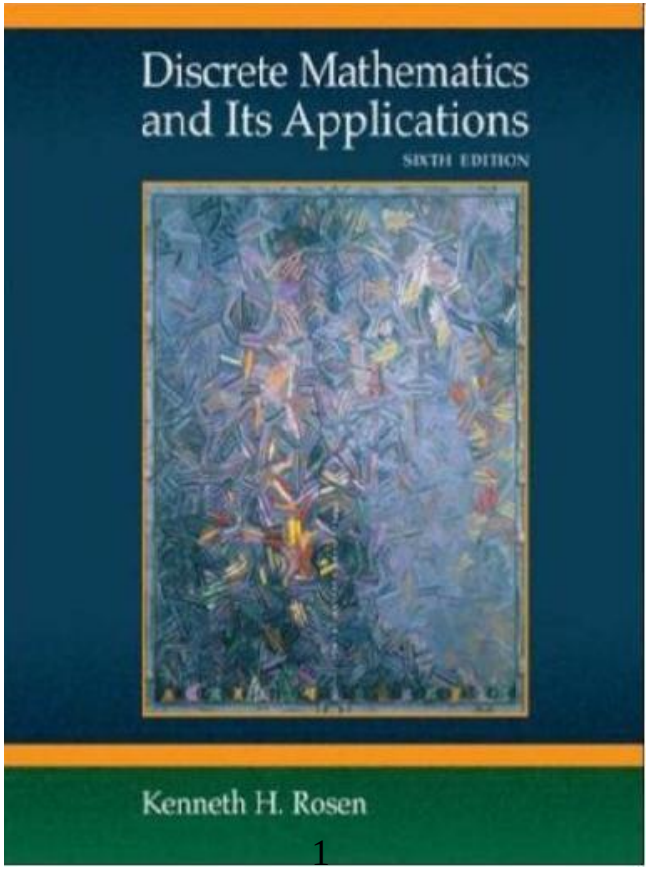
Jiangxi University of Science and Technology

Discrete Mathematics and Its Applications



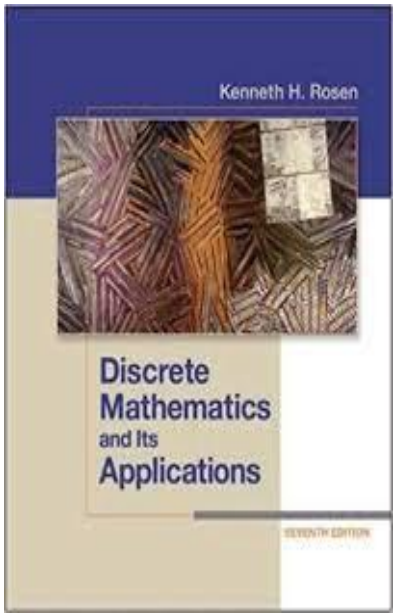
Lecture013: CHP3:

The Fundamentals: Algorithms, the Integers, and Matrices

Acknowledgement

Most of these slides are adapted from ones created by Professor Bart Selman at Cornell University ,

and Dr Johnnie Baker and **Discrete Mathematics and Its Applications** (Seventh Edition) **Kenneth H. Rosen**



3.1 Algorithms Introduction

- Algorithm: a procedure that follows a sequence of steps that leads to the desired answer.

DEFINITION 1:

An *algorithm* is a finite set of precise instructions for performing a computation or for solving a problem.

Example:

Example: Describe an algorithm for finding the maximum (largest) value in a finite sequence of integers.

Solution:

1. Set the temporary maximum equal to the first integer in the sequence.
(The temporary maximum will be the largest integer examined at any stage of the procedure.)

2. Compare the next integer in the sequence to the temporary maximum, and if it is larger than the temporary maximum,set the temporary maximum equal to this integer.

3. Repeat the previous step if there are more integers in the sequence.

4. Stop when there are no integers in the sequence. The temporary maximum at this point is the largest integer in the sequence.

3.1 Algorithms

- An algorithm can also be described using pseudocode, an intermediate step between an English language description of an algorithm and an implementation of this algorithm in a programming language.
- Having a pseudocode description of the algorithm is the starting point of writing a computer program.

