جامعة بنها

كلية الحاسبات والذكاء الإصطناعي

Discrete Mathematics

Lecture 04

Dr. Ahmed Hagag

Faculty of Computers and Artificial Intelligence
Benha University

Spring 2023

Announcement

Quiz (1)

In Lecture 5

12/3/2023

Covers: Lec 1, 2, and 3

Chapter 2: Basic Structures

- Sets.
- Functions.
- Sequences, and Summations.
- Matrices.

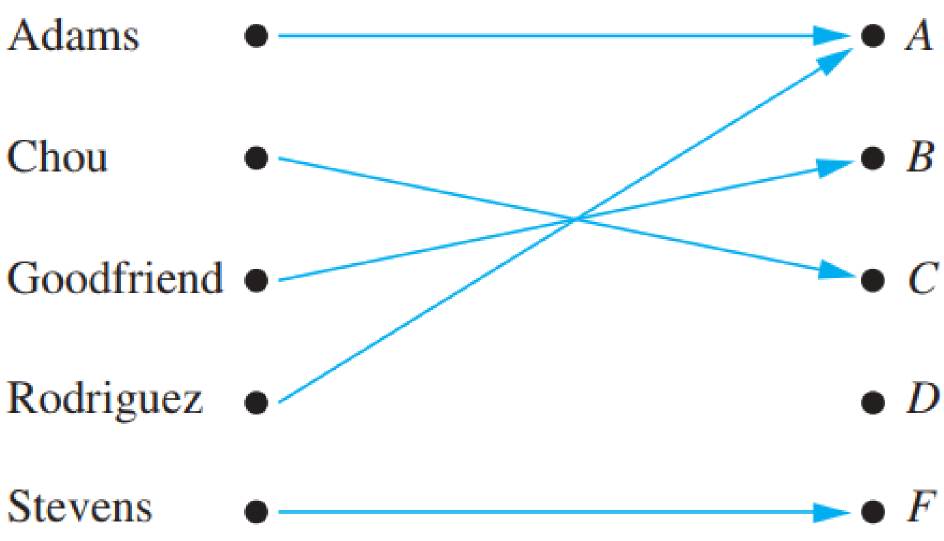
Function

Let A and B be nonempty sets. A function f from A to B is an assignment of exactly one element of B to each element of A.

We write $f(a) = b$ if b is the unique element of B assigned by the function f to the element a of A.

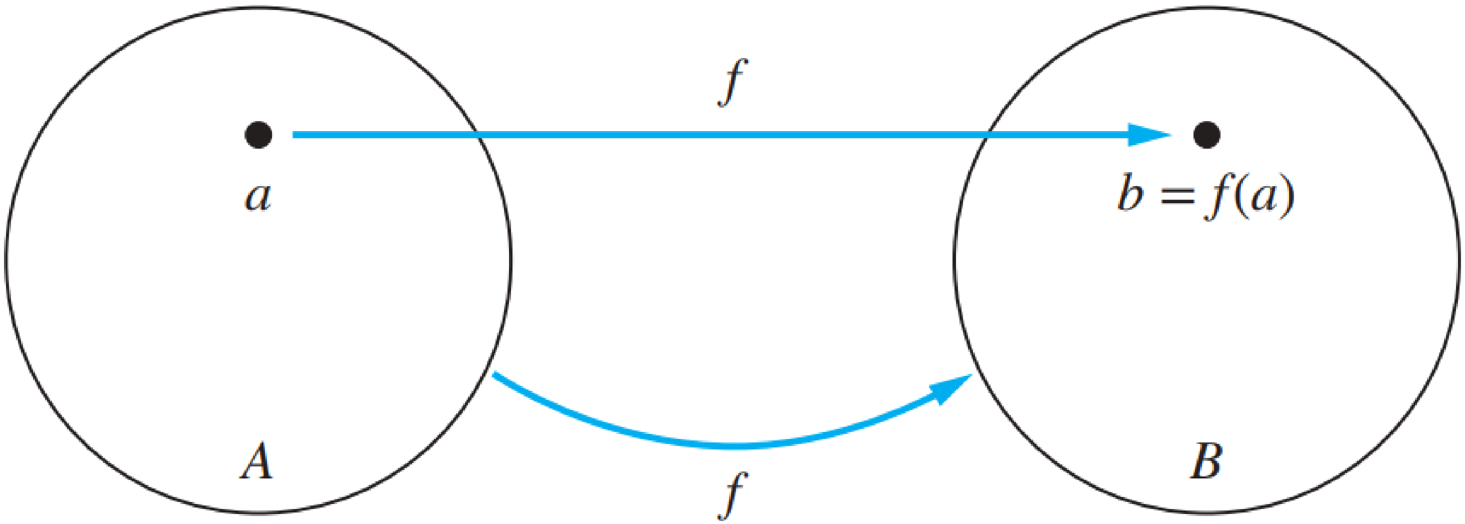
If f is a function from A to B, we write $f: A \rightarrow B$.

Function



Assignment of grades in a discrete mathematics class.

The Function $f: A \rightarrow B$



The function f maps A to B.

The Function $f: A \to B$

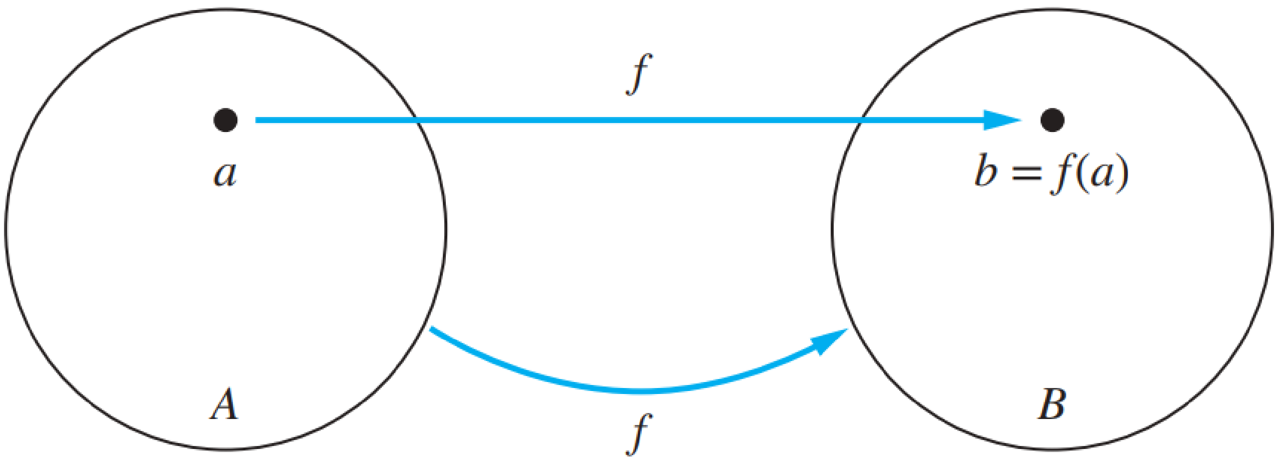
Domain: A

Co-Domain: B

$f(a) = b$

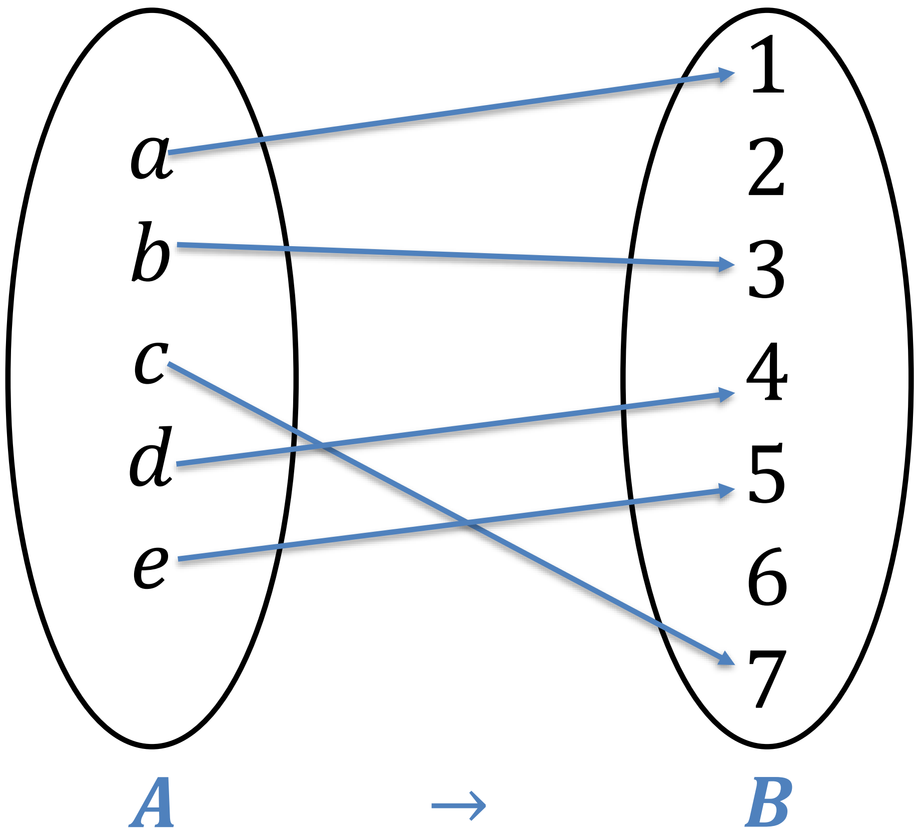
b is the ***image*** of a
a is a ***preimage*** of b

The **range**, or image, of f is the ***set of all images*** of elements of A.



The function f maps A to B.

The Function $f: A \to B$



$\text{Domain} = \{a, b, c, d, e\}$

$\text{Co-Domain} = \{1,2,3,4,5,6,7\}$

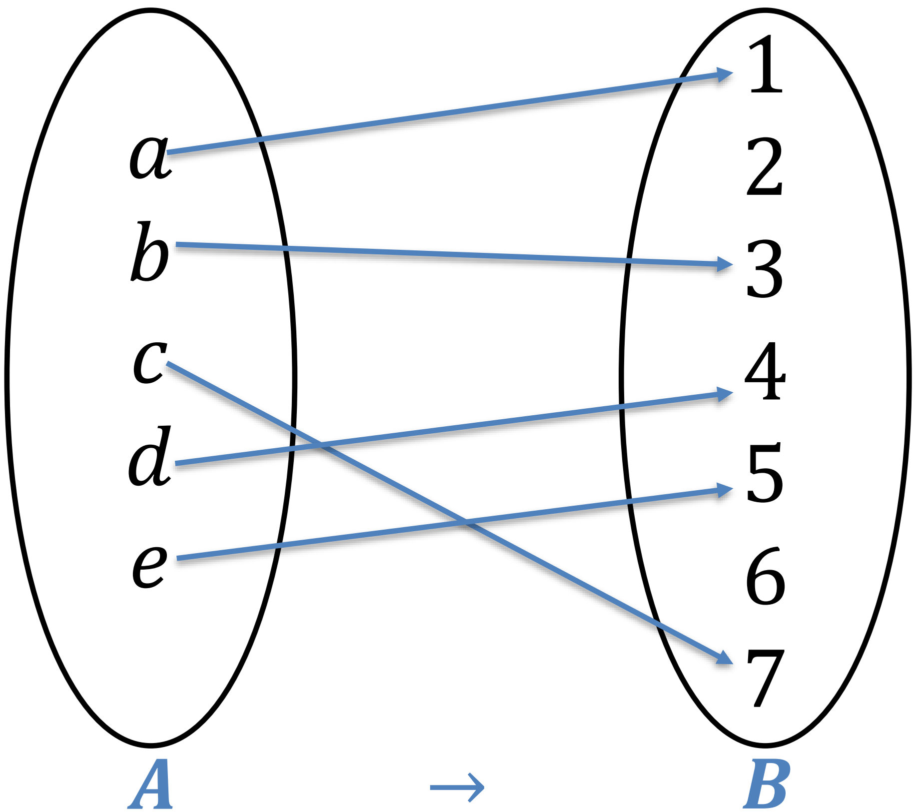
$\text{Range} = \{1,3,4,5,7\}$

One-to-One function (injective)

A function f is said to be **one-to-one**, or **injective**, if and only if $f(a) = f(b)$ implies that $a = b$ for all a and b in the domain of f.

