1st Class

2023-2024

Discrete Structures

الهياكل المتقطعة

أستاذ المادة : د. أقباس عزالدين

References

1- Theory and problems of Discrete mathematics, by Seymour Lipschutz & Marc Lars Lipson, Schaum's Outline Series, third edition 2007

2- Mathematical foundation of computer science, Y.N. Singh, 2005

3- discrete structures, Amin Witno, Revision Notes and Problems 2006, www.witno.com

4- Discrete mathematics for New technology, Rowan Garnier & John Taylor, Second Edition, 2002

5- Discrete mathematical structures for computer science by Bernard Kolman & Robert C. Busby

6- Discrete Mathematics and Its Applications, Seventh Edition, Kenneth H. Rosen, *AT&T Laboratories*, 2012

7- http://www.math.uvic.ca/faculty/gmacgill/guide

8- http://en.wikibooks.org/wiki/Discrete_mathematics/Set_theory

SETS AND ELEMENTS

A set is a collection of objects called the elements or members of the set. The ordering of the elements is not important and repetition of elements is ignored,
for example

$$\{1, 3, 1, 2, 2, 1\} = \{1, 2, 3\}.$$

One usually uses capital letters, A,B,X, Y, . . . , to denote sets, and lowercase letters, a, b, x, y, . . ., to denote elements of sets.
Below you'll see a sampling of items that could be considered as sets:

- The items in a store
- The English alphabet
- Even numbers

A set could have as many entries as you would like. It could have one entry, 10 entries, 15 entries, infinite number of entries, or even have no entries at all! For example, in the above list the English alphabet would have 26 entries, while the set of even numbers would have an infinite number of entries.
Each entry in a set is known as an **element or member**
Sets are written using curly brackets "{" and "}", with their elements listed in between.
For example:
1- the English alphabet could be written as {a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}
2- even numbers could be {0,2,4,6,8,10,...}

Principles:

$\in$ belong to
$\notin$ not belong to
$\subseteq$ subset
$\subset$ proper subset (is a non-equal subset)
For example, {a, b} is a proper subset of {a, b, c}, but {a, b, c} is not a proper subset of {a, b, c}.
$\not\subset$ not subset

So we could replace the statement: "a is belong to the alphabet" with:

$a \in \{alphabet\}$

and replace the statement "3 is not belong to the set of even numbers" with:

$3 \notin \{Even\ numbers\}$

Now if we named our sets we could go even further.
Give the set consisting of the **alphabet** the name A,
and give the set consisting of **even numbers** the name E.

We could now write
$a \in A$
and
$3 \notin E$.

Problem

Let A = {2, 3, 4, 5} and C = {1, 2, 3, . . ., 8, 9}, Show that A is a proper subset of C.

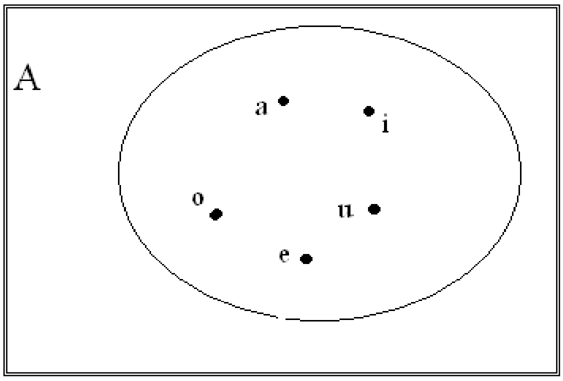
Answer

Each element of A belongs to C, so $A \subseteq C$. On the other hand,

$1 \in C$ but $1 \notin A$. Hence $A \neq C$. Therefore A is a proper subset of C.

There are three ways to specify a particular set:

1) By list its members separated by commas and contained in braces{ }, (if it is possible), for example: A= {a,e,i,o,u}
2) By state those properties which characterize the elements in the set, for example:
 A={x:x is a letter in the English alphabet, x is a vowel}
3) Venn diagram: (A graphical representation of sets).



Example (1)

A={x:x is a letter in the English alphabet, x is a vowel}

$e \in A$ (e is belong to A)

$f \notin A$ (f is not belong to A)

Example (2)

X is the set {1,3,5,7,9}

$3 \in X$ and

$4 \notin X$

Example (3)

Let $E = \{x \mid x^2 - 3x + 2 = 0\}$ $\rightarrow (x-2)(x-1)=0$ $\rightarrow x=2$ & $x=1$

$E = \{2, 1\}$, and

$2 \in E$

Empty Set

A set with no elements is called an *empty set*.

An empty is denoted by { } or ∅.

For example,

- ∅ = {*x*: *x* is an integer and $x^2 + 5 = 0$}
- ∅ = {*x*: *x* are living beings who never die}
- ∅ = {*x*: *x* is the UOT student of age below 15}
- ∅ = { *x*: *x* is the set of persons of age over 200}

Universal set:

In any application of the theory of sets, the members of all sets under investigation usually belong to some fixed large set called the universal set.

For example:

In human population studies the universal set consists of all the people in the world.

We will let the symbol U denotes the universal set.

Subsets:

Every element in a set A is also an element of a set B, then A is called a subset of B.

We also say that B contains A.

This relationship is written:

$$A \subset B \quad \text{or} \quad B \supset A$$

If A is not a subset of B, i.e. if at least one element of A does not belong to B, we write $A \not\subset B$.

Example 4:

Consider the sets:

A = {1,3,4,5,8,9}, B = {1,2,3,5,7} and C ={1,5}

Then $C \subset A$ and $C \subset B$

since 1 and 5, the element of C, are also members of A and B.

But $B \not\subset A$ since some of its elements, e.g. 2 and 7, do not belong to A.

Furthermore, since the elements of A, B and C must also belong to the universal set U,

we have that U must at least the set {1,2,3,4,5,7,8,9}.

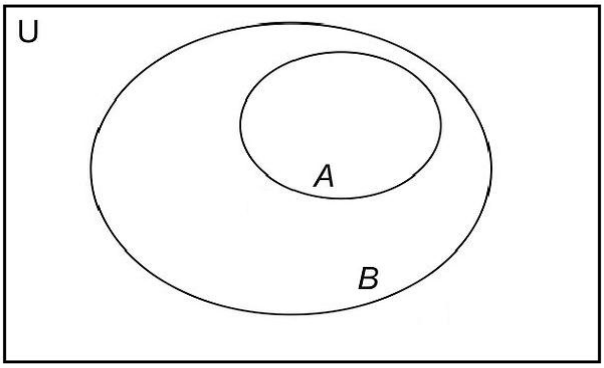
$A \subset B$: $\{\forall\ x \in A \quad \Rightarrow \quad x \in B$

$A \not\subset B$: $\{\exists\ x \in A \quad \text{but} \quad x \notin B$

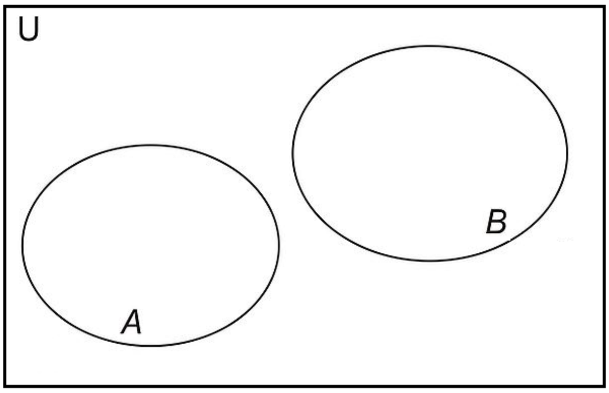
$\forall$: For all لكل

$\exists$: There exists يوجد على الاقل

The notion of subsets is graphically illustrated below:



A is entirely within B so $A \subset B$.



A and *B* are disjoint or ($A \cap B = \emptyset$) so we could write $A \not\subset B$ and $B \not\subset A$.

Set of numbers:

Several sets are used so often, they are given special symbols.

N = the set of natural numbers or positive integers

$$\mathbb{N} = \{0, 1, 2, 3, \ldots\}$$

Z = the set of all integers: . . . ,−2,−1, 0, 1, 2, . . .

$$\mathbb{Z} = \mathbb{N} \cup \{\ldots, -2, -1\}$$

Q = the set of rational numbers

$$\mathbb{Q} = \mathbb{Z} \cup \{\ldots, -1/3, -1/2, 1/2, 1/3, \ldots, 2/3, 2/5, \ldots\}$$

Where $Q = \{ a/b : a , b \in Z, b \neq 0\}$

R = the set of real numbers

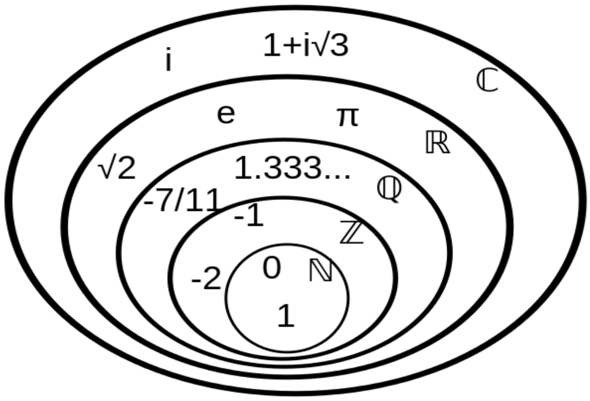
$$\mathbb{R} = \mathbb{Q} \cup \{\ldots, -\pi, -\sqrt{2}, \sqrt{2}, \pi, \ldots\}$$

C = the set of complex numbers

$$\mathbb{C} = \mathbb{R} \cup \{i, 1+i, 1-i, \sqrt{2}+\pi i, \ldots\}$$

Where $C = \{ x + iy ; x , y \in R; i = \sqrt{-1}\}$

Observe that $N \subset Z \subset Q \subset R \subset C$.



Theorem 1:

For any set A, B, C:

1. $\varnothing \subset A \subset U$.
2. $A \subset A$.
3. If $A \subset B$ and $B \subset C$, then $A \subset C$.
4. $A = B$ if and only if $A \subset B$ and $B \subset A$.

Set operations:

1) UNION:

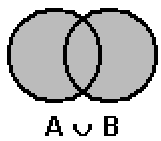
The *union* of two sets *A* and *B*, denoted by $A \cup B$, is the set of all elements which belong to *A* or to *B*;

$$A \cup B = \{ x : x \in A \text{ or } x \in B\}$$

Example

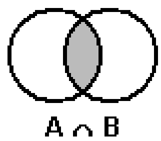
A={1,2,3,4,5} B={5,7,9,11,13}

$A \cup B$ = {1,2,3,4,5,7,9,11,13}



2) **INTERSECTION**

The intersection of two sets A and B, denoted by $A \cap B$, is the set of elements which belong to both A and B;



$$A \cap B = \{ x : x \in A \text{ and } x \in B\}.$$

Example 1

A={1,3,5,7,9} B={2,3,4,5,6}

The elements they have in common are 3 and 5

$A \cap B$ = {3,5}

Example 2

A={The English alphabet} B={vowels}

So $A \cap B$ = {vowels}

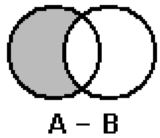
Example 3

A={1,2,3,4,5} B={6,7,8,9,10}

In this case A and B have nothing in common. $A \cap B = \varnothing$

3) THE DIFFERENCE:

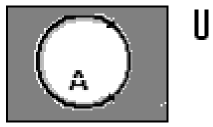
The difference of two sets A\B or A-B is those elements which belong to A but which do not belong to B.



$A \setminus B = \{x : x \in A, x \notin B\}$

4) COMPLEMENT OF SET:

Complement of set A^c or A' , is the set of elements which belong to U but which do not belong to A .



$A^c = \{x : x \in U, x \notin A\}$

Example 1:

let A={1,2,3}

B = {3,4}

U={1,2,3,4,5,6}

Find:

$A \cup B = \{1, 2, 3, 4\}$

$A \cap B = \{3\}$

$A - B = \{1, 2\}$

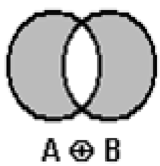
$A^c = \{4, 5, 6\}$

5) Symmetric difference of sets

The symmetric difference of sets A and B, denoted by $A \oplus B$, consists of those elements which belong to A or B but not to both. That is,

$A \oplus B = (A \cup B) \setminus (A \cap B)$ or

$A \oplus B = (A \setminus B) \cup (B \setminus A)$



Example:

Suppose U = N = {1, 2, 3, . . .} is the universal set.

Let A = {1, 2, 3, 4}, B= {3, 4, 5, 6, 7},

C = {2, 3, 8, 9}, E= {2, 4, 6, 8,. . .}

Then:

A^c = {5, 6, 7, . . .},

B^c = {1, 2, 8, 9, 10, . . .},

C^c ={1,4,5,6,7,10,…}

E^c ={1, 3, 5, 7, ...}

A\B = {1, 2},

A\C = {1, 4},

B\C = {4, 5, 6, 7},

A\E = {1, 3},

B\A = {5, 6, 7},

C\A = {8, 9},

C\B = {2, 8, 9},

E\A = {6, 8, 10, 12, . . .}.

Furthermore:

$A \oplus B = (A\backslash B) \cup (B\backslash A)$ = {1, 2, 5, 6, 7},

$B \oplus C$ = {2, 4, 5, 6, 7, 8, 9},

$A \oplus C = (A\backslash C) \cup (A\backslash C)$ = {1, 4, 8, 9},

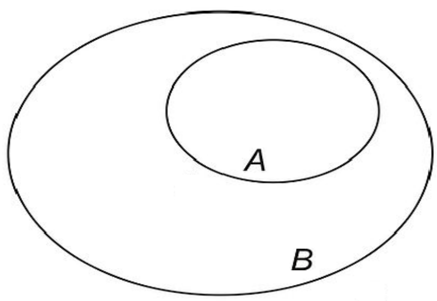
$A \oplus E$ = {1, 3, 6, 8, 10, . . .}.

Theorem 2 :

$A \subset B$,

$A \cap B = A$,

$A \cup B = B$ are equivalent



Theorem 3: (Algebra of sets)

Sets under the above operations satisfy various laws or identities which are listed below:

1- $A \cup A = A$
$A \cap A = A$

2- $(A \cup B) \cup C = A \cup (B \cup C)$ — Associative laws
$(A \cap B) \cap C = A \cap (B \cap C)$

3- $A \cup B = B \cup A$ — Commutativity
$A \cap B = B \cap A$

4- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ — Distributive laws
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

5- $A \cup \varnothing = A$ — Identity laws
$A \cap U = A$

6- $A \cup U = U$ — Identity laws
$A \cap \varnothing = \varnothing$

7- $(A^c)^c = A$ — Double complements

8- $A \cup A^c = U$ — Complement intersections and unions

$A \cap A^c = \varnothing$

9- $U^c = \varnothing$
$\varnothing^c = U$

10- $(A \cup B)^c = A^c \cap B^c$ — De Morgan's laws
$(A \cap B)^c = A^c \cup B^c$

Power set

The power set of some set S, denoted P(S), is the set of all subsets of S (including S itself and the empty set)

P(S) = {e : e $\subseteq$ S}

Example 1:

Let A = { 1,2 3}

Power set of set A = P(A)

={1},{2},{3},{1,2},{1,3},{2,3},{},A]

Example 2:

P({0,1})={{},{0},{1},{0,1}}

Classes of sets:

Collection of subset of a set with some properties

Example:

Suppose A = { 1,2 3} ,

let X2 be the class of subsets of A which contain exactly two elements of A. Then

class X0 = [{}]
class X1 = [{1},{2},{3}]
class X2 = [{1,2},{1,3},{2,3}]
class X3 = [{1,2,3}]

Cardinality

The cardinality of a set S, denoted **|S|**, is simply the number of elements a set has, so

|{a,b,c,d}| = 4,

The cardinality of the power set

Theorem:

If $|A| = n$ then $|P(A)| = 2^n$

(Every set with n elements has 2^n subsets)

Problem set

Write the answers to the following questions.

1. |{1,2,3,4,5,6,7,8,9,0}|
2. |P({1,2,3})|
3. P({0,1,2})
4. P({1})

Answers

1. 10
2. $2^3=8$
3. {{},{0},{1},{2},{0,1},0,2},{1,2},{0,1,2}}
4. {{},{1}}

The Cartesian product

The Cartesian Product of two sets is the set of all tuples made from elements of two sets.

We write the Cartesian Product of two sets A and B as A × B. It is defined as:

$$A \times B = \{(a, b) | a \in A \text{ and } b \in B\}$$

It may be clearer to understand from examples;

$$\{0, 1\} \times \{2, 3\} = \{(0, 2), (0, 3), (1, 2), (1, 3)\}$$
$$\{a, b\} \times \{c, d\} = \{(a, c), (a, d), (b, c), (b, d)\}$$
$$\{0, 1, 2\} \times \{4, 6\} = \{(0, 4), (0, 6), (1, 4), (1, 6), (2, 4), (2, 6)\}$$

Example:
If A = {1, 2, 3} and B = {x, y} then

A . B = {(1, x), (1, y), (2, x), (2, y), (3, x), (3, y)}
B . A = {(x, 1), (x, 2), (x, 3), (y, 1), (y, 2), (y, 3)}

It is clear that, the cardinality of the Cartesian product of two sets A and B is:

$$|A \times B| = |A||B|$$

A Cartesian Product of two sets A and B can be produced by making tuples of each element of A with each element of B; this can be visualized as a grid (which *Cartesian* implies) or table: if, *e.g.*,
A = { 0, 1 } and B = { 2, 3 }, the grid is

×		A	
		0	**1**
B	**2**	(0,2)	(1,2)
	3	(0,3)	(1,3)

Problem set

Answer the following questions:

1. {2,3,4}×{1,3,4}
2. {0,1}×{0,1}
3. |{1,2,3}×{0}|
4. |{1,1}×{2,3,4}|

Answers
1. {(2,1),(2,3),(2,4),(3,1),(3,3),(3,4),(4,1),(4,3),(4,4)}
2. {(0,0),(0,1),(1,0),(1,1)}
3. 3
4. 6

EXAMPLE

What is the Cartesian product $A \times B \times C$, where

$A = \{0, 1\}$,

$B = \{1, 2\}$, and

$C = \{0, 1, 2\}$?