ALGORITHM 1: Finding the Maximum Element in a Finite Sequence.

procedure $max(a_1, a_2, \ldots, a_n$: integers)

$max := a_1$

for $i := 2$ to n

    if $max < a_i$, then $max := a_i$

{max is the largest element}

3.1 Algorithms

- Several properties algorithms generally share:

- Input – **An algorithm has input values from a specified set.**
- Output – **From each set of input values an algorithm produces output values from a specified set. The output values are the solution to the problems.**
- Definiteness – **The steps of an algorithm must be defined precisely.**
- Correctness – **An algorithm should produce the correct output values for each set of input values.**
- Finiteness – **An algorithm should produce the desired output after a finite (but perhaps large) number of steps for any input in the set.**
- Effectiveness – **It must be possible to perform each step of an algorithm exactly and in a finite amount of time.**
- Generality – **The procedure should be applicable for all problems of the desired form, not just for a particular set of input values.**

3.1 Algorithms Searching Algorithms

- Locating an element x in a list of distinct elements $a_1, a_2, \ldots, a_n$, or determine that it is not in the list.
- E.g. search for a word in a dictionary
  find a student's ID in a database table
  determine if a substring appears in a string
- The linear search (sequential search)
  - Begins by comparing x and a_1. When $x = a_1$, the solution is the location of a_1, namely, 1. When $x \neq a_1$, compare x with a_2. If $x = a_2$, the solution is the location of a_2, namely, 2. When $x \neq a_2$, compare x with a_3. Continue this process, comparing x successively with each term of the list until a match is found, where the solution is the location of the term. If the entire list has been searched without locating x, the solution is 0.

ALGORITHM 2: The Linear Search Algorithm.

procedure *linear search*(*x*: integer, $a_1, a_2, \ldots, a_n$: distinct integers)

***i*: = 1**

While (*i*: <= *n* and $x \neq a_i$)

    ***i* := *i* + 1**

If *i* <= *n* then *location* : = *i*

else *location* := 0

{*location* is the subscript of the term that equals *x*, or is 0 if *x* is not found}

3.1 Algorithms

- The binary search
  - The algorithm is used when the list has terms occuring in order of increasing size **(e.g. smallest to largest for numbers; alphabetic order for words)**.
  - It proceeds by comparing the element to be located to the middle terms of the list. The list is then split into two smaller sublists of the same size, or where one of these smaller lists has one fewer term than the other.
  - The search continues by restricting the search to the appropriate sublist based on the comparison of the element to be located and the middle term.

Example: To search for 19 in the list

Example: To search for 19 in the list
1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

split the list into half –
1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
Compare 19 with the largest term in the first list,
10 < 19,
split the second half –
12 13 15 16 18 19 20 22
16 < 19, **split the second half –**
18 19 20 22
19 !< 19, **split the first half –**
18 19
18 < 19, use the second half,
only one element, compare, get the location

3.1 Algorithms

ALGORITHM 3: The Binary Search Algorithm.

procedure *binary search* (x: integer, a_1, a_2, …, a_n: increasing integers)

i: = 1 {*i* is left endpoint of search interval}

j:= n {*j* is right endpoint of search interval}

while $i < j$

begin

$m := \lfloor (i + j)/2 \rfloor$

If $x > a_m$ then $i := m + 1$

else $j := m$

end

if $x = a_i$ then *location*: = *i*

else *location*: = 0

{*location* is the subscript of the term that equals *x*, or is 0 if *x* is not found}

3.1 Algorithms Sorting

- **Putting the elements into a list in which the elements are in increasing order**

- The Bubble Sort
  - **It puts a list into increasing order by successively comparing adjacent elements, interchanging them if they are in the wrong order. The smaller elements "bubble" to the top as they are interchanged with larger elements and the larger elements "sink" to the bottom.**

Example: Use the bubble sort to put 3,2,4,1,5 into increasing order.

First pass

3	2	2	2	2	2
2	3	3	3	3	1
4	4	4	1	1	3
1	1	1	4	4	4
5	5	5	5	5	5

second pass

Third pass

2	1
1	2
3	3
4	4
5	5

Fourth Pass

1
2
3
4
5

3.1 Algorithms

ALGORITHM 4: The Bubble Sort.

