Discrete mathematical structure

1   Class

st

Contents

1- WHAT IS A SET

2- Elements

3- Universal set, empty set

4- Subsets

5- Set operations

6- Finite sets, Counting Principle.

7- Classes of sets

8- Power set

9- Cardinality

10- Partitions of set

11- Relations.

12- Product sets.

13- Representation of relations.

14- Inverse relations.

15- Composition of relations.

16- Function.

17- One-to-one , onto and invertible functions.

18- Graph of a function.

19- Composition of function.

20- By using the Graph of function.

21- Invertible Functions

22- Graphs.

23- Degree.

24- Connectivity.

25- The Bridges of konigsberg, traversable multigraphs.

26- Special graph.

27- Matrices and graphs.

28-**-** Labeled graphs.

29- Tree.

30- Finite state machines (FSM).

31- State table.

32- Finite automata.

33- An Optimistic Approach.

34- Deterministic Finite State Automata**.**

References

1. www.geocities.com/basicmathsets
2. Discrete mathematics by Seymour Lipschutz
3. Discrete mathematical structures for computer science by Bernard bKolman& Robert C. Busby
4. http://en.wikibooks.org/wiki/Discrete_mathematics/Set_theory
5. http://www.math.uvic.ca/faculty/gmacgill/guide
6. http://www.maths.mq.edu.au/~wchen/lndmfolder

WHAT IS A SET?

A set is any well-defined list or collection of things, and will be denoted by capital letters A, B, X, Y,…….

Below you'll see just a sampling of items that could be considered as sets:

- Your favorite clothes
- A coin collection
- 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** and will be denoted by lower case letters a,b,x,y,.....

Sets are written using curly brackets "{" and "}", with their elements listed in between.

For example 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}

and even numbers could be {0,2,4,6,8,10,...} (Note: the dots at the end indicating that the set goes on infinitely)

Elements

By now you know each entry in a set is called an **element**

Principles:

- $\in$ belong to
- $\notin$ not belong to
- $\subseteq$ subset
- $\subset$ proper subset
- $\not\subset$ not subset

$\in$ means "belong to"; $\notin$ means "not belong to"

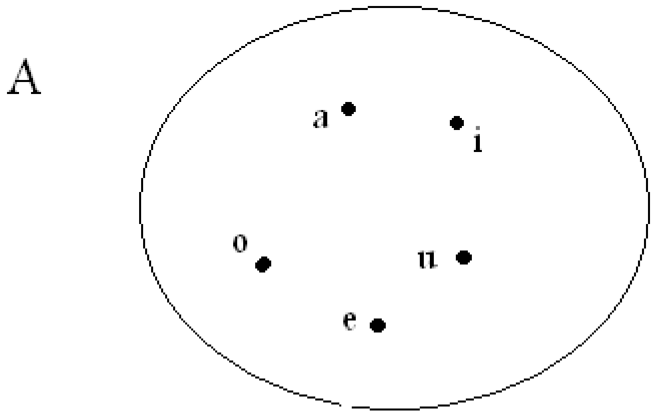
So we could replace the statement "a is belong to the alphabet" with a $\in$ {alphabet} and replace the statement "3 is not belong to the set of even numbers" with 3 $\notin$ {Even numbers}

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.

There are three ways to specify a particular set:

1) By list its members (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}

$3 \in X$
$4 \notin X$

Universal set, empty 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.
The set with no elements is called the empty set or null set and is denoted by $\varnothing$ or {}.

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:

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

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

Example:

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}.

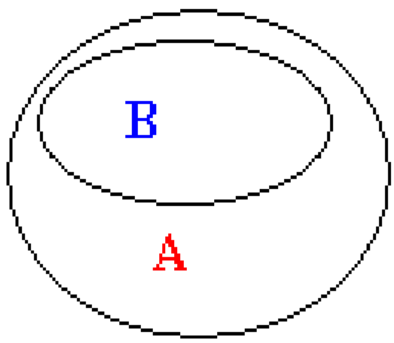
$A\subset B$: $\{\forall x\in A \quad \Rightarrow \quad x\in B$

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

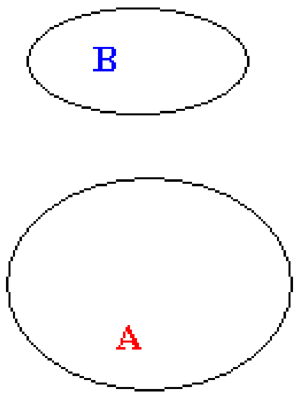
$\forall$: For all لكل

$\exists$: There exists يوجد على الاقل

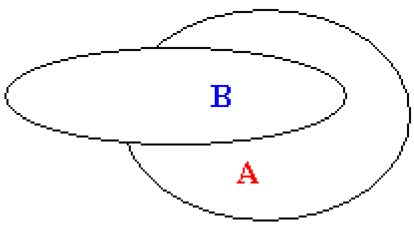
The notion of subsets is graphically illustrated below



In this first illustration B is entirely within A so $B \subset A$.



In this second illustration A and B have nothing in common ($A \cap B = \emptyset$) so we could write
$A \not\subset B$ and $B \not\subset A$.



In this last illustration some of B is in A, but not all of B is in A so we could write $B \not\subset A$.

Set of numbers:

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

The natural numbers

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

The integers

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

The rational numbers

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

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

The real numbers

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

The **complex** numbers

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

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

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:

A union of two or more sets is another set that contains everything contained in the previous sets.

If A and B are sets then $A \cup B$ represents the union of A and B:

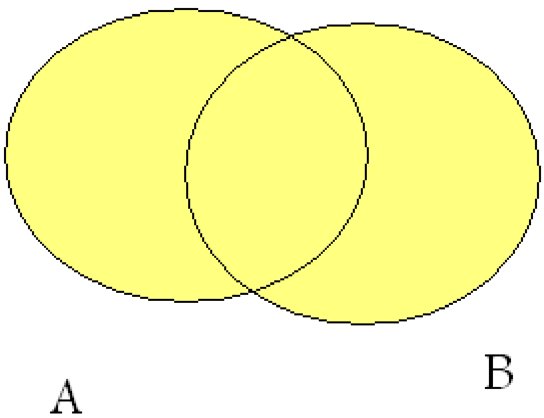
$A \cup B = \{ x : x \in A \text{ or } x \in B\}$

Example

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

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

Notice that when I wrote out the united set I did not write "5" twice. I simply listed all of the new sets elements.

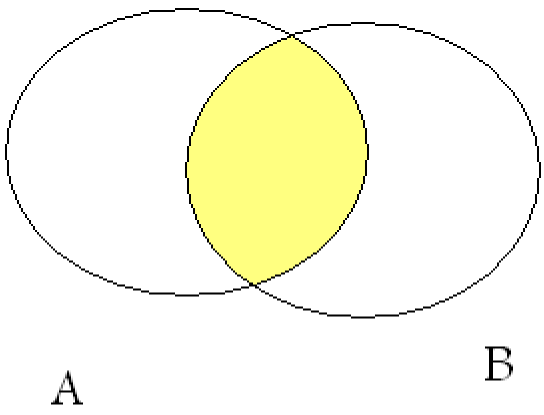


2) **INTERSECTION**

The intersection of two (or more) sets is those elements that they have in common.

So if A and B are sets then the intersection (the elements they both have in common) is denoted by $A \cap B$.

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



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

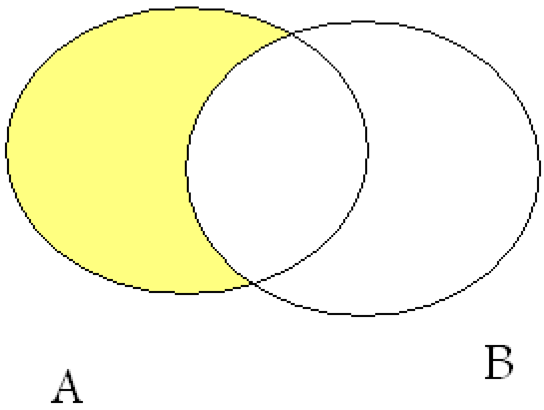
A={1,2,3,4,5}　　　　　　B={6,7,8,9,10}
In this case A and B have nothing in common.

$A \cap B = \varnothing$

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.

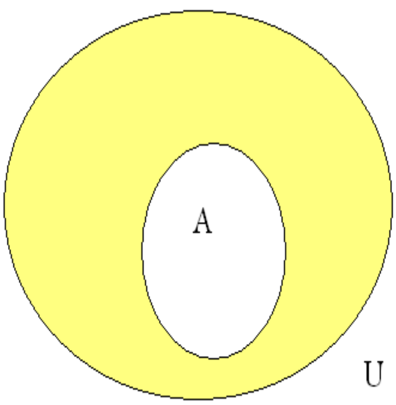
$$A \backslash B = \{x : x \in A, x \notin B\}$$



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 :

let $A=\{1,2,3\}$ $B=\{3,4\}$ $U=\{1,2,3,4,5,6\}$

Find :

$A \cup B = \{1, 2, 3, 4\}$
$A \cap B = \{3\}$
$A - B = \{1, 2\}$
$A^c = \{4, 5, 6\}$

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- $A \cup B = B \cup A$ Commutatively
$A \cap B = B \cap A$

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

$A \cap A^c = \varnothing$ and unions

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

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

We discuss two methods of proving equations involving set operations. The first is to break down what it means for an object x to be an element of each side, and the second is to use Venn diagrams.
For example, consider the first of De Morgan's laws :

$$(A \cup B)^c = A^c \cap B^c$$

We must prove: 1) $(A \cup B)^c \subset A^c \cap B^c$
2) $A^c \cap B^c \subset (A \cup B)^c$

We first show that $(A \cup B)^c \subset A^c \cap B^c$

Let's pick an element at random $x \in (A \cup B)^c$. We don't know anything about x, it could be a number, a function, or indeed an elephant. All we do know about x, is that

$x \in (A \cup B)^c$, so

$x \notin A \cup B$

because that's what complement means. Therefore

$x \notin A$ and $x \notin B$,

by pulling apart the union. Applying complements again we get

$x \in A^c$ and $x \in B^c$

Finally, if something is in 2 sets, it must be in their intersection, so

$$x \in A^c \cap B^c$$

So, any element we pick at random from : $(A \cup B)^c$ is definitely in,

$A^c \cap B^c$, so by definition

$$(A \cup B)^c \subset A^c \cap B^c$$

Next we show that $(A^c \cap B^c) \subset (A \cup B)^c$.

This follows a very similar way. Firstly, we pick an element at random from the first set, $x \in (A^c \cap B^c)$

Using what we know about intersections, that means

$$x \in A^c \text{ and } x \in B^c$$

Now, using what we know about complements,

$$x \notin A \text{ and } x \notin B.$$

If something is in neither A nor B, it can't be in their union, so

$$x \notin A \cup B,$$

And finally

$$\therefore \quad x \in (A \cup B)^c$$

We have prove that every element of $(A \cup B)^c$ belongs to $A^c \cap B^c$ and that every element of $A^c \cap B^c$ belongs to $A \cup B)^c$. Together, these inclusions prove that the sets have the same elements, i.e. that $(A \cup B)^c = A^c \cap B^c$

EXERCISE:

1- LET A={1,2,4,a,b,c}. Answer each of the following as True or False.

a. $2 \in A$
b. $3 \in A$
c. $c \notin A$
d. $\varnothing \in A$
e. $\varnothing \notin A$
f. $A \in A$

2- Let A={a,b,c,g}
B={d, e,f,g}
C={a,c,f}
D={f,h,k}
U={a,b,c,d,e,f,g,h,k}

Compute: $A \cup B$, $B \cup C$, $A \cap B$, $A \cap C$, $A - B$, $B \cap D$, A^c .