Solution:

The Cartesian product $A \times B \times C$ consists of all ordered triples (a, b, c), where $a \in A$, $b \in B$, and $c \in C$. Hence,

$A \times B \times C = \{(0, 1, 0), (0, 1, 1), (0, 1, 2), (0, 2, 0), (0, 2, 1), (0, 2, 2), (1, 1, 0), (1, 1, 1), (1, 1, 2), (1, 2, 0), (1, 2, 1), (1, 2, 2)\}$.

EXAMPLE

Suppose that $A = \{1, 2\}$. It follows that

$A^2 = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$ and

$A^3 = \{(1,1,1), (1,1,2), (1,2,1), (1,2, 2), (2,1,1), (2,1,2), (2, 2,1), (2, 2, 2)\}$.

Computer Representation of Sets

There are various ways to represent sets using a computer. One method is to store the elements of the set in an unordered fashion. However, if this is done, the operations of computing the union, intersection, or difference of two sets would be time-consuming, because each of these operations would require a large amount of searching for elements. We will present a method for storing elements using an arbitrary ordering of the elements of the universal set. This method of representing sets makes computing combinations of sets easy.

Assume that the universal set U is finite (and of reasonable size so that the number of elements of U is not larger than the memory size of the computer being used). First, specify an arbitrary ordering of the elements of U, for instance:

$a1, a2, \ldots, an$. Represent a subset A of U with the bit string of length n, where the ith bit in this string is 1 if ai belongs to A and is 0 if ai does not belong to A.

Example

Let U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, and the ordering of elements of U has the elements in increasing order; that is, $a_i = i$. What bit strings represent

1- the subset of all odd integers in U,
2- The subset of all even integers in U, and
3- the subset of integers not exceeding 5 in U?

Solution:

1- The bit string that represents the set of odd integers in U, namely, {1, 3, 5, 7, 9}, has a one bit in the first, third, fifth, seventh, and ninth positions, and a zero elsewhere. It is:

   10 1010 1010.

2- we represent the subset of all even integers in U, namely, {2, 4, 6, 8, 10}, by the string 01 0101 0101.

3- The set of all integers in U that do not exceed 5, namely, {1, 2, 3, 4, 5}, is represented by the String 11 1110 0000.

Using bit strings to represent sets, it is easy to find complements of sets and unions, intersections, and differences of sets. To find the bit string for the complement of a set from the bit string for that set, we simply change each 1 to a 0 and each 0 to 1, because $x \in A$ if and only if $x \notin \bar{A}$. Note that this operation corresponds to taking the negation of each bit when we associate a bit with a truth value—with 1 representing true and 0 representing false.

Example

We have seen that the bit string for the set {1, 3, 5, 7, 9} (with universal set {1, 2, 3, 4,5, 6, 7, 8, 9, 10}) is 10 1010 1010.What is the bit string for the complement of this set?

Solution:

The bit string for the complement of this set is obtained by replacing 0s with 1s and vice versa. This yields the string 01 0101 0101, which corresponds to the set {2, 4, 6, 8, 10}.

To obtain the bit string for the union and intersection of two sets we perform bitwise Boolean operations on the bit strings representing the two sets.
The bit in the *i*th position of the bit string of the **union** is 1 if either of the bits in the *i*th position in the two strings is 1 (or both are 1), and is 0 when both bits are 0. Hence, the bit string for the union is the bitwise *OR* of the bit strings
for the two sets. The bit in the *i*th position of the bit string of the **intersection** is 1 when the bits in the corresponding position in the two strings are both 1, and is 0 when either of the two bits is 0 (or both are). Hence, the bit string for the intersection is the bitwise *AND* of the bit strings for the two sets.

EXAMPLE
The bit strings for the sets {1, 2, 3, 4, 5} and {1, 3, 5, 7, 9} are 11 1110 0000 and 10 1010 1010, respectively. Use bit strings to find the union and intersection of these sets.
Solution:
The bit string for the **union** of these sets is:

11 1110 0000 ∨ 10 1010 1010 = 11 1110 1010, which corresponds to the set {1, 2, 3, 4, 5, 7, 9}.
The bit string for the **intersection** of these sets is
11 1110 0000 ∧ 10 1010 1010 = 10 1010 0000, which corresponds to the set {1, 3, 5}.

Finite Sets and Counting Principle:

A set is said to be finite if it contains exactly m distinct elements, where m denotes some nonnegative integer. Otherwise, a set is said to be infinite.

For example:
- The empty set ∅ and the set of letters of English alphabet are finite sets,
- The set of even positive integers, {2,4,6,…..}, is infinite.

If a set A is finite, we let n(A) or #(A) denote the number of elements of A.

Example:
If A ={1,2,a,w} then
$n(A) = \#(A) = |A| = 4$

Lemma: If A and B are finite sets and disjoint Then $A \cup B$ is finite set and:
$$n(A \cup B) = n(A) + n(B)$$

Theorem (Inclusion–Exclusion Principle): Suppose A and B are finite sets. Then
$A \cup B$ and $A \cap B$ are finite and
$$|A \cup B| = |A| + |B| - |A \cap B|$$

That is, we find the number of elements in A or B (or both) by first adding n(A) and n(B) (inclusion) and then subtracting $n(A \cap B)$ (exclusion) since its elements were counted twice.
We can apply this result to obtain a similar formula for three sets:

Corollary:
If A, B, C are finite sets then
$$|A\cup B\cup C| = |A| + |B| + |C| - |A\cap B| - |A\cap C| - |B\cap C| + |A\cap B \cap C|$$

Example (1) :
A= {1,2,3}
B= {3,4}
C= {5,6}
$A \cup B \cup C = \{1,2,3,4,5,6\}$
$|A \cup B \cup C| = 6$

$|A| = 3$, $|B| = 2$, $|C| = 2$
$A \cap B = \{3\}$, $|A \cap B| = 1$
$A \cap C = \{\ \}$, $|A \cap C| = 0$

$B \cap C = \{\ \}$, $|B \cap C| = 0$

$A \cap B \cap C = \{\ \}$, $|A \cap B \cap C| = 0$

$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$

$|A \cup B \cup C| = 3 + 2 + 2 - 1 - 0 - 0 + 0 = 6$

Example (2):

Suppose a list A contains the 30 students in a mathematics class, and a list B contains the 35 students in an English class, and suppose there are 20 names on both lists. Find the number of students:

(a) only on list A

(b) only on list B

(c) on list $A \cup B$

Solution:

(a) List A has 30 names and 20 are on list B; hence 30 − 20 = 10 names are only on list A.

(b) Similarly, 35 − 20 = 15 are only on list B.

(c) We seek $n(A \cup B)$. By inclusion–exclusion,

$$n(A \cup B) = n(A) + n(B) - n(A \cap B) = 30 + 35 - 20 = 45.$$

Example (3):

Suppose that 100 of 120 computer science students at a college take at least one of languages: French, German, and Russian:

65 study French (F).

45 study German (G).

42 study Russian (R).

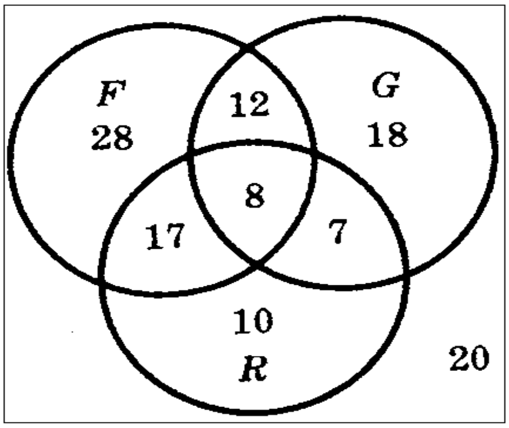
20 study French & German $F \cap G$.

25 study French & Russian $F \cap R$.

15 study German & Russian $G \cap R$.

Find the number of students who study:

1) All three languages ($F \cap G \cap R$)
2) The number of students in each of the eight regions of the Venn diagram



Solution:

|F ∪ G ∪ R| = |F| + |G| + |R| - |F ∩ G| - |F ∩ R| - |G ∩ R| + |F ∩ G ∩ R|

100 = 65+ 45 + 42 - 20 - 25 - 15 + |F ∩ G ∩ R|

100 = 92 + |F ∩ G ∩ R|

∴ |F ∩ G ∩ R| = 8 students study the 3 languages

20 – 8 = 12 (F ∩ G) - R

25 – 8 = 17 (F ∩ R) - G

15 – 8 = 7 (G ∩ R) - F

65 – 12 – 8 – 17 = 28 students study French only

45 – 12 – 8 7 = 18 students study German only

42 – 17 – 8 7 = 10 students study Russian only

120 – 100 = 20 students do not study any language

Mathematic induction:

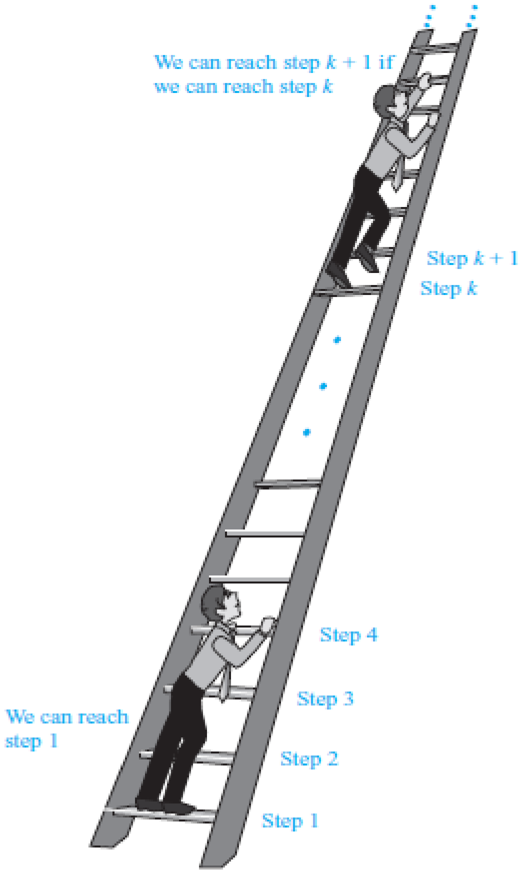
Suppose that we have an infinite ladder and we want to know whether we can reach every step on this ladder. We know two things:

l. We can reach the first rung of the ladder.

2. If we can reach a particular rung of the ladder, then we can reach the next rung.

Can we conclude that we can reach every rung? By (1), we know that we can reach the first rung of the ladder. Moreover, because we can reach the first rung, by (2), we can also reach the second rung; it is the next rung after the first rung. Applying

(2) again, because we can reach the second rung, we can also reach the third rung. Continuing in this way, we can show that we can reach the fourth rung, the fifth rung, and so on. For example, after 100 uses of (2), we know that we can reach the 101 st rung.



We can verify using an important proof technique called mathematical induction. That is, we can show that *P(n)* is true for every positive integer *n,* where *P(n)* is the statement that we can reach the *nth* rung of the ladder.

Mathematical induction is an important proof technique that can be used to prove assertions of this type. Mathematical induction is used to prove results about a large variety of discrete objects. For example, it is used to prove results about the complexity of algorithms, the correctness of certain types of computer programs, theorems about graphs and trees, as well as a wide range of identities and inequalities.

In general, mathematical induction can be used to prove statements that assert that *P(n)* is true for all positive integers *n*, where *P(n)* is a propositional function.

PRINCIPLE OF *MATHEMATICAL INDUCTION*
To prove that *P(n)* is true for all positive integers *n*, where *P(n)* is a propositional function, we complete two steps:

(i)BASIS *STEP:* We verify that *P(1)* is true.
(ii)INDUCTIVE STEP*:* We show that the conditional statement *P(k)→P(k* + I) is true for all positive integers *k*.

EXAMPLE1:
Show that if *n* is a positive integer, then

$$1+2+ \dots +n = \frac{n(n+1)}{2}.$$

Prove P (for n ≥ 1)

Solution*:*

Let *P(n)* be the proposition that the sum of the first *n* positive integers is *n(n* + 1)/2
We must do two things to prove that *P(n)* is true for
n = 1, 2, 3, ...
Namely, we must show that *P(1)* is true and that the conditional statement *P(k)* implies *P(k* + 1) is true for *k* = 1,2.3, ...

(i)BASIS STEP: P(1) IS true, because $1 = \frac{1(1+1)}{2}$

left side =1 & Right side =2/2 = 1

left side = Right side

(ii)INDUCTIVE STEP: For the inductive hypothesis we assume that *P(k)* holds for an arbitrary positive integer *k*. That is, we assume that P(k) is true

$$1+2+ \dots +k = \frac{k(k+1)}{2}$$

Under this assumption, it must be shown that *P(k* + 1) is true, namely, that

to prove that P(k+1) is true

$$1 + 2 + 3 + 4 + \dots + k + (k+1) = 1/2 * k * (k+1) + (k+1)$$

$$= \frac{k(k+1) + 2(k+1)}{2}$$

$$= \frac{(k+1)(k+2)}{2}$$

$$= 1/2 (k + 1)(k + 2)$$

So P is true for all n ≥ k

Example 2:

Conjecture a formula for the sum of the first *n* positive odd integers. Then prove your conjecture using mathematical induction.

Solution*:*

The sums of the first *n* positive odd integers for
n = 1,2,3,4,5 are:

1 = 1, 1 + 3 = 4, 1 + 3 + 5 = 9,
1+3+5+7=16, 1 + 3 + 5 + 7 + 9 = 25.

From these values it is reasonable to conjecture that the sum of the first *n* positive odd integers is n^2, that is,

$$1 + 3 + 5 + ... + (2n - 1) = n^2$$

We need a method to *prove* that this *conjecture* is correct, if in fact it is.
Let *P(n)* denote the proposition that the sum of the first *n* odd positive integers is n^2

(i)BASIS STEP: P(1) states that the sum of the first one odd positive integer is 1^2. This is true because the sum of the first odd positive integer is 1.

(ii)INDUCTIVE STEP:
we first assume the inductive hypothesis.
The inductive hypothesis is the statement that *P(k)* is true, that is,

$$1 + 3 + 5 + ... + (2k - 1) = k^2$$

(ii) n=k; Assuming P(k) is true,
We add (2k-1)+2 = 2K + 1 to both sides of P(k), obtaining:

$$1 + 3 + 5 + \dots + (2k - 1) + (2k + 1) = k^2 + (2k + 1)$$
$$= (k + 1)^2$$

Which is P(k + 1).
That is, P(k + 1) is true whenever P(k) is true.
By the principle of mathematical induction:
P is true for all n ≥.k.

Example 3:
Prove the following proposition (for $n \geq 0$):

$$P(n) : 1 + 2 + 2^2 + 2^3 + \dots + 2^n = 2^{n+1} - 1$$

solution :
(i) P(0) : left side =1
Right side $=2^1-1=1$

(ii) Assuming P(k) is true ; n=k

$$P(k) : 1 + 2 + 2^2 + 2^3 + \ldots +2^k = 2^{k+1} - 1$$

We add 2^{k+1} to both sides of P(k), obtaining

$$1 + 2 + 2^2 + 2^3 + \ldots +2^k + 2^{k+1} = 2^{k+1} - 1 + 2^{k+1}$$
$$= 2(2^{k+1}) - 1 = 2^{k+2} - 1$$

which is P(k +1). That is, P(k +1) is true whenever P(k) is true.
By the principle of induction:
P(n) is true for all n.

Homework:
Prove by induction:
1) 2 + 4 + 6 + ……. + 2n = n (n + 1)
2) 1 + 4 + 7 + ……. + (3n – 2) = 1/2 n (3n - 1)

Relations

The important aspect of the any set is the relationship between its elements. The association of relationship established by sharing of some common feature proceeds comparing of related objects. For example, assume a set of students, where students are related with each other if their sir names are same.

Conversely, if set is formed a class of students then we say that students are related if they belong to same class etc.
Relation is a predefined alliance of objects. The examples of relations are viz. brother and sister, and mathematical relation such as less than, greater than, and equal etc.

The relations can be classifying on the basis of its association among the objects. For example, relations said above are all association among two objects so these relations are called binary relation. Similarly, relations of parent to their children, boss and subordinates, brothers and sisters etc. are the examples of relations among three/more objects known as tertiary relation, quadratic relations and so on. In general an *n*-ary relation is the relation framed among *n* objects.

Product sets:
Consider two arbitrary sets A and B. The set of all ordered pairs (a,b) where $a \in A$ and $b \in B$ is called the product, or Cartesian product, of A and B.

$$A \times B = \{(a,b) : a \in A \text{ and } b \in B\}$$

Example
R denotes the set of real numbers and so :
$\mathbf{R}^2 = \mathbf{R} \times \mathbf{R}$ is the set of ordered pairs of real numbers.
The geometrical representation of $\mathbf{R}^2$ as points in the plane as in Fig.-1. Here each point *P* represents an ordered pair *(a, b)* of real numbers and vice versa; the vertical line through *P* meets the *x*-axis at *a*, and the horizontal line through *P* meets the *y*-axis at *b*. $\mathbf{R}^2$ is called the *Cartesian plane*.

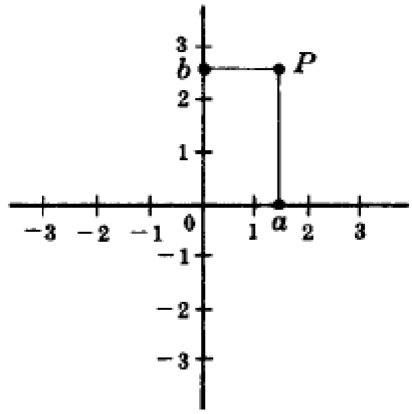


Fig. -1

Example:

a)Let A = {1,2} and B = {a ,b ,c} then

A × B = {(1,a), (1,b),(1,c),(2,a),(2,b),(2,c)}, Also,

A × A = {(1, 1), (1, 2), (2, 1), (2, 2)}

The order in which the sets are considered is important, so

$A \times B \neq B \times A$.

$n(A \times B) = n(A) \times n(B) = 2 \times 3 = 6$

Binary relation:

A relation between two objects is a binary relation and it is given by a set of ordered couples.

Let *A* and *B* be sets. A *binary relation* from *A* to *B* is a subset of *A* × *B*.

Suppose *R* is a relation from *A* to *B*. Then *R* is a set of ordered pairs where each first element comes from *A* and each second element comes from *B*. That is, for each pair $a \in A$ and $b \in B$, exactly one of the following is true:

(i) $(a, b) \in R$; we then say "*a* is *R*-related to *b*", written *aRb*.

(ii) $(a, b) \notin R$; we then say "*a* is not *R*-related to *b*", written $a \not R\, b$.