Functions (9/21)

One-to-One function (injective)



$f(a) = 1$

$f(b) = 3$

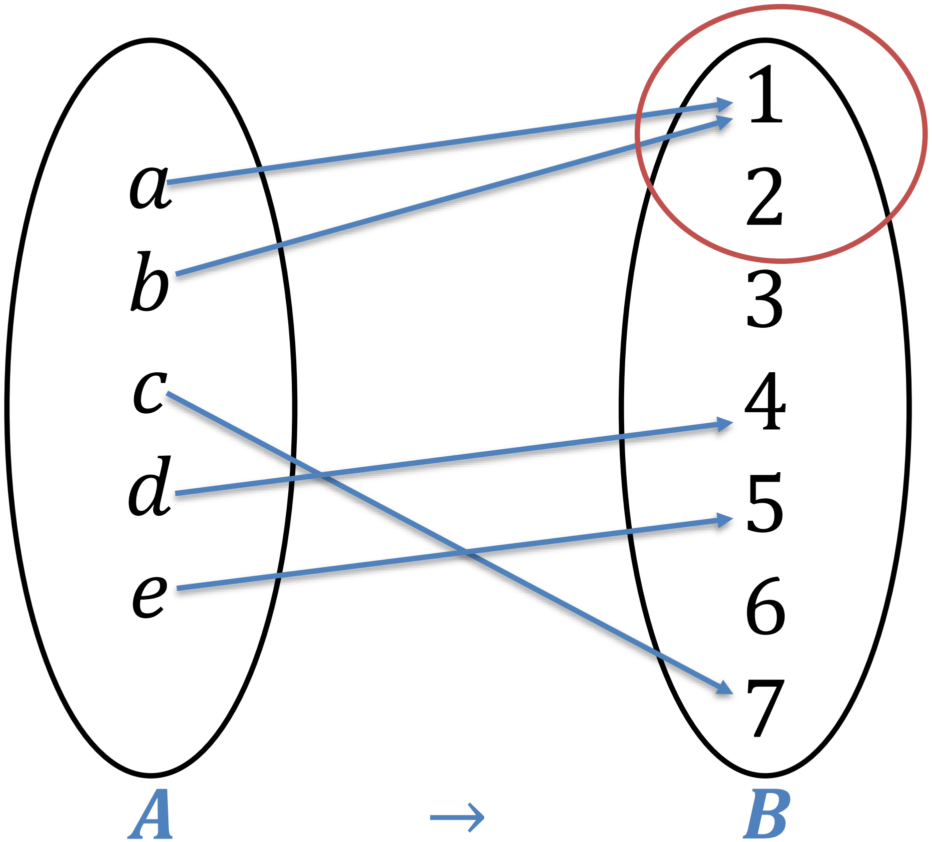
$f(c) = 7$

$f(d) = 4$

$f(e) = 5$

Functions (9/21)

NOT *One-to-One* function (Not injective)



$$\boxed{\begin{array}{c} f(a) = 1 \\ f(b) = 1 \end{array}}$$

$$f(c) = 7$$

$$f(d) = 4$$

$$f(e) = 5$$

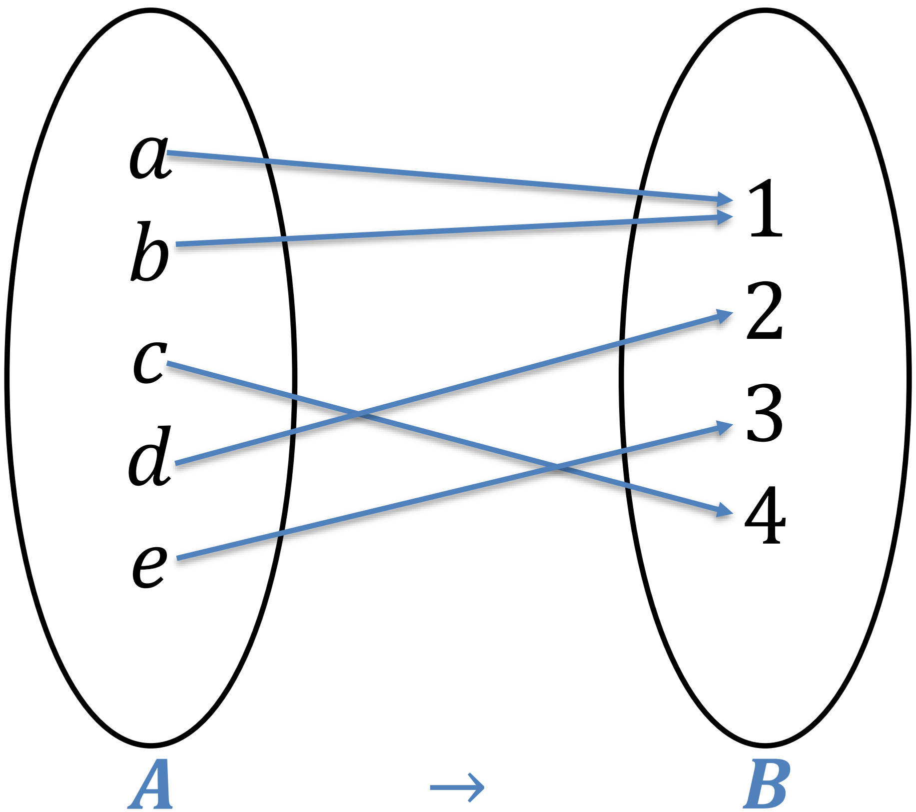
Functions (10/21)

onto function (surjective)

A function f from A to B is called **onto**, or **surjective**, if and only if for every element $b \in B$ there is an element $a \in A$ with $f(a) = b$.

Functions (10/21)

onto function (surjective)



$f(a) = 1$

$f(b) = 1$

$f(c) = 4$

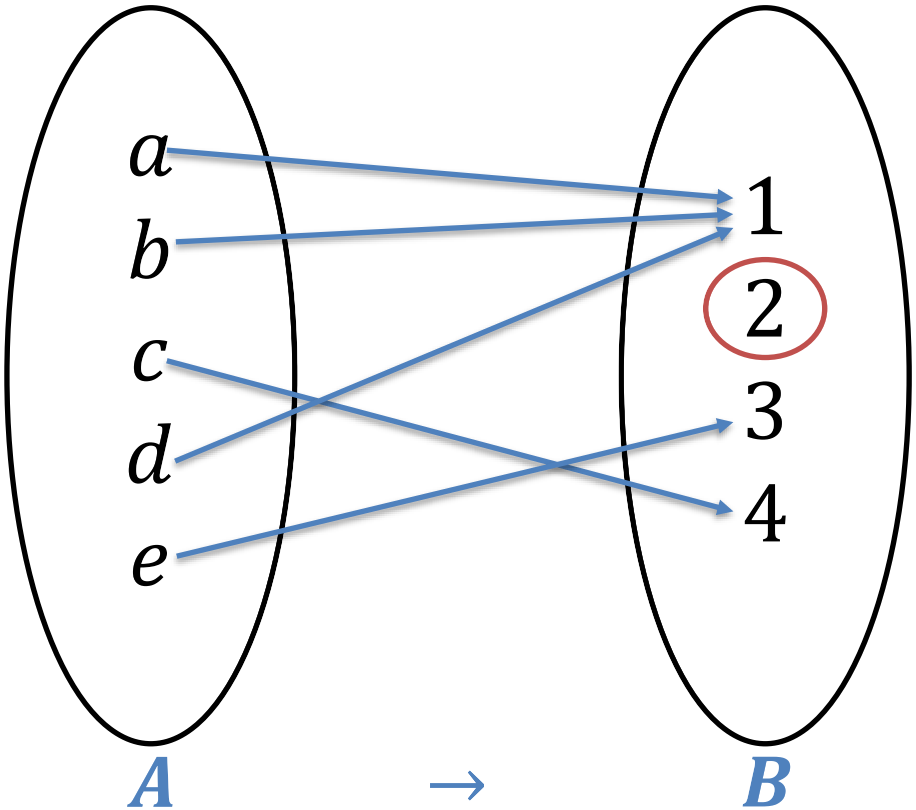
$f(d) = 2$

$f(e) = 3$

Co-Domain = {1,2,3,4}

Range = {1,2,3,4}

NOT *onto* function (Not surjective)



$f(a) = 1$

$f(b) = 1$

$f(c) = 4$

$f(d) = 1$

$f(e) = 3$

Co-Domain = {1,2,3,4}

Range = {1,3,4}

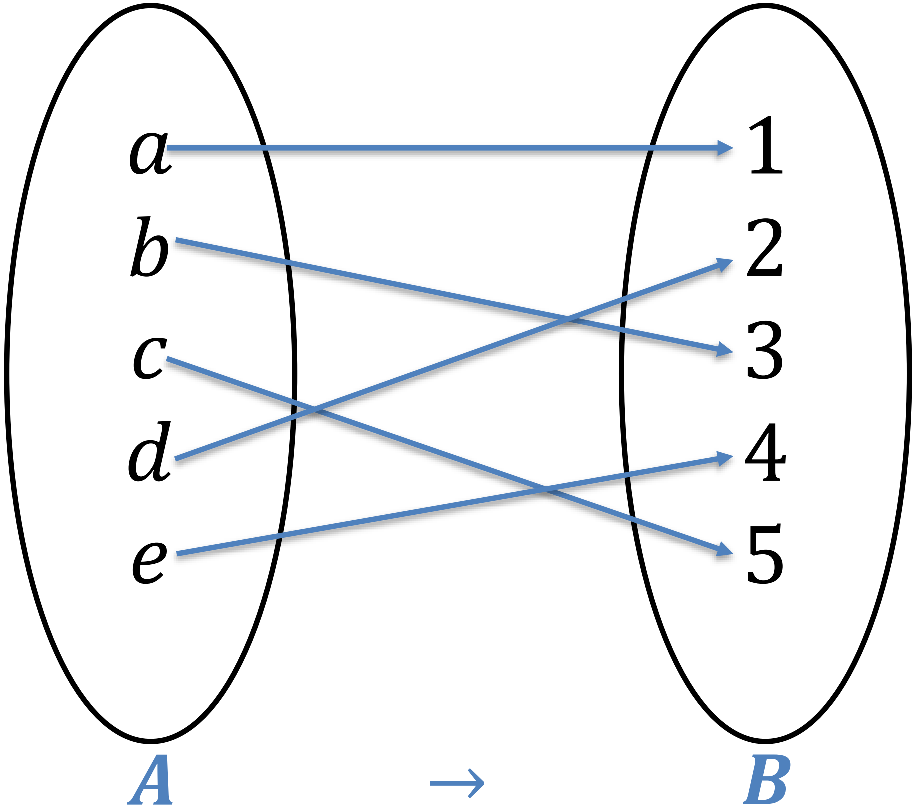
One-to-one correspondence (bijection)

The function f is a **one-to-one correspondence**, or a **bijection**, if it is both one-to-one and onto.

Functions (11/21)

One-to-one correspondence (bijection)

$|A| = |B|$



$f(a) = 1$

$f(b) = 3$

$f(c) = 5$

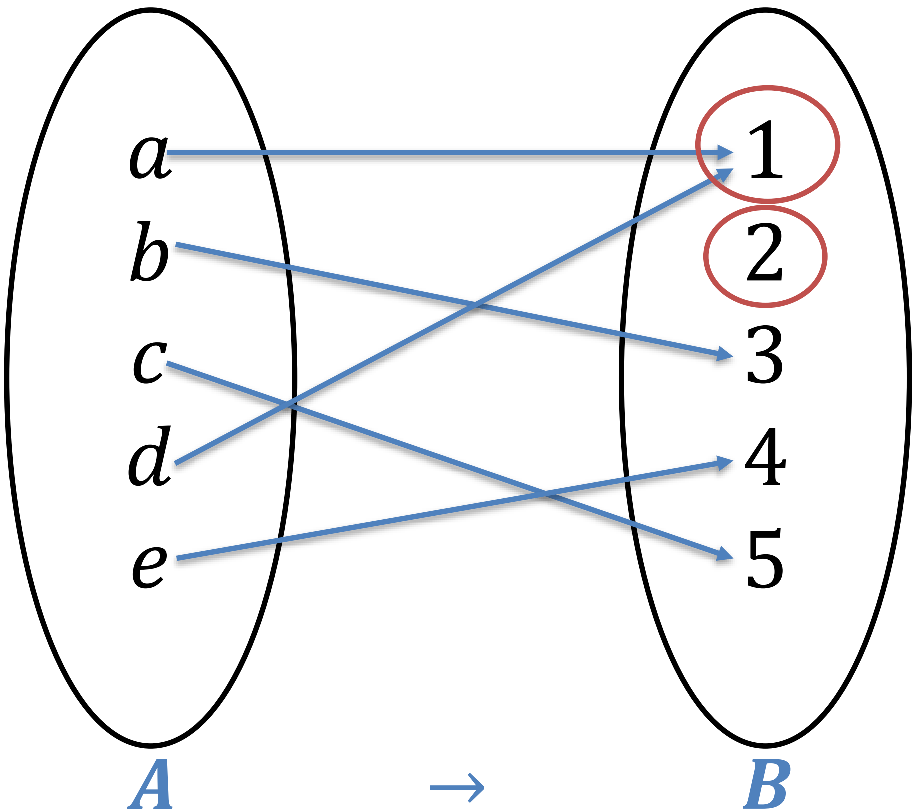
$f(d) = 2$

$f(e) = 4$

Co-Domain = {1,2,3,4,5}

Range = {1,2,3,4,5}

NOT *One-to-one correspondence* (Not bijection)