FINITE SETS, COUNTING PRINCPLE:

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 $\varnothing$ and the set of letters of English alphabet are finite sets, whereas 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 set and disjoint Then $A \cup B$ is finite set and:

$$n(A \cup B) = n(A) + n(B)$$

Theorem (4):

If A and B are finite sets then $A \cup B$ and $A \cap B$ are finite and :

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

Theorem (5):

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}

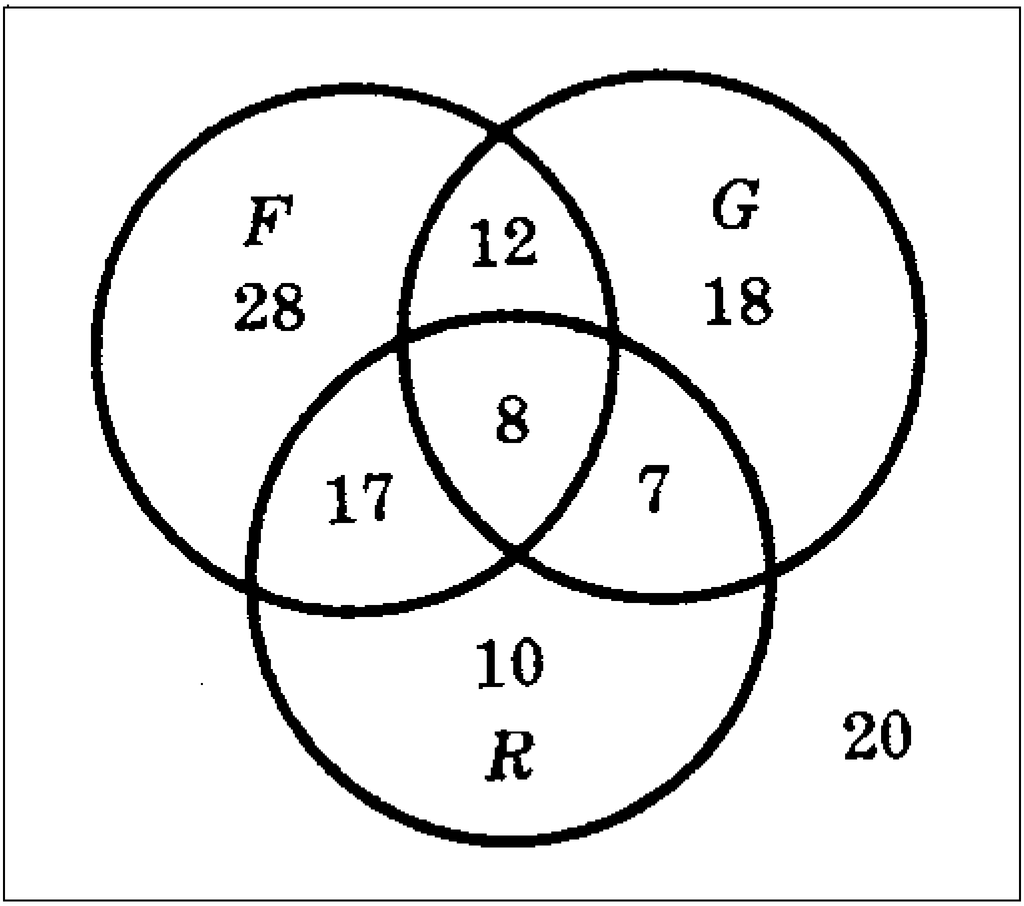
$A \cup B \cup C = \{1,2,3,4,5,6\}$
$|A \cup B \cup C| = 6$

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

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

Example (2):

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



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:

$|F \cup G \cup R| = |F| + |G| + |R| - |F \cap G| - |F \cap R| - |G \cap R| + |F \cap G \cap R|$

$100 = 65 + 45 + 42 - 20 - 25 - 15 + |F \cap G \cap R|$

$100 = 92 + |F \cap G \cap R|$

$\therefore |F \cap G \cap R| = 8$ students study the 3 languages

$20 - 8 = 12$ $(F \cap G) - R$

$25 - 8 = 17$ $(F \cap R) - G$

$15 - 8 = 7$ $(G \cap R) - F$

$65 - 12 - 8 - 17 = 28$ 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

Classes of sets :

Given a set A, we might wish to talk about some of its subsets. Thus we would be considering a set of sets. Whenever such a situation occures, to avoid confusion we will speak of **class of sets** or collection of sets rather than a set of sets.

Example : Suppose A = { 1,2 3} , let X be the class of subsets of A which contain exactly three elements of A. Then

$$X = [\{1,2,3\},\{1,2,4\},\{1,3,4\},\{2,3,4\}]$$

The elements of X are the sets $\{1,2,3\}$, $\{1,2,4\}$, $\{1,3,4\}$ and $\{2,3,4\}$.

Power set

The *power set* of some set S, denoted P(S), is the set of *all* subsets of S (including S itself). (The empty set is a subset of **all** sets.)

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

Example : Let A = { 1,2 3}

Power set of set A = P(A) = [{1},{2},{3},{1,2},{1,3},{2,3},{},A]

Cardinality

The *cardinality* of a set S, denoted |S|, is simply the number of elements a set has. So |{a,b,c,d}|=4, and so on. The cardinality of a set need not be finite: some sets have infinite cardinality.

The cardinality of the power set

If P(S)=T, then $|T|=2^{|S|}$.

This is because T contains sets representing all possible combinations of existence or nonexistence of the elements of P, meaning that each element can be in two states: in the subset, or not in the subset.

Since the number of possible combinations of different states of objects is the multiple of all the number of possible states of each object, and since each element in P can have exactly two states for each subset of P(in the subset or not in the subset), it is therefore inferable that the number of subsets for P is $2^{|S|}$.

Problem set:

Based on the above information, 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},{0,1,2},{0,2},{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;

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

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

Based on the above information, answer the following questions:

1. $\{2,3,4\}\times\{1,3,4\}$
2. $\{0,1\}\times\{0,1\}$
3. $|\{1,2,3\}\times\{0\}|$
4. $|\{1,1\}\times\{2,3,4\}|$

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

Partitions of set:

Let S be a any nonempty set. A partition (∏) of S is a subdivision of S into nonoverlapping, nonempty subsets. A partition of S is a collection {Ai} of non-empty subsets of S such that:

1) $Ai \neq \emptyset$, where i=1,2,3,……
2) $Ai \cap Aj = \emptyset$ where $i \neq j$.
3) $UA_i = S$ where $A_1 \cup A_2 \cup .... \cup A_i = S$

Example 1:

let A = {1,2,3,n}
A1 = {1}, A2 = {3,n}, A3 = {2}

∏ = {A1, A2, A3} is a partition on A because it satisfy the three above conditions

Example 2 :

Consider the following collections of subsets of S = {1,2,3,4,5,6,7,8,9}

(i) [{1,3,5},{2,6},{4,8,9}]
(ii) [{1,3,5},{2,4,6,8},{5,7,9}]
(iii) [{1,3,5},{2,4,6,8},{7,9}]

Then

(i) is not a partition of S since 7 in S does not belong to any of the subsets.
(ii) is not a partition of S since {1,3,5} and {5,7,9} are not disjoint.
(iv) is a partition of S.

Mathematic induction:

It is useful for proving propositions that must be true for all integers or for a range of integer.

Proposition: is any statement P(n) which can be either true or false for each n in N. Suppose P has the following two properties.

(i) P(1) is true
(ii) P(k+1) is true whenever P(k) is true

Then P is true for every positive integer $\forall\ n \geq k$.

Example 1:

Let P be the proposition that the sum of the first n odd numbers is n^2; that is,

$$P(n): \quad 1 + 3 + 5 + \dots + (2n - 1) = n^2$$

Solution:

(The nth odd number is 2n – 1, and the next odd number is 2n +1.)
Observe that P(n) is true foe n = 1,

(i) n=1; P(1): $2*n\text{-}1 = 1^2$
(ii) n=k; Assuming P(k) is true,
we add (2k-1)+2 = 2K + 1 to both sides of P(k),

obtaining:

$$\begin{aligned} 1 + 3 + 5 + \dots + (2k - 1) + (2k + 1) &= k^2 + (2k + 1) \\ &= (k + 1)^2 \end{aligned}$$

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 2:.

$$P(n): \quad 1 + 2 + 3 + 4 + \dots\dots + n = 1/2\ n(n+1)$$

$$\sum_{i=1}^{n} i = 1/2\ n\ (\ n + 1)$$

solution :

n =1

(i) P(1) : left side =1 Right side =1/2 * 1 * (2) =1
(ii) let P(k) is true ; n=k

1 + 2 + 3 + 4 + … + k = 1/2 * k * (k+1) to prove that P(k+1) is true

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

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

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

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

So P is true for all $n \geq k$

EXERCISE:

Prove by induction:

1) $2 + 4 + 6 + \ldots + 2n = n(n+1)$
2) $1 + 4 + 7 + \ldots + (3n - 2) = 1/2\, n(3n - 1)$

Relations

There are many relations in mathematics :"less than" , "is parallel to ","is a subset of", etc. In a certain sense, these relations consider the existence or nonexistence of a certain connection between pairs of objects taken in a definite order. We define a relation simply in terms of ordered pairs of objects.

Product sets:

Consider two arbitrary sets **A** and **B** . The set of all ordered pairs (a ,b) where $\mathbf{a} \in \mathbf{A}$ and $\mathbf{b} \in \mathbf{B}$ 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:

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

$A \text{ x } B = \{(1,a), (1,b),(1,c),(2,a),(2,b),(2,c)\}$

$B \text{ x } A = \{(a,1),(a,2),(b,1),(b,2),(c,1),(c,2)\}$

Relations

Let **A** and **B** be sets. **A** binary relation, **R**, from **A** to **B** is a subset of **A**$\times$**B.** If $(x,y) \in \mathbf{R}$, we say that x is R-related to y and denote this by x**R**y

if $(x,y) \notin \mathbf{R}$, we write $x \not\mathbf{R} y$ and say that x is not **R**-related to y .

if R is a relation from **A** to **A** ,i.e. **R** is a subset of **A** $\times$ **A**, then we say that **R** is a relation on **A.**

The **domain** of a relation **R** is the set of all first elements of the ordered pairs which belong to **R**, and the **range** of **R** is the set of second elements.

Example 1:

Let $A = \{1, 2, 3, 4\}$. Define a relation R on A by writing $(x, y) \in R$ if $x < y$. Then

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

Example 2:

let $A = \{1,2,3\}$ and $R = \{(1,2),(1,3),(3,2)\}$. Then R is a relation on A since it is a subset of AxA with respect to this relation:

1R2, 1R3, 3R2 but $(1,1) \notin R$ & $(2,1) \notin R$

The domain of R is {1,3} and
the range of R is {2,3}

Example 3:
Let A = {1, 2, 3}. Define a relation R on A by writing $(x, y) \in R$, such that $a \geq b$, list the element of R

$$aRb \leftrightarrow a \geq b \ , \ a,b \in A$$

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

Representation of relations:

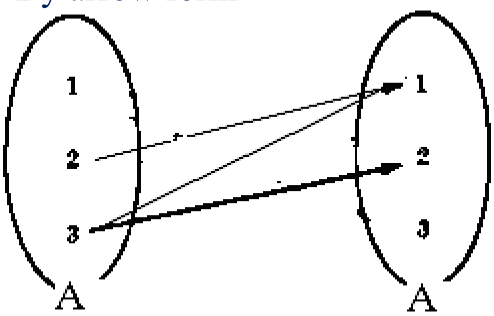
1) By language
2) By ordered pairs
3) By arrow form
4) By matrix form
5) By coordinates
6) By graph form

Example 4:
Let A = {1,2,3}, the relation R on A such that: $aRb \leftrightarrow a>b; \quad a,b \in A$

1) By language:
   $R = \{(a,b) : a,b \in A \text{ and } aRb \leftrightarrow a>b\}$

2) By ordered pairs
   R = {(2,1),(3,1),(3,2)}

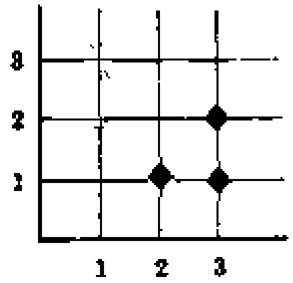
3) By arrow form



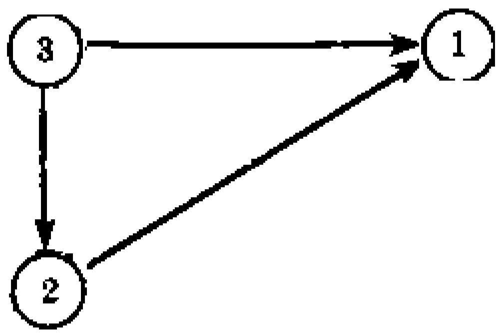
4) By matrix form

	1	2	3
1	0	0	0
2	1	0	0
3	1	1	0

5) By coordinates



6) By graph form



Properties of relations :

Let R be a relation on the set A

1) Reflexive : R is reflexive if : $\forall$ a $\in$A $\rightarrow$ aRa or (a,a) $\in$ R ; $\forall$ a, b $\in$A
2) Symmetric : aRb $\rightarrow$ bRa $\forall$ a,b $\in$A
3) Transitive : aRb $\wedge$ bRc $\rightarrow$ aRc
4) Equivalence relation : it is Reflexive & Symmetric & Transitive
5) Irreflexive : $\forall$ a $\in$A (a,a) $\notin$ R
6) AntiSymmetric : if aRb and bRa $\rightarrow$ a=b
   the relations $\geq$,$\leq$ and $\subseteq$ are antisymmetric

Example 5:

Consider the relation of C of set inclusion on any collection of sets:

1) A $\subset$ A for any set, so $\subset$is reflexive
2) A $\subset$ B dose not imply B $\subset$ A, so $\subset$ is not symmetric
3) If A $\subset$ B and B $\subset$ C then A $\subset$ C, so $\subset$ is transitive
4) $\subset$ is reflexive, not symmetric & transitive, so $\subset$ is not equivalence relations
5) A $\subset$ A, so $\subset$ is not Irreflexive
6) If A $\subset$ B and B $\subset$ A then A = B, so $\subset$ is anti-symmetric

Example 6:

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 $(2,2) \notin R$, so R is not reflexive
2) $(2,3) \in R$ but $(3,2) \notin R$, so R is not symmetric
3) $(1,2) \in R$ and $(2,3) \in R$ but $(1,3) \notin R$, so R is not transitive

So R is not Equivalence relation

Example 7 :

What is the properties of the relation =?