Example

(a) A = (1, 2, 3) and B = {x, y, z}, and let

R = {(1, y), (1, z), (3, y)}. Then R is a relation from A to B since R is a subset of A × B.

With respect to this relation

1Ry, 1Rz, 3Ry but $(1,x) \notin R$ & $(2,x) \notin R$

(b) Set inclusion $\subseteq$ is a relation on any collection of sets. For, given any pair of set *A* and *B*, either $A \subseteq B$ *or* or $A \nsubseteq B$.

(c) Consider the set *L* of lines in the plane. Perpendicularity, written "⊥," is a relation on *L*. That is, given any pair of lines *a* and *b*, either $a \perp b$ or *a* not⊥ *b*. Similarly, "is parallel to," written "||" is a relation on *L* since either $a \parallel b$ or $a \nparallel b$.

(d) Let *A* be any set. Then $A \times A$ and ∅ are subsets of $A \times A$ and hence are relations on *A* called the *universal relation* and *empty relation*, respectively.

Example :
Let A = {1, 2, 3}. Define a relation R on A by writing
(x, y) ∈ R , such that a≥b, list the element of R
aRb ↔ a≥b , a,b∈A
∴ R = {(1,1),(2,1), (2,2), (3,1), (3,2), (3,3)}.

Pictorial representation of relations

There are various ways of picturing relations:

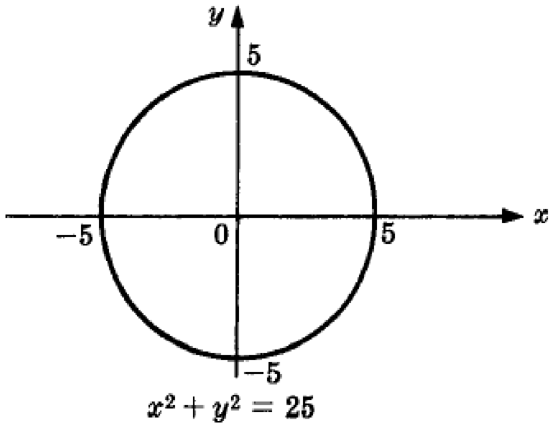
I - By coordinate plane

Let *S* be a relation on the set R of real numbers; that is, *S* is a subset of R^2 = **R** × **R**. Frequently, *S* consists of all ordered pairs of real numbers which satisfy some given equation

$E(x, y) = 0$ (such as $x2 + y2 = 25$).

Since R^2 can be represented by the set of points in the plane, we can picture *S* by emphasizing those points in the plane which belong to *S*. The pictorial representation of the relation is called the *graph* of the relation.

For example,
the graph of the relation $\boldsymbol{x^2+y^2 = 25}$ is a circle having its center at the origin and radius 5.



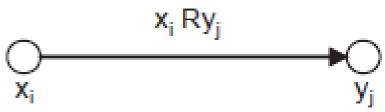
II -Directed Graphs of Relations on Sets

Relation can be represented pictorially by drawing its *graph* (directed graph). Consider a relation R be defined between two sets:

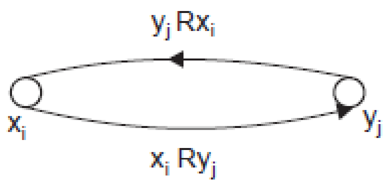
X = {*x*1, *x*2, …….., *x*l} and

Y = { *y*1, *y*2, …….., *ym*}

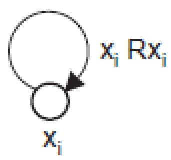
i.e., *xi* R *yj* , that is ordered couple (*xi*, *yj*) ∈ R where 1 ≤ *i* ≤ l and 1 ≤ *j* ≤ *m*. The elements of sets X and Y are represented by small circle called nodes. The existence of the ordered couple such as (*xi*, *yj*) is represented by means of an edge marked with an arrow in the direction from *xi* to *yj*.



While all nodes related to the ordered couples in R are connected by proper arrows, we get a directed graph of the relation R. For the ordered couples *xi* R*yj* and *yj* R*xi* we draw two arcs between nodes *xi* and *yj*,



If ordered couple is like *xi*R*xi* or (*xi*, *xi*) ∈ R then we get self loop over the node *xi*.



Example,

Relation R on the set A = {1, 2, 3, 4}:

R = {(1, 2), (2, 2), (2, 4), (3, 2), (3, 4), (4, 1), (4, 3)}

Fig. 3 shows the directed graph of R

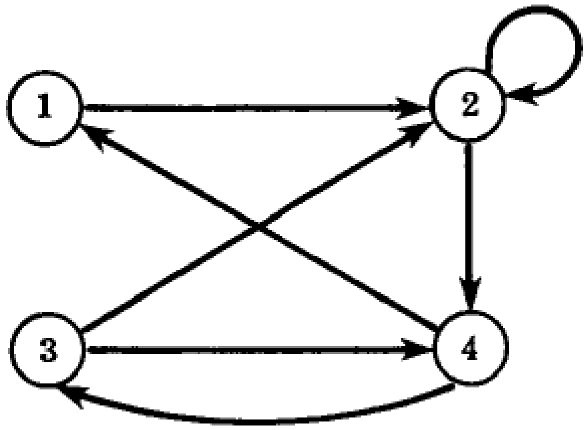


Fig. -3

III - matrix

Form a rectangular array (matrix) whose rows are labeled by the elements of *A* and whose columns are labeled by the elements of *B*. Put a 1 or 0 in each position of the array according as $a \in A$ is or is not related to $b \in B$. This array is called the *matrix of the relation*.

Example,

let A = {1, 2, 3} and B = {x, y, z}.

R = {(1,y),(1,z),(3,y)}

Fig. 4 shows the matrix of *R*.

	x	y	z
1	0	1	1
2	0	0	0
3	0	1	0

Fig. 4

IV - arrow from

Write down the elements of A and the elements of B in two disjoint disks, and then draw an arrow from a ∈ A to b ∈ B whenever a is related to b. This picture will be called the arrow diagram of the relation.

Fig. 5 pictures the relation R in the previous example by the arrow form.

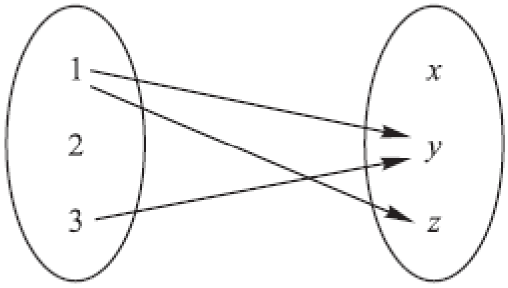


Fig. 5

Properties of binary relations (Types of relations)

Let R be a relation on the set A

1)Reflexive :

R is said to be *reflexive* if ordered couple $(x, x) \in R$ for $\forall x \in X$.

$\forall$ a $\in$A $\rightarrow$ aRa or (a,a) $\in$ R ; $\forall$ a, b $\in$A. .

Thus R is not reflexive if there exists a $\in$ A such that (a, a) $\notin$ R.

Example i:

Consider the following five relations on the set $A = \{1, 2, 3, 4\}$:

$R1 = \{(1, 1), (1, 2), (2, 3), (1, 3), (4, 4)\}$

$R2 = \{(1, 1)(1, 2), (2, 1), (2, 2), (3, 3), (4, 4)\}$

$R3 = \{(1, 3), (2, 1)\}$

$R4 = \emptyset$, the empty relation

$R5 = A \times A$, the universal relation

Determine which of the relations are reflexive.

Since *A* contains the four elements 1, 2, 3, and 4,
a relation *R* on *A* is reflexive if it contains the four pairs
$(1, 1), (2, 2), (3, 3)$, and $(4, 4)$.

Thus only *R*2 and the universal relation $R5 = A \times A$ are reflexive.

Note that *R*1,*R*3*R*3, and *R*4 are not reflexive since, for example, $(2, 2)$ does not belong to any of them.

Example ii

Consider the following five relations:

(1) Relation $\leq$ (less than or equal) on the set **Z** of integers.

(2) Set inclusion $\subseteq$ on a collection *C* of sets.

(3) Relation ⊥ (perpendicular) on the set L of lines in the plane.
(4) Relation ∥ (parallel) on the set L of lines in the plane.

Determine which of the relations are reflexive.

The relation (3) is not reflexive since no line is perpendicular to itself.
Also (4) is not reflexive since no line is parallel to itself.
The other relations are reflexive; that is,
$x \leq x$ for every $x \in \mathbf{Z}$,
$A \subseteq A$ for any set $A \in C$, and

2) **Symmetric** :
R is said to be *symmetric* if, ordered couple $(x, y) \in R$ and also ordered couple $(y, x) \in R$ for $\forall x, \forall y \in X$.

aRb → bRa ∀ a,b ∈A. [if whenever (a, b) ∈ R then (b, a) ∈ R.]

Thus R is not symmetric if there exists a, b ∈ A such that
(a, b) ∈ R but (b, a) ∉R.

Example
(a) Determine which of the relations in Example i are symmetric
R1 = {(1, 1), (1, 2), (2, 3), (1, 3), (4, 4)}
R2 = {(1, 1)(1, 2), (2, 1), (2, 2), (3, 3), (4, 4)}
R3 = {(1, 3), (2, 1)}
R4 = ∅, the empty relation
R5 = A × A, the universal relation

R1 is not symmetric since (1, 2) ∈ R1 but (2, 1) ∉ R1.
R3 is not symmetric since (1, 3) ∈ R3 but (3, 1) ∉ R3.
The other relations are symmetric.

(b) Determine which of the relations in Example ii are symmetric.

(1) Relation ≤ (less than or equal) on the set **Z** of integers.
(2) Set inclusion ⊆ on a collection *C* of sets.
(3) Relation ⊥ (perpendicular) on the set *L* of lines in the plane.
(4) Relation ∥ (parallel) on the set *L* of lines in the plane.

The relation ⊥ is symmetric since if line *a* is perpendicular to line *b* then *b* is perpendicular to *a*.
Also, ∥ is symmetric since if line *a* is parallel to line *b* then *b* is parallel to line *a*.
The other relations are not symmetric. For example:
3 ≤ 4 but 4 not≤ 3; {1, 2} ⊆ {1, 2, 3} but {1, 2, 3} not ⊆{1, 2}.

3**) Transitive** :
R is said to be *transitive* if ordered couple (*x*, *z*) ∈ R whenever both ordered couples (*x*, *y*) ∈ R and (*y*, *z*) ∈ R.
aRb ∧ bRc → aRc. that is, if whenever *(a, b), (b, c)* ∈ *R* then *(a, c)* ∈ *R*.
Thus *R* is not transitive if there exist *a, b, c* ∈ *R* such that *(a, b), (b, c)* ∈ *R* but *(a, c)* ∉ *R*.

Example
(a) Determine which of the relations in example i are transitive.

R1 = {(1, 1), (1, 2), (2, 3), (1, 3), (4, 4)}
R2 = {(1, 1)(1, 2), (2, 1), (2, 2), (3, 3), (4, 4)}
R3 = {(1, 3), (2, 1)}
R4 = ∅, the empty relation
R5 = A × A, the universal relation

The relation R3 is not transitive since (2, 1), (1, 3) ∈ R3 but (2, 3) ∉ R3. All the other relations are transitive.

(b) Determine which of the relations in example ii are transitive.

(1) Relation ≤ (less than or equal) on the set Z of integers.
(2) Set inclusion ⊆ on a collection C of sets.

(3) Relation ⊥ (perpendicular) on the set L of lines in the plane.
(4) Relation ∥ (parallel) on the set L of lines in the plane.

The relations ≤, ⊆, and | are transitive, but certainly not ⊥.
Also, since no line is parallel to itself, we can have
a ∥ b and b ∥ a, but a ∥ a. Thus ∥ is not transitive.

4)Equivalence relation :
A binary relation on any set is said an equivalence relation if it is **reflexive**, **symmetric**, and **transitive**.

R is an equivalence relation on *S* if it has the following three properties:

a - For every $a \in S$, aRa. (reflexive)
b- If aRb, then bRa. (symmetric)
c- If aRb and bRc, then aRc. (transitive)

5) **Irreflexive** :
$\forall a \in A \ (a,a) \notin R$

6) **AntiSymmetric** :
if $(x, y) \in R$ but $(y,x) \notin R$ unless $x = y$.
or
if aRb and bRa then a=b,
that is, **if $a \neq b$ and aRb then $(b,a) \notin R$**.
Thus R is not antisymmetric if there exist distinct elements a and b in A such that aRb and bRa.

the relations ≥,≤ and ⊆ are antisymmetric

Example
(a) Determine which of the relations in Example i are antisymmetric.

$R1 = \{(1, 1), (1, 2), (2, 3), (1, 3), (4, 4)\}$
$R2 = \{(1, 1)(1, 2), (2, 1), (2, 2), (3, 3), (4, 4)\}$
$R3 = \{(1, 3), (2, 1)\}$

R4 = ∅, the empty relation
R5 = A × A, the universal relation

*R*2 is not antisymmetric since *(*1, 2*)* and *(*2, 1*)* belong to *R*2, but 1 ≠ 2. Similarly,
the universal relation *R*3 is not antisymmetric.
All the other relations are antisymmetric.

(b) Determine which of the relations in Example ii are antisymmetric.
(1) Relation ≤ (less than or equal) on the set **Z** of integers.
(2) Set inclusion ⊆ on a collection *C* of sets.
(3) Relation ⊥ (perpendicular) on the set *L* of lines in the plane.
(4) Relation ∥ (parallel) on the set *L* of lines in the plane.

The relation≤ is antisymmetric since whenever $a \le b$ and $b \le a$ then $a = b$.
Set inclusion ⊆ is antisymmetric since whenever
$A \subseteq B$ and $B \subseteq A$ then $A = B$. Also,
The relations ⊥ and ∥ are not antisymmetric.

7) **Compatible** :
if a relation is only **reflexive** and **symmetric** then it is called a *compatibility* relation. So, we can say that: every equivalence relation is a compatibility relation, but not every compatibility relation is an equivalence relation.

Example:
Determine the properties of the relation ⊂ of set (inclusion on any collection of sets):
1) A ⊂ A for any set, so ⊂ is reflexive
2) A ⊂ B does not imply B ⊂ A, so ⊂ is not symmetric
3) If A ⊂ B and B ⊂ C then A ⊂ C, so ⊂ is transitive
4) ⊂ is reflexive, not symmetric & transitive, so ⊂ is not equivalence relations
5) A ⊂ A, so ⊂ is not Irreflexive

6) If A ⊂ B and B ⊂ A then A = B, so ⊂ is anti-symmetric
7) ⊂ *is reflexive and not symmetric then it is not compatibility* relation.

Example:
If A ={1,2,3} and R={(1,1),(1,2),(2,1),(2,3)}, is R equivalence relation ?
1) 2 is in A but (2,2) ∉ R, so R is not reflexive
2) (2,3) ∈ R but (3,2) ∉ R, so R is not symmetric
3) (1,2) ∈ R and (2,3) ∈ R but (1,3) ∉ R, so R is not transitive
So R is not Equivalence relation.

Example:
What is the properties of the relation **=** ?
1) a=a for any element a ∈ A, so = is reflexive
2) If a = b then b = a, so = is symmetric
3) If a = b and b = c then a = c, so = is transitive
4) = is (reflexive + symmetric + transitive), so = is equivalence
5) a = a, so = is not Irreflexive
6) If a = b and b = a then a = b, so = is anti-symmetric
7) = *is reflexive and symmetric then it is compatibility* relation.

Remark:
The properties of being symmetric and being antisymmetric are not negatives of each other.
For example,
the relation R = {(1, 3), (3, 1), (2, 3)} is neither symmetric nor antisymmetric.
On the other hand, the relation R = {(1, 1), (2, 2)} is both symmetric and antisymmetric.

From the directed graph of a relation we can easily examine some of its properties. For example if a relation is reflexive,

then we must get a self-loop at each node. Conversely if a relation is irreflexive, then there is no self-loop at any node.
For symmetric relation if one node is connected to another, thenthere must be a return arc from second node to the first node.
For antisymmetric relation there is no such direct return arc exist. Similarly we examine the transitivity of the relation in the directed graph.

8) Partial ordered relation
A binary relation R is said to be partial ordered relation if it is:
reflexive, antisymmetric, and transitive.

Example,
R={(w,w), (x, x), (y, y), (z, z),(w, x), (w, y), (w, z),(x, y), (x, z)}
In a partial ordered relation objects are related through superior/inferior criterion..

Example
In the arithmetic relation less than or equal to" ≤" (or greater than or equal to "≥") are partial ordered relations.

Since,
(1) Every number is equated to itself so it is reflexive.
(2) Also, if m and n are two numbers then ordered couple $(m, n) \in R$ if $m = n \Rightarrow n \nleq m$ so $(n, m) \notin R$ hence, relation is antisymmetric.
(3) if $(m, n) \in R$ and $(n, k) \in R \Rightarrow m = n$ and $n = k \Rightarrow m = k$ so $(m, k) \in R$ hence, R is transitive.

Example
The relation ⊆ of set inclusion is a partial ordering on any collection of sets since set inclusion has the three desired properties. That is,
(1) $A \subseteq A$ for any set A (reflexive).
(2) If $A \subseteq B$ and $B \subseteq A$, then $A = B$ (antisymmetric).
(3) If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$ (transitive).

Composition of relations:

When a relation is formed over stages such that let R be one relation defined from set X to Y, and S be another relation defined from set Y to Z,

then a relation W denoted by R ° S is a composite relation, i.e

W = R ° S ={(x,z) : ∃ x ∈ X for which (x,y) ∈ R and (y,z) ∈ S}

Composite relation W can also represented by a diagram.