$f(a) = 1$

$f(b) = 3$ **NOT one-to-one**

$f(c) = 5$ **NOT onto**

$f(d) = 1$

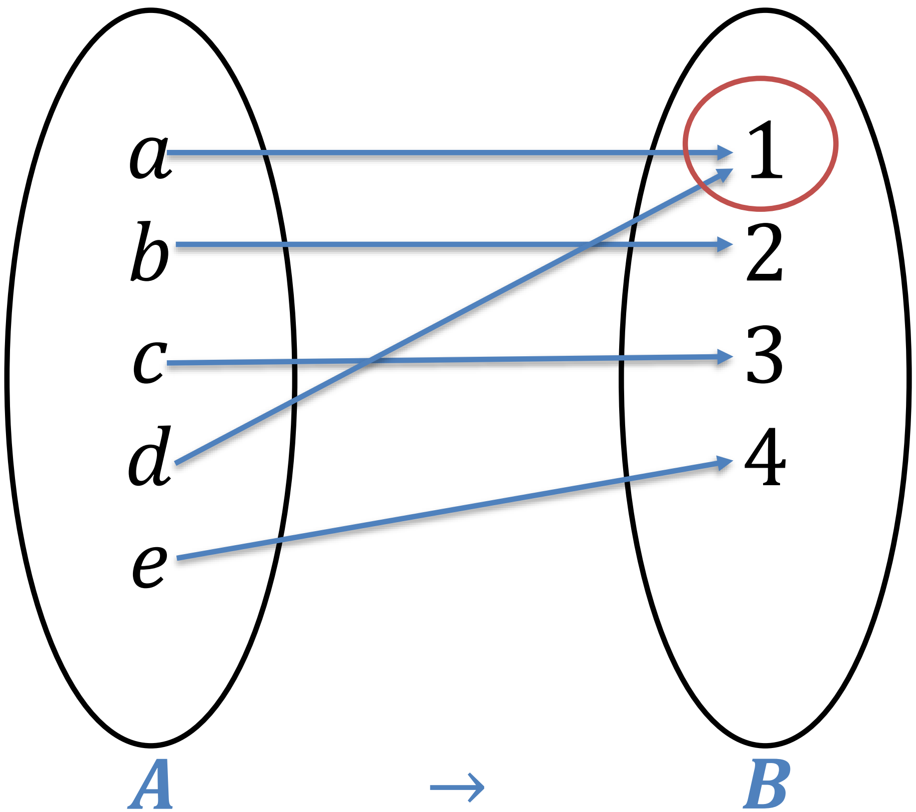
$f(e) = 4$

Co-Domain $= \{1,2,3,4,5\}$

Range $= \{1,3,4,5\}$

Functions (11/21)

NOT *One-to-one correspondence* (Not bijection)



$f(a) = 1$

$f(b) = 2$

$f(c) = 3$

$f(d) = 1$

$f(e) = 4$

Onto

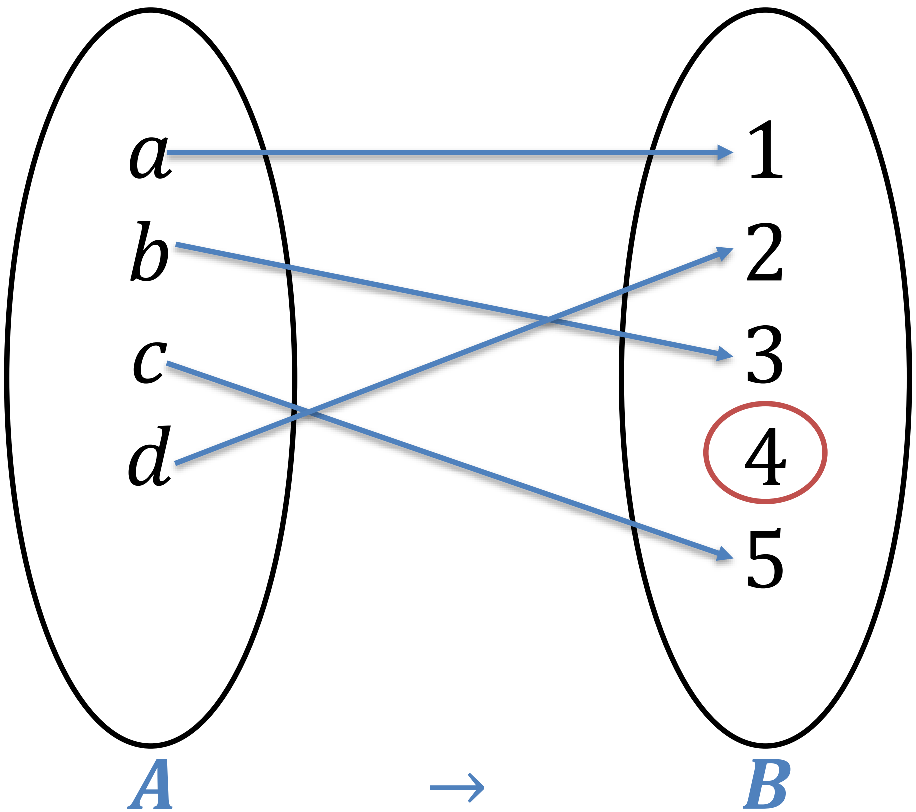
NOT one-to-one

Co-Domain $= \{1,2,3,4\}$

Range $= \{1,2,3,4\}$

Functions (11/21)

NOT *One-to-one correspondence* (Not bijection)



$f(a) = 1$

$f(b) = 3$

$f(c) = 5$

$f(d) = 2$

One-to-one

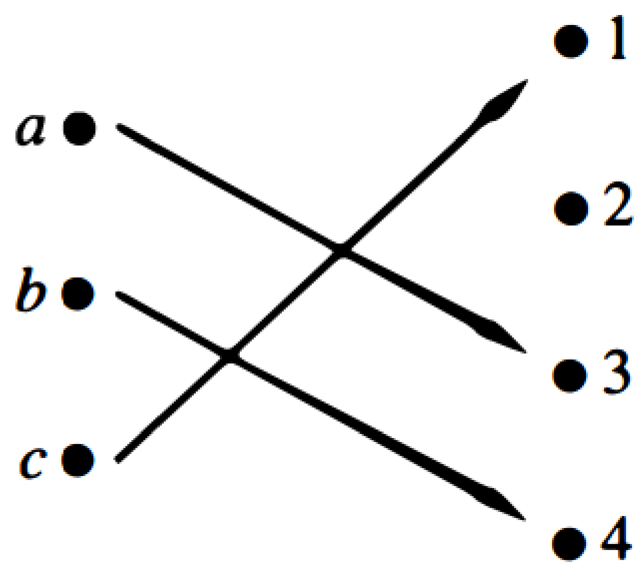
NOT onto

Co-Domain = {1,2,3,4,5}

Range = {1,2,3,5}

Functions (12/21)

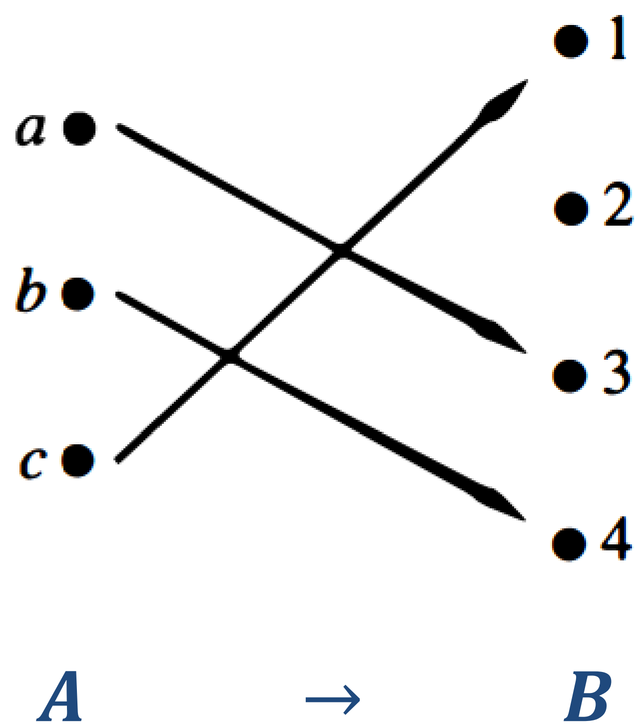
Examples



$A \rightarrow B$

Functions (12/21)

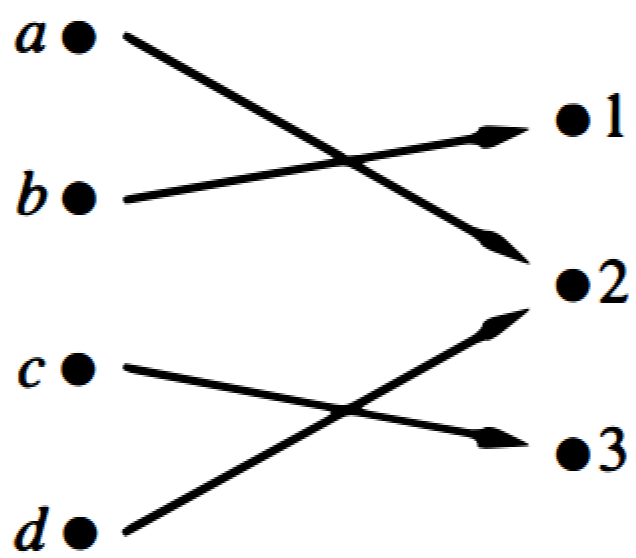
Examples



One-to-one

NOT onto

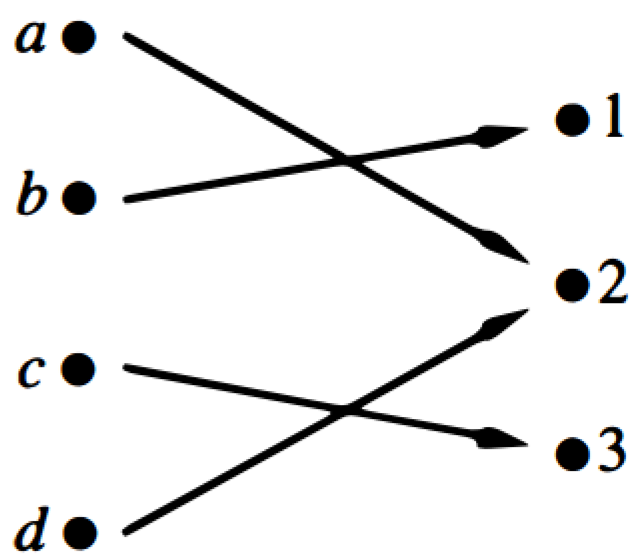
Examples



$A \rightarrow B$

Functions (12/21)

Examples

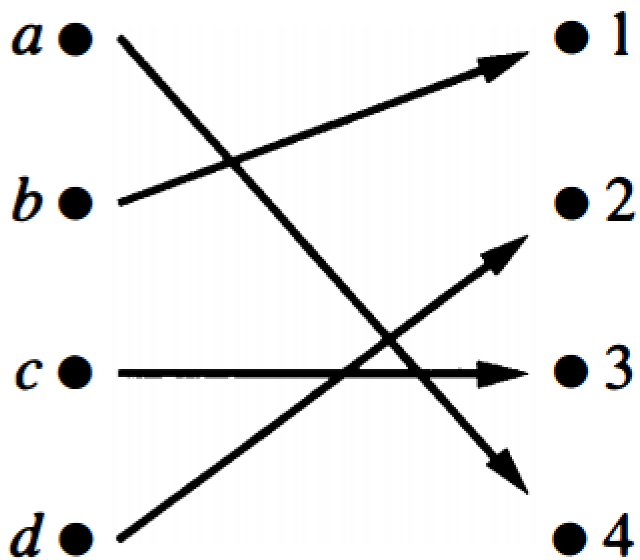


NOT One-to-one

Onto

$A \rightarrow B$

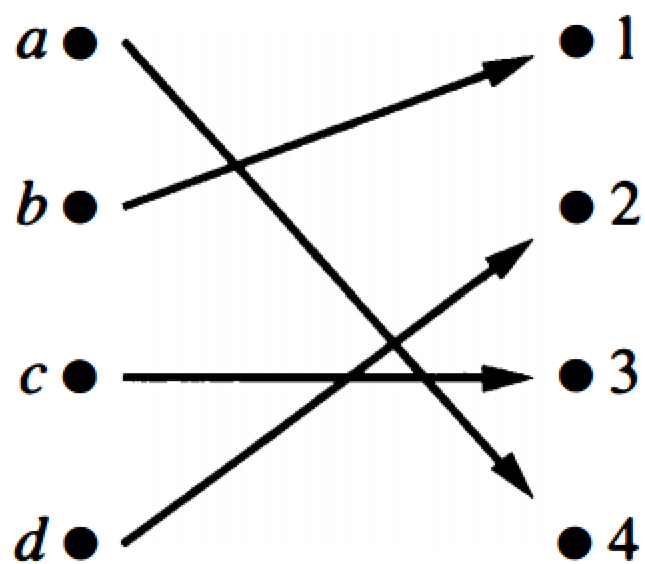
Examples



$A \rightarrow B$

Functions (12/21)

Examples



$A \rightarrow B$

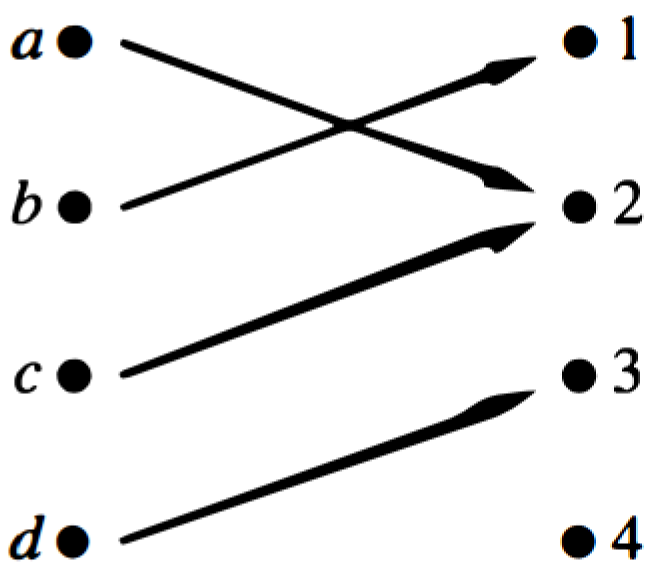
One-to-one

Onto

∴ bijection

Functions (12/21)