1) a=a for any element $a \in A$, 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

Inverse relations:

$$R^{-1} = \{(b,a) : (a,b) \in R\}$$

Example 1 :

Let R be the following relation on A ={1,2,3}

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

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

The matrix for R :

$$MR = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix},$$

and

$$MR-1 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix},$$

MR^{-1} is the transpose of matrix R

Composition of relations:

Let A, B, C be sets and let :

$R : A \to B$ $\quad$ $(R \subset A \times B)$

$S : B \to C$ $\quad$ $(S \subset B \times C)$

There is a relation from A to C denoted by

$R \circ S$ (composition of R and S)

$R \circ S : A \to C$

$R \circ S = \{(a,c) : \exists\, b \in B$ for which $(a,b) \in R$ and $(b,c) \in S\}$

Example :

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

$B = \{a, b, c, d\}$

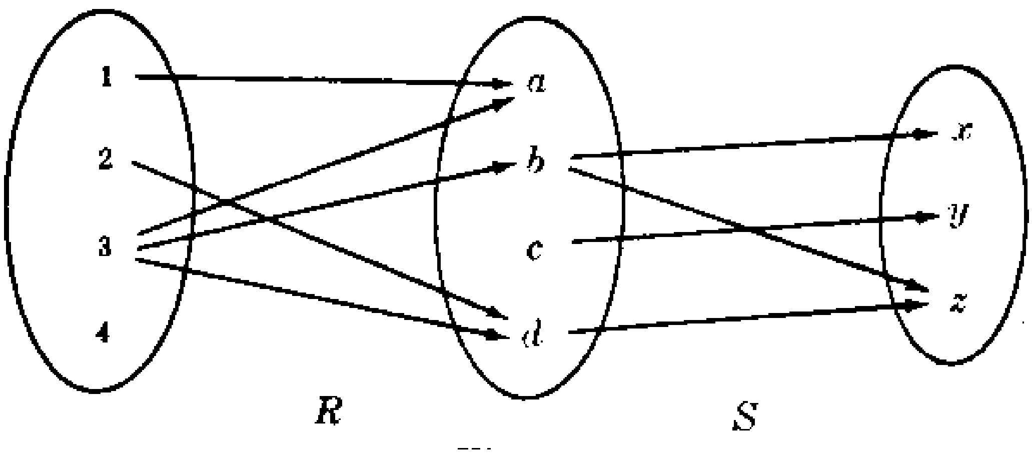
$C = \{x, y, z\}$

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

$S = \{(b,x),(b,z),(c,y),(d,z)\}$. $\quad$ Find $R \circ S$?

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 $dSz \Rightarrow 2(R \circ S)\, z$

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

2) The second way by matrix:

$$
MR = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}
\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}
$$

$$
MS = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}
\begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\
\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}
$$

$R \circ S = M_R \cdot M_S =$

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

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

EXERCISE:

1- Let A={a,b} and B={4,5,6}

a. List the elements in $A \times B$
b. List the elements $B \times A$
c. List the elements $A \times A$
d. List the elements $B \times B$

2- If A={a,b,c} B={1,2} and C={d,e} List all elements in $A \times B \times C$

3- If $A \subseteq C$ and $B \subseteq D$, show that $A \times B \subseteq C \times D$.

4- A=Z: a R b and only if a+b is even.

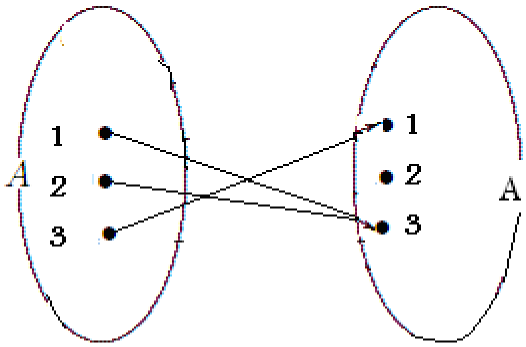
5- Consider the following arrays

VERT={1,2,6,4} , TAIL={1,2,2,4,4,3,4,1} , HEAD={2,2,3,3,4,4,1,3}, NEXT={8,3,0,5,0,0,0}, these describe a relation R on the set A={1,2,3,4}. Compute both the digraph of R and the matrix M_g .

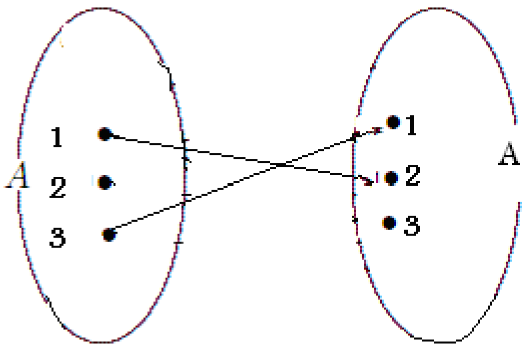
Function:

Function is an important class of relation.
Definition:
 Let A,B be two nonempty sets, a function F: A→B is a rule which associates with **each** element of A a **unique** element in B.
 The set A is called the **domain** of the function, and the set B is called the **range** of the function.

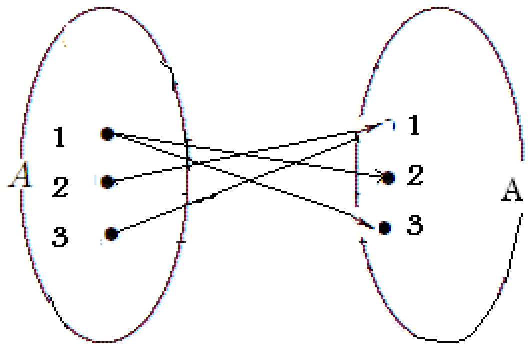
Example:
 consider the following relation on the set A={1,2,3}
F = {(1,3),(2,3),(3,1)}
F is a function



G = {1,2},(3,1)}
G is not a function from A to A



H = {(1,٣),(2,1),(1,2),(3,1)}
H is not a function

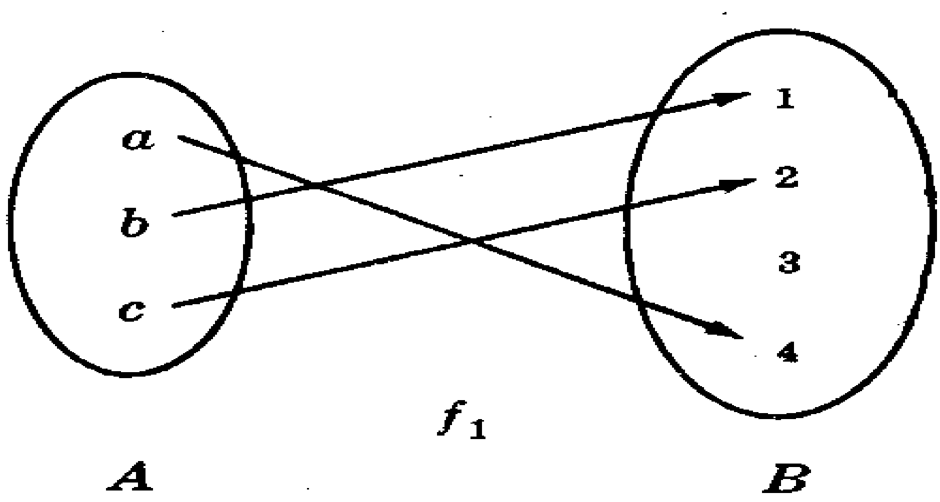


One-to-one ,onto and invertible functions :

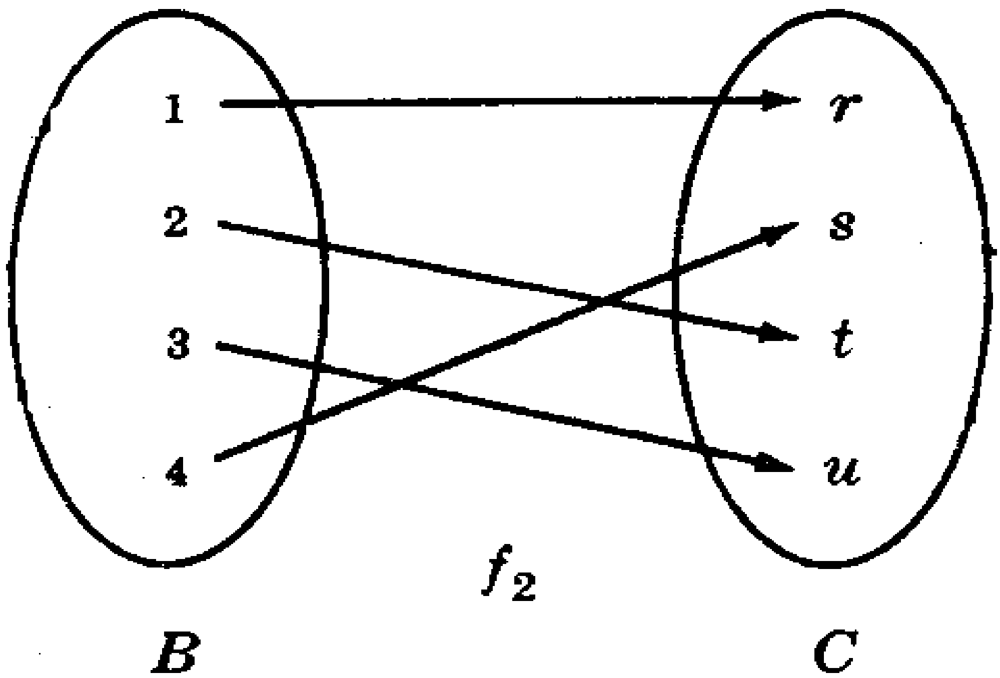
1) One –to-one : a function $F:A \rightarrow B$ is said to be one-to-one if different elements in the domain (A) have distinct images.
   If $F(a) = F(a') \Rightarrow a = a'$
   $F:A \rightarrow B$ is one-to-one if different elements in A have distinct images

2) Onto
   $F:A \rightarrow B$ is said to be an onto function if each element of B is the image of some element of A.
   $\forall\ b \in B \quad \exists \quad a \in A : F(a) = b$