Example :

let A ={1,2,3,4}

B = {a, b, c, d}

C = {x, y, z}

And

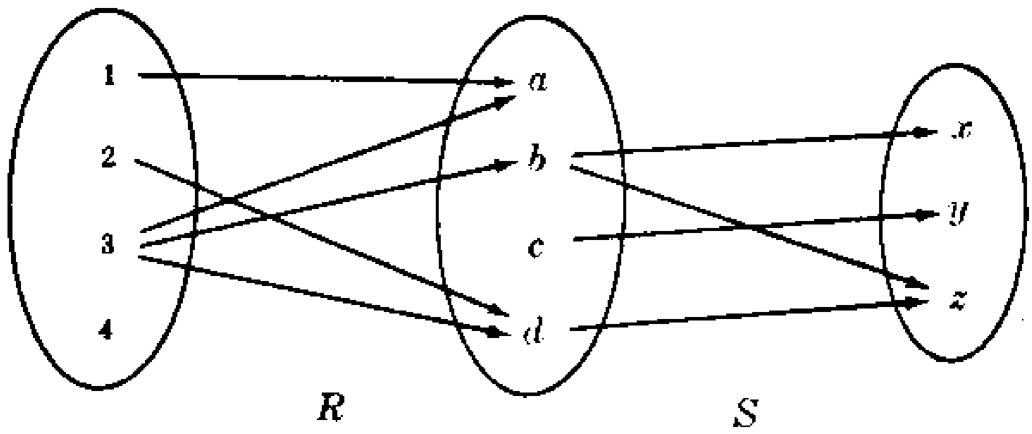
R = {(1,a),(2,d),(3,a),(3,d),(3,b)}

S = {(b,x),(b,z),(c,y),(d,z)}

Find R ° S ?

Solution :

1) **The first way by arrow form**



There is an arrow (path) from 2 to d which is followed by an arrow from d to z

2Rd and dSz ⇒ 2(R ° S) z

And 3(R◦S)x and 3(R◦S)z

So **R ° S = {(3,x),(3,z),(2,z)}**

2) The second way by matrix:

$$M_R = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{matrix} a & b & c & d \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad M_S = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{matrix} x & y & z \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

$R \circ S = M_R \,.\, M_S =$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{matrix} x & y & z \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

$R \circ S = \{(2,z),(3,x),(3,z)\}$

Example,
let R1 = {(*p*, *q*), (*r*, *s*), (*t*, *u*), (*q*, *s*)} and
R2 = {(*q*, *r*), (*s*, *v*), (*u*, *w*)} are two relations then,

R1 ∘ R2 = {(*p*, *r*), (*r*, *v*), (*t*, *w*), (*q*, *v*)}, and

R2 ∘ R1 = {(*q*, *s*)}

Home work:

Consider the following relations on the set $A = \{1, 2, 3\}$:

$R = \{(1, 1), (1, 2), (1, 3), (3, 3)\}$,

$S = \{(1, 1)(1, 2), (2, 1)(2, 2), (3, 3)\}$,

$T = \{(1, 1), (1, 2), (2, 2), (2, 3)\}$

$\varnothing$ = empty relation

$A \times A$ = universal relation

- Determine whether or not each of the above relations on *A* is:
  (*1*) reflexive;
  (*2*) symmetric;
  (*3*) transitive;
  (*4*) antisymmetric.
  (5) Irreflexive
  (6) compatibility
  7) Partial ordered relation

Function:

In many instances we assign to each element of a set a particular element of a second set. For example, suppose that each student in a discrete mathematics class is assigned a letter grade from the set {*A*,*B*,*C*,*D*, *F*}. And suppose that the grades are *A* for Adams, *C* for Chou, *B* for Goodfriend, *A* for Rodriguez, and *F* for Stevens. This assignment is an example of a function. This assignment is an example of a function. The concept of a function is extremely important in mathematics and computer science.

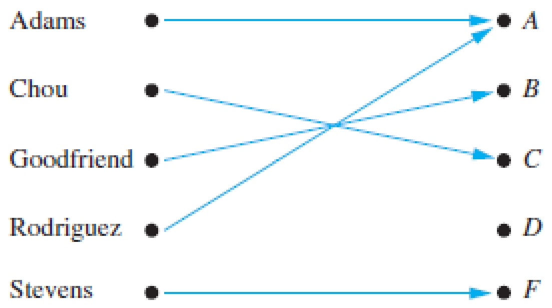


Fig. 1 Assignment of Grades in a Discrete Mathematics Class.

Function is a class of relation. it establishes the relationship between objects. For example, in computer system input is fed to the system in form of data or objects and the system generates the output that will be the function of input. So, function is the mapping or transformation of objects from one form to other.

Definition:

Let *A* and *B* be nonempty sets. A *function F: A→B is a rule which associates with* ***each*** *element of A a* ***unique*** *element in B.*

EXAMPLE 1

Let *R* be the relation with ordered pairs (Abdul, 22), (Brenda, 24), (Carla, 21), (Desire, 22), (Eddie, 24), and (Felicia, 22). Here each pair consists of a graduate student and this student's age. Specify a function determined by this relation.

Solution:

If *f* is a function specified by *R*, then

f (Abdul) = 22,

f (Brenda) = 24,
f (Carla) = 21,
f (Desire) = 22,
f (Eddie) = 24, and
f (Felicia) = 22. (Here, *f (x)* is the age of *x*, where *x* is a student.)

EXAMPLE 2

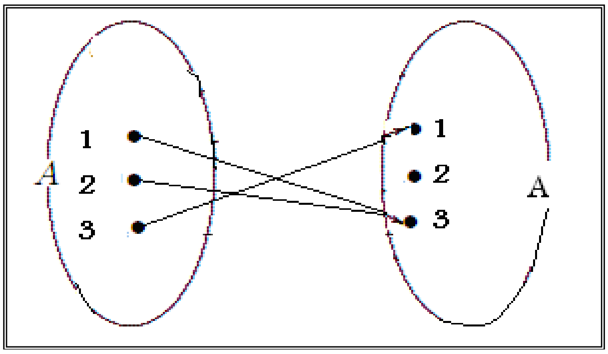
Consider the function f (x) = x^3, i.e., f assigns to each real number its cube. Then the image of 2 is 8, and so we may write f (2) = 8.

Example 3 :

consider the following relation on the set A={1,2,3}

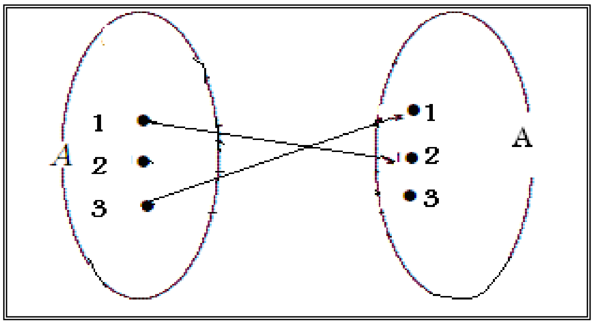
F = {(1,3),(2,3),(3,1)}

F is a function

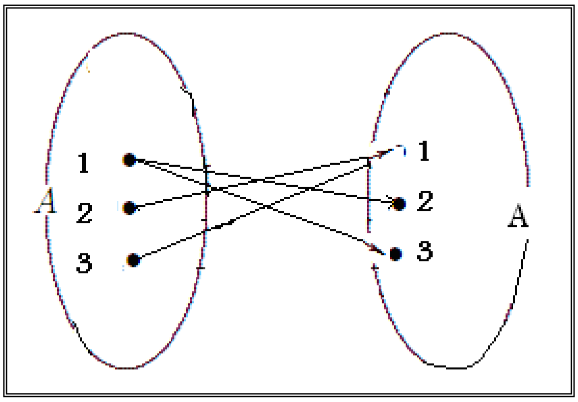


G = {1,2},(3,1)}

G is not a function from A to A



H = {(1,3),(2,1),(1,2),(3,1)}
H is not a function .



Classification of functions:
(One-to-one ,onto and invertible functions) :

Some functions never assign the same value to two different domain elements. These functions are said to be one-to-one.

1) **One –to-one** :

a function F:A→B is said to be one-to-one if different elements in the domain (A) have distinct images.

Or If F(a) =F(a') ⇒ a = a'

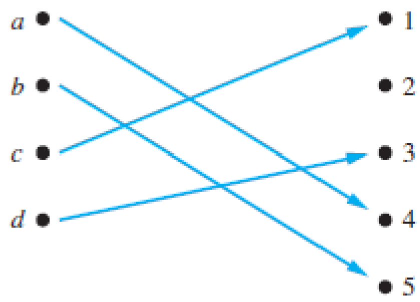


Fig 2: A One-to-One Function.

2) **Onto** :

F:A→B is said to be an onto function if each element of B is the image of some element of A.

$\forall\ b \in B \quad \exists \quad a \in A : F(a) = b$

EXAMPLE

Let *f* be the function from {*a*, *b*, *c*, *d*} to {1, 2, 3} defined by

f (*a*) = 3,
f (*b*) = 2,
f (*c*) = 1, and
f (*d*) = 3.

Is *f* an **onto** function?

***Solution**:*

Because all three elements of the codomain are images of elements in the domain, we see that *f* is onto. This is illustrated in Figure 3.

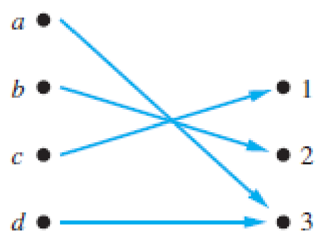


Fig. 3 An Onto Function

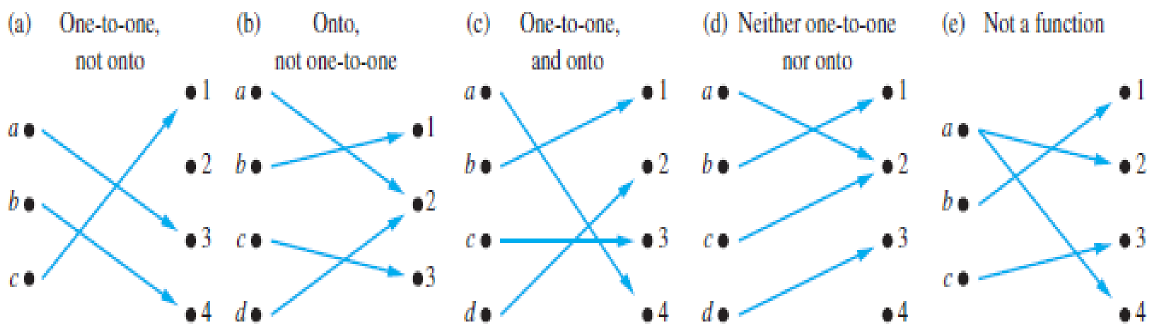


Fig 4 . Examples of Different Types of Correspondences.

3) **Invertible (One-to-one correspondence)**

F:A→ B is invertible if and only if F is **both** one-to-one and onto.

F:A→ B is invertible if its inverse relation f^{-1} is a function F:B →A

$$F^{-1} :\{(b,a) \ \forall \ (a,b) \in F\}$$

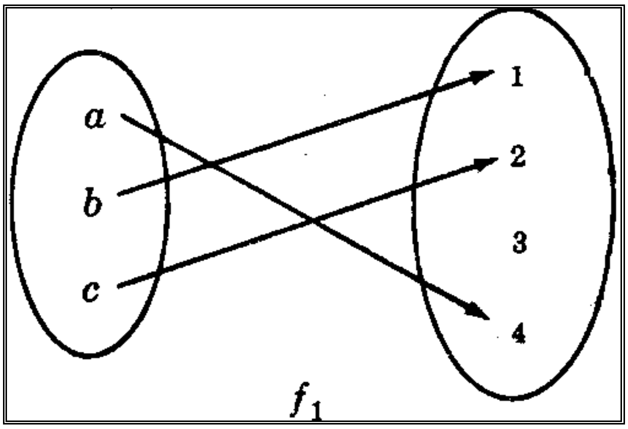
EXAMPLE

Let *f* be the function from {*a, b, c, d*} to {1, 2, 3, 4} with
f (a) = 4,
f (b) = 2,
f (c) = 1, and
f (d) = 3. Is *f* an **invertible**?

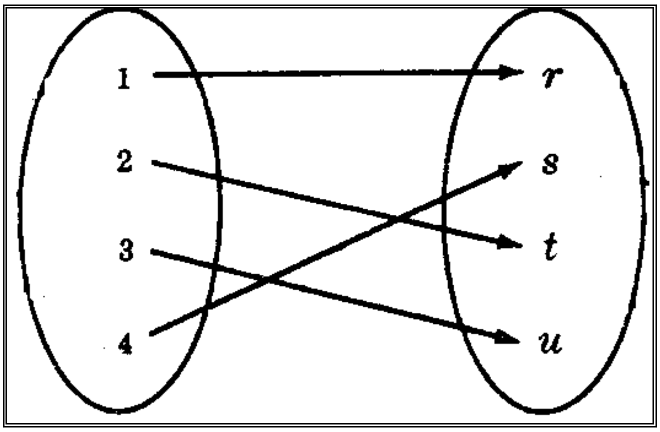
Solution:

The function *f* is one-to-one and onto.
It is one-to-one because no two values in the domain are assigned the same function value.
It is onto because all four elements of the codomain are images of elements in the domain. Hence, *f* is a invertible.

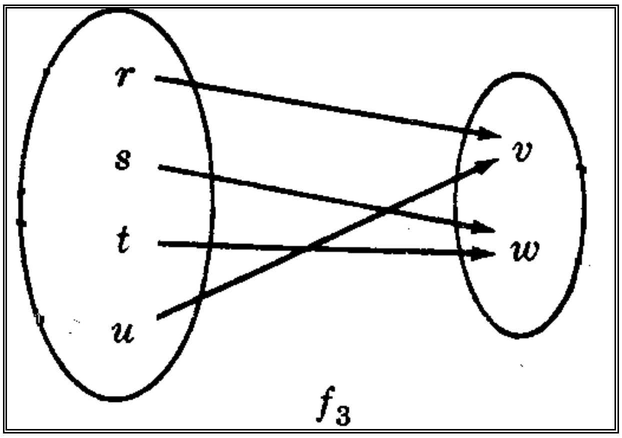
Figure 4 displays four functions where
the first is one-to-one but not onto,
the second is onto but not one-to-one,
the third is both one-to-one and onto, and
the fourth is neither one-to-one nor onto.
The fifth correspondence in Figure 4 is not a function, because it sends an element to two different elements.



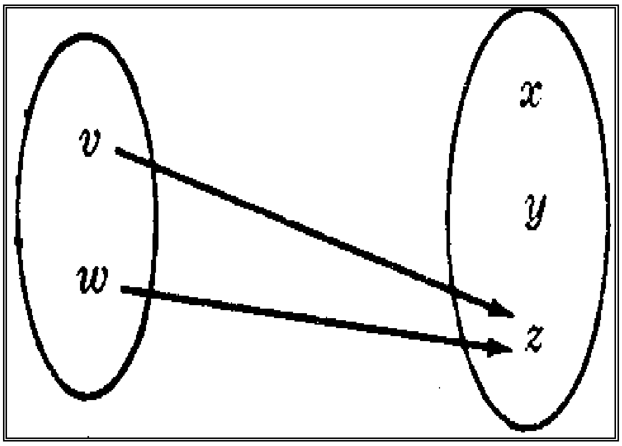
one to one but not onto
(3∈B but it is not the image under f1)



both one to one & onto
(or one to one correspondence between A and B)



not one to one & onto



not one to one & not onto

Graph of a function:

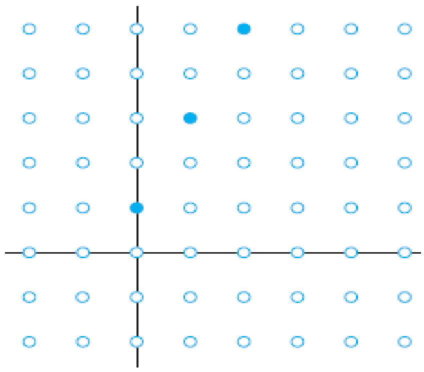
We can associate a set of pairs in $A \times B$ to each function from A to B. This set of pairs is called the **graph** of the function and is often displayed pictorially to aid in understanding the behavior of the function.

EXAMPLE

Display the graph of the function $f(n) = 2n + 1$ from the set of integers to the set of integers.

Solution:

The graph of f is the set of ordered pairs of the form $(n, 2n + 1)$, where n is an integer.

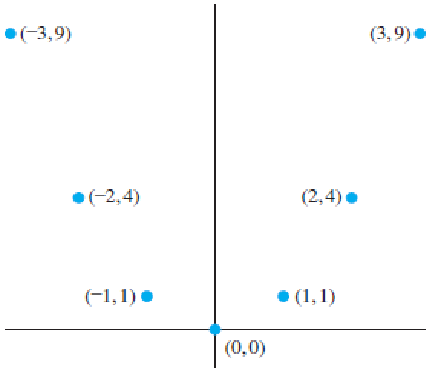


EXAMPLE

Display the graph of the function $f(x) = x^2$ from the set of integers to the set of integers.

Solution:

The graph of f is the set of ordered pairs of the form $(x, f(x)) = (x, x^2)$, where x is an integer.



By a *real polynomial function*, we mean a function *f*: **R** → **R** of the form

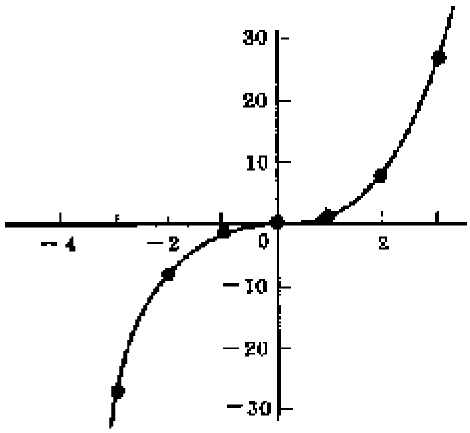
$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

where the a_i are real numbers. Since **R** is an infinite set, it would be impossible to plot each point of the graph. However, the graph of such a function can be approximated by first plotting some of its points and then drawing a smooth curve though these points. The table points are usually obtained from a table where various values are assigned to x and the corresponding value of f(x) computed.