procedure *bubblesort*(a_1, a_2, ..., a_n: real numbers with n >=2)

for i: = 1 to n – 1

    for j: = 1 to n - i

        if $a_j > a_{j+1}$ *then interchange* a_j *and* a_{j+1}

{a_1, a_2, ..., a_n is in increasing order}

3.1 Algorithms: The Insertion Sort

- To sort a list with *n* elements, the insertion sort begins with the second element. The insertion sort compare this second element with the first element and inserts it before the first element if it does not exceed the first element and after the first element if it exceeds the first element. At this point, the first two elements are in the correct order. The third element is then compared with the first element, and if it is larger than the first element, it is compared with the second element; it is then inserted into the correct position among the first three elements.
- In general, in the *j*th step of the insertion sort, the *j*th element of the list is inserted into the correct position in the list of the previously sorted *j* -1 elements.
- Example: 3, 2, 1, 4, 5

3.1 Algorithms

ALGORITHM 5: The Insertion Sort.

procedure *insertion sort*(a_1, a_2, ..., a_n: real numbers with n >=2)

for j: = 2 to n

begin

i:= 1

while $a_j > a_i$

$i := i + 1$

$m := a_j$

for k: = 0 to $j - i - 1$

$a_{j-k} = a_{j-k-1}$

$a_i := m$

end {a_1, a_2, ..., a_n are sorted}

3.6 Integers and Algorithms Representation of Integers

- Integers can be represented in decimal, binary, octal(base 8) or hexadecimal(base 16) notations.

THEOREM 1:

Let *b* be a positive integer greater than 1. Then if *n* is a positive integer, it can be expressed uniquely in the form

$$n = a_k b^k + a_{k-1} b^{k-1} + \ldots + a_1 b + a_0$$

where *k* is a nonnegative integer, $a_0, a_1, \ldots, a_k$ are nonnegative integers less than *b*, and $a_k \neq 0$.

- Example: $(965)_{10} = 9 * 10^2 + 6*10^1 + 5*10^0$

$$(1\ 0101\ 1111)_2 = 1*2^8 + 0*2^7 + 1*2^6 + \ldots + 1*2^1 + 1*2^0 = (351)_{10}$$

Integers and Algorithms 3.6

- Hexadecimal digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
  (16 digits or letters representing 0 – 15)

Example:

$(2AE0B)_{16} = 2*16^4 + 10*16^3 + 14*16^2 + 0*16^1 + 11*16^0 = (175627)_{10}$

Convert an integer from base *10* to base *b*:

Divide *n* by *b* to obtain a quotient and a remainder. The remainder is the rightmost digit in the result. Keep dividing the quotient, and write down the remainder, until obtaining a quotient equal to zero.

Example: convert $(12345)_{10}$ to base 8:

$12345 = 8*1543 + 1$
$1543 = 8*192 + 7$
$192 = 8*24 + 0$
$24 = 8*3 + 0$
$3 = 8*0 + 3$

$(12345)_{10} = (30071)_8$

Integers and Algorithms 3.6

- Example:

  Find the hexadecimal expansion of $(177130)_{10}$:

  177130 = 16*11070 + 10

  11070 = 16*691 + 14

  691 = 16*43 + 3

  43 = 16*2 + 11

  2 = 16*0 + 2

  $(177130)_{10} = (2B3EA)_{16}$

  Find the binary expansion of (241)10:

3.6 Integers and Algorithms

Algorithms for Integer Operations

- Example: Add $a = (1110)_2$ and $b = (1011)_2$

```
      1 1 1 0
      1 0 1 1
  --------------
    1 1 0 0 1
```

- Algorithm:

  The binary expansion of a and b are:

  $a = (a_{n-1}a_{n-2}\ldots a_1a_0)_2$ $\quad$ $b = (b_{n-1}b_{n-2}\ldots b_1b_0)_2$

  a and b each have n bits (putting bits equal to 0 at the beginning of one of these expansions if necessary).

  To add a and b, first add their rightmost bits. This gives

  $a_0 + b_0 = c_0*2 + s_0$ where s_0 is the rightmost bit and c_0 is the carry.

  Then add the next pair of bits and the carry,

  $a_1 + b_1 + c_0 = c_1*2 + s_1$ where s_1 is the next bit from the right.

  Continue the process. At the last stage, add a_{n-1}, b_{n-1} and c_{n-2} to obtain $c_{n-1}*2 + s_{n-1}$. The leading bit of the sum is $s_n=c_{n-1}$.