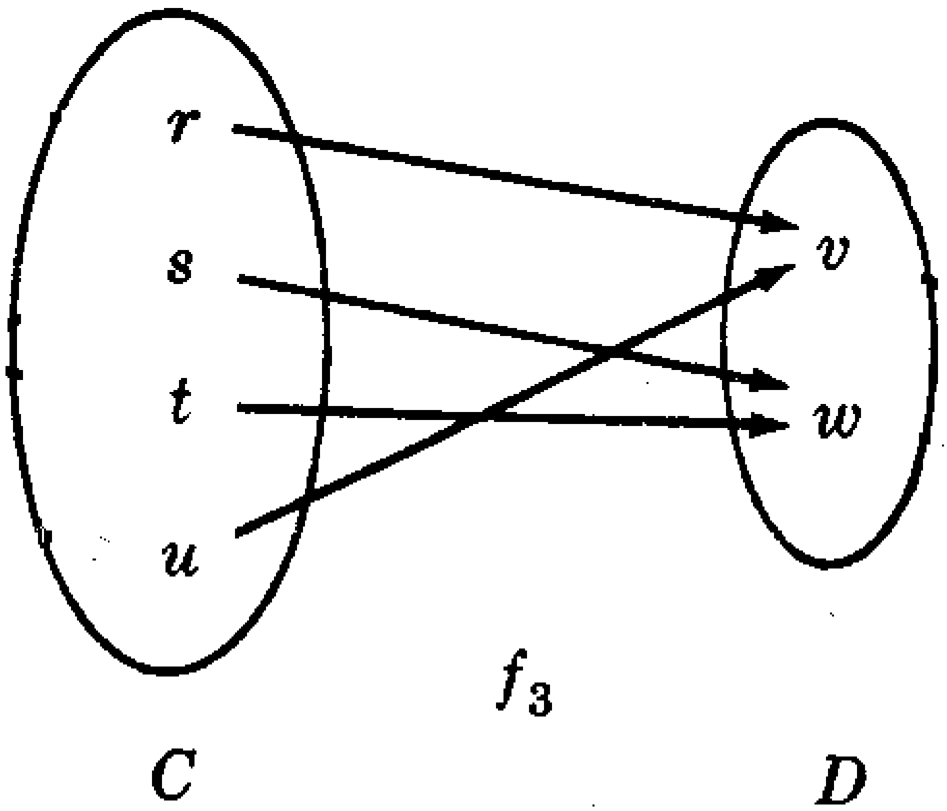
3) Invertible (One-to-one correspondence)
   $F:A \rightarrow B$ is invertible if its inverse relation f^{-1} is a function $F:A \rightarrow B$
   $F:A \rightarrow B$ is invertible if and only if F is **both** one-to-one and onto
   $F^{-1} : \{(b,a)\ \forall\ (a,b) \in F\}$



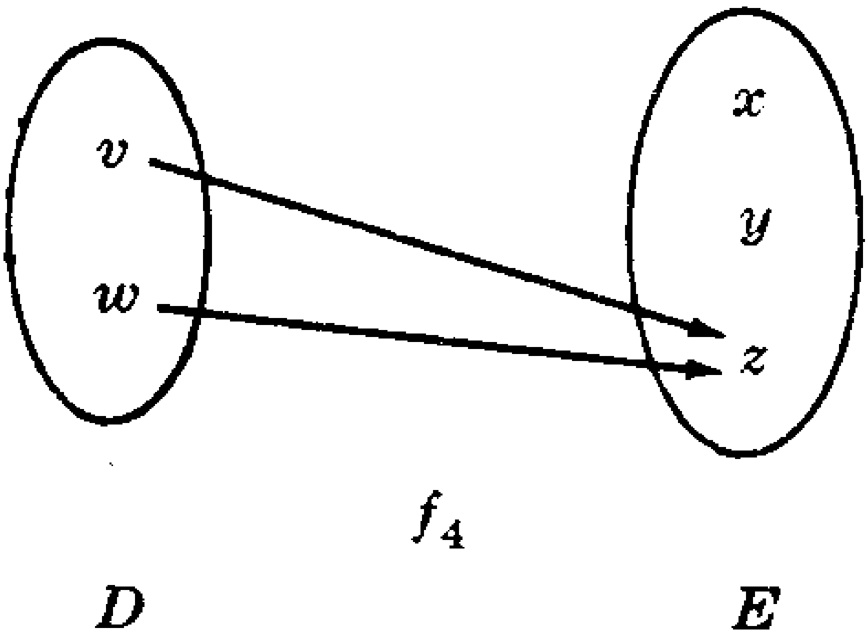
one to one & not onto [3 ∈ B but it is not the image under f1]



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 plot a function by plot 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 $f:R\rightarrow R$ and $f(x)= x^3$, find f(x)

$f(3) = 3^3 = 27$

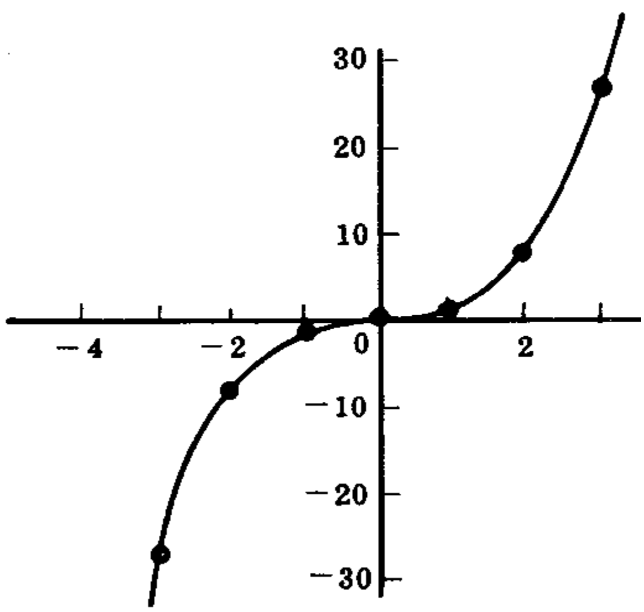
$f(-2) = (-2)^3 = -8$

$f(y) = y^3$

$f(y+1) = (y+1)^3 = y^3 + 3y^2 + 3y + 1$

$f(x+h) = (x+h)^3 = x^3 + 3x^2h + 3xh^2 + h^3$

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



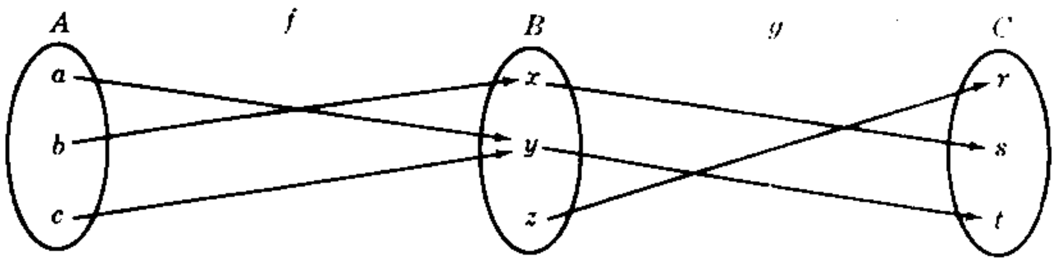
Composition of function:

Let $f:A\rightarrow B$ and $g:B\rightarrow C$, to find the composition function $gof:A\rightarrow C$

(gof)(a) = g(f(a)) = g(y) = t

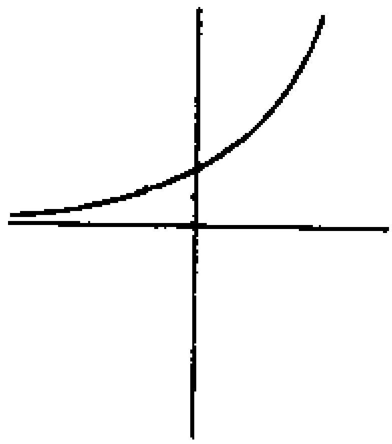
(gof)(b) = g(f(b)) = g(x) = s

(gof)(c) = g(f(c)) = g(y) = t



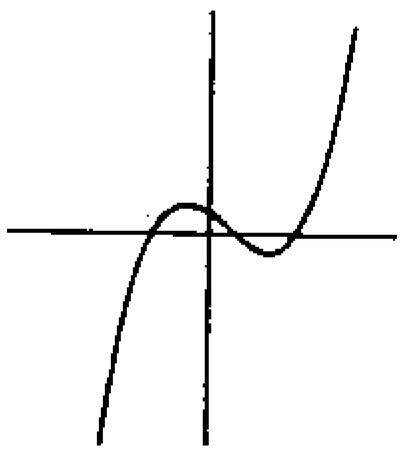
By using the Graph of function : One-to-one ,onto and invertible functions :

One-to-one : A function f:A→B is said to be one-to-one if there are no 2 distinct pairs (a1,b) and (a2,b) in the graph. Each horizontal line can intersect the graph of f in <u>at most</u> one point.



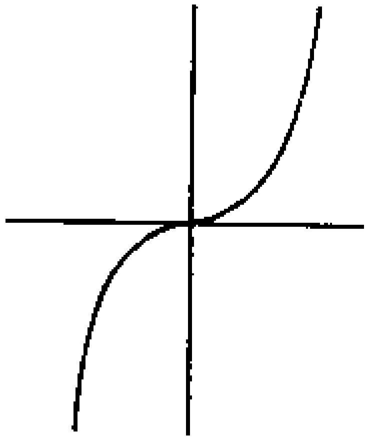
$f_2(x) = 2^x$

Onto : For every b∈B there must be at most one a∈A such that (a,b) belong to the graph of f. Each horizontal line must intersect the graph of f <u>at least once</u>

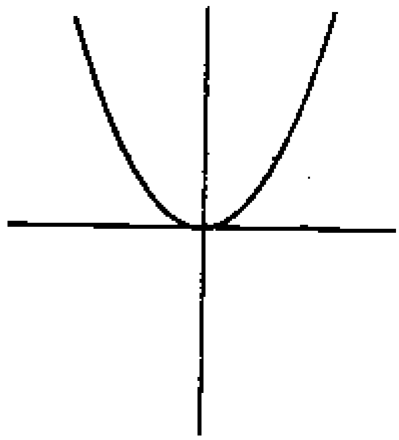


$f_3(x) = x^3 - 2x^2 - 5x + 6$

Invertible: f is both one-to-one and onto. Each horizontal line will intersect the graph of f in exactly one point.



$f_4(x) = x^3$



$f\ (x) = x^2$

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

EXERCISE:

1- Let A={1,2,3,4} and B-{a,b,c,d} and let f ={(1,a),(2,a),3,d),(4,c)}.

2- Let ={1,2,3} B={4,5,6} C={7,8} D={a,b,c,d}, consider the following four functions from A→B, A→D,B→C and A→B,each defined by its one- to-one.

3- Let f : R→R and $f(x) = x^3 + 1$ and x= [1,2,3,0,-1,-2,-3] , find f(x)

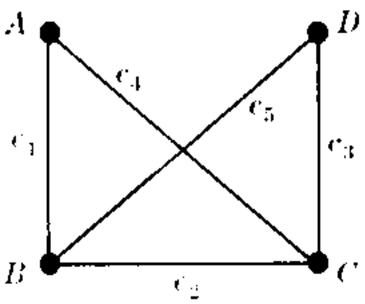
4- Let R be the set of all real numbers, and let f : R→R be defined by f=x^2 ,Is f invertible .

Graphs:

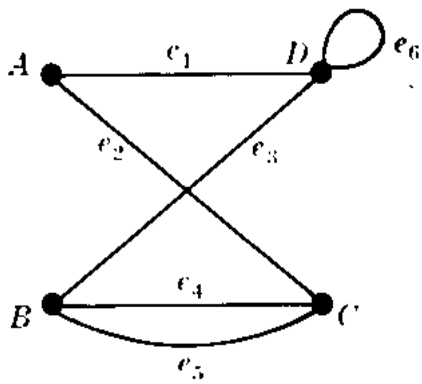
A graph G consists of two things:
(i) A set V whose elements are vertices, points or nodes.
(ii) A set E of unordered pairs of distinct vertices called edges.

We denote such a graph by G(V,E) .
Vertices u and v are said to be adjacent if there is an edge {u,v}.
Graphs are the most useful model with computer science such as logical design, formal languages, communication network, compiler writing and retrieval.

G(V,E)
V = {V1, V2, V3,V4}
E = {e1, e2, e3, e4}
E = {(v1,v2),(v2,v3),(v3,v1),(v3,v4)}



(a) Graph



(b) Multigraph

For example we have in (a) the graph G(V,E) where (i) V consists of four vertices A, B, C, D ; and (ii) E consists of five edges e1 ={A,B}, e2 = {B,C}, e3 = {C, D}, e4 = {A, C}, e5 = {B, D}.
The diagram in (b) is not a graph but a multigraph . The reason is that e4 and e5 are multiple edges, i.e. edges connecting the same endpoints, and e6 is a loop, i.e. an edge whose endpoints are the same vertex. The definition of a graph does not permit such multiple edges or loops.
Let G(V,E) be a graph. Let V' be a subset of V and let E' be subset of E whose end-points belong to V'. Then G(V',E') is a graph and is called a subgraph of G(V,E). If E' contains all the edges of E whose endpoints lie in V', then G(V',E') is called the subgraph generated by V'.