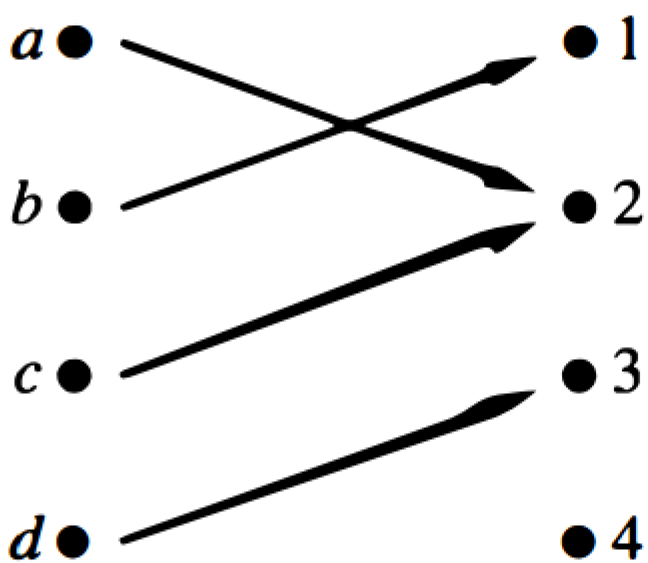
Examples



$A \rightarrow B$

Functions (12/21)

Examples

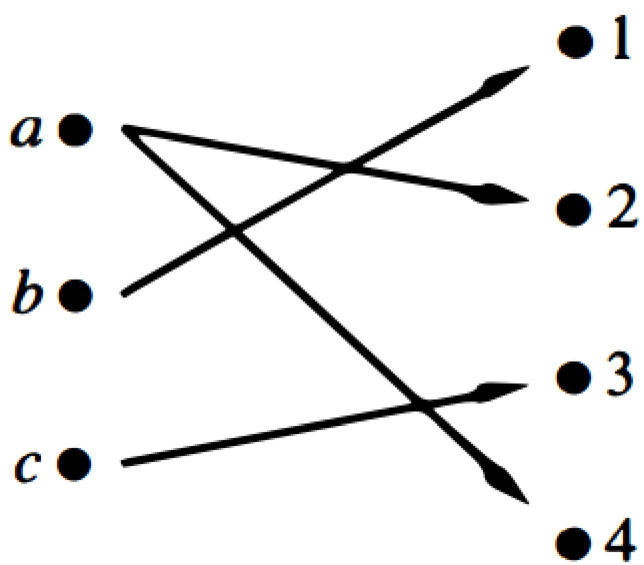


NOT One-to-one

NOT Onto

$A \rightarrow B$

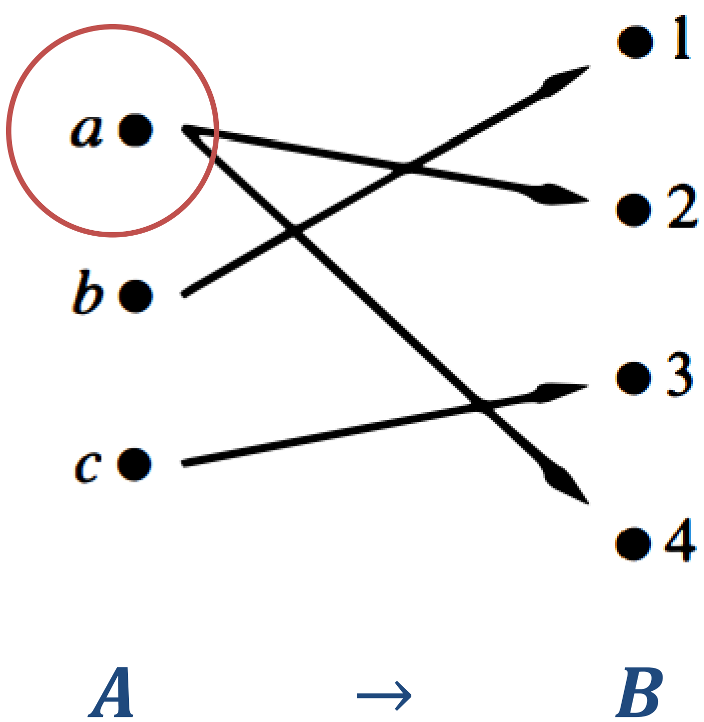
Examples



$A \rightarrow B$

Functions (12/21)

Examples



NOT a function from *A* to *B*

Examples

Determine whether the function $f(x) = x + 1$ from the set of integers to the set of integers is one-to-one.

Examples (Answer)

Determine whether the function $f(x) = x + 1$ from the set of integers to the set of integers is one-to-one.

$f(a) = a + 1$ and $f(b) = b + 1$

$f(x)$ is one–to–one (if $f(a) = f(b)$ and a equal b then).

$$a + 1 = b + 1$$

$$a = b$$

$$\therefore f(x) \text{ is one–to–one}$$

Functions (14/21)

Examples

Determine whether the function $f(x) = x^2$ from the set of integers to the set of integers is one-to-one.

Examples (Answer)

Determine whether the function $f(x) = x^2$ from the set of integers to the set of integers is one-to-one.

$f(a) = a^2$ and $f(b) = b^2$

$f(x)$is one–to–one (if $f(a) = f(b)$ and a equal b then).

$$a^2 = b^2$$

$$\pm a = \pm b$$

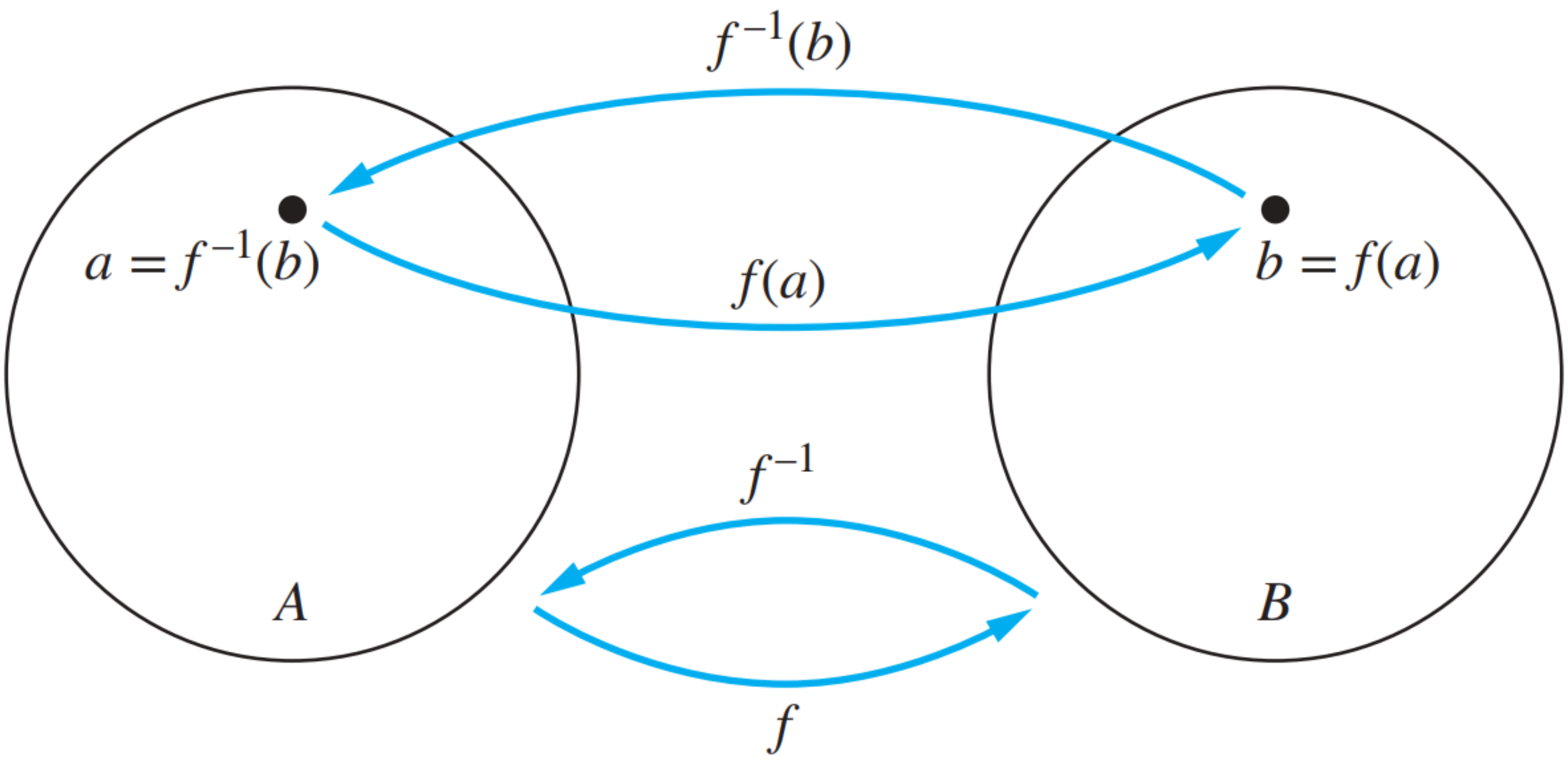
a may be not equal b

$\therefore f(x)$ is NOT one–to–one

Inverse Functions

Let f be a *one-to-one correspondence* from the set A to the set B. The **inverse** function of f is the function that assigns to an element b belonging to B the unique element a in A such that $f(a) = b$. The inverse function of f is denoted by $\boldsymbol{f^{-1}}$. Hence, $f^{-1}(b) = a$ when $f(a) = b$.

Inverse Functions



Invertible

A one-to-one correspondence is called **invertible** because we can define an inverse of this function. A function is **not invertible** if it is not a one-to-one correspondence, because the inverse of such a function does not exist.

Invertible – Example

Let f be the function from $\{a, b, c\}$ to $\{1, 2, 3\}$ such that $f(a) = 2$, $f(b) = 3$, and $f(c) = 1$. Is f invertible, and if it is, what is its inverse?

Invertible – Example

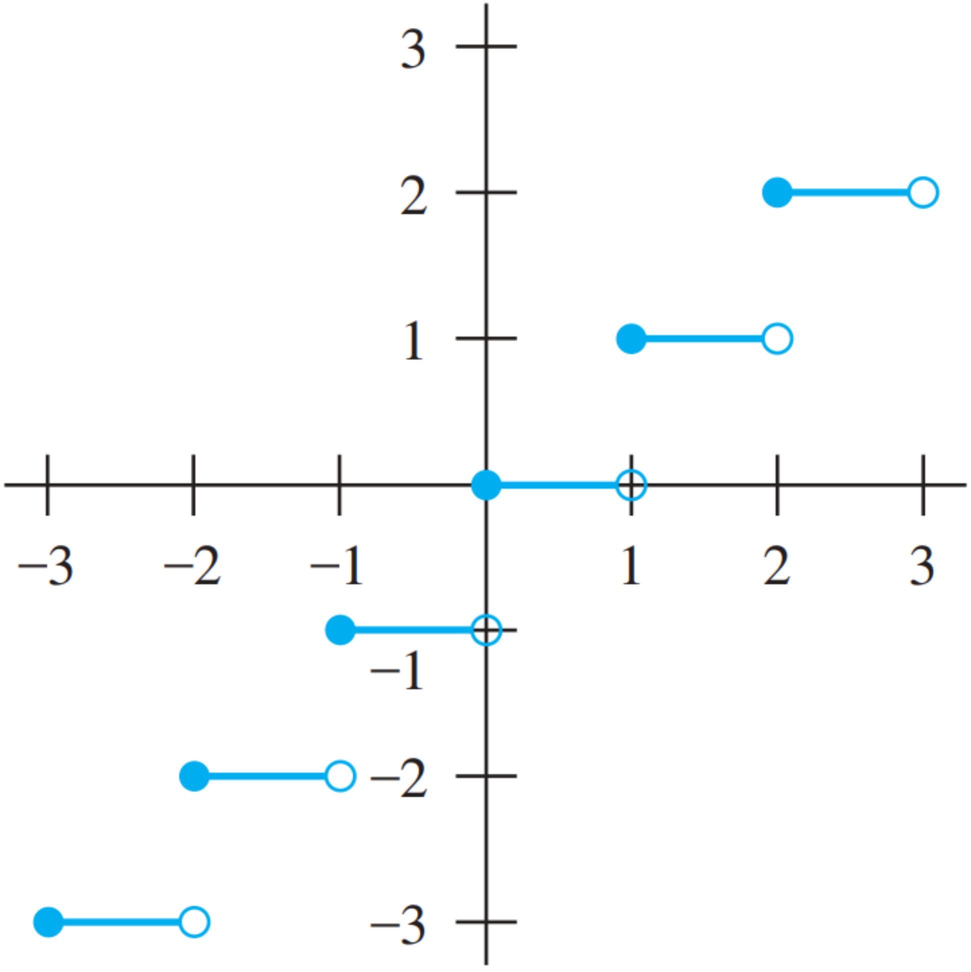
Let f be the function from $\{a, b, c\}$ to $\{1, 2, 3\}$ such that $f(a) = 2$, $f(b) = 3$, and $f(c) = 1$. Is f invertible, and if it is, what is its inverse?

Answer:

The function f is invertible because it is a one-to-one correspondence. The inverse function f^{-1} reverses the correspondence given by f, so $f^{-1}(1) = c$, $f^{-1}(2) = a$, and $f^{-1}(3) = b$.