3.6 Integers and Algorithms

- Example: Following the procedure specified in the algorithm to add $a = (1110)_2$ and $b = (1011)_2$

$a_0 + b_0 = 0 + 1 = 0*2 + 1$ $(c_0 = 0, s_0 = 1)$

$a_1 + b_1 + c_0 = 1 + 1 + 0 = 1*2 + 0$ $(c_1=1, s_1 = 0)$

$a_2 + b_2 + c_1 = 1 + 0 + 1 = 1*2 + 0$ $(c_2=1, s_2 = 0)$

$a_3 + b_3 + c_2 = 1 + 1 + 1 = 1*2 + 1$ $(c_3=1, s_3 = 1)$

$(s_4= c_3=1)$

result: 11001

3.6 Integers and Algorithms

- Example: Find the product of $a = (110)_2$ and $b = (101)_2$

```
          1 1 0
          1 0 1
    ------------------
          1 1 0
        0 0 0
      1 1 0
    ------------------
      1 1 1 1 0
```

- Algorithm:

  $ab = a(b_0 2^0 + b_1 2^1 + \ldots + b_{n-1} 2^{n-1})$

  $= a(b_0 2^0) + a(b_1 2^1) + a(b_{n-1} 2^{n-1})$

  Each time we mulply a term by 2, we shift its binary expansion one place to the left and add a zero at the tail end. Consequently we obtain $(ab_j)2^j$ by shifting the binary expansion of ab_j j places to the left, adding j zero bits at the tail end. Finally we obtain ab by adding the n integers $ab_j 2^j$, $j = 0,1,2,\ldots, n\text{-}1$

3.6 Integers and Algorithms

- Example: Find the product of $a = (110)_2$ and $b = (101)_2$

  $ab_0 2^0 = (110)_2 * 1 * 2^0 = (110)_2$

  $ab_1 2^1 = (110)_2 * 0 * 2^1 = (0000)_2$

  $ab_2 2^2 = (110)_2 * 1 * 2^2 = (11000)_2$

  $ab = (110)_2 + (0000)_2 + (11000)_2 = (11110)_2$

3.8 Matrices Introduction

DEFINITION 1:
A *matrix* is a rectangular array of numbers. A matrix with *m* rows and *n* columns is called an *m* x *n* matrix. The plural of matrix is *matrices*. A matrix with the same number of rows as columns is called *square*. Two matrices are equal if they have the same number of rows and the same number of columns and the corresponding entries in every position are equal.

- Example: The matrix $\begin{bmatrix} 1 & 1 \\ 0 & 2 \\ 1 & 3 \end{bmatrix}$ is a 3 x 2 matrix.

Matrices 3.8

DEFINITION 2:

Let

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

The *i*th *row* of A is the 1 x *n* matrix $[a_{i1}, a_{i2}, \dots, a_{in}]$. The *j*th *column* of A is then *n* x 1 matrix

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

The (*i*,*j*)th *element* or *entry* of A is the element a_{ij}, that is, the number in the *i*th row and *j*th column of A.

A convenient shorthand notation for expressing the matrix A is to write $A = [a_{ij}]$, which indicates that A is the matrix with its (*i*,*j*)th element equal to a_{ij}.

3.8 Matrices Matrix Arithmetic

DEFINITION 3:

Let A = [a_{ij}] and B = [b_{ij}] be m x n matrices. The *sum* of A and B, denoted by A + B, is the m x n matrix that has $a_{ij} + b_{ij}$ as its (i,j)th element. In other words, A + B = [$a_{ij} + b_{ij}$].

- The sum of two matrices of the same size is obtained by adding elements in the corresponding positions.
- Matrices of different sizes cannot be added

Example:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 2 & -3 \\ 3 & 4 & 0 \end{bmatrix} + \begin{bmatrix} 3 & 4 & -1 \\ 1 & -3 & 0 \\ -1 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 & -2 \\ 3 & -1 & -3 \\ 2 & 5 & 2 \end{bmatrix}$$

3.8 Matrices

DEFINITION 4:
Let A be an m x k matrix and B be a k x n matrix. The product of A and B, denoted by AB, is the m x n matrix with its (i,j)th entry equal to the sum of the products of the corresponding elements from the ith row of A and the jth column of B. In other words, if AB = [cij], then

$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{ik}b_{kj}$.