Degree :

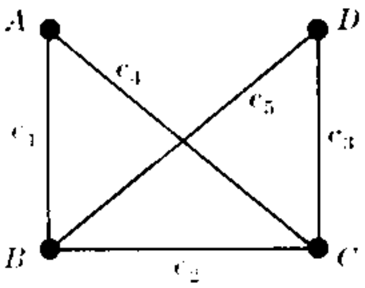
The degree of a vertex v, written deg(v), is equal to the number of edges which are incident on v. since each edge is counted twice in counting the degrees of the vertices of a graph, we have the following result.
Theorem: The sum of the degrees of the vertices of a graph is equal to twice the number of edges.

For example, in the figure (a) we have

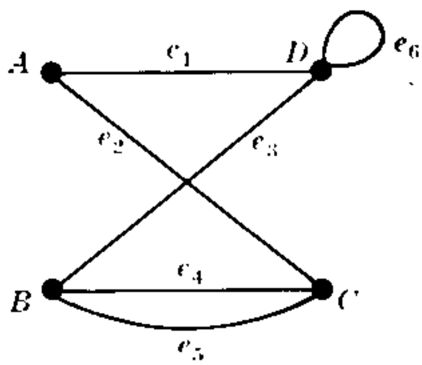
deg(A) = 2,
deg(B) = 3,
deg(C) = 3,
deg(D) = 2

The sum of the degrees equals ten which, as expected, is twice the number of edges.

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 (b) 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.



(a) Graph



(b) Multigraph

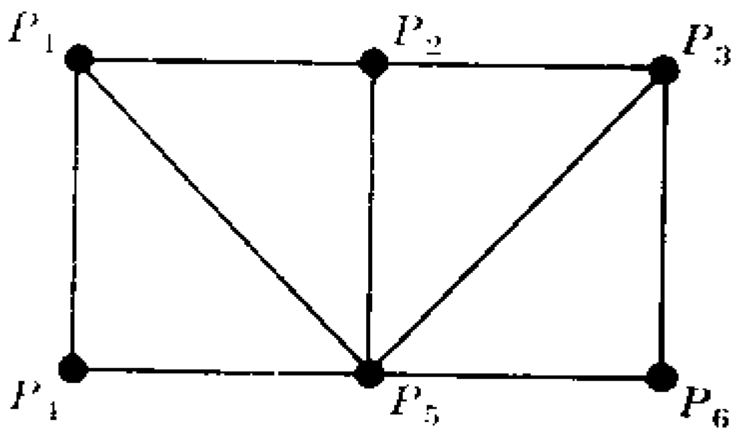
Connectivity

A **walk** in a multigraph consists of an alternating sequence of vertices and edges of the form

$$v_0, e_1, v_1, e_2, v_2, \ldots\ldots, e_{n-1}, v_{n-1}, e_n, v_n$$

Length of walk: is the number n of edges.
Closed walk: the walk is said to be closed if $v_0 = v_n$. Otherwise, we say that the walk is from v_0 to v_n.
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 $v_1 = v_n$

Example: Consider the following graph, then



(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, P5, P6)
Is not a walk since there is no edge {P2, P6}.

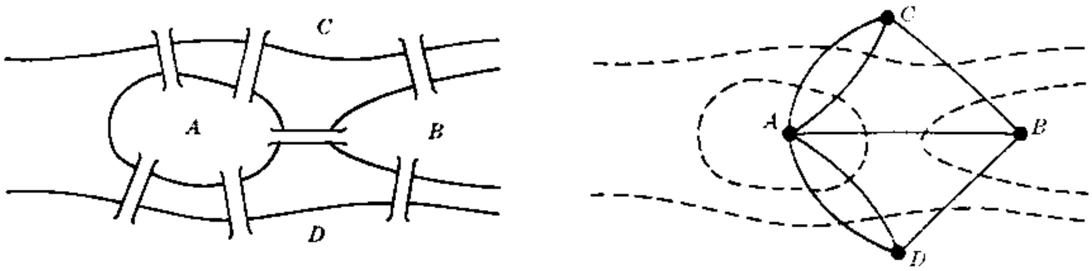
The sequence: (P4, P1, P5, P2, P3, P5, P6)
Is a trail since no edge is used twice; but it is not a path since the vertex P5 is used twice.

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

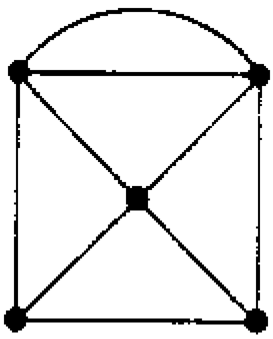
The Bridges of konigsberg, traversable multigraphs

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 (b).

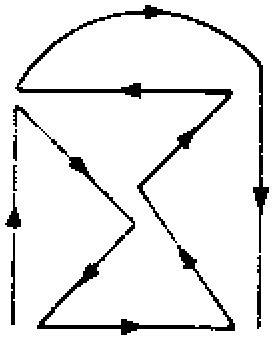


(a) Königsberg in 1736 (b) Euler's graphical representation

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)



(a)



(b)

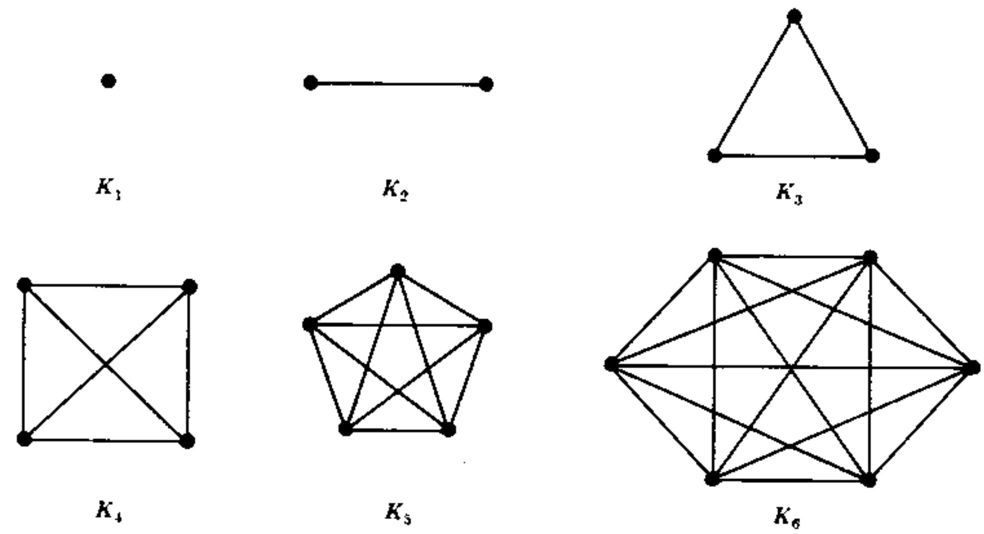
We now show how Euler proved that the konigsberg multigraph is not traversable and the walk in it is impossible. Suppose a multigraph is traversable and that a traversable trail does not begin or end at vertex P. thus the edges in the trail incident with P must appear in pairs, and so P is an even vertex.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

Theorem (Euler) : A finite connected graph is eulerian if and only if each vertex has even degree.

Corollary: Any finite connected graph with two odd vertices is traversable. A traversable trail may begin at either odd vertex and will end at the other odd vertex.

Special graph:

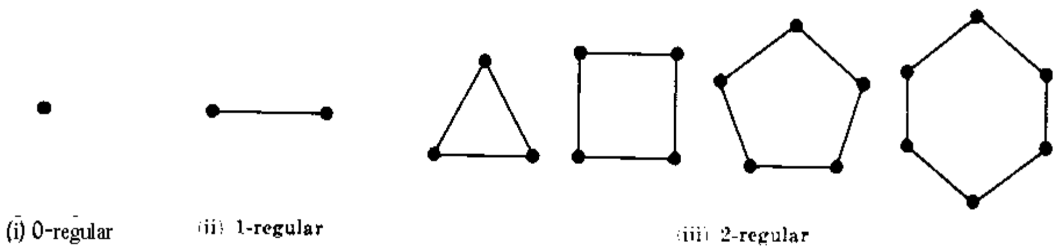
1- **Complete graph:**
   Each vertex is connected to every other vertex. The complete graph with n vertices is denoted by Kn

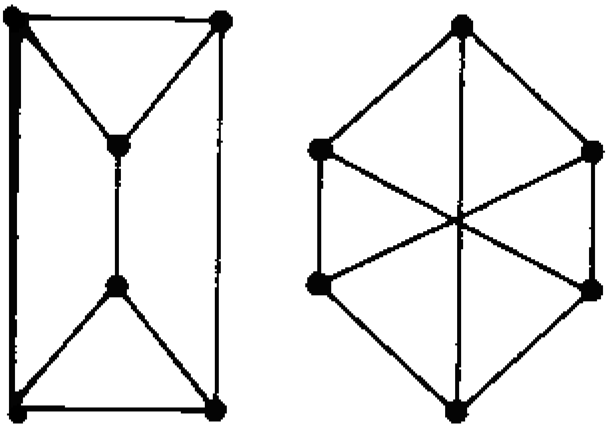


2- Regular Graph

Every vertex has the same degree. A graph G is regular of degree K or K- regular if every vertex has degree K.

Example: 2-regular graph with every vertex has degree 2.

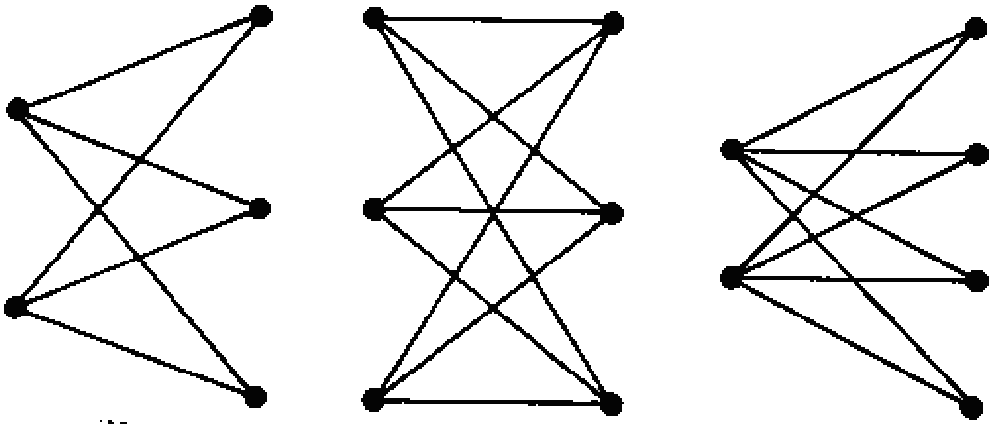




(vi) 3-regular

3- **Bipartite graph :**

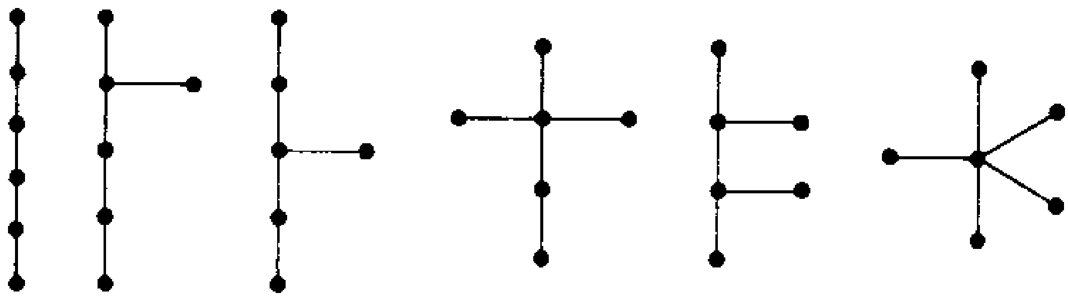
Graph G is said to be bipartite if its vertices V can be partitioned into two subsets M and N such that each edge of G connects a vertex of M to a vertex of N.