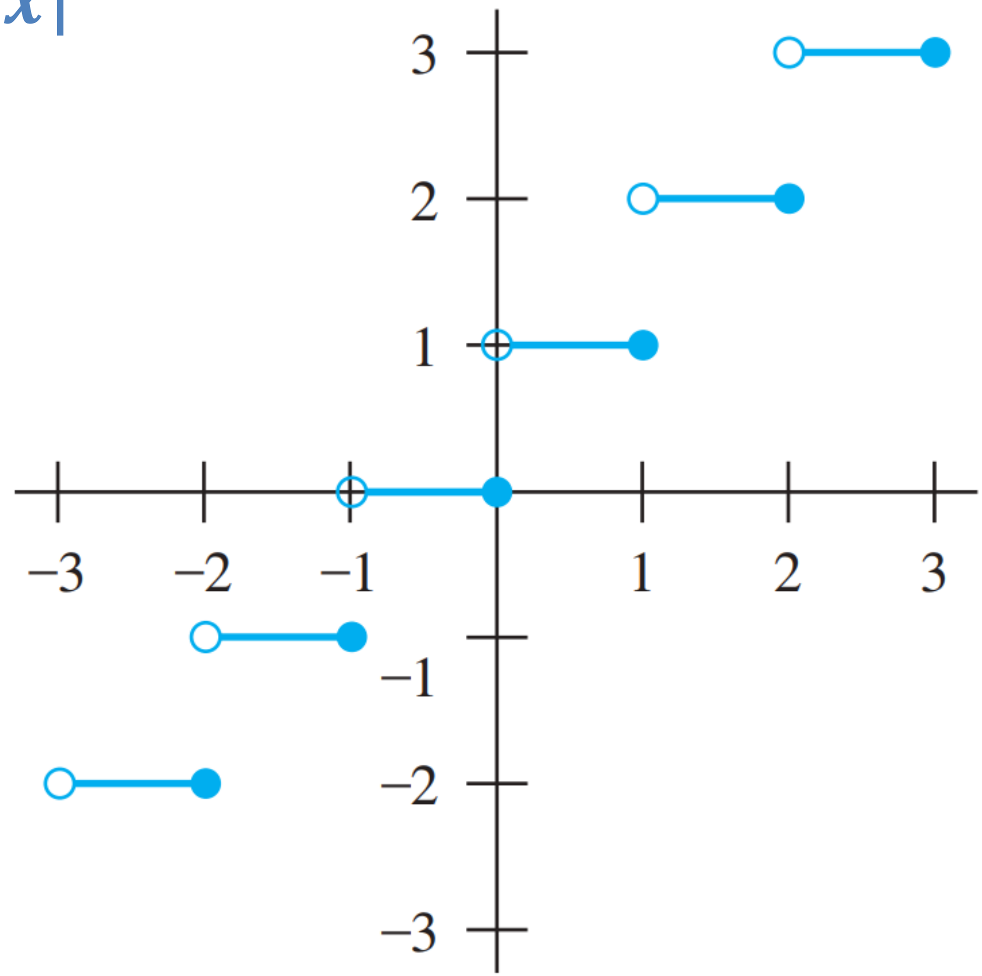
Some Important Functions (1/4)

Floor function $y = \lfloor x \rfloor$



Some Important Functions (2/4)

Ceiling function $y = \lceil x \rceil$



Some Important Functions (3/4)

Useful Properties

$$\lfloor -x \rfloor = -\lceil x \rceil$$
$$\lceil -x \rceil = -\lfloor x \rfloor$$

$$\lfloor x + n \rfloor = \lfloor x \rfloor + n$$
$$\lceil x + n \rceil = \lceil x \rceil + n$$

Some Important Functions (4/4)

Examples

$\lfloor 0.5 \rfloor =$

$\lceil 0.5 \rceil =$

$\lceil 3 \rceil =$

$\lfloor -0.5 \rfloor =$

$\lceil -1.2 \rceil =$

$\lfloor 1.1 \rfloor =$

$\lfloor 0.3 + 2 \rfloor =$

$\left\lceil 1.1 + \lceil 0.5 \rceil \right\rceil =$

Some Important Functions (4/4)

Examples-Answer

$$\lfloor 0.5\rfloor = 0$$

$$\lceil 0.5\rceil = 1$$

$$\lceil 3\rceil = 3$$

$$\lfloor -0.5\rfloor = -\lceil 0.5\rceil = -1$$

$$\lceil -1.2\rceil = -1$$

$$\lfloor 1.1\rfloor = 1$$

$$\lfloor 0.3 + 2\rfloor = 2$$

$$\left\lceil 1.1 + \lceil 0.5\rceil \right\rceil = 3$$

Chapter 2: Basic Structures

- Sets.
- Functions.
- Sequences, and Summations.
- Matrices.

Sequences (1/13)

Definition

- A sequence is a set of things (usually numbers) that are in order.
  - For example, 1 , 2, 3, 5, 8 is a sequence with five terms and 1, 3, 9, 27, 81, . . . , 30, . . . is an infinite sequence.
- We use the notation a_n to denote the image of the integer n. We call a_n a term of the sequence.
- We use the notation $\{a_n\}$ to describe the sequence.

$$\{a_n\} = \{a_1, a_2, a_3, \dots\}$$

Sequences (2/13)

Example

- Consider the sequence $\{a_n\}$, where

$$a_n = \frac{1}{n}$$

The list of the terms of this sequence, beginning with a_1, namely,

$$a_1, a_2, a_3, a_4, \dots ,$$

Starts with

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots$$

Sequences (3/13)

Geometric

A *geometric progression* is a sequence of the form

$$a, ar, ar^2, \ldots, ar^n, \ldots$$

where the $\boxed{\textit{initial term } a}$ and the $\boxed{\textit{common ratio } r}$ are real numbers.

$$2, 10, 50, 250, \ldots$$

Geometric – Example1

$$1, -1, 1, -1, 1, \ldots;$$

$$\{ar^n\}, \qquad n = 0,1,2,\ldots$$

$$a = 1$$

$$r = -1$$

Sequences (5/13)

Geometric – Example2

$2, 10, 50, 250, 1250, \ldots;$

$\{ar^n\}, \qquad n = 0,1,2, \ldots$

$a = 2$

$r = 5$

Geometric – Example3

$$6, 2, \frac{2}{3}, \frac{2}{9}, \frac{2}{27}, \ldots$$

$$\{ar^n\}, \qquad n = 0,1,2, \ldots$$

$$a = 6$$
$$r = 1/3$$

Geometric – Example4

Find $\boldsymbol{a}, \boldsymbol{r}$? $\{\mathbf{3} * \mathbf{4}^{\boldsymbol{n}}\}, \quad \boldsymbol{n} = \mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots$

$$\{ar^n\}, \qquad n = 0,1,2, \ldots$$

$$a = 3$$
$$r = 4$$

Geometric – Example5

Find $\boldsymbol{a}, \boldsymbol{r}$? $\{\mathbf{3} * \mathbf{4}^{\boldsymbol{n}}\}, \quad \boldsymbol{n} = \mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots$

Geometric – Example5

Find $\boldsymbol{a}, \boldsymbol{r}?\ \{\mathbf{3} * \mathbf{4}^{\boldsymbol{n}}\},\ \ \boldsymbol{n} = \mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots$

$$a = 12$$
$$r = 4$$

Arithmetic

An *arithmetic progression* is a sequence of the form

$$a, a + d, a + 2d, \ldots, a + nd, \ldots$$

where the $\boxed{\textit{initial term } a}$ and the $\boxed{\textit{common difference } d}$ are real numbers.

Arithmetic – Example1

$$-1, 3, 7, 11, \ldots,$$

$$\{a + nd\}, \qquad n = 0,1,2, \ldots$$

$$a = -1$$
$$d = 4$$

Arithmetic – Example2

$$7, 4, 1, -2, \ldots$$

$$\{a + nd\}, \qquad n = 0,1,2, \ldots$$

$$a = 7$$
$$d = -3$$

Sequences (12/13)

Notes:

- Are terms obtained from previous terms by adding the same amount or an amount that depends on the position in the sequence?
- Are terms obtained from previous terms by multiplying by a particular amount?
- Are terms obtained by combining previous terms in a certain way?
- Are there cycles among the terms?

Fibonacci Sequence

The *Fibonacci sequence*, $f_0, f_1, f_2, \ldots,$ is defined by the initial conditions $f_0 = 0, f_1 = 1$, and the recurrence relation

$$f_n = f_{n-1} + f_{n-2}$$

for $n = 2, 3, 4, \ldots$.

$$0, 1, 1, 2, 3, 5, 8, \ldots$$

Summations (1/8)

Next, we introduce **summation notation.**
We begin by describing the notation used to express the sum of the terms

$$a_m, a_{m+1}, \ldots, a_n$$

from the sequence $\{a_n\}$. We use the notation

$$\sum_{j=m}^{n} a_j, \qquad \textstyle\sum_{j=m}^{n} a_j, \qquad \text{or} \qquad \sum_{m \le j \le n} a_j$$

(read as the sum from $j = m$ to $j = n$ of a_j)

to represent

Here, the variable j is called the **index of summation**

$$a_m + a_{m+1} + \cdots + a_n.$$

Summations (1/8)

$$\sum_{j=m}^{n} a_j = \sum_{i=m}^{n} a_i = \sum_{k=m}^{n} a_k$$

Here, the index of summation runs through all integers starting with its **lower limit** m and ending with its **upper limit** n. A large uppercase Greek letter sigma, $\sum$, is used to denote summation.

Summations (2/8)

Example 1

Express the sum of the first 100 terms of the sequence $\{a_n\}$, where $a_n = 1/n$ for $n = 1, 2, 3, \ldots$

Summations (3/8)

Example 1

Express the sum of the first 100 terms of the sequence $\{a_n\}$, where $a_n = 1/n$ for $n = 1, 2, 3, \ldots$

Answer

$$\sum_{n=1}^{100} 1/n$$

Summations (4/8)

Example 2

What is the value of $\sum_{j=1}^{5} j^2$?

Summations (4/8)

Example 2

What is the value of $\sum_{j=1}^{5} j^2$?

Answer

$$\sum_{j=1}^{5} j^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2$$

$$= 1 + 4 + 9 + 16 + 25$$

$$= 55.$$

Summations (5/8)

Example 3

What is the value of $\sum_{s \in \{0,2,4\}} s$?

Summations (5/8)

Example 3

What is the value of $\sum_{s \in \{0,2,4\}} s$?

$$\sum_{s \in \{0,2,4\}} s = 0 + 2 + 4 = 6.$$

Summations (6/8)

Example 4

Suppose we have the sum

$$\sum_{j=1}^{5} j^2$$

but want the index of summation to run between 0 and 4

$$\sum_{j=1}^{5} j^2 = \sum_{k=0}^{4} (k+1)^2$$

It is easily checked that both sums are $1 + 4 + 9 + 16 + 25 = 55$.

Summations (7/8)

Double Summation

Find $$\sum_{i=1}^{4}\sum_{j=1}^{3} ij$$

Summations (8/8)

Double Summation

Find

$$\sum_{i=1}^{4}\sum_{j=1}^{3} ij = \sum_{i=1}^{4}(i + 2i + 3i)$$

$$= \sum_{i=1}^{4} 6i$$

$$= 6 + 12 + 18 + 24 = 60.$$

Matrices (1/14)

Definition:

A ***matrix*** is a rectangular array of numbers. A matrix with m rows and n columns is called an $m \times n$ matrix. A matrix with the same number of rows as columns is called ***square***.

$$\begin{array}{c c} & \begin{array}{cc} 1 & 2 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 1 & 1 \\ 0 & 2 \\ 1 & 3 \end{bmatrix} \end{array}$$

Matrices (1/14)

Definition:

A ***matrix*** is a rectangular array of numbers. A matrix with m rows and n columns is called an $m \times n$ matrix. A matrix with the same number of rows as columns is called ***square***.

$$\text{The matrix} \begin{bmatrix} 1 & 1 \\ 0 & 2 \\ 1 & 3 \end{bmatrix} \text{is a } 3 \times 2 \text{ matrix.}$$

Matrices (2/14)

$m \times n$ matrix

Let m and n be positive integers and let

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

Matrices (3/14)

$m \times n$ matrix

The (2, 1)th ***element*** or ***entry*** of **A** is the element $\boldsymbol{a_{21}}$, means 2nd *row* and 1st *column* of **A**.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

Matrix Arithmetic (Sum.)

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

$$\underset{\mathbf{A}}{\begin{bmatrix} 1 & 0 & -1 \\ 2 & 2 & -3 \\ 3 & 4 & 0 \end{bmatrix}} + \underset{\mathbf{B}}{\begin{bmatrix} 3 & 4 & -1 \\ 1 & -3 & 0 \\ -1 & 1 & 2 \end{bmatrix}} = \underset{\mathbf{A+B}}{\begin{bmatrix} 4 & 4 & -2 \\ 3 & -1 & -3 \\ 2 & 5 & 2 \end{bmatrix}}.$$

Matrices (4/14)

Matrix Arithmetic (Sum.)

Note: matrices of *different sizes* can **not** be added.

Let $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$ be $m \times n$ matrices.

