      for(j = 1; j <= bCols; j++)
           c[i][j] = 0;

/*continued on next slide */
```

Remember
bRows must equal aCols
c is a aRows x bCols matrix

Matrix multiplication



Multiply Matrices and Print the Result

```
/* multiply both the matrices and store in matrix c */
for(i =1; i <= aRows; i++)
      for(j = 1; j <= bCols; j++)
          for(k = 1; k <=  aCols; k++)
               c[i][j] += a[i][k]*b[k][j];


/* print matrix c */
for(i = 1; i <= aRows; i++){
       for(j = 1; j <= bCols;  j++)    /* print a row */
           printf("%d  ", c[i][j]);    /* notice missing \n */
       printf("\n");                      /* print a newline at the end a row */
    }
}                                       /* End of main program */
```

Points to Ponder

- Some repetition in the program
  - Reading in matrix A seems to be similar to reading in matrix B
    - Except for changes in the number of rows and columns
- You will learn how to write functions in C later
  - Functions are written to avoid repeated actions
  - Example C program would look like

```
readMat(A, aRows, aCols);
readMat(B, bRows, bCols);
```

  - Function readMat( ) must perform the operations desired

Celebrity Problem

(More efficient Solution)

There are n persons in a room. A person is called a celebrity iff everyone knows him but he does not know anyone. So by definition if at all there is a celebrity there can be only one. The problem is to find if there is a celebrity among the people in the room and if there is one identify him, by asking questions. The only kind of question allowed is “Does A know B?”.

Celebrity Problem

(Naivest Solution)

'Naivest' Solution:

For each pair of persons A and B, ask the questions, does A know B and does B know A. Build the nxn matrix $\{a_{ij}\}$. is 1 if the ith person knows the jth person and it is 0 otherwise. Now if there is a column i, which contains all 1s, and if row i contains all 0s except the diagonal, then the person i is a celebrity. If there is no such i, then there is no celebrity.

Number of questions asked = n*(n-1)

Celebrity Problem

(Better Solution)

Choose two persons A and B. Ask if A knows B. If answer is yes, then A cannot be the celebrity. If answer is no, then B cannot be the celebrity. So one person gets eliminated. Now if at all there is a celebrity he will be in the remaining n-1 persons. Keep eliminating in this fashion. After n-1 questions, only one person X, will be left. For each person Y ≠ X, ask if Y knows X and X knows Y. Hence we can find if X is actually a celebrity.

Number of questions asked = n-1 + 2n-2 = 3n-3

Celebrity Problem

(Best Solution)

Simplifying assumption: n = 2^k

Group the persons into pairs. So we will have n/2 pairs. In each pair (A, B) eliminate one person by asking if A knows B. Now we will be left with n/2 persons. Repeat the process till we are left with only one person.

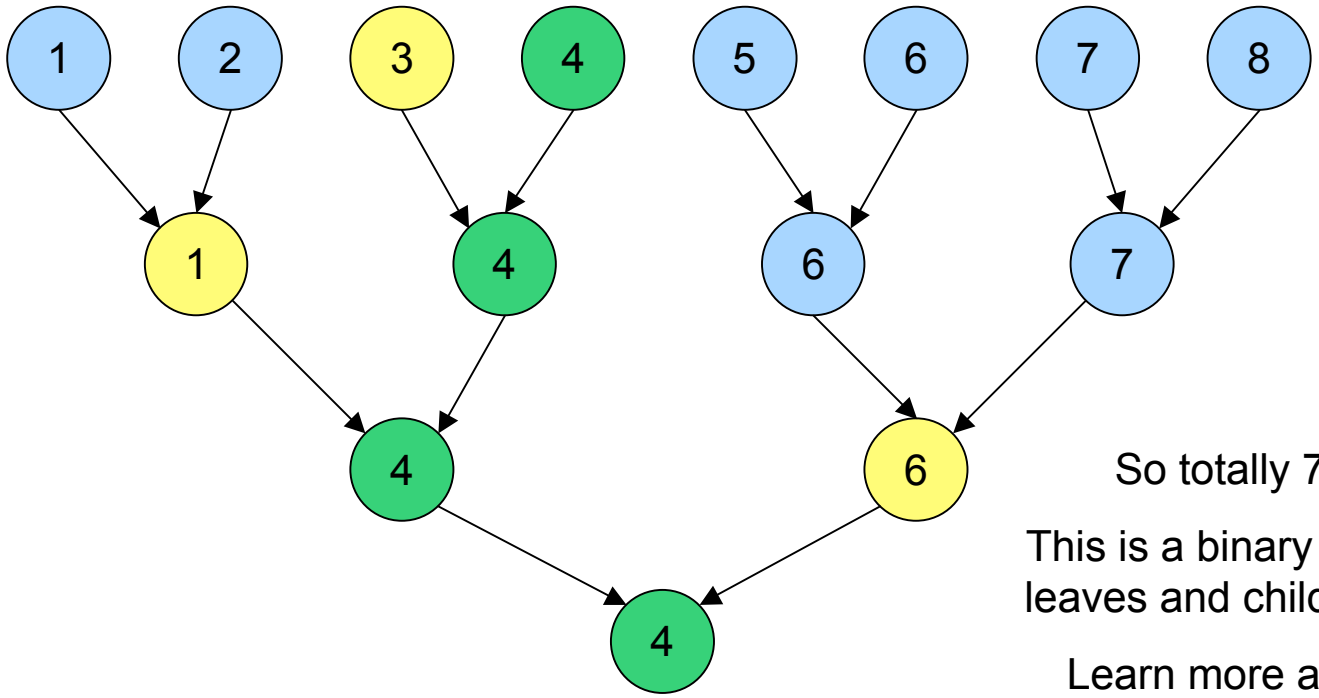
Number of times this process in repeated is obviously k, since each time the number reduces by half.

(cont.)

Celebrity Problem

(Best Solution)

We can keep track of the tree thus obtained. Let us take a case having 8 persons. The tree looks likes something below.



So totally 7 Qs asked.

This is a binary tree with a root, leaves and children (at most 2).

Learn more about this later

Generalizing above

- Given N people
  - N/2 Qs in level 1
  - N/4 Qs in level 2
  - 1 Question in level k, where, $N = 2^k$
  - So total is $N(1/2 + 1/4 + \ldots 1/2^k)$
  - $N\ (1/2)\ (\ 1 - 1/2^k)/(1/2)$
  - $N - N/2^k = N - 1$

Celebrity Problem

(Best Solution)

Clearly, if at all there is a celebrity, it can be only 4. Also note that out of the 2n-2(= 6) questions asked to confirm whether 4 is actually a celebrity, we have already asked $\log_2 n$ (= 3) questions (Notice the yellow circles). So once we maintain this tree, the number of additional questions that we need to ask is $2n - 2 - \log_2 n$.

Therefore the total number of questions asked in this case is $n-1 + 2n - 2 - \log_2 n = 3n - 3 - \log_2 n$.

There exists no algorithm which can do better than this, the proof of which is beyond the scope of this lecture. Extending it for n not a power of 2 is also interesting.

Thank You