$K_{2,3}$ $K_{3,3}$ $K_{2,4}$

4- Tree graph:

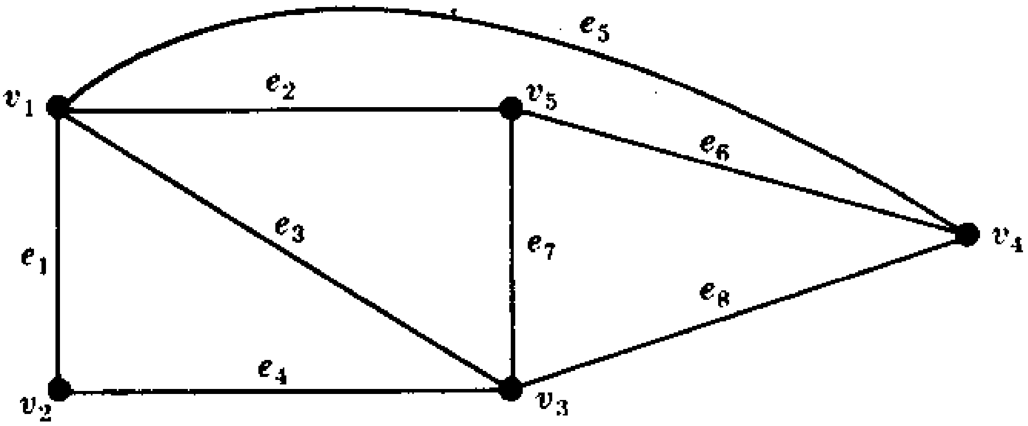
A graph is said to be cycle-free or acyclic if it has no cycle. A connected graph with no cycle is called a tree.



EXERCISE:

MATRICES AND GRAPHS:

Let G be a graph with vertices v1, v2, …….,vm and edges e1, e2, ….,en. It is sometimes practical, especially for computational reasons, to represent G by a matrix. Note that the edges of G can be represented by an $n \times 2$ integer matrix B where each row of B denotes an edge of G, e.g. the row (3,4) would denote the edge (v3,v4). This edge matrix B does not completely describe G unless we are also given the number m of vertices of G. We do discuss two other widely used matrix representations of G.



$$B = \begin{pmatrix} 1 & 2 \\ 1 & 5 \\ 1 & 3 \\ 2 & 3 \\ 1 & 4 \\ 4 & 5 \\ 3 & 5 \\ 3 & 4 \end{pmatrix}$$

(1) Adjacency matrix. Let A = (aij) be the m × m matrix defined by:

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge, i.e. if } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

Then A is called the adjacency matrix of G. Observe that $a_{ij} = a_{ji}$;hence A is a symmetric matrix.

$$A = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

(2) Incidence matrix. Let M = (mij) be the m × n matrix defined by

$$m_{ij} = \begin{cases} 1 & \text{if the vertex } v_i \text{ is incident on the edge } e_j \\ 0 & \text{otherwise} \end{cases}$$

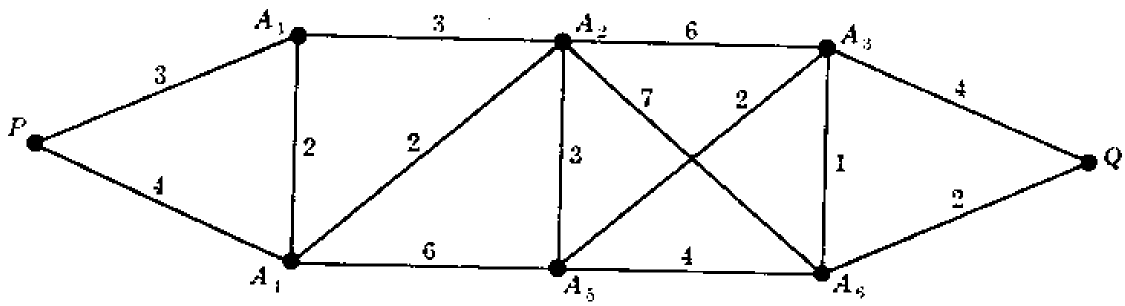
$$M = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0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

Labeled 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.

One important problem in graph theory is to find a minimum path between two given points.

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)

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

Tree:

Tree is a connected graph with no cycle

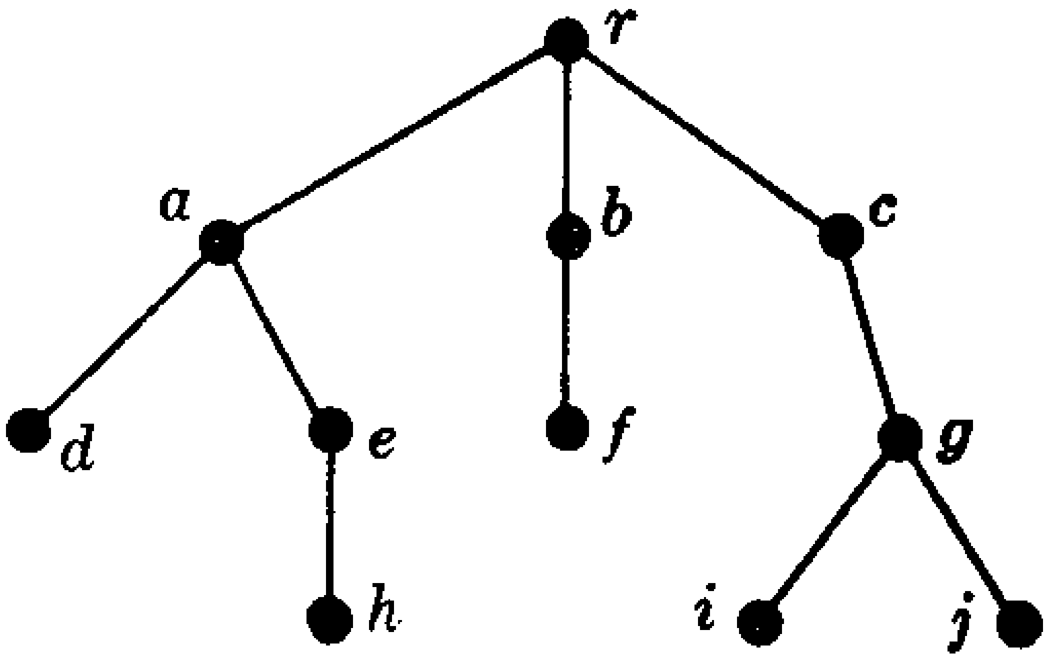
Theorem:

Let G be graph with more than one vertex. Then the following are equivalence:

1) G is a tree.
2) G is cycle-free with (n-1) edges.
3) G is connected and has (n-1) edges. (i.e: if any edge is deleted then the resulting graph is not connected)

Rooted tree:

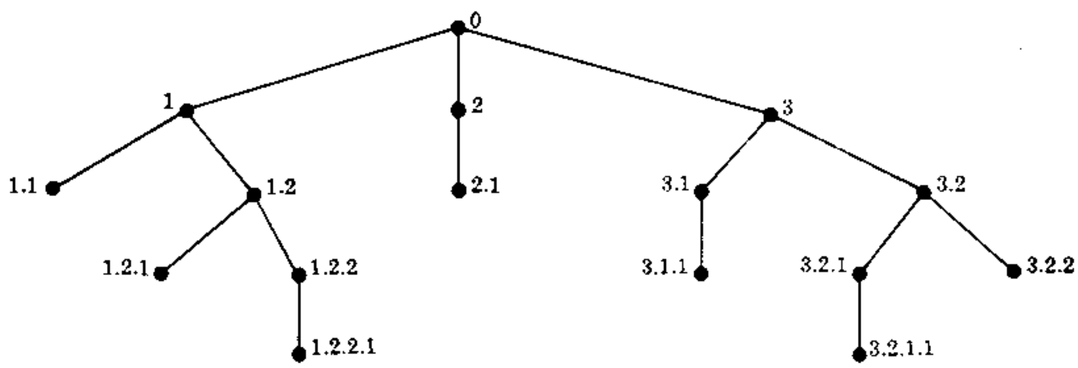
A rooted tree R consists of a tree graph together with vertex r called the root of the tree.



Height or depth: The number of levels of a tree

Leaves: The vertices of the tree that have no child (vertices with degree one)

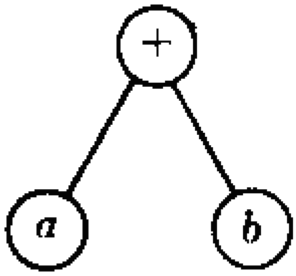
Order Rooted Tree (ORT): Whenever draw the digraph of a tree, we assume some ordering at each level, by arranging children from left to right.



Degree of tree: The largest number of children in the vertices of the tree
Binary tree : every vertex has at most 2 children

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

the binary rooted tree for a+b is :

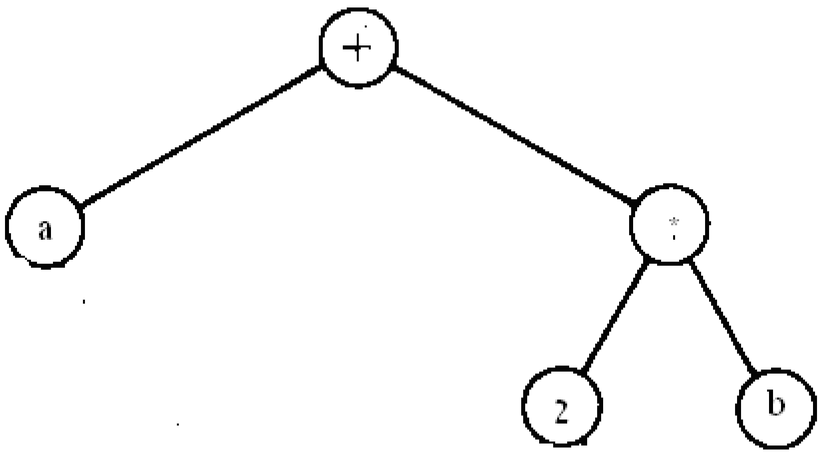


The variable in the expression a & b appear as leaves and the operations appear as the other vertices.

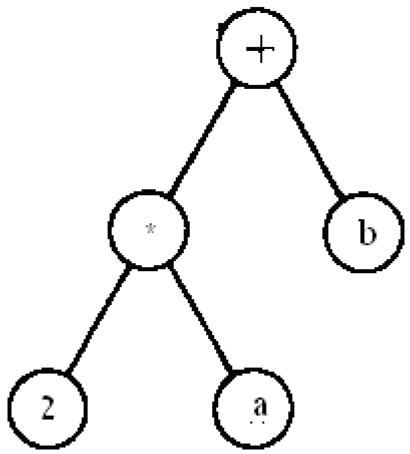
Polish notation:
The polish notation form of an algebraic expression represents the expression unambiguously with out the need for parentheses
1) a + b (infix)
2) + a b (prefix)
3) a b + (postfix)

example 1: infix polish notation is : a + b
prefix polish notation : + a b

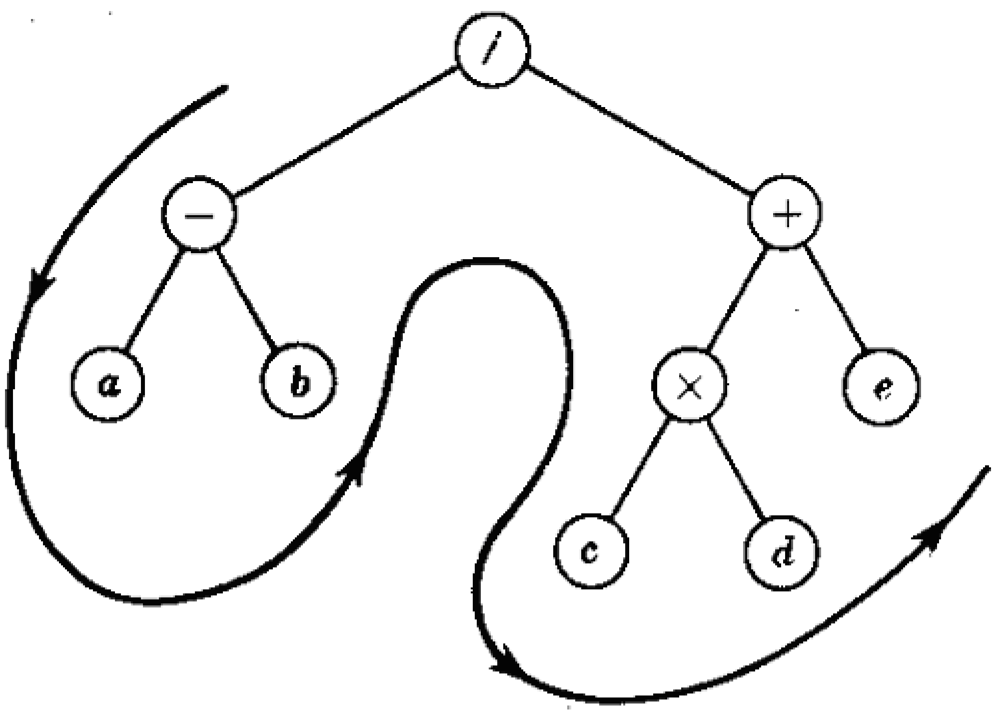
example 2: infix polish notation is : a + 2 * b
prefix polish notation : + a * 2 b