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

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

A **B** **A+B**

Matrix Arithmetic (Product/Multiplication)

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ik} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mk} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1j} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2j} & \cdots & b_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ b_{k1} & b_{k2} & \cdots & b_{kj} & \cdots & b_{kn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & c_{ij} & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix}$$

$$\mathbf{A}_{mk} \qquad \mathbf{B}_{kn} \qquad \mathbf{AB} = \mathbf{C}_{mn}$$

Matrix Arithmetic (Product/Multiplication)

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ik} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mk} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1j} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2j} & \cdots & b_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ b_{k1} & b_{k2} & \cdots & b_{kj} & \cdots & b_{kn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & c_{ij} & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix}$$

$$\mathbf{A}_{mk} \qquad \mathbf{B}_{kn} \qquad \mathbf{AB} = \mathbf{C}_{mn}$$

Example1 (1/2)

$$\mathbf{A}_{3\times3} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & -1 \end{bmatrix}_{3\times3} \qquad \mathbf{M}_{3\times2} = \begin{bmatrix} 1 & 2 \\ 3 & -1 \\ 1 & 1 \end{bmatrix}_{3\times2}$$

$$\mathbf{A}_{3\times3} \times \mathbf{M}_{3\times2} = \mathbf{B}_{3\times2}$$

$$\begin{array}{c c} & \\ \boxed{1} & \\ \boxed{2} & \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & -1 \end{bmatrix} \\ \boxed{3} & \end{array} \times \begin{array}{c} \begin{array}{cc} \boxed{1} & \boxed{2} \end{array} \\ \begin{bmatrix} 1 & 2 \\ 3 & -1 \\ 1 & 1 \end{bmatrix} \end{array} = \begin{bmatrix} & \\ & \\ & \end{bmatrix}$$

Example1 (2/2)

$$\mathbf{A}_{3\times3} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & -1 \end{bmatrix}_{3\times3} \qquad \mathbf{M}_{3\times2} = \begin{bmatrix} 1 & 2 \\ 3 & -1 \\ 1 & 1 \end{bmatrix}_{3\times2}$$

$$\mathbf{A}_{3\times3} \times \mathbf{M}_{3\times2} = \mathbf{B}_{3\times2}$$

$$\boldsymbol{a_{11}} = \mathbf{6} = (\mathbf{1} \times \mathbf{1} + \mathbf{1} \times \mathbf{3} + \mathbf{2} \times \mathbf{1})$$

$$\begin{array}{cc} \boxed{1} & \\ \boxed{2} & \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & -1 \end{bmatrix} \\ \boxed{3} & \end{array} \times \begin{array}{c} \begin{array}{cc} \boxed{1} & \boxed{2} \end{array} \\ \begin{bmatrix} 1 & 2 \\ 3 & -1 \\ 1 & 1 \end{bmatrix} \end{array} = \begin{bmatrix} 6 & 3 \\ 10 & 3 \\ 9 & -2 \end{bmatrix}$$

Matrices (6/14)

Example1 (2/2)

$$\mathbf{A}_{3\times3} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & -1 \end{bmatrix}_{3\times3} \qquad \mathbf{M}_{3\times2} = \begin{bmatrix} 1 & 2 \\ 3 & -1 \\ 1 & 1 \end{bmatrix}_{3\times2}$$

$$\mathbf{A}_{3\times3} \times \mathbf{M}_{3\times2} = \mathbf{B}_{3\times2}$$

$$\boldsymbol{a_{31}} = \mathbf{9}$$
$$= (\mathbf{1} \times \mathbf{1} + \mathbf{3} \times \mathbf{3} + (-\mathbf{1}) \times \mathbf{1})$$

$$\begin{array}{cc} & \begin{array}{cc} \boxed{1} & \boxed{2} \end{array} \\ \begin{array}{c} \boxed{1} \\ \boxed{2} \\ \boxed{3} \end{array} \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & -1 \end{bmatrix} \times & \begin{bmatrix} 1 & 2 \\ 3 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 3 \\ 10 & 3 \\ 9 & -2 \end{bmatrix} \end{array}$$

Matrices (7/14)

Example2 (1/2)

Let

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}.$$

Does $\mathbf{AB} = \mathbf{BA}$?

Example2 (2/2)

Let

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}.$$

Solution: We find that

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

Hence, $\mathbf{AB} \neq \mathbf{BA}$.

Identity matrix ($\mathbf{I}_n$)

The *identity matrix* of order n is the $n \times n$ matrix $\mathbf{I}_n = [\delta_{ij}]$, where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$.

$$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{3\times 3}$$

$\mathbf{A}$ is an $m \times n$ matrix, we have

$$\mathbf{A}\mathbf{I}_n = \mathbf{I}_m\mathbf{A} = \mathbf{A}.$$

Matrices (9/14)

Powers of square matrices (A^r)

When $\mathbf{A}$ is an $n \times n$ matrix, we have

$$\mathbf{A}^0 = \mathbf{I}_n, \qquad \mathbf{A}^r = \underbrace{\mathbf{AAA} \cdots \mathbf{A}}_{r \text{ times}}.$$

Transpose of A (A^t)

Interchanging the rows and columns of **A**

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

A

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

$\mathbf{A}^t$

Transpose of A (A^t)

Interchanging the rows and columns of **A**

$$\begin{bmatrix} \boxed{1 \quad 2 \quad 3} \\ 4 \quad 5 \quad 6 \end{bmatrix} \qquad \begin{bmatrix} \boxed{\begin{matrix}1\\2\\3\end{matrix}} & \begin{matrix}4\\5\\6\end{matrix} \end{bmatrix}$$

$$\mathbf{A} \qquad\qquad \mathbf{A}^t$$

Matrices (11/14)

Symmetric

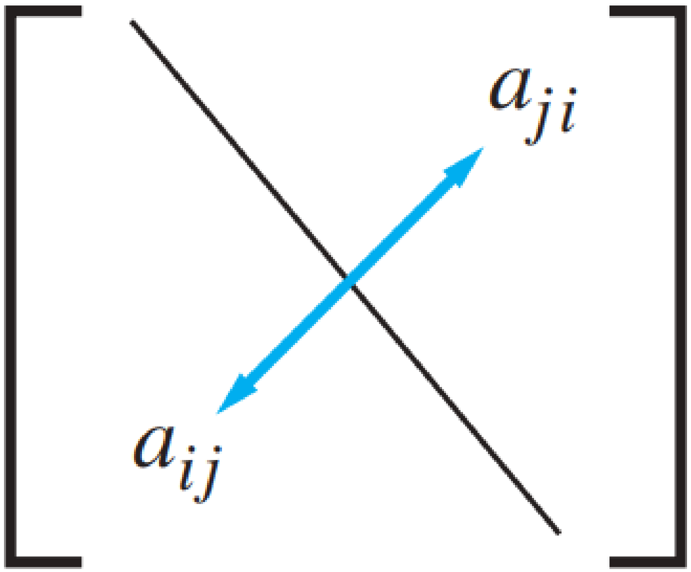
A square matrix $\mathbf{A}$ is called *symmetric* if $\mathbf{A} = \mathbf{A}^t$

$$\underset{\mathbf{A}}{\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}} = \underset{\mathbf{A}^t}{\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}}$$

Matrices (11/14)

Symmetric

A square matrix **A** is called *symmetric* if $\mathbf{A} = \mathbf{A}^t$



Zero–One Matrices

A matrix all of whose entries are either **0** or **1**

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}.$$

join and *meet* (Zero–One Matrices)

meet

$$b_1 \wedge b_2 = \begin{cases} 1 & \text{if } b_1 = b_2 = 1 \\ 0 & \text{otherwise,} \end{cases}$$

join

$$b_1 \vee b_2 = \begin{cases} 1 & \text{if } b_1 = 1 \text{ or } b_2 = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Example (1/3)

Find the join and meet of the zero–one matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}.$$

Example (2/3)

Find the join and meet of the zero–one matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}.$$

Solution: We find that the join of **A** and **B** is

$$\mathbf{A} \vee \mathbf{B} = \begin{bmatrix} 1 \vee 0 & 0 \vee 1 & 1 \vee 0 \\ 0 \vee 1 & 1 \vee 1 & 0 \vee 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

Example (3/3)

Find the join and meet of the zero–one matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}.$$

Solution:

The meet of **A** and **B** is

$$\mathbf{A} \wedge \mathbf{B} = \begin{bmatrix} 1 \wedge 0 & 0 \wedge 1 & 1 \wedge 0 \\ 0 \wedge 1 & 1 \wedge 1 & 0 \wedge 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Chapter 3: Algorithms

- Concept of Algorithms.
- Linear Search Algorithm.

Concept of Algorithms (1/12)

Introduction (1/2)

There are many general classes of problems that arise in discrete mathematics. For instance: given a sequence of integers, find the largest one; given a set, list all its subsets; given a set of integers, put them in increasing order; given a network, find the shortest path between two vertices.

Introduction (2/2)

When presented with such a problem, the first thing to do is to construct a model that translates the problem into a mathematical context. To complete the solution, a method is needed that will solve the general problem using the model. Ideally, what is required is a procedure that follows a sequence of steps that leads to the desired answer. Such a sequence of steps is called an **algorithm**.

Concept of Algorithms (2/12)

Definition 1

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

MOHAMMED IBN MUSA AL-KHOWARIZMI
"Uzbekistan"

Concept of Algorithms (3/12)

EXAMPLE

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

Concept of Algorithms (4/12)

EXAMPLE

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

10	5	7	25	2	14
a_1	a_2	a_3	a_4	a_5	a_6

Concept of Algorithms (4/12)

EXAMPLE

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

10	5	7	25	2	14
a_1	a_2	a_3	a_4	a_5	a_6

$\max = 25$

return 25

EXAMPLE

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

a_1	a_2	a_3	a_4	a_5	a_6

Concept of Algorithms (5/12)

EXAMPLE

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

10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
a_1	a_2	a_3	a_4	a_5	a_6

Concept of Algorithms (5/12)

EXAMPLE

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

	5				
a_1	a_2	a_3	a_4	a_5	a_6

Concept of Algorithms (5/12)

EXAMPLE

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

[redacted]	[redacted]	7	[redacted]	[redacted]	[redacted]
a_1	a_2	a_3	a_4	a_5	a_6

Concept of Algorithms (6/12)

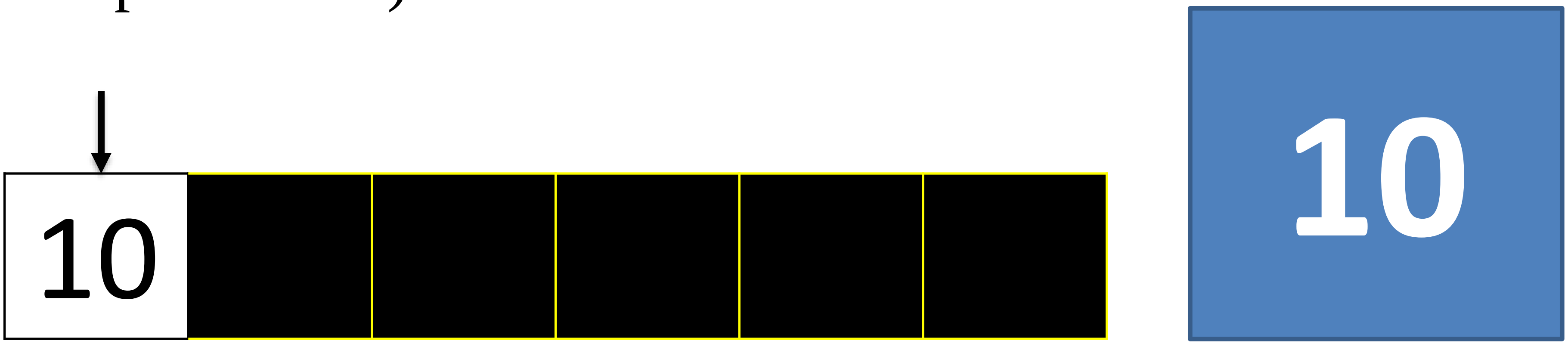
Solution: We perform the following steps:

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.)

Concept of Algorithms (7/12)

Solution: We perform the following steps:

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.)



temporary maximum

If you start from the **left**.

Concept of Algorithms (8/12)

Solution: We perform the following steps:

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.