- The product of the two matrices is not defined when the number of columns in the first matrix and the number of rows in the second matrix is not the same.
- Example:

Let $A = \begin{bmatrix} 1 & 0 & 4 \\ 2 & 1 & 1 \\ 3 & 1 & 0 \\ 0 & 2 & 2 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & 4 \\ 1 & 1 \\ 3 & 0 \end{bmatrix}$

Find AB if it is defined. $AB = \begin{bmatrix} 14 & 4 \\ 8 & 9 \\ 7 & 13 \\ 8 & 2 \end{bmatrix}$

3.8 Matrices

- If A and B are two matrices, it is not necessarily true that AB and BA are the same. E.g. if A is 2 x 3 and B is 3 x 4, then AB is defined and is 2 x 4, but BA is not defined.
- Even when A and B are both *n* x *n* matrices, AB and BA are not necessarily equal.
- Example:

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

Does AB = BA?

Solution:

$AB = \begin{bmatrix} 3 & 2 \\ 5 & 3 \end{bmatrix}$ and $BA = \begin{bmatrix} 4 & 3 \\ 3 & 2 \end{bmatrix}$

3.8 Matrices

ALGORITHM 1: Matrix Multiplication.

procedure *matrix multiplication*(A, B: matrices)

for *i: = 1 to m*

    for *j:* = 1 to *n*

    begin

        $c_{ij} := 0$

        for *q:= 1 to k*

            $c_{ij} := c_{ij} + a_{iq}b_{qj}$

    end

{C=[c_{ij}] is the product of A and B}

3.8 Matrices: Transposes and Powers of Matrices

DEFINITION 5:

The identity matrix of order n is the n x n matrix $I_n = [\delta_{ij}]$, where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$. Hence

$$I_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

- Multiplying a matrix by an appropriately sized identity matrix does not change this matrix. In other words, when A is an m x n matrix, we have

  $AI_n = I_m A = A$
- Powers of square matrices can be defined. When A is an n x n matrix, we have
- $A^0 = I_n$, $A^r = AAAA\dots A$ (r times)

3.8 Matrices

DEFINITION 6:
Let A = [a_{ij}] be an m x n matrix. The transpose of A, denoted by A^t, is the n x m matrix obtained by interchanging the rows and columns of A. In other words, if $A^t = [b_{ij}]$, then $b_{ij} = a_{ji}$, for $i = 1,2,\ldots,n$ and $j = 1,2,\ldots,m$.

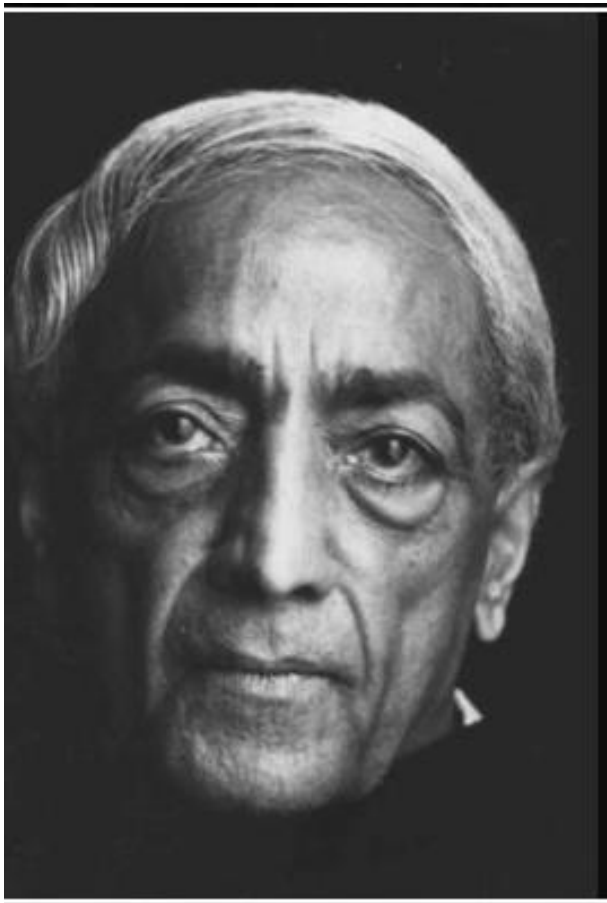
- Example:
  The transpose of the matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ is the matrix $\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$

3.8 Matrices

DEFINITION 7:

A square matrix A is called *symmetric* is $A = A^t$. Thus $A = [a_{ij}]$ is symmetric if $a_{ij} = a_{ji}$ for all i and j with $1 <= i <= n$ and $1 <= j <= n$.

- Example:
  The matrix $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ is symmetric.



There is no end to education. It is not that you read a book, pass an examination, and finish with education. The whole of life, from the moment you are born to the moment you die, is a process of learning.

— Jiddu Krishnamurti —

AZ QUOTES