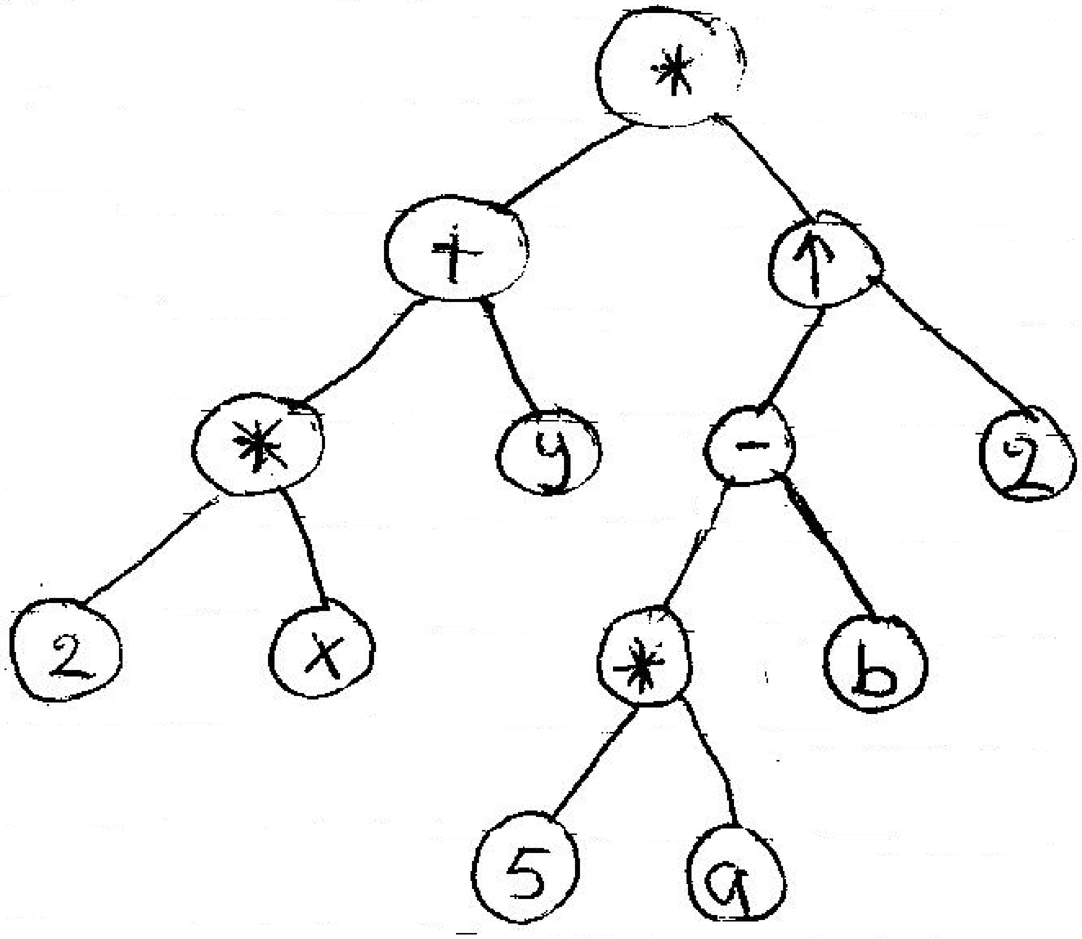
example 3: infix polish notation is : 2 * a + b
prefix polish notation : + * 2 a b



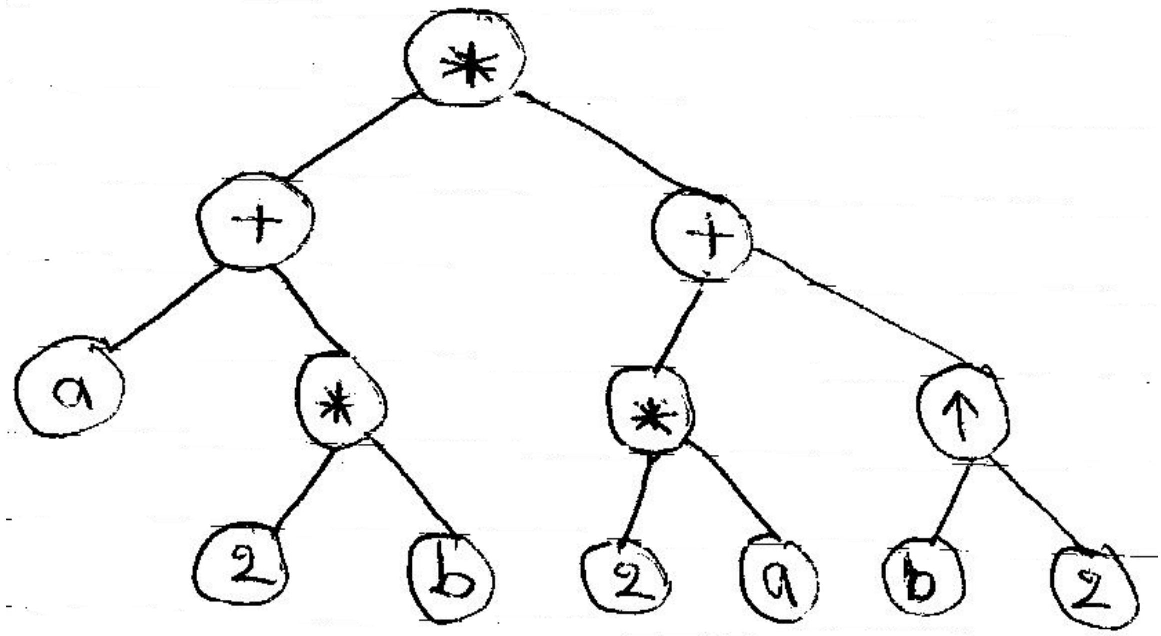
example 4: infix polish notation is : (a – b) / (c * d) + e)
prefix polish notation : / - a b + * c d e



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



example 6: infix polish notation is : (a + 2 * b) (2 * a + b^2)
prefix polish notation : * + a * 2 b + * 2 a ^ b 2



EXERCISE:

1- If possible, compute:

a. $(2C-3F)^T B$
b. $A^T(D+F)$
c. $(3E)A^T$
d. $(BC)^T$ and $C^T B^T$
e. $(B^T + A)C$
f. $(D^T + E)F$

2- Each relation R is defined on the set A ,in each case determine if R is a tree, and if it ism, find the root.

a. A={a,b,c,d,e,f}
R={(a,d),(b,c),(c,a),(d,e)}
b. A={a,b,c,d,e}
R={(a,b),(b,e), (c,d)),(d,b),(c,a)}
c. A={a,b,c,d,e,f}
R={(a,b),(c,e),(f,a),(f,c),(f,d)}
d. A={1,2,3,4,5,6}
R={(2,1),(3,4),(5,2),(6,5),(6,3)}
e. A={1,2,3,4,5,6}
R={(1,1),(3,1),(2,3),(3,4),(4,5).(4,6)}

3- Defined Polish notation ,and with examples.

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) A next-state function f

$$f: S \times A \rightarrow S$$

(5) An output function g

$$g: S \times A \rightarrow Z$$

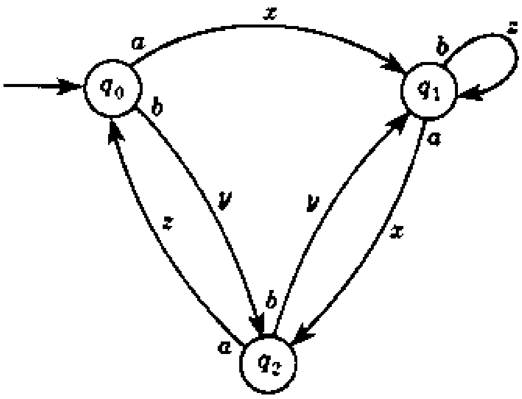
This machine M is denoted by M = (A, S, Z, q0, f, g) where q0 is the initial state.

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 x A→ 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 x A → 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.



	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 diagram **State table**

Example 2:

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

Example 3:

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

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

Output :1, 1, 1, 0, 1, 1,b

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

The input symbols are : A={00, 01, 10, 11,b}

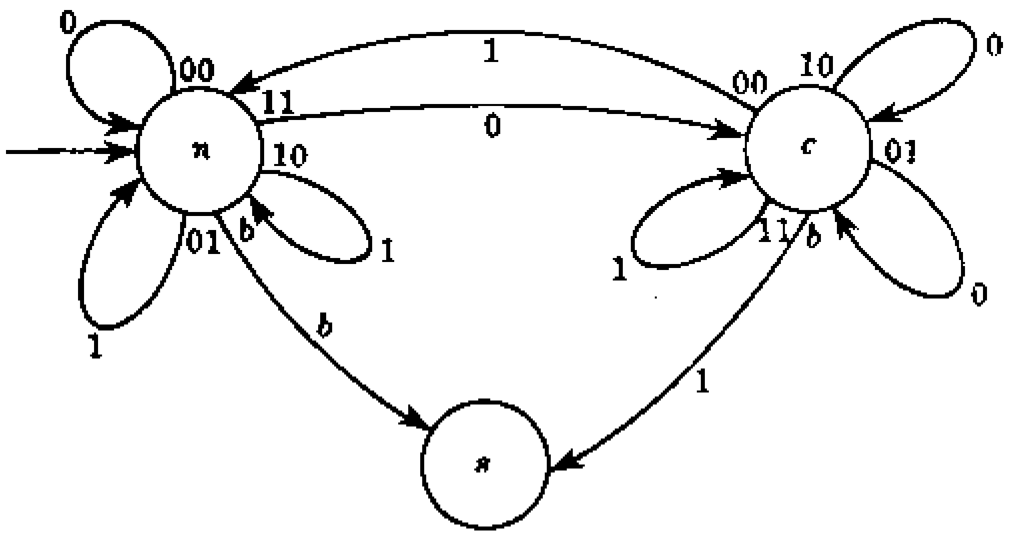
The output symbols are : Z={0, 1,b}

We have 3 states:

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

n is the initial state

	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



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 $\in$T. We will let L(M) denote the set of all strings which are recognized by M.

Example

Example:

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 the table :

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

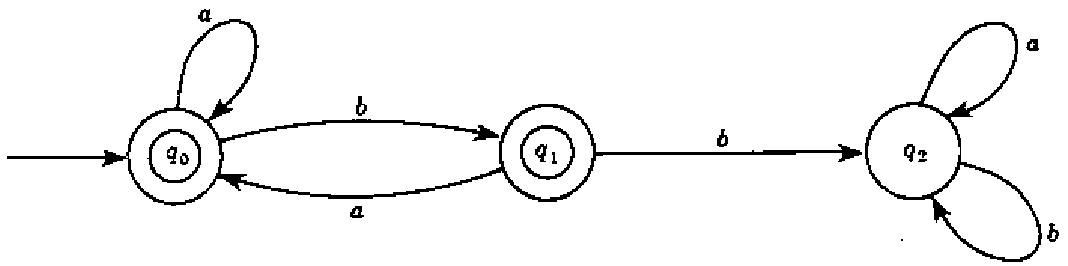
The automaton M will recognize those strings which do not have two successive b's.

