af2

The Queen Mother says

"Please turn your phone off"

Matrices/Arrays

Section 3.8

Matrices

- A matrix is a rectangular array, possibly of numbers
- it has m rows and n columns
- if m = n then the matrix is square
- two matrices are equal if
  - they have same number of rows
  - they have same number of columns
  - corresponding entries are equal

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \end{bmatrix}$$

A is an m by n array

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ . & . & \ldots & \\ a_{m,1} & a_{m,2} & \ldots & a_{m,n} \end{bmatrix}.$$

The ith row of A is a 1 by n matrix (a vector)

$$[a_{i,1}, a_{i,2}, \cdots, a_{i,n}]$$

The jth column of A is a m by 1 matrix (a vector)

$$\begin{bmatrix} a_{1,j} \\ a_{2,j} \\ . \\ . \\ . \\ a_{m,j} \end{bmatrix}$$

A is an m by n array

We also sometimes say that A is an array using the following shorthand

$$A = [a_{ij}]$$

i.e. A is an array with its (i,j)th element equal to a_{ij}

addition

- To add two arrays/matrices A and B
  - they must both have the same number of rows
  - they must both have same number of columns

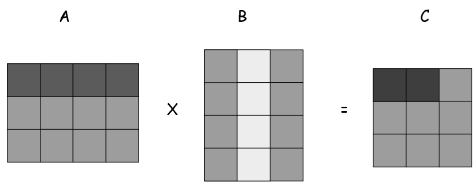
$$C_{i,j} = a_{i,j} + b_{i,j}$$

C = A + B

- for i := 1 to n do
  - for j := 1 to m do
    - C[i][j] := A[i][j] + B[i][j]

- *How many array references are performed?*
- *How is an array reference made? What is involved?*
- *How many additions are performed?*
- *Would it matter if the loops were the other way around?*

Multiplication



Note: A is m × k, B is k × n, and C is m × n

- When
  - A is m × k
  - B is k × n
  - C = A.B
  - C is m × n
    - to compute an element of C

$$C_{i,j} = a_{i,1}.b_{1,j} + a_{i,2}.b_{2,j} + ... + a_{i,k}.b_{k,j}$$

$$C_{i,j} = \sum_{x=1}^{k} a_{i,x} b_{x,j}$$

- for i = 1 to m do
  - for j = 1 to n do
    - C[i][j] := 0
    - for x = 1 to k do
      - C[i][j] := C[i][j] + A[i][x] * B[x][j]

- *Could we make it more efficient?*
- *How many array access do we do?*
- *How many multiplications and divisions are performed?*
- *What is the complexity of array multiplication?*
- *Is the ordering of the loops significant?*

Do an example on the blackboard!

$$A = \begin{bmatrix} 1 & 0 & 4 \\ 2 & 1 & 1 \\ 3 & 1 & 0 \\ 0 & 2 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 2 & 4 \\ 1 & 1 \\ 3 & 0 \end{bmatrix} \quad C = AB$$

Is multiplication commutative?

- *Does A x B = B x A?*
  - *It might not be defined!*
  - *Can you show that A x B might not be B x A?*

Show that A x B might not be same as B x A!

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\text{Does} \quad AB = BA \, ?$$

Multiplication is associative

(AB)C = A(BC)

Does it matter?

- Assume
  - A is 30 x 20
  - B is 20 x 40
  - C is 40 x 10

- BC takes 20 x 10 x 40 operations = 8,000
- the result array is 20 x 10
  - call it R
- AR takes 30 x 10 x 20 operations = 6,000
- A(BC)takes 14,000 operations (mults)

- AB takes 30 x 40 x 20 operations = 24,000
- the result array is 30 x 40
  - call it R
- RC takes 30 x 40 x 10 operations = 12,000
- (AB)C takes 36,000 operations (mults)

Identity Matrix

$$I_n$$

$$I_4 = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$I_n = [\delta_{ij}]$$
$$i = j \leftrightarrow \delta_{ij} = 1$$
$$i \neq j \leftrightarrow \delta_{ij} = 0$$

$$A.I_n = A = I_n.A$$

No change!

Raising a Matrix to a power

$$A^0 = I$$
$$A^1 = A$$
$$A^2 = A.A$$
$$A^3 = A.A.A$$
$$\vdots$$
$$A^r = \underbrace{A.A.A.\cdots.A}_{r \quad times}$$

Transpose of a Matrix

Interchange rows with columns

$$A = [a_{ij}]$$
$$A^t = [b_{ij}] \ \ where \ \ b_{ij} = a_{ji} \wedge 1 \le i \le n \wedge 1 \le j \le m$$

$$X^t = \begin{matrix} a & c \\ b & e \\ c & f \end{matrix} \qquad X = \begin{matrix} a & b & c \\ d & e & f \end{matrix}$$

Symmetric Matrix

$$A = A^t$$

$$a_{ij} = a_{ji}$$

Must be square

2	3	5	7
3	9	6	4
5	6	1	8
7	4	8	3

Think of distance

Zero-One Matrices/arrays

- 1 *might* be considered as true
- 0 *might* be considered as false
- the array/matrix *might* then be considered as
  - a relation
  - a graph
  - whatever

Join of A and B (OR)

$$C = A \vee B$$

$$c_{i,j} = a_{i,j} \vee b_{i,j}$$

So, it is *like* addition

join

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

- Isn't this like add?
  - Just replace x + y with max(x,y)?

join

```
• for i := 1 to n do
    • for j := 1 to m do
    • C[i][j] := max(A[i][j],B[i][j])
```

meets of A and B (AND)

$$C = A \wedge B$$

$$c_{i,j} = a_{i,j} \wedge b_{i,j}$$

So, it is like addition too?

meets

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

- Isn't this like add?
  - Just replace x + y with min(x,y)?

meets

- for i := 1 to n do
  - for j := 1 to m do
  - C[i][j] := min(A[i][j],B[i][j])

Boolean product

$$c_{i,j} = (a_{i,1} \wedge b_{1,j}) \vee (a_{i,2} \wedge b_{2,j}) \vee ... \vee (a_{i,k} \wedge b_{k,j})$$

- Like multiplication but
  - replace times with and
  - replace plus with or
- The symbol is an O with a dot in the middle

Boolean product

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

$$A \otimes B = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Boolean product

- for i = 1 to m do
  - for j = 1 to n do
    - C[i][j] := 0
    - for x = 1 to k do
      - C[i][j] := C[i][j] OR A[i][x] AND B[x][j]

Can we make this more efficient?

Raising a 0/1 array to a power

The r^{th} boolean power

$$A^{[r]} = \underbrace{A \otimes A \otimes A \otimes \cdots \otimes A}_{r\ \ times}$$

applications

- matrices
  - tables
  - weight and height
  - distance from a to b (and back again?)
- zero-one matrices
  - adjacency in a graph
  - relations
  - constraints

- Imagine matrix A
  - each row is an airline
  - column is flight numbers
  - A[i][j] gives us jth flight number for ith airline
- Imagine matrix B
  - each row is a flight number
  - each column is departure time
- join(A,B) gives
  - for each airline the departure times
- in constraint programming a zero-one matrix is a constraint
  - boolean product gives us path consistency