Concept of Algorithms (8/12)

Solution: We perform the following steps:

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.

temporary maximum

if $(value > 10)$ **then** (temporary maximum = $value$)

Concept of Algorithms (9/12)

Solution: We perform the following steps:

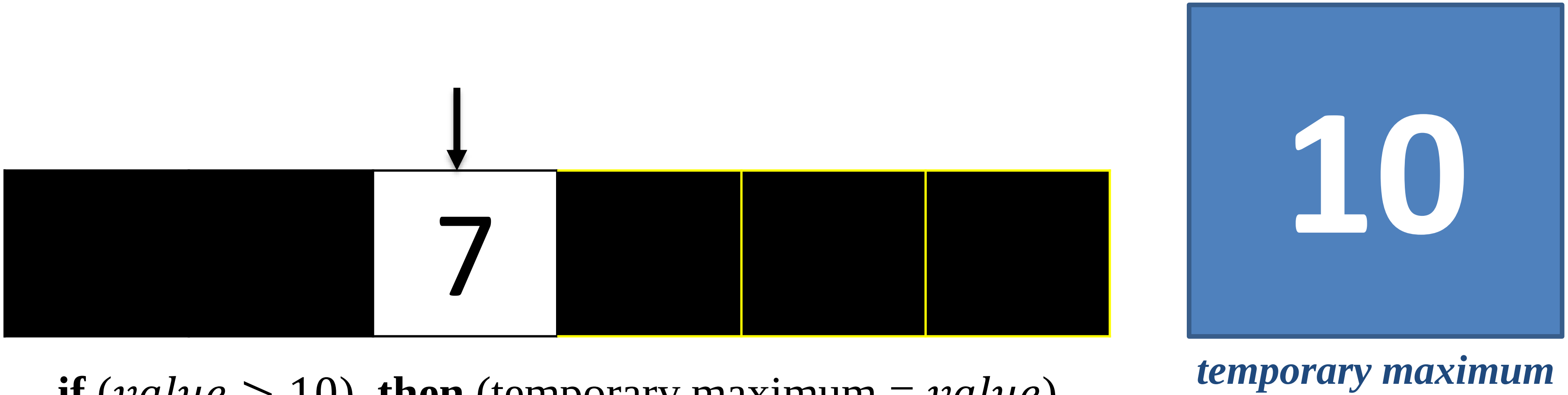
3. Repeat the previous step if there are more integers in the sequence.

if ($value > 10$) **then** (temporary maximum = $value$)

Concept of Algorithms (9/12)

Solution: We perform the following steps:

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

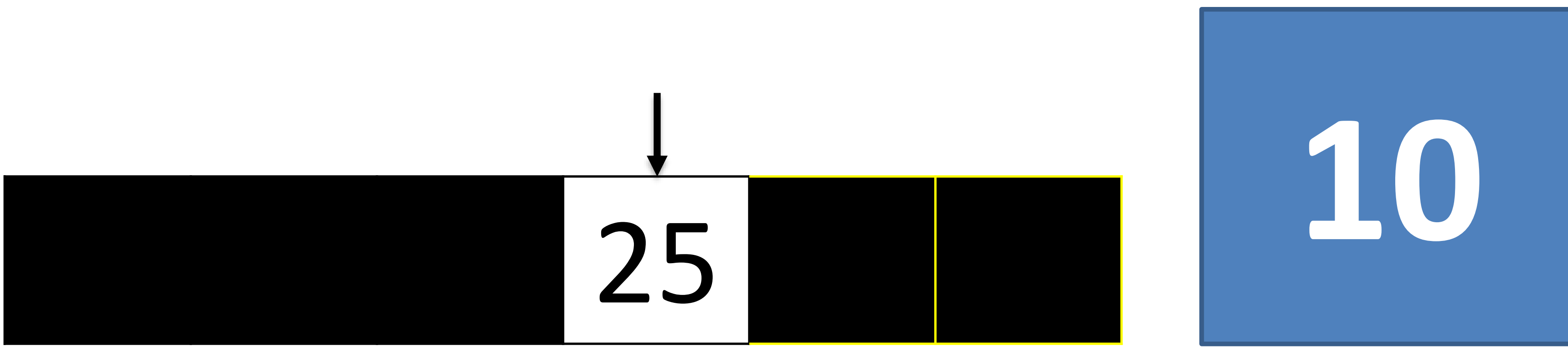


if $(value > 10)$ **then** (temporary maximum = $value$)

Concept of Algorithms (9/12)

Solution: We perform the following steps:

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



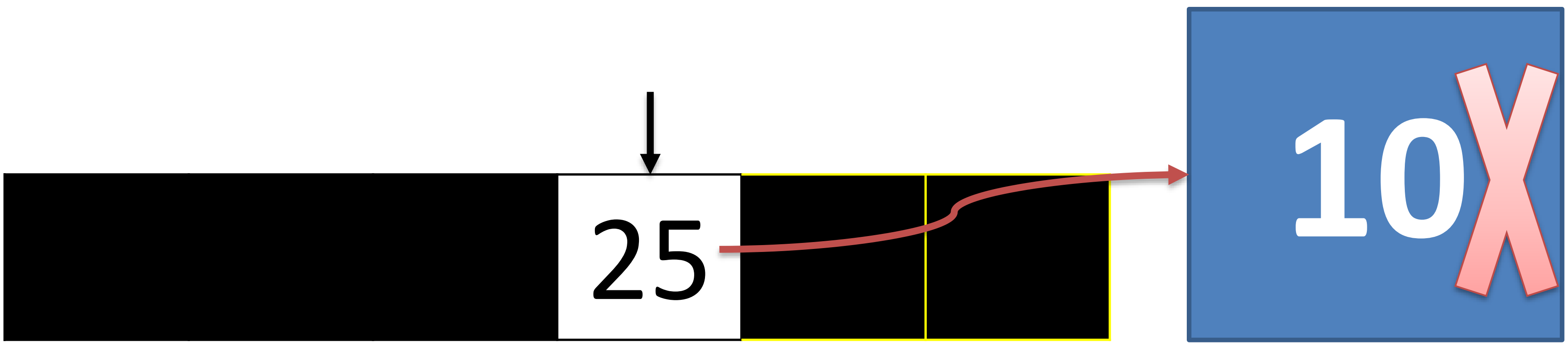
temporary maximum

if $(value > 10)$ **then** (temporary maximum = $value$)

Concept of Algorithms (9/12)

Solution: We perform the following steps:

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



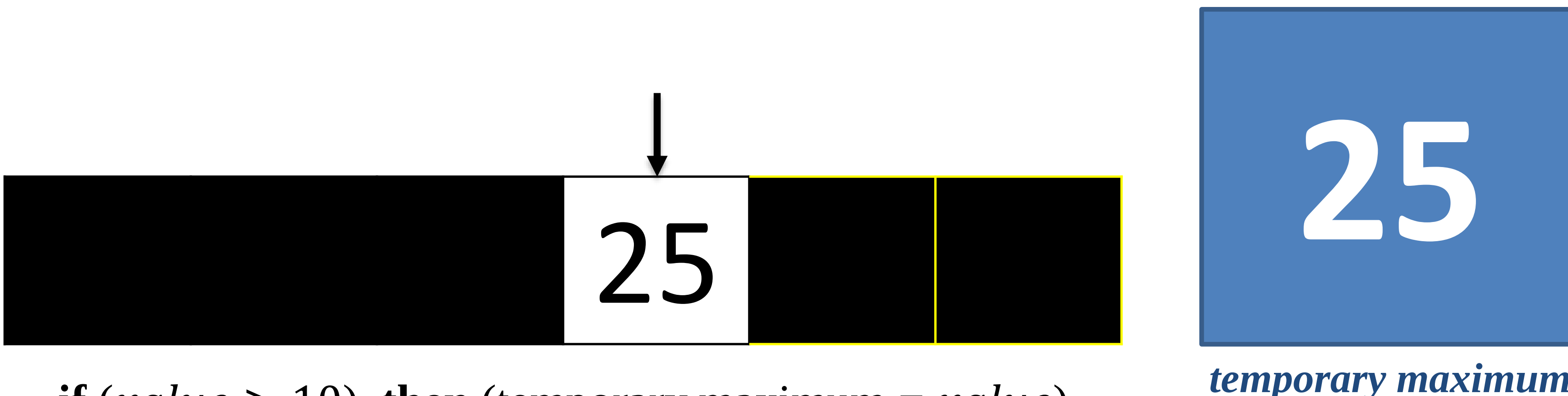
temporary maximum

if ($value$ > 10) **then** (temporary maximum = $value$)

Concept of Algorithms (9/12)

Solution: We perform the following steps:

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

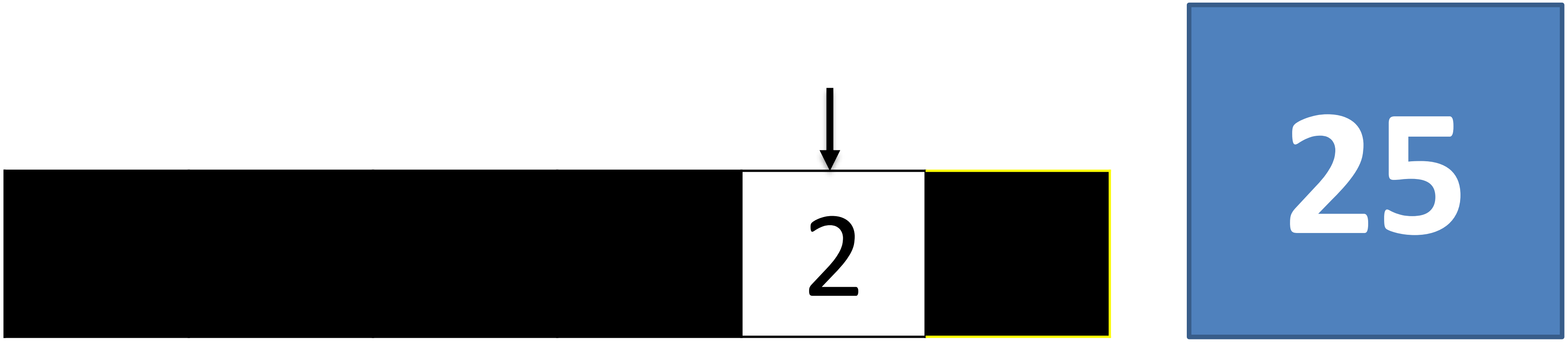


if $(value > 10)$ **then** (temporary maximum = $value$)

Concept of Algorithms (9/12)

Solution: We perform the following steps:

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



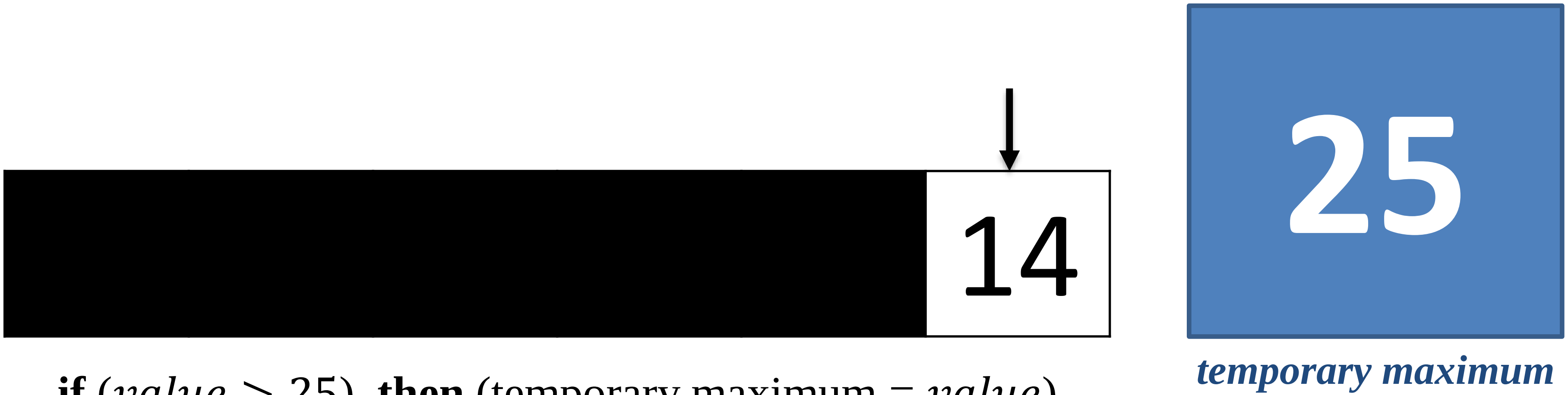
temporary maximum

if ($value > 25$) **then** (temporary maximum = $value$)

Concept of Algorithms (9/12)

Solution: We perform the following steps:

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



if $(value > 25)$ **then** (temporary maximum = $value$)

Concept of Algorithms (10/12)

Solution: We perform the following steps:

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

[redacted]

25

temporary maximum

Stop

Concept of Algorithms (10/12)

Solution: We perform the following steps:

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

return 25

[redacted]

Stop

25

temporary maximum

=

largest integer

Concept of Algorithms (11/12)

Solution: pseudocode

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** $max := a_i$
return max\{max is the largest element\}

PROPERTIES OF ALGORITHMS (1/2)

- ***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 problem.
- ***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.

PROPERTIES OF ALGORITHMS (2/2)

- ***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.

Introduction (1/3)

Locate the ***value* = 2** or determine that it is not in the list.

10	5	7	25	2	14
a_1	a_2	a_3	a_4	a_5	a_6

Introduction (2/3)

Locate the ***value* = 2** or determine that it is not in the list.

10	5	7	25	**2**	14
a_1	a_2	a_3	a_4	a_5	a_6

The *value* 2 is founded in the location of a_5, namely, 5.

Introduction (3/3)

Locate the ***value* = 2** or determine that it is not in the list.

Location	1	2	3	4	5	6
Value	10 a_1	5 a_2	7 a_3	25 a_4	**2** a_5	14 a_6

The *value* 2 is founded in the location of a_5, namely, 5.

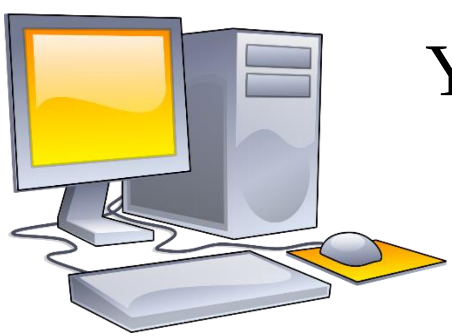
return 5

Searching Algorithms (2/18)

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list

Location	1	2	3	4	5	6
Value						



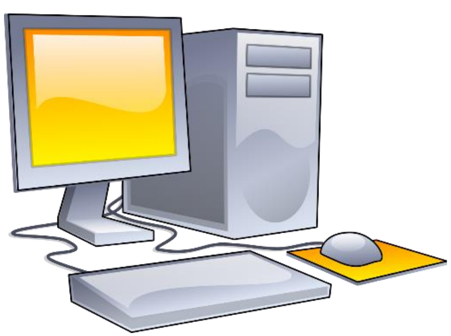
You can start from the right, left, or middle.