Thus M will accept:

aababaaba, aaa, baab, abaaababab, b, aabaaab

But will reject :

aabaabba, bbaaa, ababbaab, bb, abbbbaa



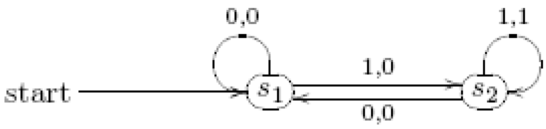
Some Examples of FSM

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. 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:

$$x = 10111010101111110101$$
$$y = 00011000000111110000$$

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.2.1:

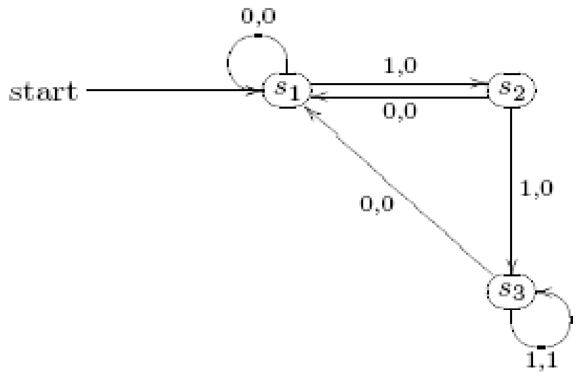
Suppose again that *A (input)* = Z (output) = *{*0*,* 1*}*, and that we want to design a finite state machine that recognizes the sequence pattern 111 in the input string $x \in A*$. An example of the same input string $x \in A*$ and its corresponding output string $y \in Z*$ is shown below:

$$x = 101110101011111110101$$
$$y = 000010000000111110000$$

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 $s1$ the "passive" state, by $s2$ the "expectant" state and by $s3$ the "excited" state, then we have the state diagram below:



We then have the corresponding transition table:

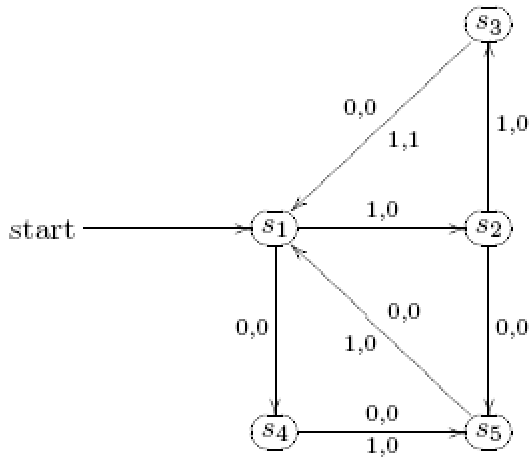
	g 0	g 1	f 0	f 1
$+s_1+$	0	0	s_1	s_2
s_2	0	0	s_1	s_3
s_3	0	1	s_1	s_3

Example.3.1:

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 $x \in A*$, but only when the third 1 in the sequence pattern 111 occurs at a position that is a multiple of 3. An example of the same input string $x \in A*$ and its corresponding output string $y \in Z*$ is shown below:

$$x = 1011101010111110101$$
$$y = 0000000000000100000$$

In order to achieve this, the finite state machine must as before 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. However, this is not enough, as we also need to keep track of the entries to ensure that the position of the first 1 in the sequence pattern 111 occurs at a position immediately after a multiple of 3. It follows that if we permit the machine to be at its "passive" state only either at the start or after 3k entries, where $k \in$ N, then it is necessary to have two further states to cater for the possibilities of 0 in at least one of the two entries after a "passive" state and to then delay the return to this "passive" state. We can have the state diagram below:



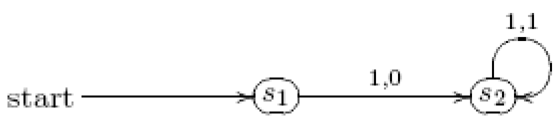
We then have the corresponding transition table:

	g 0	g 1	f 0	f 1
$+s_1+$	0	0	s_4	s_2
s_2	0	0	s_5	s_3
s_3	0	1	s_1	s_1
s_4	0	0	s_5	s_5
s_5	0	0	s_1	s_1

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 1.2:
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:

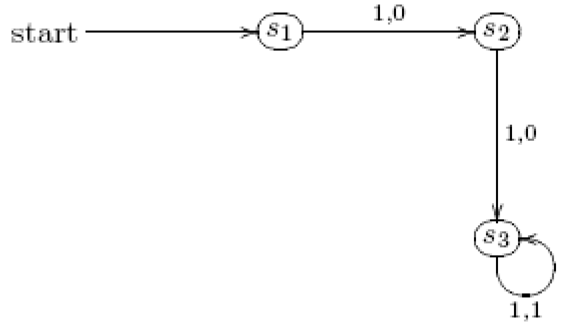


It now remains to study the situation when we have the "wrong" input at each state. Naturally, with a "wrong" input, the output is always 0, so the only unresolved question is to determine the next state.

Note that whenever we get an input 0, the process starts all over again; in other words, we must return to state $s1$.. We therefore obtain the state diagram as in Example.1.1.

Example 2.2:

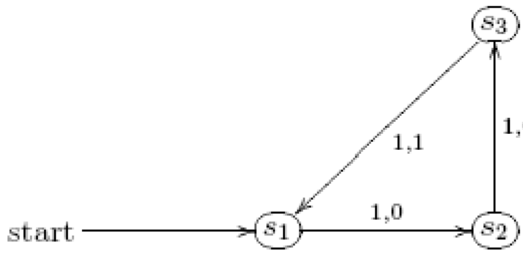
Suppose again that A *(input)* = Z (output) = $\{0, 1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 111 in the input string $x \in A*$. Consider first of all the situation when the required pattern occurs repeatly and without interruption. In other words, 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 $s1$. We therefore obtain the state diagram as Example 2.1.

Example.3.2:

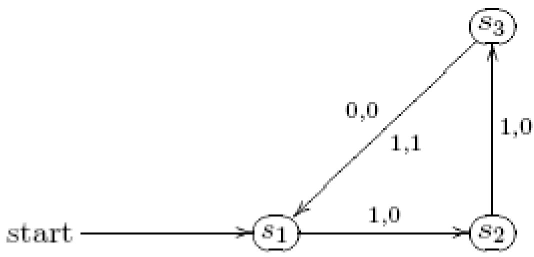
Suppose again that A *(input)* = Z (output) = $\{0, 1\}$, and that we want to design a finite state machine that recognizes the sequence pattern 111 in the input string $x \in A*$, but only when the third 1 in the sequence pattern 111 occurs at a position that is a multiple of 3. Consider first of all 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:



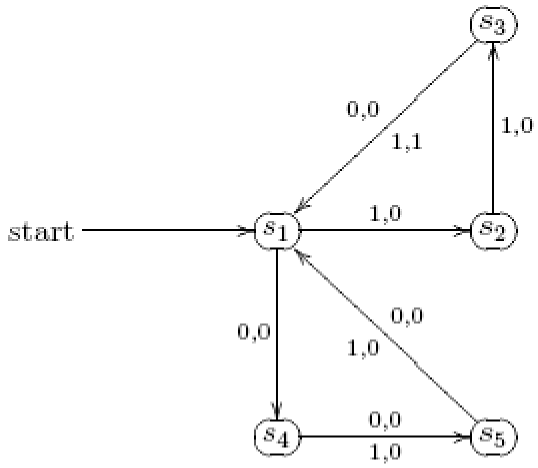
It now remains to study the situation when we have the “wrong” input at each state. As before, with “wrong” input, the output is always 0, so the only unresolved question is to determine the next state. Suppose that the machine is at state $s3$, and the wrong input 0 is made. We note that if the next three entries are 111, then that pattern should be recognized. It follows that $f(s3, 0) = s1$. We now 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.

Suppose that the machine is at state $s3$, and the wrong input 0 is made. We note that if the next three entries are 111, then that pattern should be recognized. It follows that $f(s3, 0) = s1$. We now have the following incomplete state diagram:



Suppose that the machine is at state $s1$, and the wrong input 0 is made. We note that the next two inputs cannot contribute to any pattern, so we need to delay by two steps before returning to $s1$. We now have the following incomplete state diagram:



Finally, it remains to examine $f(s2, 0)$. We therefore obtain the state diagram as Example 3.1.
Note that in Example 3.2, in dealing with the input 0 at state *s*1, we have actually introduced the extra states *s*4 and *s*5. These extra states are not actually involved with positive identification of the desired pattern, but are essential in delaying the return to one of the states already present. It follows that at these extra states, the output should always be 0. However, we have to investigate the situation for input 0 and 1.

EXERCISE

1-Consider the machine state transition table is shown below:

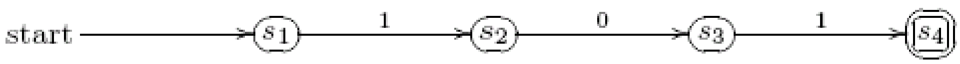
	0	1
1	1	4
2	3	2
3	2	3
4	4	1

Here S={1,2,3,4}, Show that
R={(1,1),(1,4),(4,1),(4,4),(2,2),(2,3),(3,2),(,3,3)}
Is a machine congruence and construct the state transition table for the corresponding quotient machine.

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:



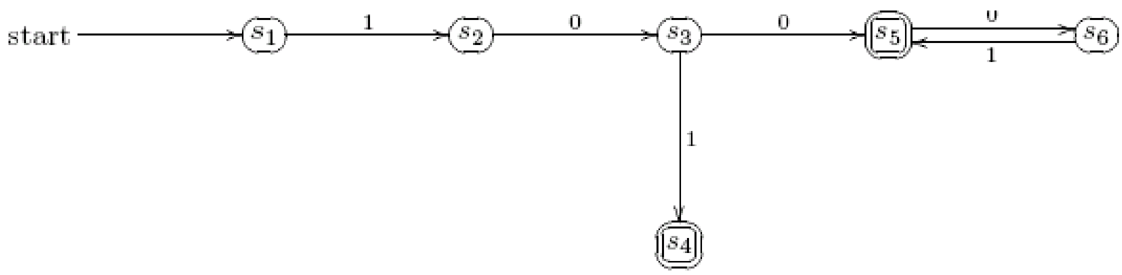
We can also describe the same information in the following transition table:

	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 $A = (S, I, v, T, s1)$, where
(a) S is the finite set of states for A;
(b) I is the finite input alphabet for A;
(c) $v : S \times I \rightarrow S$ 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.

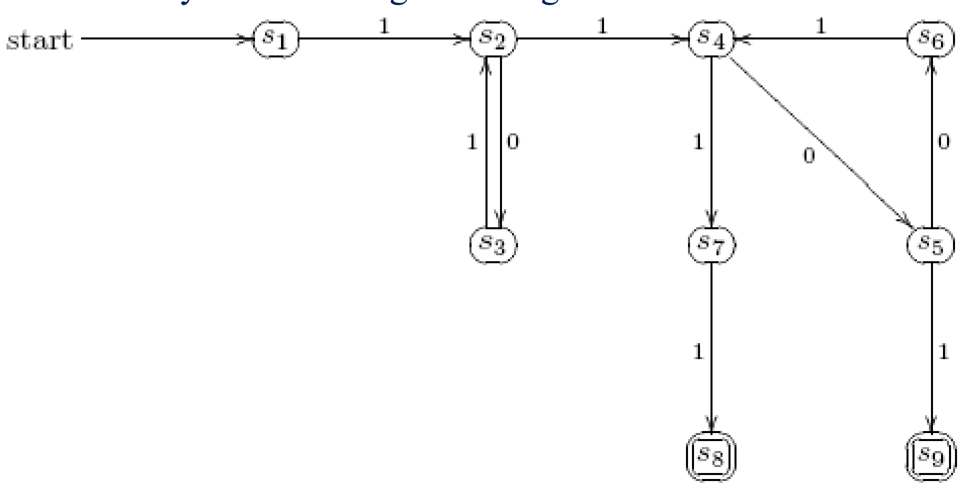
Example.2. We shall construct a deterministic finite state automaton which will recognize the
input strings 101 and 100(01)*and nothing else. This automaton can be described by the following state diagram:



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

Example .3.
We shall construct a deterministic finite state automaton which will recognize the
input strings 1(01)*1(001)*(0+1)1 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	ν 1
$+s_1+$	–	s_2
s_2	s_3	s_4
s_3	–	s_2
s_4	s_5	s_7
s_5	s_6	s_9
s_6	–	s_4
s_7	–	s_8
$-s_8-$	–	–
$-s_9-$	–	–

EXERCISE:

1-Consider a deterministic finite automata which will recognize the input string 1(01) *and 1(11)*(0+1) ,and nothing else, then decide which of the following string are accepted by this automata:
111,1111,11111,10110,1101,101110.