Example: let f: R→R and **f(x)= x³** , find f(x)

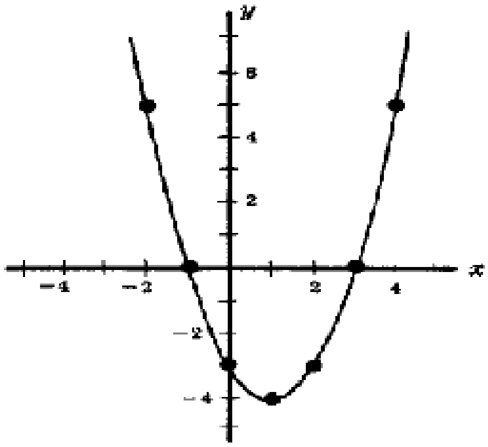
f(3) = 3^3 = 27
f(-2) = $(-2)^3$ = -8

x	$f(x)$
−3	−27
-2	8
1	−1
0	0
1	1
2	8
3	27



Example : let f: R→R and ***f (x)* = *x*² − 2*x* – 3**, , find f(x)

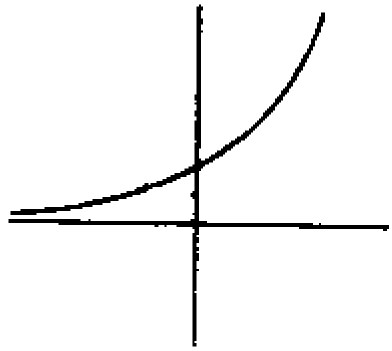
x	$f(x)$
−2	5
−1	0
0	−3
1	−4
2	−3
3	0
4	5



Geometrical Characterization of One-to-One and Onto Functions

For the functions of the form f : R → R. the graphs of such functions may be plotted in the Cartesian plane and functions may be identified with their graphs, so the concepts of being one-to-one and onto have some geometrical meaning :

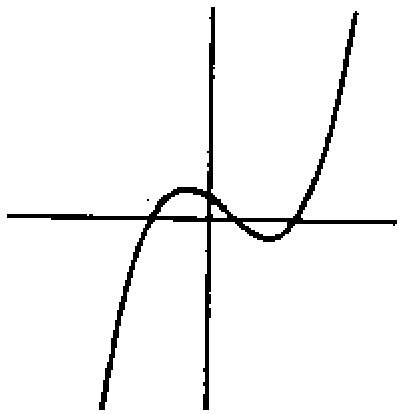
(1) f :R → R is said to be <u>one-to-one</u> if there are no 2 distinct pairs (a1,b) and (a2,b) in the graph one-to-one or if each horizontal line intersects the graph of f in <u>at most one point.</u>



$f_2(x) = 2^x$

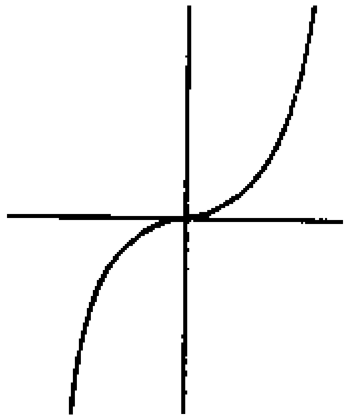
---

(2) f :R → R is an <u>onto</u> function if each horizontal line intersects the graph of f <u>at one or more points (at least once)</u>

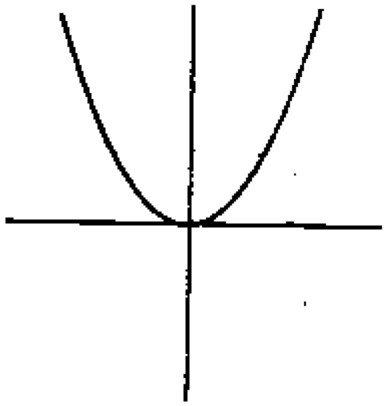


$f_3(x) = x^3 - 2x^2 - 5x + 6$

(3) if f is both one-to-one and onto, i.e. invertible, then each horizontal line will intersect the graph of f **<u>at exactly one</u> point**.



$f_4(x) = x^3$



$f_1(x) = x^2$

f(x)NOT (ONE-TO-ONE) & NOT (ONTO)

Sequences of sets

A sequence is a discrete structure used to represent an ordered list.

For example,

1, 2, 3, 5, 8 is a sequence with five terms (called a *list*)

1, 3, 9, 27, 81 , . . . , 3*n*, . . . is an infinite sequence.

A *sequence* is a function from subset of the set of integers (usually either the set {0, 1, 2, . . .} or the set {1, 2, 3, . . .}) to a set *S*.. The notation a_n is used to denote the image of the integer n that called the term of the sequence and used to describe the sequence . Thus a sequence is usually denoted by

*a*1, *a*2, *a*3, . . .

We describe sequences by listing the terms of the sequence in order of increasing subscripts.

EXAMPLE 1
Consider the sequence $\{a_n\}$, where

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

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

$a_1, a_2, a_3, a_4, \ldots ,$

starts with

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

EXAMPLE 2
a-The sequences $\{b_n\}$ with $b_n = (-1)^n$
if we start at n = 0, the list of terms begins with 1,−1, 1,−1, 1, . . .

b-The sequences $\{c_n\}$ with $c_n = 2 \times 5^n$
if we start at n = 0, the list of terms begins with

2, 10, 50, 250, 1250, . . .

c- The sequences $\{d_n\}$ with $d_n = 6 \times (1/3)^n$
if we start at n = 0, The list of terms begins with

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

d- The sequences $\{b_n\}$ with $b_n = 2^{-n}$
if we start at n = 0, The list of terms begins with

$$1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$$

e- The sequences {bn} with $a_n = \frac{1}{n}$
if we start at n = 1, The list of terms begins with

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

RECURSIVELY DEFINED FUNCTIONS

A function is said to be *recursively defined* if the function definition refers to itself. In order for the definition not to be circular, the function definition must have the following two properties:

(1) There must be certain arguments, called *base values*, for which the function does not refer to itself.

(2) Each time the function does refer to itself, the argument of the function must be closer to a base value.

A recursive function with these two properties is said to be *well-defined*.

Factorial Function

The product of the positive integers from 1 to *n*, inclusive, is called "*n* factorial" and is usually denoted by *n*!. That is,

$$n! = n(n-1)(n-2) \cdot \cdot \cdot 3 \cdot 2 \cdot 1$$

where

0! = 1, so that the function is defined for all nonnegative integers. Thus:

We have: f(0) = 0! = 1

f (1) = 1! = 1,

f (2) = 2! = 1 · 2 = 2,

f (6) = 6! = 1 · 2 · 3 · 4 · 5 · 6 = 720,

and

f (20) = 1x 2 x3x 4 x 5 x 6 x 7 x 8 x 9 x 10 x 11x 12 x 13 x 14 x 15x16 x 17 x 18 x 19 x20 = 2,432,902,008,176,640,000.

the factorial function grows extremely rapidly as *n* grows.

This is true for every positive integer n; that is,

$$n! = n \cdot (n - 1)!$$

Accordingly, the factorial function may also be defined as follows:

Definition of Factorial Function:

(a) If $n = 0$, then $n! = 1$.

(b) If $n > 0$, then $n! = n \cdot (n - 1)!$

The definition of $n!$ is recursive, since it refers to itself when it uses $(n - 1)!$. However:
(1) The value of $n!$ is explicitly given when $n = 0$ (thus 0 is a base value).
(2) The value of $n!$ for arbitrary n is defined in terms of a smaller value of n which is closer to the base value 0.

Accordingly, the definition is not circular, or, in other words, the function is well-defined.

EXAMPLE 7: the 4! Can be calculated in 9 steps using the recursive definition .

(1) $4! = 4 \cdot 3!$
(2) $3! = 3 \cdot 2!$
(3) $2! = 2 \cdot 1!$
(4) $1! = 1 \cdot 0!$
(5) $0! = 1$
(6) $1! = 1 \cdot 1 = 1$
(7) $2! = 2 \cdot 1 = 2$
(8) $3! = 3 \cdot 2 = 6$
(9) $4! = 4 \cdot 6 = 24$

Fibonacci Sequence

The Fibonacci sequence is a particularly useful sequence that is important for many applications, including modeling the population growth of rabbits. It is usually denoted by F0, F1, F2, . . and can be defined by:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . .

That is, F0 = 0 and F1 = 1 and each succeeding term is the sum of the two preceding terms. For example, the next two terms of the sequence are

34 + 55 = 89 and
55 + 89 = 144

Fibonacci Sequence can be defined:

(a) If n = 0, or n = 1, then Fn = n.
(b) If n > 1, then Fn = $F_{n-1} + F_{n-2}$.

Where : The base values are 0 and 1, and the value of Fn is defined in terms of smaller values of n which are closer to the base values.
Accordingly, this function is well-defined.

Graphs:

Graphs are discrete structures consisting of vertices and edges that connect these vertices, so a graph G(V,E) consists of:

(i) V, a nonempty set of *vertices* (or *nodes*).

(ii) *E*, a set of *edges*. Each *edge* has either one or two vertices associated with it, called its *endpoints*.

Graphs are used in a wide variety of models with computer science such as communication network, logical design, transportation networks, formal languages, compiler writing and retrieval.

For example: in a communication network, where computers can be represented by vertices and communication links by edges. A graph in which each edge connects two different vertices and where no two edges connect the same pair of vertices is called a **simple graph**.

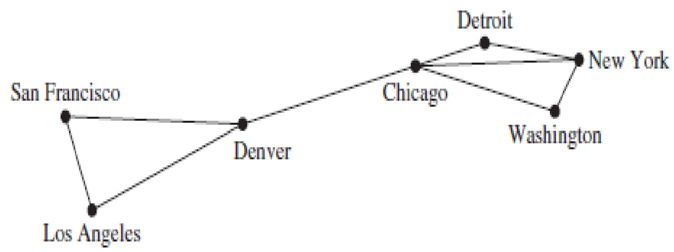


Figure (1): simple graph

A computer network may contain multiple links between data centers, as shown in Figure 2. To model such networks we need graphs that have more than one edge connecting the same pair of vertices. Graphs that may have **multiple edges** connecting the same vertices are called **multigraphs**.

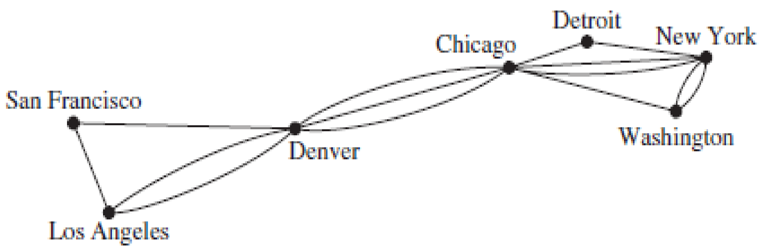


Figure (2): multigraphs

Sometimes a communications link connects a data center with itself, perhaps a feedback loop for diagnostic purposes. Such a network is illustrated in Figure 3. To model this network we need to include edges that connect a vertex to itself. Such edges are called **loops**,

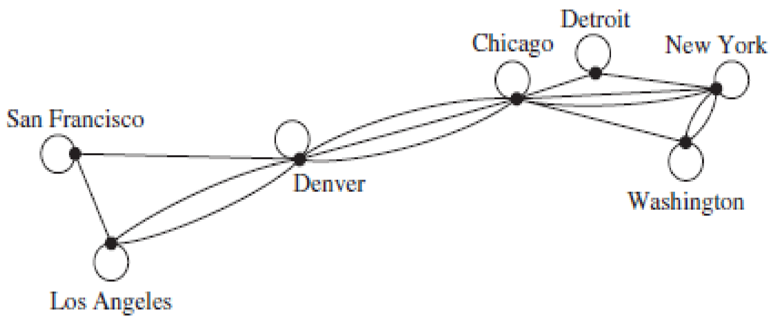


Figure (3): multigraphs with loops

In a computer network, some links may operate in only one direction (such links are called single duplex lines). This may be the case if there is a large amount of traffic sent to some data centers, with little or no traffic going in the opposite direction. Such a network is shown in Figure 4. To model such a computer network we use a **directed graph**. Each edge of a directed graph is associated to an ordered pair.

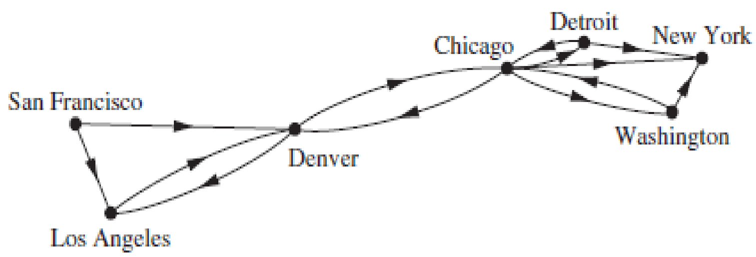


Figure (4): directed graph

For example we have in Figure (5) the graph G(V,E) where: V consists of four vertices A, B, C, D ; and, E consists of five edges

e1 ={A,B},
e2 = {B,C},
e3 = {C, D},

e4 = {A, C},
e5 = {B, D}.

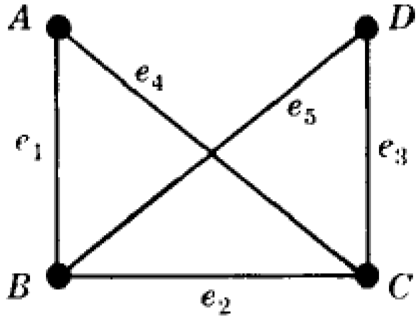


Figure (5)

Vertices u and v are said to be **adjacent** if there is an edge e={u,v}. In such a case, u and v are called the endpoints of e, and e is said to connect u and v. Also, the edge e is said to be **incident** on each of its endpoints u and v.

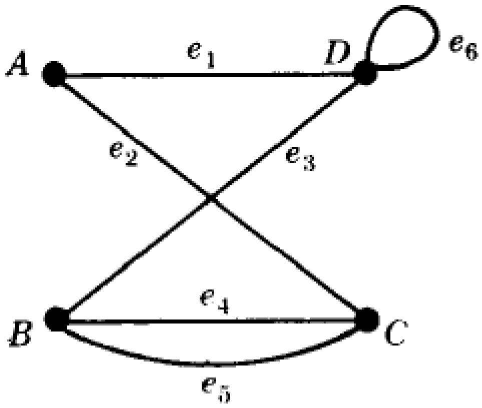


Figure 6: multigraph with: 1) multiple edges e4 & e5
2) a loop e6

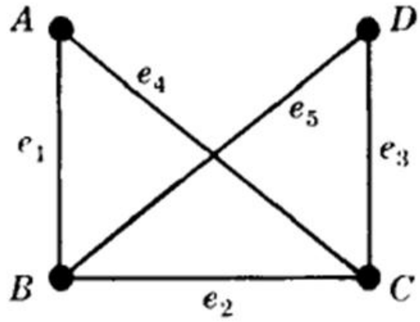
Degree :

The degree of a vertex v [deg(v)], is equal to the number of edges which are incident on v. since each edge is counted twice in counting the degrees of the vertices of a graph.

Theorem: The sum of the degrees of the vertices of a graph is equal to twice the number of edges. Let *G = (V ,E)* be an undirected graph with *m* edges. Then

$$2m = \sum_{v \in V} deg(v).$$

For example, in the figure (5) we have



deg(A) = 2,
deg(B) = 3,
deg(C) = 3,
deg(D) = 2

The sum of the degrees = twice the number of edges = 2×5=10

EXAMPLE 1: How many edges are there in a graph with 10 vertices each of degree six?
Solution: Because the sum of the degrees of the vertices is
$6 \times 10 = 60$, it follows that $2m = 60$
where m is the number of edges. Therefore, $m = 30$.

A vertex is said to be **even** or **odd** according as its degree is an even or odd number. Thus A and D are even vertices whereas B and C are odd vertices.
This theorem also holds for multigraphs where a loop is counted twice towards the degree of its endpoint. For example, in Fig (6) we have deg (D) = 4 since the edge e6 is counted twice; hence D is an even vertex.
A vertex of degree zero is called an isolated vertex.

Subgraphs
Consider a graph G = G(V,E) and a graph H = H(V', E') is called a subgraph of G if the vertices and edges of H are contained in the vertices and edges of G, that is, if $V' \subseteq V$ and $E' \subseteq E$.

Sometimes we need only part of a graph to solve a problem. For instance, we may care only about the part of a large computer network that involves the computer centers in New York, Denver, Detroit, and Atlanta. Then we can ignore the other computer centers and all telephone lines not linking two of these specific four computer centers. In the graph model for the large network, we can remove the vertices corresponding to the computer centers other than the four of interest, and we can remove all edges incident with a vertex that was removed. When edges and vertices are removed from a graph, without removing endpoints of any remaining edges, a smaller graph is obtained. Such a graph is called a **subgraph** of the original graph.

EXAMPLE 2: The graph *G* shown in Figure 7 is a subgraph of *K*5. If we add the edge connecting a, b, *c* and *e* to *G*, we obtain the subgraph induced by *W* = {*a*, *b*, *c*, *e*}.

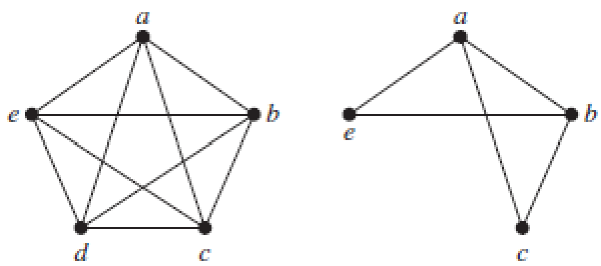


Figure 7

Connectivity :

Many problems can be modeled with paths formed by traveling along the edges of graphs. For instance, the problem of determining whether a message can be sent between two computers using intermediate links can be studied with a graph model. Problems of efficiently planning routes for mail delivery, garbage pickup, diagnostics in computer networks, and so on can be solved using models that involve paths in graphs.

a walk is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph. As

the path travels along its edges, it visits the vertices along this walk, that is, the endpoints of these edges.
A **walk** in a multigraph G consists of an alternating sequence of vertices and edges of the form:

v0, e1, v1, e2, v2,……., en-1,vn-1,en,vn

where each edge e_i contains the vertices v_{i-1} and v_i (which appear on the sides of ei in the sequence).

Length of walk : is the number n of edges. When there is no ambiguity, we denote a path by its sequence of vertices

(v0, v1, . . . , vn).

Closed walk: the walk is said to be closed if v0 = vn .
Otherwise, we say that the walk is from v0 to vn.

Trail: is a walk in which all edges are distinct.
Path: is a walk in which all vertices are distinct.
Cycle: is a closed walk such that all vertices are distinct except v1 = vn, A cycle of length k is called a k-cycle.

EXAMPLE 1

In the simple graph shown in Figure 8:

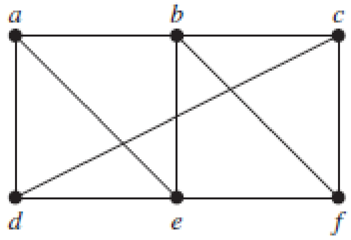


Figure 8

a, *d*, *c*, *f* , *e* is a path of length 4, because {*a, d*}, {*d, c*}, {*c, f* }, and {*f, e*} are all edges. However,

d, *e*, *c*, *a* is not a path, because {*e, c*} is not an edge. Note that

b, *c*, *f* , *e*, *b* is a circuit of length 4 because {*b, c*}, {*c, f* }, {*f, e*}, and {*e, b*} are edges, and this path begins and ends at *b*.