Searching Algorithms (2/18)

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list

Location	1	2	3	4	5	6
Value						

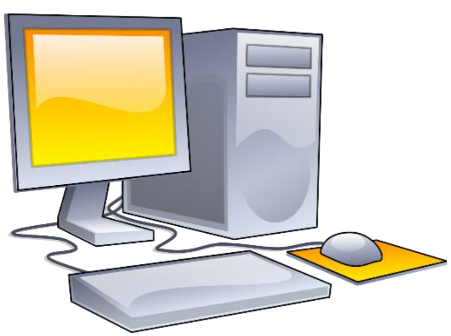


If you start from the **left**.

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list

Location	1	2	3	4	5	6
Value	10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]



Not found in the 1st location

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list

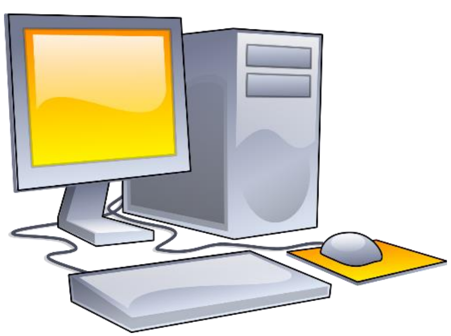
Location	1	2	3	4	5	6
Value	10	5				

Not found in the 2nd location

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list

Location	1	2	3	4	5	6
Value	10	5	7			



Not found in the 3rd location

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list

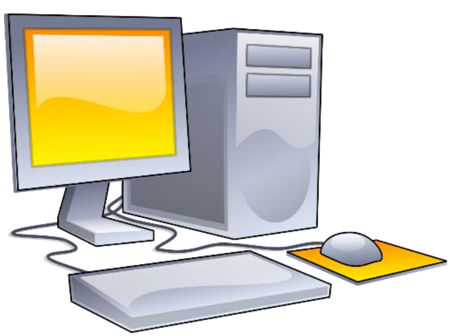
Location	1	2	3	4	5	6
Value	10	5	7	25		

Not found in the 4th location

Linear Search Algorithm (or Sequential Search)

Locate an element **2** in this list

Location	1	2	3	4	5	6
Value	10	5	7	25	2	



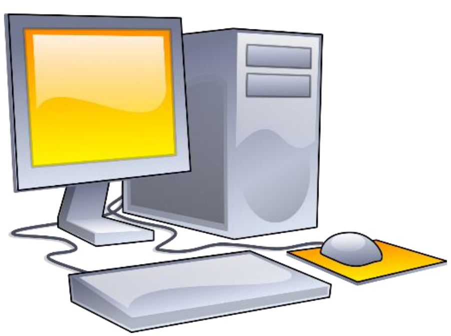
Founded in the **5**$^{\text{th}}$ location

return 5

Linear Search Algorithm (or Sequential Search)

Locate an element **99** in this list

Location	1	2	3	4	5	6
Value						

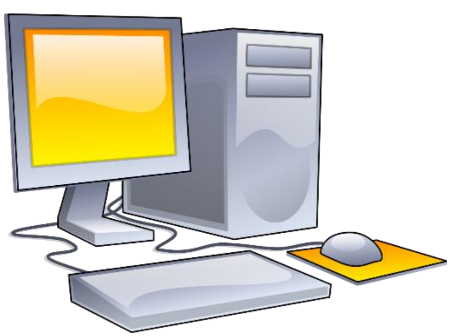


If you start from the **left**.

Linear Search Algorithm (or Sequential Search)

Locate an element **99** in this list

Location	1	2	3	4	5	6
Value	10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]



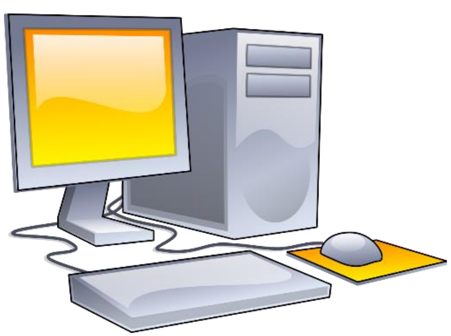
Not found in the 1st location

Searching Algorithms (3/18)

Linear Search Algorithm (or Sequential Search)

Locate an element **99** in this list

Location	1	2	3	4	5	6
Value	10	5	[redacted]	[redacted]	[redacted]	[redacted]



Not found in the 2nd location

Searching Algorithms (3/18)

Linear Search Algorithm (or Sequential Search)

Locate an element **99** in this list

Location	1	2	3	4	5	6
Value	10	5	7			

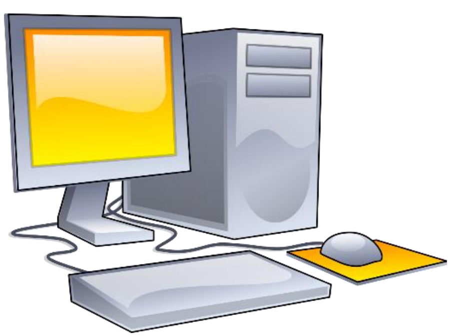
Not found in the 3rd location

Searching Algorithms (3/18)

Linear Search Algorithm (or Sequential Search)

Locate an element **99** in this list

Location	1	2	3	4	5	6
Value	10	5	7	25		

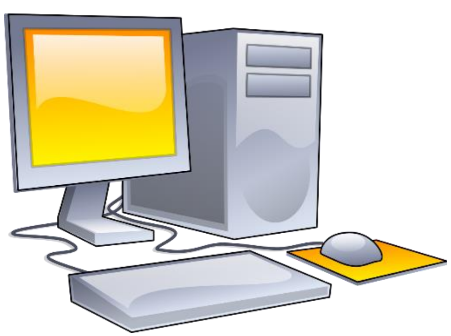


Not found in the 4th location

Linear Search Algorithm (or Sequential Search)

Locate an element **99** in this list

Location	1	2	3	4	5	6
Value	10	5	7	25	2	



Not found in the 5th location

Linear Search Algorithm (or Sequential Search)

Locate an element **99** in this list

Location	1	2	3	4	5	6
Value	10	5	7	25	2	14

Not found in the 6th location

Linear Search Algorithm (or Sequential Search)

Locate an element **99** in this list

Location	1	2	3	4	5	6
Value	10	5	7	25	2	14

Not Founded in all the list

return 0

Linear Search Algorithm (or Sequential Search)

Locate an element **33** in this list

Linear Search Algorithm (or Sequential Search)

Locate an element **33** in this list

1	2	3	4	5	6	7	8	9	10
10	14	19	26	27	31	33	35	42	44
= 33									

return 7

Searching Algorithms (5/18)

Linear Search Algorithm (or Sequential Search)

1. Comparing x and a_1.
   When $x = a_1$, **return** the location of a_1, namely, 1.

2. When $x \neq a_1$, compare x with a_2.
   If $x = a_2$, **return** the location of a_2, namely, 2.

Searching Algorithms (6/18)

Linear Search Algorithm (or Sequential Search)

3. When $x \neq a_2$, compare x with a_3, and so on. Continue this process, comparing x successively with each term of the list until a match is found, where the solution is the location of that term, unless no match occurs. If the entire list has been searched without locating x, **return 0**.

Linear Search Algorithm (or Sequential Search)

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** $location := i$
else $location := 0$
return $location$ {$location$ is the subscript of the term that equals x, or is 0 if x is not found}

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	2	3
	10	5	7
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5		3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	2	3
	[redacted]		
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	1	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
  $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	$\boxed{1}$	2	3
	a_1	a_2	a_3

Searching Algorithms (8/18)

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	1	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* := i
else *location* := 0
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	$\boxed{1}$	2	3
	10	[redacted]	
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 1

ALGORITHM 2 The Linear Search Algorithm.

x	i	n
5	1	3

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$) ***True***
  $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	$\boxed{1}$	2	3
	10	[redacted]	
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

i =	1	$\boxed{2}$	3
		5	
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \le n$ and $x \ne a_i$)
    $i := i + 1$
if $i \le n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	$\boxed{2}$	3
	a_1	5 a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$) ***False***
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	$\boxed{2}$	3
		5	
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	$\boxed{2}$	3
		5	
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 1

x	i	n
5	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
  $i := i + 1$
True
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	$\boxed{2}$	3
	[redacted]	5	[redacted]
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 1

location

x	i	n
5	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
True
if $i \leq n$ **then** $\boxed{location := i}$
~~**else** *location* := 0~~
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	$\boxed{2}$	3
		5	
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

***Example* 1**

x	i	n
5	2	3

location

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
  $i := i + 1$
True
if $i \leq n$ **then** $\boxed{location := i}$
~~**else** *location* := 0~~
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}
2

$i =$	1	$\boxed{2}$	3
	[redacted]	5	[redacted]
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \le n$ and $x \ne a_i$)
　　$i := i + 1$
if $i \le n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	2	3
	10	5	7
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8		3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	2	3
	[redacted]		
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8	1	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	$\boxed{1}$	2	3
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8	1	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	$\boxed{1}$	2	3
	10	[redacted]	[redacted]
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8	1	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \le n$ and $x \ne a_i$) *True*
  $i := i + 1$
if $i \le n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	$\boxed{1}$	2	3
	10	[redacted]	
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
        $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	$\boxed{2}$	3
		5	
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8	2	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
  $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	$\boxed{2}$	3
		5	
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

ALGORITHM 2 The Linear Search Algorithm.

x	i	n
8	2	3

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

$i := 1$

True

while ($i \leq n$ and $x \neq a_i$)

    $i := i + 1$

if $i \leq n$ **then** *location* $:= i$

else *location* $:= 0$

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

$i =$	1	$\boxed{2}$	3
	[redacted]	5	[redacted]
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

ALGORITHM 2 The Linear Search Algorithm.

x	i	n
8	3	3

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \le n$ and $x \ne a_i$)
    $i := i + 1$
if $i \le n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	2	$\boxed{3}$
	[redacted]	[redacted]	7
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8	3	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
 $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	2	$\boxed{3}$
	[redacted]	[redacted]	7
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

ALGORITHM 2 The Linear Search Algorithm.

x	i	n
8	3	3

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \le n$ and $x \ne a_i$) *True*
    $i := i + 1$
if $i \le n$ **then** *location* := i
else *location* := 0
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	2	$\boxed{3}$
	[redacted]	[redacted]	7
	a_1	a_2	a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8	4	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$ 1 2 3 $\boxed{4}$

[redacted]

a_1 a_2 a_3

Linear Search Algorithm (or Sequential Search)

Example 2

ALGORITHM 2 The Linear Search Algorithm.

x	i	n
8	4	3

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

$i := 1$

while ($i \le n$ and $x \ne a_i$)

$\quad i := i + 1$

if $i \le n$ **then** *location* $:= i$

else *location* $:= 0$

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

i **=** **1** **2** **3** $\boxed{\textbf{4}}$

[redacted]

a_1 a_2 a_3

Linear Search Algorithm (or Sequential Search)

Example 2

ALGORITHM 2 The Linear Search Algorithm.

x	i	n
8	4	3

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
False
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$ 1 2 3 $\boxed{4}$

[redacted]

a_1 a_2 a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8	4	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
    $i := i + 1$
if $i \leq n$ **then** *location* $:= i$
else *location* $:= 0$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

$i =$ 1 2 3 $\boxed{4}$

[redacted]

a_1 a_2 a_3

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8	4	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \le n$ and $x \ne a_i$)
  $i := i + 1$
False
if $i \le n$ **then** ~~$location := i$~~
else $\boxed{location := 0}$
return $location$ {$location$ is the subscript of the term that equals x, or is 0 if x is not found}

$i =$	1	2	3	$\boxed{4}$
	a_1	a_2	a_3	

Searching Algorithms (9/18)

Linear Search Algorithm (or Sequential Search)

Example 2

x	i	n
8	4	3

ALGORITHM 2 The Linear Search Algorithm.

procedure *linear search*(x: integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \leq n$ and $x \neq a_i$)
  $i := i + 1$
False
if $i \leq n$ **then** ~~*location* := *i*~~
else $\boxed{location := 0}$
return *location*{*location* is the subscript of the term that equals x, or is 0 if x is not found}

0

$i =$ 1 2 3 $\boxed{4}$

[redacted]

a_1 a_2 a_3

Video Lectures

كلية الحاسبات والذكاء الإصطناعي

All Lectures: https://www.youtube.com/playlist?list=PLxIvc-MGOs6gZlMVYOOEtUHJmfUquCjwz

Lectures #4: https://www.youtube.com/watch?v=OE8ek2FSxWE&list=PLxIvc-MGOs6gZlMVYOOEtUHJmfUquCjwz&index=13

https://www.youtube.com/watch?v=S7jh1BH_UU8&list=PLxIvc-MGOs6gZlMVYOOEtUHJmfUquCjwz&index=14 Up to time 00:05:57

https://www.youtube.com/watch?v=Wc7RW5EVaBw&list=PLxIvc-MGOs6gZlMVYOOEtUHJmfUquCjwz&index=15

https://www.youtube.com/watch?v=MFRVt2zwfOY&list=PLxIvc-MGOs6gZlMVYOOEtUHJmfUquCjwz&index=16

https://www.youtube.com/watch?v=A4dq1rVwcF4&list=PLxIvc-MGOs6gZlMVYOOEtUHJmfUquCjwz&index=17

https://www.youtube.com/watch?v=Isp3IHOAJWQ&list=PLxIvc-MGOs6gZlMVYOOEtUHJmfUquCjwz&index=18

https://www.youtube.com/watch?v=M3IROxIWPYM&list=PLxIvc-MGOs6gZlMVYOOEtUHJmfUquCjwz&index=19

Thank You

Dr. Ahmed Hagag

ahagag@fci.bu.edu.eg