The walk *a*, *b*, *e*, *d*, *a*, *b*, which is of length 5, is not path because it contains the edge {*a*, *b*} twice.

Example: Consider the graph in figure (9), then

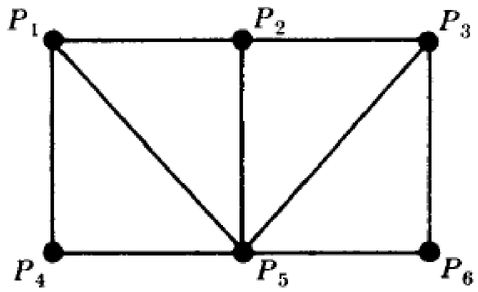


Figure (9)

The sequence: (P4, P1, P2, P5, P1, P2, P3, P6) is a walk from P4 to P6. It is not a trail since the edge {P1,P2} is used twice.

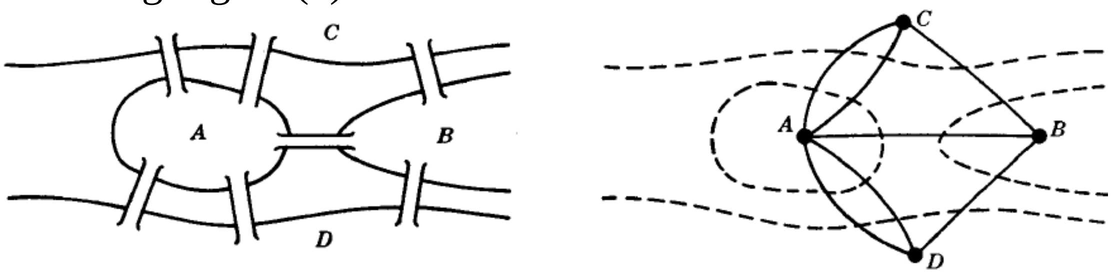
The sequence: (P4, P1, P5, P3, P2, P6) Is not a walk since there is no edge {P2, P6}.

The sequence: (P4, P1, P5, P2, P3, P5, P6) is a trail since no edge is used twice; but it is not a path since the vertex P5 is used twice.
The sequence: (P4, P1, P5, P3, P6) Is a path from P4 to P6.

The shortest path from P4 to P6 is (P4, P5, P6) which has length = 2 (2 edges only)
The distance between vertices u & v d(u,v) is the length of the shortest path d(P4,P6) = 2.

The Bridges of konigsberg, traversable multigraphs

The eighteenth-century East Prussian town of Konigsberg included two islands and seven bridges. Question: beginning anywhere and ending anywhere, can a person walk through town crossing all seven bridges but not crossing any bridge twice? The people of Konigsberg wrote to the celebrated Swiss mathematician L. Euler about this question. Euler proved in 1736 that such a walk is impossible. He replaced the islands and two side of the river by points and the bridges by curves, obtaining Fig 12 (b).



(a) Königsberg in 1736 (b) Euler's graphical representation

Fig. 12

Konigsberg graph is a multigraph, A multigraph is said to traversable if it can be drawn without any breaks and without repeating any edge. That is if there is a walk includes all vertices and uses each edge exactly once. Such a walk must be a trail (no edge is used twice) and will be called a *traversable trail*.

We now show how Euler proved that the konigsberg multigraph is not traversable and the walk in it is impossible. Suppose a multigraph is traversable and that a traversable trail does not begin or end at vertex P. thus the edges in the trail incident with P must appear in pairs, and so P is an even vertex. Therefore if a vertex Q is odd, the traversable trail must begin or end at Q.

Consequently, a multigraph with more than two odd vertices cannot be traversable. Observe that the multigraph corresponding to the Konigsberg bridge problem has four odd vertices. Thus one cannot walk through Konigsberg so that each bridge is crossed exactly once.

Tree graph:

A graph *T* is called a *tree* if *T* is connected and *T* has no cycles. Consider a tree *T* . Clearly, there is only one simple path between two vertices of *T* ; otherwise, the two paths would form a cycle. Also:

(a) Suppose there is no edge {u, v} in T and we add the edge e = {u, v} to T . Then the simple path from u to v in T and e will form a cycle; hence T is no longer a tree.

(b) suppose there is an edge e = {u, v} in T , and we delete e from T . Then T is no longer connected; hence T is no longer a tree.

Theorem: Let *G* be a graph with $n > 1$ vertices. Then the following are equivalent:

(i) *G* is a tree.

(ii) *G* is a cycle-free and has $n - 1$ edges.

(iii) *G* is connected and has $n - 1$ edges.

This theorem also tells us that a finite tree *T* with *n* vertices must have $n-1$ edges. For example, the tree in Fig. 13*(a)* has 9 vertices and 8 edges, and the tree in Fig. 13*(b)* has 13 vertices and 12 edges.

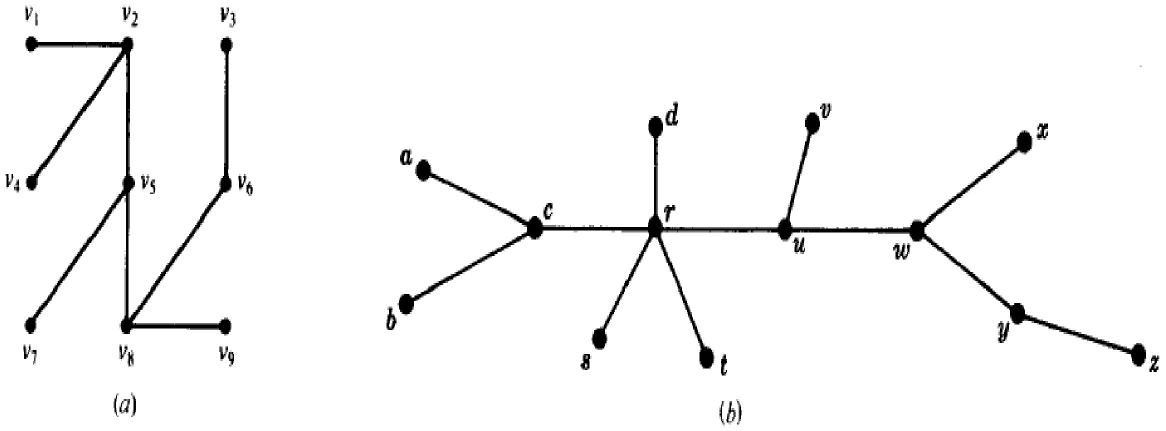


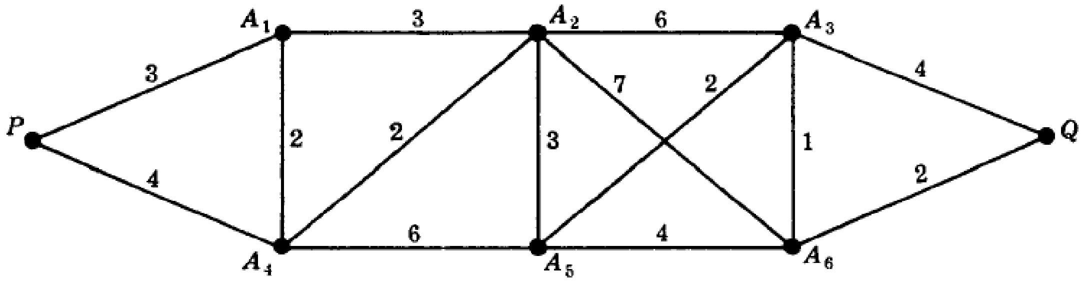
Figure 13

Labeled And weighted graphs:

A graph G is called a labeled graph if its edges and/or vertices are assigned data. If each edge (e) is assigned a non-negative number L(e). Then L(e) is called the weight or length of e. The weight of a path in such a weighted graph G is defined to be the sum of the weights of the edges in the path.

One important problem in graph theory is to find a shortest path, that is, a path of minimum weight (length), between any two given vertices.

Example: find the minimum path between P & Q:



(P, A1, A2, A5, A3, A6, Q)

$$\sum_{P}^{Q} L(e) = 3 + 3 + 3 + 2 + 1 + 2 = 14$$

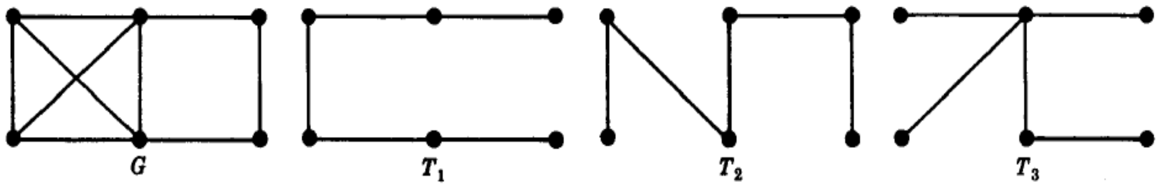
Another minimum path:

(P, A4, A2, A5, A3, A6, Q)

$$\sum_{P}^{Q} L(e) = 4+ 2 + 3 + 2 + 1 + 2 = 14$$

Spanning Trees

A subgraph T of a connected graph G is called a spanning tree of G if T is a tree and T includes all the vertices of G.



Minimum Spanning Trees

Suppose G is a connected weighted graph. That is, each edge of G is assigned a nonnegative number called the weight of the edge. Then any spanning tree T of G is assigned a total weight obtained by adding the weights of the edges in T . A minimal spanning tree of G is a spanning tree whose total weight is as small as possible.

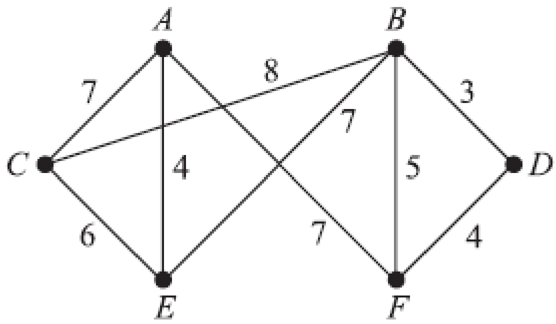
Algorithm 1 : The input is a connected weighted graph G with n vertices.

Step 1. Arrange the edges of G in the order of decreasing weights.

Step 2. Proceeding sequentially, delete each edge that does not disconnect the graph until $n - 1$ edges remain.

Step 3. Exit.

EXAMPLE 5: Find a minimal spanning tree of the weighted graph Q, Note that Q has six vertices, so a spanning tree will have five edges.



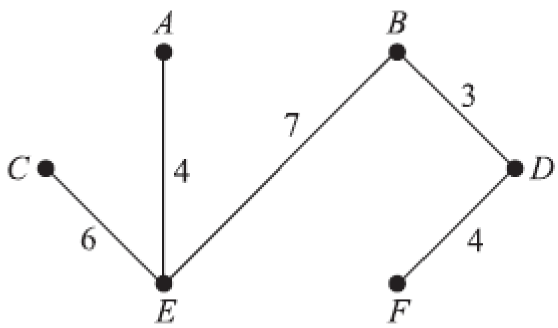
(*a*)

First we order the edges by decreasing weights, and then we successively delete edges without disconnecting Q until five edges remain. This yields the following data:

Edges:	BC	AF	AC	BE	CE	BF	AE	DF	BD
Weight	8	7	7	7	6	5	4	4	3
Delete	Yes	Yes	Yes	No	No	Yes			

Thus the minimal spanning tree of Q which is obtained contains the edges:

BE, CE, AE, DF, BD The spanning tree has weight 24



(*b*)

Algorithm 2: (Kruskal): The input is a connected weighted graph G with n vertices.

Step 1. Arrange the edges of G in order of increasing weights.

Step 2. Starting only with the vertices of G and proceeding sequentially, add each edge which does not result in a cycle until $n - 1$ edges are added.

Step 3. Exit.

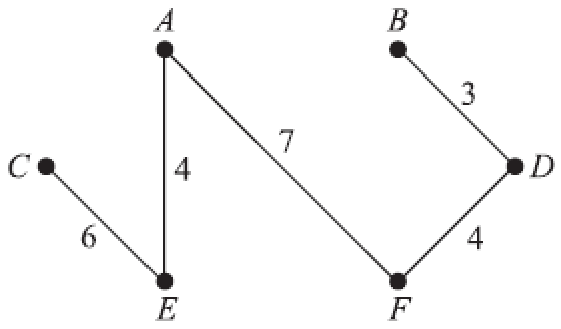
First we order the edges by increasing weights, and then we successively add edges without forming any cycles until five edges are included. This yields the following data:

Edges	BD	AE	DF	BF	CE	AC	AF	BE	BC
Weight	3	4	4	5	6	7	7	7	8
Add?	Yes	Yes	Yes	No	Yes	No	Yes		

Thus the minimal spanning tree of Q which is obtained contains the edges:

BD, AE, DF, CE, AF

Observe that this spanning tree is not the same as the one obtained using Algorithm 1 as expected it also has weight 24.



(c)

REPRESENTING GRAPHS IN COMPUTER MEMORY:

There are many useful ways to represent graphs where in working with a graph it is helpful to be able to choose its most convenient representation.

(1)adjacency lists

EXAMPLE 6 Use adjacency lists to describe the simple graph given in Figure 14.

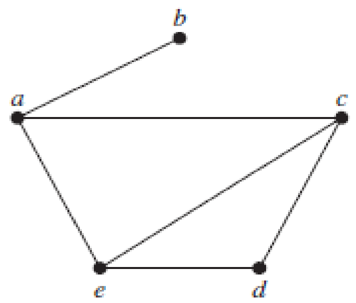


Figure 14

Solution: Table 1 lists those vertices adjacent to each of the vertices of the graph.

TABLE 1 An Adjacency List for a Simple Graph.

Vertex	*Adjacent Vertices*
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d

EXAMPLE 7

Represent the directed graph shown in Figure 15 by adjacency lists

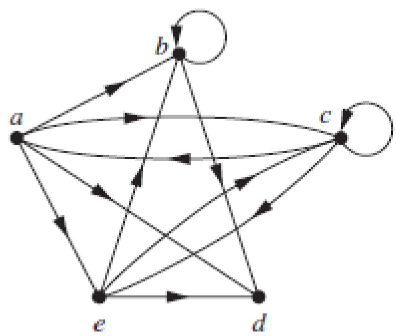


Figure 15

Solution: Table 2 represents the directed graph shown in Figure 15.

TABLE 2 An Adjacency List for a Directed Graph.

Initial Vertex	*Terminal Vertices*
a	b, c, d, e
b	b, d
c	a, c, e
d	
e	b, c, d

(2)Adjacency Matrices

Carrying out graph algorithms using the representation of graphs by adjacency lists, can be cumbersome if there are many edges in the graph. To simplify computation, graphs can be represented using matrices. Two types of matrices commonly used to represent graphs will be presented here. One is based on the adjacency of vertices, and the other is based on incidence of vertices and edges.

Suppose that $G = (V ,E)$ is a simple graph where $|V| = n$. The **adjacency matrix A** of G, is the n x n zero–one matrix with 1 as its (i, j)th entry when v_i and v_j are adjacent, and 0 as its (i, j)th entry when they are not adjacent.

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

EXAMPLE 8 Use an adjacency matrix to represent the graph shown in Figure 16.

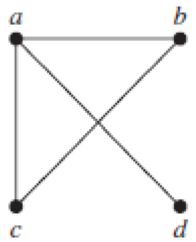


Figure 16

Solution:

We order the vertices as a, b, c, d. The matrix representing this graph is

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

EXAMPLE 9: Draw a graph with the following adjacency matrix

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

***Solution**:* A graph with this adjacency matrix is shown in Figure 17.

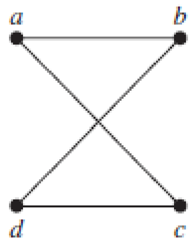


Figure 17

The adjacency matrix of a simple graph is symmetric, that is, $a_{ij} = a_{ji}$, because both of these entries are 1 when v_i and v_j are adjacent, and both are 0 otherwise. Furthermore, because a simple graph has no loops, each entry a_{ii}, $i = 1, 2, 3, \ldots, n$, is 0.

Adjacency matrices can also be used to represent undirected graphs with loops and with multiple edges. A loop at the vertex v_i is represented by a 1 at the (i, i)th position of the adjacency matrix. When multiple edges connecting the same pair of vertices v_i and v_j, or multiple loops at the same vertex, are present, the adjacency matrix is no longer a zero–one matrix, because the (i, j)th entry of this matrix equals the number of edges that are associated to $\{v_i , v_j\}$.

EXAMPLE 10: Use an adjacency matrix to represent the multigraph shown in Figure 18.

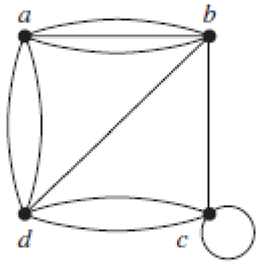


Figure 18

Solution: The adjacency matrix using the ordering of vertices *a, b, c, d* is:

$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}.$$

The adjacency matrix for a directed graph does not have to be symmetric, because there may not be an edge from *vj* to *vi* when there is an edge from *vi* to *vj* .

TRADE-OFFS BETWEEN ADJACENCY LISTS AND ADJACENCY MATRICES

When a simple graph contains relatively few edges, that is, when it is sparse, it is usually preferable to use adjacency lists rather than an adjacency matrix to represent the graph.

(3) Incidence Matrices

Another common way to represent graphs is to use **incidence matrices**. Let *G = (V ,E)* be an undirected graph. Suppose that *v*1, *v*2, . . . , *vn* are the vertices and *e*1, *e*2, . . . , *em* are the edges of *G*. Then the incidence matrix with respect to this ordering of *V* and *E* is the *n*×*m* matrix **M** = [*mij*], where:

$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

EXAMPLE 11: Represent the graph shown in Figure 19 with an incidence matrix.

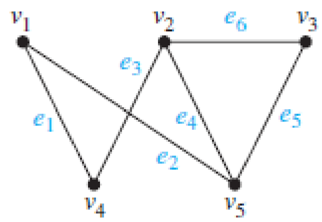
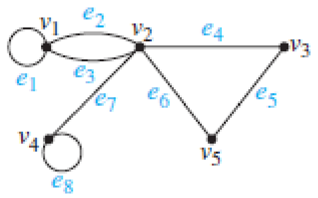


Figure 19

Solution: The incidence matrix is

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}.$$

EXAMPLE 12: Represent the multigraph shown in the following figure using an incidence matrix.



Solution: The incidence matrix for this graph is

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}.$$

Rooted tree:

Recall that a tree graph is a connected cycle-free graph, that is, a connected graph without any cycles. A *rooted tree T* is a tree graph with a designated vertex *r* called the *root* of the tree.

Consider a rooted tree *T* with root *r*. The length of the path from the root *r* to any vertex *v* is called the *level* (or *depth*) of *v*, and the maximum vertex level is called the *depth* of the tree.

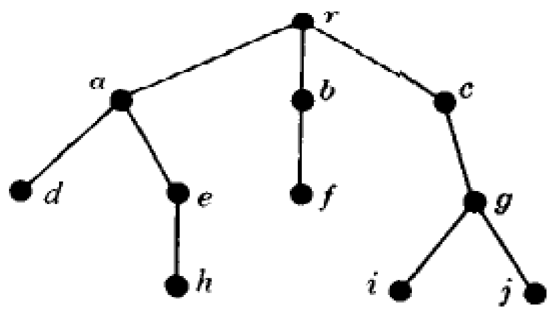


Figure 20

Those vertices with degree 1, other than the root *r*, are called the *leaves* of *T*.

One usually draws a picture of a rooted tree *T* with the root at the top of the tree.

Figure 20 shows a rooted tree *T* with root *r* and 10 other vertices. The tree has five leaves, *d,f, h, i,* and *j* . Observe that: *level(a)* = 1, *level(f)* = 2, *level(j)* = 3. Furthermore, the depth of the tree is 3.

EXAMPLE 13:

Suppose Marc and Erik are playing a tennis tournament such that the first person to win two games in a row or who wins a total of three games wins the tournament. Find the number of ways the tournament can proceed.

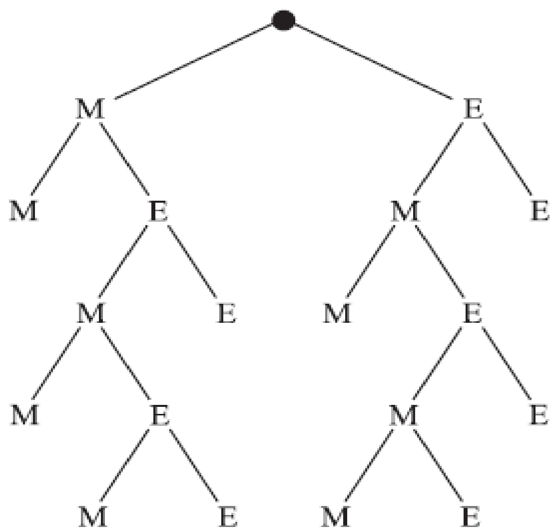


Figure 21

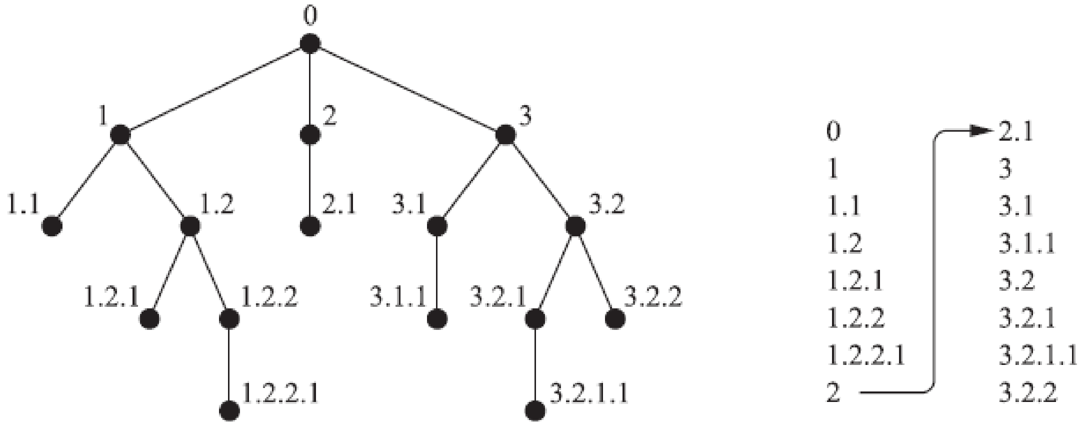
The rooted tree in Fig.21 shows the various ways that the tournament could proceed. There are 10 leaves which correspond to the 10 ways that the tournament can occur:

MM, **MEMM**, **MEMEM**, **MEMEE**, **MEE**, **EMM**, **EMEMM**, **EMEME**, **EMEE**, **EE**

Specifically, the path from the root to the leaf describes who won which games in the particular tournament.

Order Rooted Tree (ORT):

Whenever draw the digraph of a tree, we assume some ordering at each level, by arranging children from left to right. Where identical to the order obtained by moving down the leftmost branch of the tree, then the next branch to the right, then the second branch to the right, and so on.



Degree of tree: The largest number of children in the vertices of the tree

Binary tree : every vertex has at most 2 children

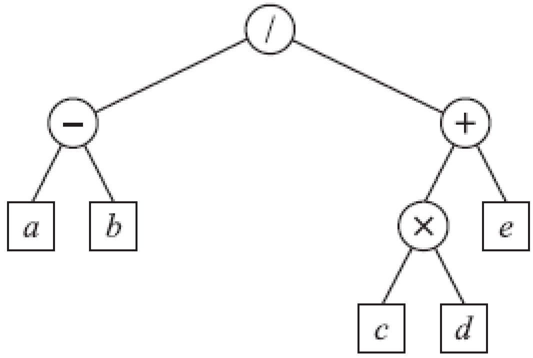
Algebraic Expressions and Polish Notation

Any algebraic expression involving binary operations +, -, $\times$, ÷ can be represented by an order rooted tree (ORT).

Let E be any algebraic expression which uses only binary operations, such as:

$$E = (a - b)/((c \times d) + e)$$

Then E can be represented by a tree as



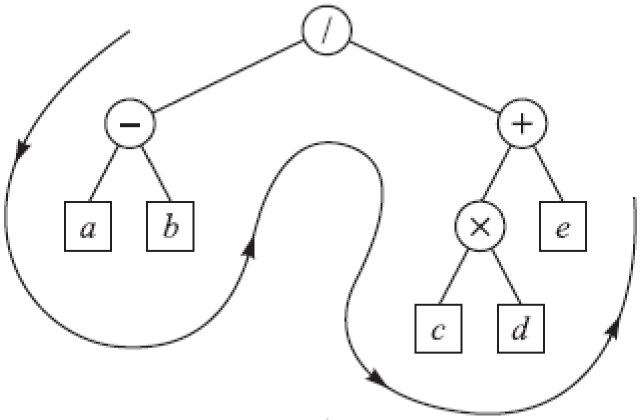
where the variables in *E* appear as the external nodes, and the operations in *E* appear as internal nodes.
The Polish mathematician Lukasiewicz observed that by placing the binary operation symbol before its arguments, e.g.:

$+ab$ instead of $a + b$ and $/cd$ instead of c/d

one does not need to use any parentheses. This notation is called *Polish notation in prefix form*. (one can place the symbol after its arguments, called *Polish notation in postfix form*.) Rewriting *E* in prefix form we obtain:

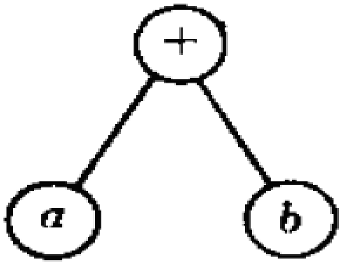
$$E = / - a\ b + \times c\ d\ e$$

Observe that this is precisely the order of the vertices in its tree which can be obtained by scanning the tree as :

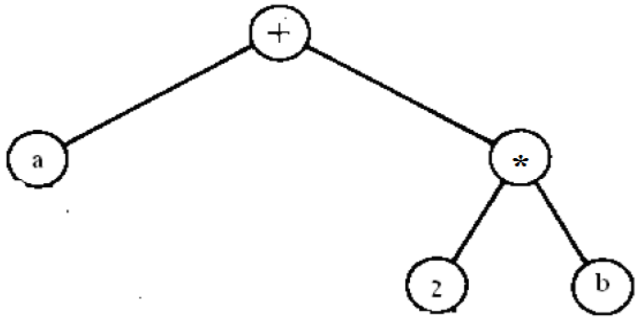


The polish notation form of an algebraic expression represents the expression unambiguously without the need for parentheses
1) a + b (infix)
2) + a b (prefix)
3) a b + (postfix)

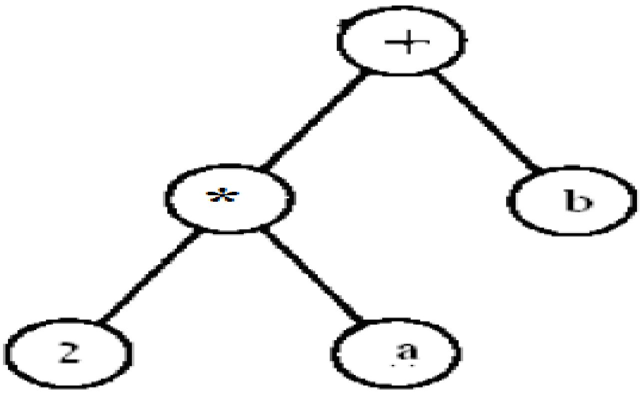
Example 14:
infix polish notation is : a + b
prefix polish notation : + a b



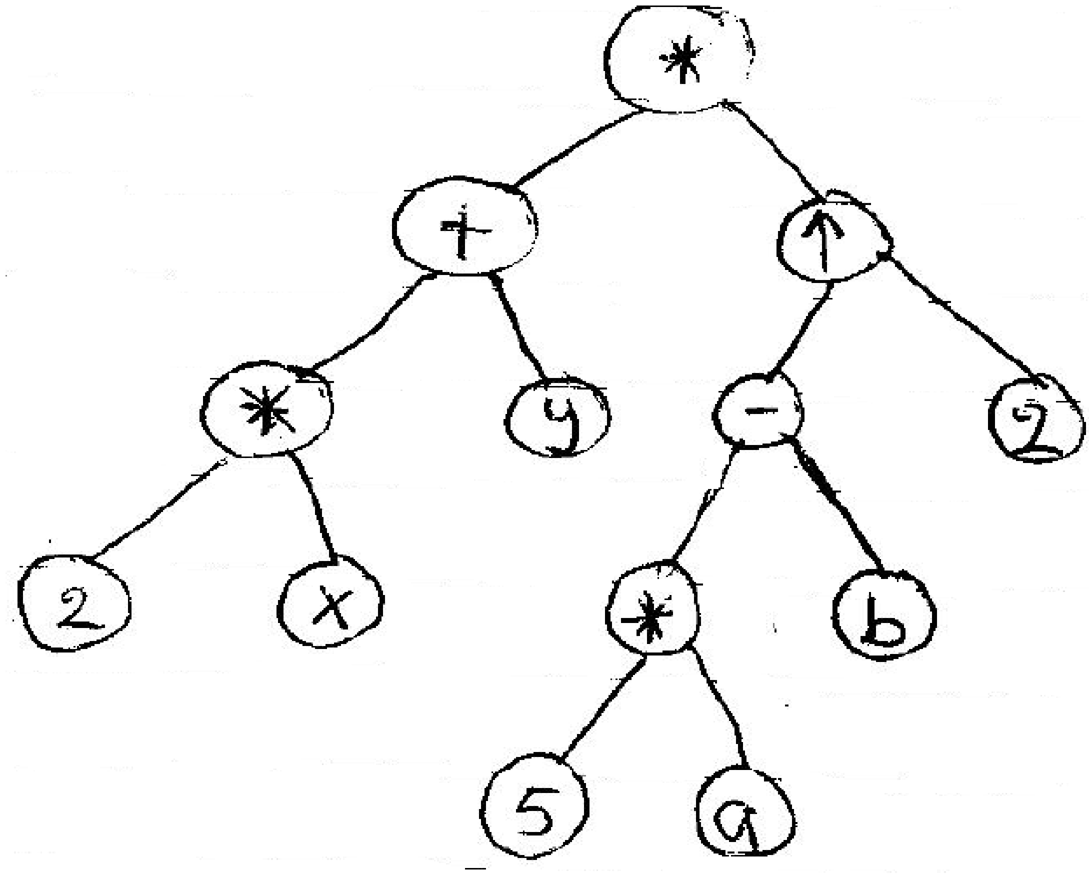
example 15:
infix polish notation is : a + 2 * b
prefix polish notation : + a * 2 b



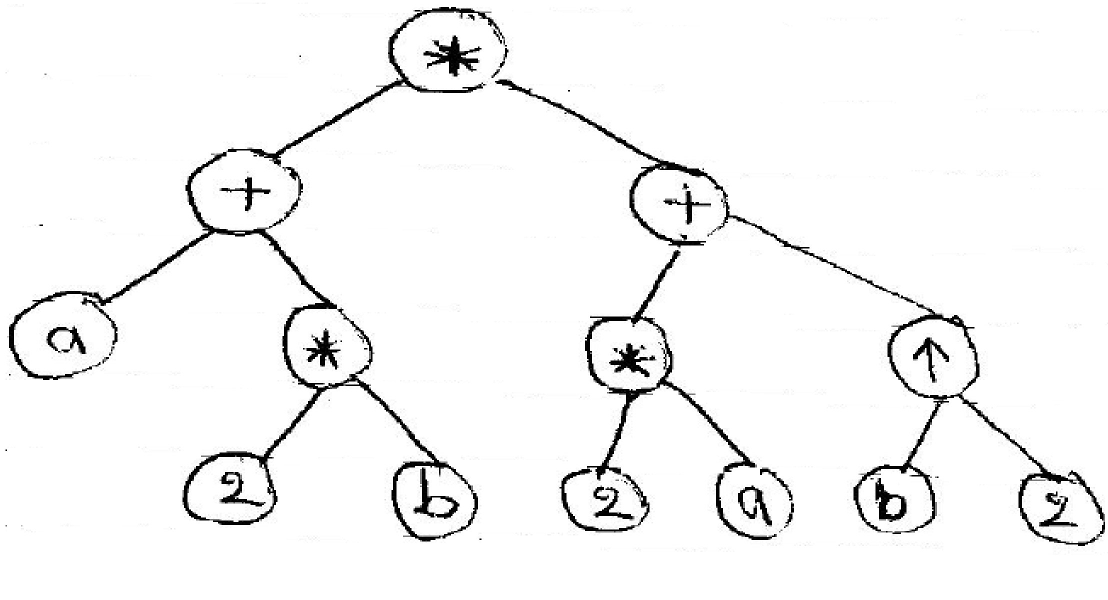
example 16:
infix polish notation is : 2 * a + b
prefix polish notation : + * 2 a b



example 17:
infix polish notation is : (2 * x + y).(5 * a – b)^2
prefix polish notation : * + * 2 x y ^ - * 5 a b 2



example 18:
infix polish notation is : (a + 2 * b) (2 * a + b^2)
prefix polish notation : * + a * 2 b + * 2 a ^ b 2



To evaluate an expression in polish form proceed as follows:

a) move from left to right until we find a simple string of the form Pxy, where P is the symbol for a binary operations: (+,-,× , /) and x & y are numbers.
b) Evaluate xPy and substitute the answer.
c) Continue this procedure until only one number remains.

Example:
evaluate the value of the expression (a-b)×(c+(d/e)), if a=6, b=4, c=5, d=2 and e=2

Prefix: * - a b+ c /d e
To evaluate: * - 6 4+ 5 /2 2

a) *- 6 4 +5 / 2 2
b) *2 + 5 / 2 2
c) * 2 + 5 1
d) * 2 6
e) 12

Homework:
Rewrite the following expressions into prefix polish notation form, construct their corresponding ORT and evaluate their value

(3*(1-x))/((4+(7-(y+2)))*(7+(x/y)))
(3-(2+x))+((x-2) –(3+x))

Finite state machines (FSM):

We may view a digital computer as a machine which is in a certain “internal state” at any given moment. The computer “reads” an input symbol, and then “prints” an output symbol and changes its “state”. The output symbol depends solely upon the input symbol and the internal state of the machine, and the internal state of the machine depends solely upon the preceding state of the machine and the preceding input symbol.

A finite state machine FSM (or complete sequential machine) M consists of five things:

(1) A finite set A of input symbols.
(2) A finite set S of internal states.
(3) A finite set Z of output symbols.
(4) An initial state *s*0 in *S*.
(5) A next-state function f $f: S \times A \rightarrow S$
(6) An output function g $g: S \times A \rightarrow Z$

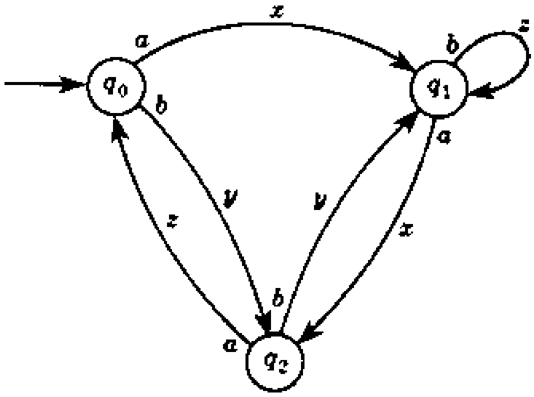
This machine M is denoted by M = (A, S, Z, q0, f, g) where q0 is the initial state.

Example 1:

The following defines a FSM with two input symbols, three internal states and three output symbols:

(1) A = {a, b}
(2) S = {q0, q1, q2}
(3) Z = {x, y, z}
(4) Next-state function $f: S \times A \rightarrow S$ defined by :
 f(q0, a) = q1 f(q1,a) = q2
 f(q2,a) = q0 f(q0, b) = q2
 f(q1, b) = q1 f(q2, b) = q1
(5) Output function $g: S \times A \rightarrow Z$ defined by
 g(q0,a) = x g(q1,a) = x
 g(q2,a) = z g(q0, b) = y
 g(q1, b) = z g(q2, b) = y

There are two ways of representing a finite state machine in compact form. One way is by a table called the **state table** of machine, and the other way is by a labeled directed graph called the **state diagram** of the machine.



State diagraph

	a	b
q_0	q_1, x	q_2, y
q_1	q_2, x	q_1, z
q_2	q_0, z	q_1, y

State table

We visualize these symbols on an "input tape." The machine *M* "reads" these input symbols one by one and, simultaneously, changes through a sequence of states.

If the input string**:** **abaab** **,** is given to the machine in example (1), and suppose q0 is the initial state of the machine.

We calculate the string of states and the string of output symbols from the state diagram by beginning at the vertex q0 and following the arrows which are labeled with the input symbols:

$$q_0 \xrightarrow{a,\,x} q_1 \xrightarrow{b,\,z} q_1 \xrightarrow{a,\,x} q_2 \xrightarrow{a,\,z} q_0 \xrightarrow{b,\,y} q_2$$

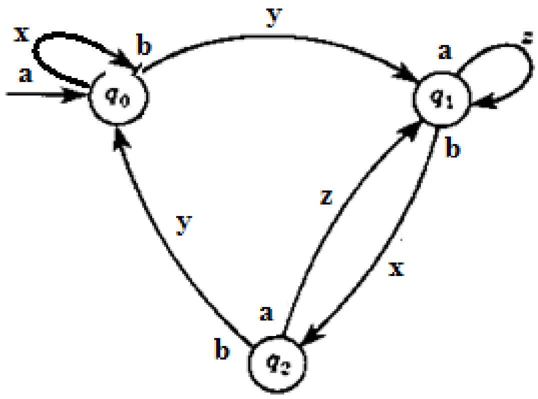
This yields the following strings of states and output symbols:

State : q0 q1 q1 q2 q0 q2

Output symbols : x z x z y

Homework:

Draw the state table for the following FSM and Trace it with the input: **aaabbb**, and **abaab**.



Example 2:

Design a **FSM** which can do binary addition

we can assume that our numbers have the same number of digits. If the machine is given the input:

1101011

\+ 0111011

then we want the output to be the binary sum 10100110.

Specifically, the input is the string of pairs of digits to be added:

11, 11, 00, 11, 01, 11, 10, b (where b denotes blank spaces)

and the output should be the string:

0, 1, 1, 0, 0, 1, 0, 1

We also want the machine to enter a state called "stop" when the machine finishes the addition.

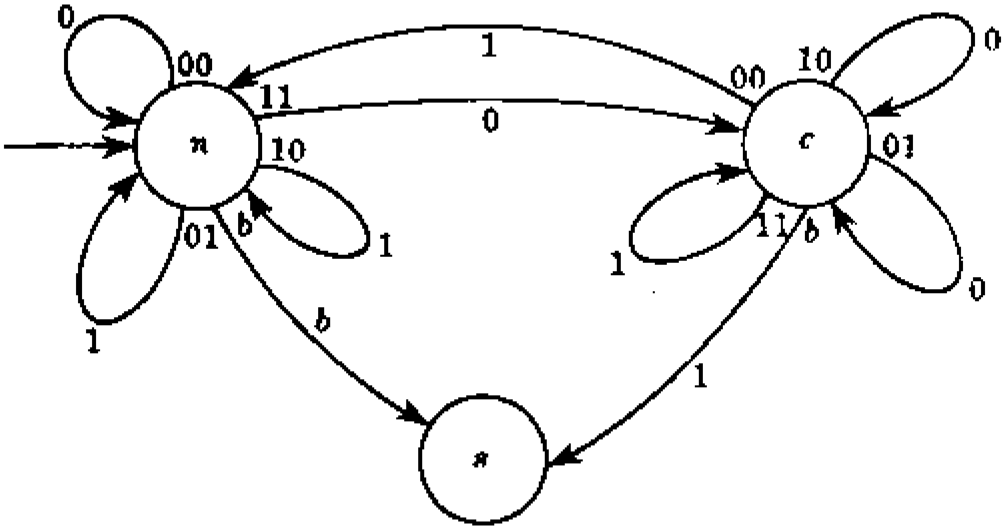
The input symbols and output symbols are, respectively, as follows:

A = {00, 01, 10, 11, b} and Z = {0, 1, b}

The machine M that we "construct" will have three states:

S = {carry *(c)*, no carry *(n)*, stop *(s)*}

Here n is the initial state.



FINITE AUTOMATA

A finite automaton is similar to a finite state machine except that an automation has “accepting” and rejecting” states rather than an output. Specifically, a finite automaton M consists of five things:

(1) A finite set A of input symbols
(2) A finite set S of internal states
(3) A subset T of S (whose elements called accepting states)
(4) An initial state q0 in S
(5) A next-state function f from S × A ino S.

The automaton M is denoted by M = (A, S, T, q0, f) when we want to designate its five parts

We can concisely describe a finite automaton M by its state diagram as was done with finite state machines, except that here we use double circles for accepting states and each edge is labeled only by the input symbol. Specifically, the state diagram D of M is a labeled directed graph whose vertices are the states of S where accepting states are labeled by having a double circle, and if f(qj, ai) = qk then there is an arc from qj to qk which is labeled with ai. Also the initial state q0 is denoted by having an arrow entering the vertex q0.

We say that M recognizes or accepts the string W if the final state sn is an accepting state, i. e. if sn ∈T. We will let L(M) denote the set of all strings which are recognized by M.

Example 1

The following defines a finite automaton with two input symbols and three states:

(1) A = {a,b}, input symbols
(2) S = {q0, q1, q2}, states
(3) T = {q0, q1}, accepting states
(4) q0, the initial state.
(5) Next-state function f : S × A → S defined by:

f(q0,a) = q0, f(q1,a)=q0, f(q2,a)=q2
f(q0,b)=q1, f(q1,b)=q2, f(q2,b)=q2

or by the table:

f	a	b
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_2	q_2

The automaton M will recognize those strings which do not have two successive b's. Thus M will accept:

aababaaba, aaa, baab, abaaababab, b, aabaaab

But will reject :

aabaabba, bbaaa, ababbaab, bb, abbbbaa

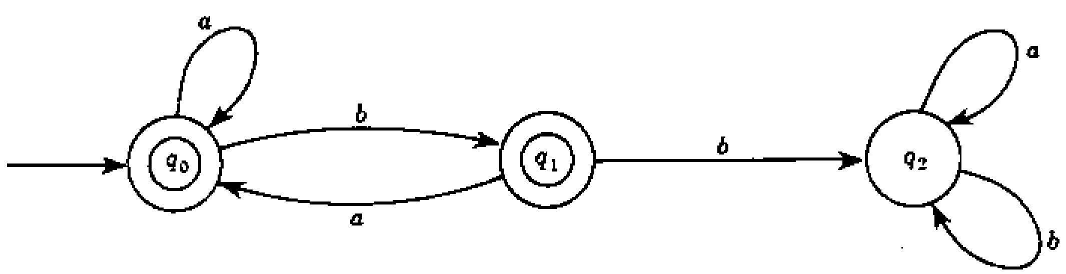


Fig. 1

Language *L(M)* Determined by an Automaton *M*

Each automaton *M* with input alphabet *A* defines a language over *A*, denoted by *L(M)*. We say that *M recognizes* the word *w* if the final state s_m is an accepting state in *Y* . The language *L(M)* of *M* is the collection of all words from *A* which are accepted by *M*.

EXAMPLE2

Determine whether or not the automaton *M* in Fig. 1 accepts the words:

$w1 = ababba$; $w2 = baab$; $w3 = \lambda$ (empty word)

Use Fig. 1 and the words *w*1 and *w*2 to obtain the following respective paths:

$$P1 = s0 \xrightarrow{a} s0 \xrightarrow{b} s1 \xrightarrow{a} s0 \xrightarrow{b} s1 \xrightarrow{b} s2 \xrightarrow{a} s2$$

$$P2 = s0 \xrightarrow{b} s1 \xrightarrow{a} s0 \xrightarrow{a} s0 \xrightarrow{b} s1$$

The final state in P_1 is s_2 which is not in Y ; hence w_1 is not accepted by *M*. On the other hand, the final state in P_2 is s_1 which is in Y ; hence w_2 is accepted by *M*. The final state determined by w_3 is the initial state s_0 since $w_3 = \lambda$ is the empty word. Thus w_3 is accepted by *M* since $s_0 \in Y$.

EXAMPLE 3

Describe the language *L(M)* of the automaton *M* in Fig. 1.

L(M) will consist of all words *w* on *A* which do not have two successive *b*'s. This comes from the following facts:

(1) We can enter the state q_2 if and only if there are two successive *b*'s.

(2) We can never leave *q*2.

(3) The state q_2 is the only rejecting (nonaccepting) state.

EXAMPLE4

Construct an automaton M with input symbols a and b, which only accept those string such that the number of b's is divisible by 3.

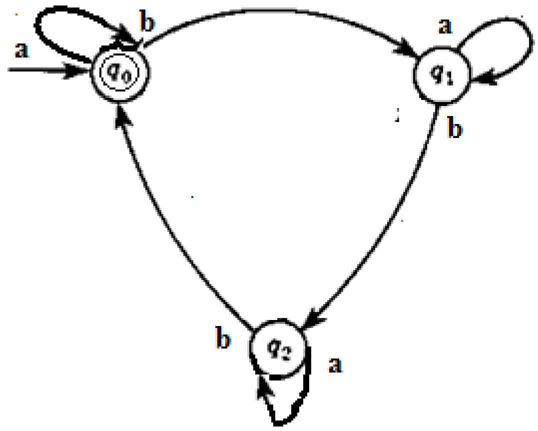
A= {a, b}

S= {q0, q1, q2}

T= {q0}

Accepted symbols: abab aab, baabab, bbabbbba, aa, aabbaab

Rejected symbols: ab, ababbb



	a	b
q0	q0	q1
q	q1	q2
q2	q2	q0

Some Examples of FSM

We study examples of finite state machines that are designed to recognize given patterns.

As there is essentially no standard way of constructing such machines, we shall illustrate the underlying ideas by examples.

Example 1:

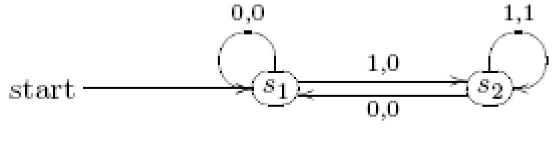
Suppose that *A (input)* = Z (output) = {0, 1}, and that we want to design a finite state machine that recognizes the sequence pattern 11 in the input string $x \in A*$. An example of an input string $x \in A*$ and its corresponding output string $y \in Z*$ is shown below:

x = 1011101010111110101
y = 0001100000011110000

Note that the output digit is 0 when the sequence pattern 11 is not detected and 1 when the sequence pattern 11 is detected. In order to achieve this, we must ensure that the finite state machine has at least two states, a "passive" state when the previous entry is 0 (or when no entry has yet been made), and an "excited" state when the previous entry is 1. Furthermore, the finite state machine has to observe the following and take the corresponding actions:

(1) If it is in its "passive" state and the next entry is 0, it gives an output 0 and remains in its "passive" state.
(2) If it is in its "passive" state and the next entry is 1, it gives an output 0 and switches to its "excited" state.
(3) If it is in its "excited" state and the next entry is 0, it gives an output 0 and switches to its "passive" state.
(4) If it is in its "excited" state and the next entry is 1, it gives an output 1 and remains in its "excited" state.

It follows that if we denote by $s1$ the "passive" state and by $s2$ the "excited" state, then we have the state diagram below:



We then have the corresponding transition table:

	g		f	
	0	1	0	1
$+s_1+$	0	0	s_1	s_2
s_2	0	1	s_1	s_2

Example 2:

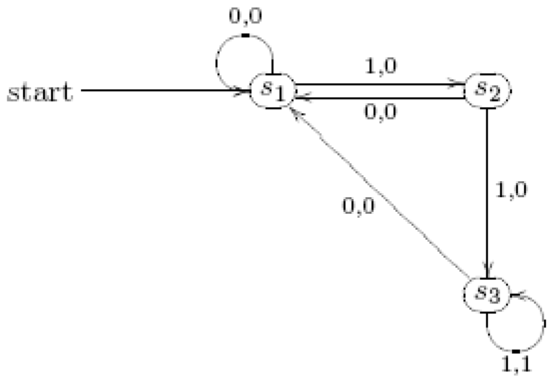
Suppose again that *A (input)* = Z (output) = {0, 1}, and that we want to design a finite state machine that recognizes the sequence pattern 111 in the input string $x \in A*$. An example of the same input string $x \in A*$ and its corresponding output string $y \in Z*$ is shown below:

x = 10111010101111110101
y = 00001000000011110000

In order to achieve this, the finite state machine must now have at least three states, a "passive" state when the previous entry is 0 (or when no entry has yet been made), an "expectant" state when the previous two entries are 01 (or when only one entry has so far been made and it is 1), and an "excited" state when the previous two entries are 11. Furthermore, the finite state machine has to observe the following and take the corresponding actions:

(1) If it is in its "passive" state and the next entry is 0, it gives an output 0 and remains in its "passive" state.
(2) If it is in its "passive" state and the next entry is 1, it gives an output 0 and switches to its "expectant" state.
(3) If it is in its "expectant" state and the next entry is 0, it gives an output 0 and switches to its "passive" state.
(4) If it is in its "expectant" state and the next entry is 1, it gives an output 0 and switches to its "excited" state.
(5) If it is in its "excited" state and the next entry is 0, it gives an output 0 and switches to its "passive" state.
(6) If it is in its "excited" state and the next entry is 1, it gives an output 1 and remains in its "excited" state.

If we now denote by *s*1 the "passive" state, by *s*2 the "expectant" state and by *s*3 the "excited" state, then we have the state diagram below:



We then have the corresponding transition table:

	g		f	
	0	1	0	1
$+s_1+$	0	0	s_1	s_2
s_2	0	0	s_1	s_3
s_3	0	1	s_1	s_3

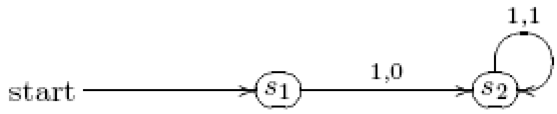
An Optimistic Approach

We construct first of all the part of the machine to take care of the situation when the required pattern occurs repeatly and without interruption. We then complete the machine by studying the situation when the "wrong" input is made at each state.

.

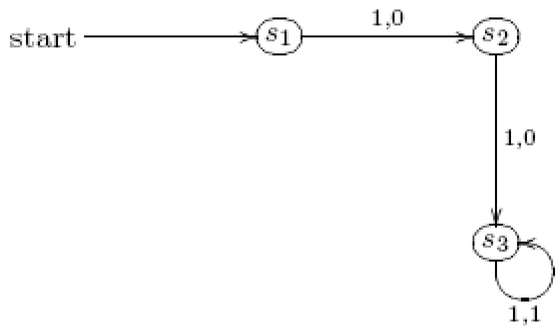
Example 1:

Suppose that *A (input)* = Z (output) = {0, 1}, and that we want to design a finite state machine that recognizes the sequence pattern 11 in the input string $x \in A*$. Consider first of all the situation when the required pattern occurs repeatly and without interruption. In other words, consider the situation when the input string is 111111 . . . . To describe this situation, we have the following incomplete state diagram:



It now remains to study the situation when we have the "wrong" input at each state. Naturally, with a "wrong" input, the output is always 0, so the only unresolved question is to determine the next state.
Note that whenever we get an input 0, the process starts all over again; in other words, we must return to state *s*1.. We therefore obtain the state diagram as in Example 1.

Example 2:
Suppose again that *A (input)* = Z (output) = *{*0, 1*}*, and that we want to design a finite state machine that recognizes the sequence pattern 111 in the input string $x \in A*$. Consider first of all the situation when the required pattern occurs repeatly and without interruption. In other words, consider the situation when the input string is 111111 . . . . To describe this situation, we have the following incomplete state diagram:

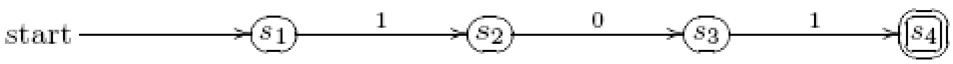


It now remains to study the situation when we have the "wrong" input at each state. As before, with a "wrong" input, the output is always 0, so the only unresolved question is to determine the next state.
Note that whenever we get an input 0, the process starts all over again; in other words, we must return to state *s*1. We therefore obtain the state diagram as Example 2.

Deterministic Finite State Automata

we discuss a slightly different version of finite state machines which is closely related to regular languages. We begin by an example which helps to illustrate the changes.

Example.1. We shall construct a deterministic finite state automaton which will recognize the input strings 101 and nothing else. This automaton can be described by the following state diagram:



We can also describe the same information in the following transition table:

	ν	
	0	1
$+s_1+$	–	s_2
s_2	s_3	–
s_3	–	s_4
$-s_4-$	–	–

We now modify our definition of a finite state machine accordingly.

Definition. A deterministic finite state automaton is a 5-tuple $A = (S, I, \nu, T, s1)$, where

(a) S is the finite set of states for A;

(b) I is the finite input alphabet for A;

(c) $\nu : S \times I \rightarrow S$ is the next-state function;

(d) T is a non-empty subset of S; and

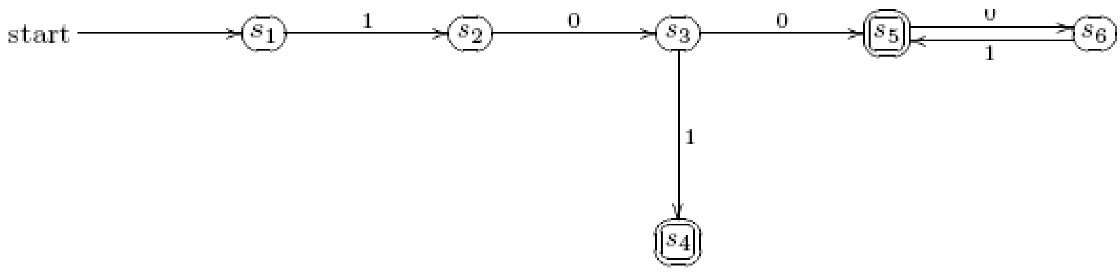
(e) $s1 \in S$ is the starting state.

Remarks.

(1) The states in T are usually called the accepting states.

(2) If not indicated otherwise, we shall always take state $s1$ as the starting state.

Example.2. We shall construct a deterministic finite state automaton which will recognize the input strings 101 and 100(01)$*$ and nothing else. This automaton can be described by the following state diagram:



We can also describe the same information in the following transition table:

	ν 0	1
$+s_1+$	—	s_2
s_2	s_3	—
s_3	s_5	s_4
$-s_4-$	—	—
$-s_5-$	s_6	—
s_6	—	s_5