



Discrete Structure Notes

Elements Of Discrete Mathematics (Pondicherry University)



Scan to open on Studocu

Proof Methods

A proof is a valid argument that establishes the truth of a mathematical statement, using the hypotheses of the theorem, if any, axioms assumed to be true, and previously proven theorems.

Using these ingredients and rules of inference, the proof establishes the truth of the statement being proved.

We move from formal proofs, as seen in the previous section, to informal proofs, where more than one inference rule may be used at each step, where steps may be skipped, and where axioms and rules of inference used are not explicitly stated.

Some Terminology

Definition

A precise and unambiguous description of the meaning of mathematical term.

Theorem

A statement that can be shown to be true using vigorous (strong) mathematical reasoning.

A statement that can be shown to be true (sometimes referred to as facts or results).

Less important theorems are often called propositions.

Lemma :

A lemma is a less important theorem, used as an auxiliary result to prove a more important theorem.

It is used as an auxiliary result to prove a theorem. It is stepping stone to prove theorem.

Corollary

It is a theorem proven as an easy consequence of a theorem.

Conjecture

A conjecture is a statement that has been proposed a true statement. If proved later then it can be used as a theorem but it may be false

∨ Axiom (or postulates) are statements that we assume to be true ∨ A proof is a valid argument that establishes the truth of a theorem.

. Rules of inference,

Axiom (or postulates)

Axiom or postulates are statements that we assume to be true (algebraic axioms specify rules for arithmetic like commutative laws) without proof. These are building blocks to prove theorem.

Proof

A proof is a valid argument that establishes the truth of a theorem. The statements used in a proof include axioms, hypotheses (or premises), previously proven theorems and definitions. Rules of inferences, together with definition of terms, are used to draw conclusions from other assertions, tying together the steps of a proof.

Three methods of showing statements of the type $p \rightarrow q$ are true:

(1) **Direct proofs:** Assume p is true; the last step establishes q is true.

(2) **Proof by Contraposition:** Uses a direct proof of the contrapositive of $p \rightarrow q$, which is $\neg q \rightarrow \neg p$. That is, assume $\neg q$ is true; the last step established $\neg p$ is true.

(3) **Proof by Contradiction:** To prove that P is true, we assume $\neg P$ is true and reach a contradiction, that is that $(r \wedge \neg r)$ is true for some proposition r . In particular, to prove $(p \rightarrow q)$, we assume $(p \rightarrow q)$ is false, and get as a consequence a contradiction. Assuming that $(p \rightarrow q)$ is false $= (\neg p \vee q)$ is false $= (p \wedge \neg q)$ is true.

Formal and Informal Proof

A formal direct proof of a conditional statement $p \rightarrow q$ works as follows:

Assume p is true, build steps using inference rules, with the final step showing that q is true.

In informal direct proof, we assume that p is true, and use axioms, definitions and previous theorems, together with rules of inference to show that q must be true.

Direct proof

Assume p is true and the last step establishes q is true.

Before we give example of direct proof, we need a definition.

Definition :

The integer n is even if there exists an integer k such that $n=2k$ and n is odd if there exists an integer k such that $n=2k+1$.

Theorem (use direct proof)

1) *If n is an odd integer, then n^2 is odd.*

Let $p(n)$: ' n is an odd integer' is true
then,

$n=2k+1$ (from definition)
 then,
 $n^2=(2k+1)^2$
 $n^2=4k^2+4k+1$
 $n^2=2(2k^2+2k)+1$
 here, $(2k^2+2k)$ is an integer, so,
 $2(2k^2+2k)$ is even
 n^2 is odd.

Definition

An integer is a perfect square if there is an integer b such that $a=b^2$

Theorem

2) *If m and n are both perfect square then mn is also a perfect square.*

Let m and n be perfect square, then there exist integers s and t such that,

$$m = s^2 \text{ and}$$

$$n = t^2$$

$$mn=s^2t^2$$

$$=(st)^2$$

Hence, R.H.S, is

mn is a perfect square.

Indirect proof

Direct proofs lead from the hypothesis of a theorem to the conclusion. They begin with the premises, continue with a sequence of deductions, and end with the conclusion. We need other methods of proving theorems of the form $\forall x(P(x) \rightarrow Q(x))$

Proofs of theorem of this type that are not direct proofs, i.e., that do not start with the hypothesis and end with the conclusion, are called indirect proofs.

Example of indirect proofs are proof by contraposition, proof by contradiction.

Proof by contraposition

(1) *If n is an integer and $3n+2$ is odd, then n is odd.*

$p \rightarrow q$ (if $3n+2$ is odd then n is odd)

$\neg q \rightarrow \neg p$ (if n is not odd then $(3n+2)$ is even)

let n is even,

$$n=2k$$

$$\text{now, } 3n+2 = 3 \cdot 2k+2$$

$$=6k+2$$

$$=2(3k+1)$$

Thus $(3n+2)$ is even

The negation of conclusion of conditional statement implies that the hypothesis is false, so, the original conditional statement is true.

(2) Prove that if $n=ab$ where a and b are positive integers then $a \leq \sqrt{n}$ or $b \leq \sqrt{n}$

Here,

If $(n=ab)$ then $(a \leq \sqrt{n} \text{ or } b \leq \sqrt{n})$

The contrapositive of $p \rightarrow q$ is $\neg q \rightarrow \neg p$

So,

$\neg q$ means $(a > \sqrt{n} \vee b > \sqrt{n})$

Now, $ab > \sqrt{n}\sqrt{n}$

$ab > n$

$ab \neq n$, which contradicts the original conditional statement

$n=ab$

Proof strategy

When we want to prove a statement of the form $\forall x (p(x) \rightarrow Q(x))$, first we try to prove by direct proof. If a direct proof does not seem to go anywhere, try the same thing with a proof by contraposition.

Definition

A real number is rational if there exist integers p and q with $q \neq 0$ such that $r=p/q$. A real number that is not rational is called irrational.

1) Prove that the sum of two rational numbers is rational.

If r and s are rational numbers then to prove $r+s$ is also rational

Let $r=p/q$ and $s=t/u$ where p, q, t, u are integers such that $q \neq 0$ and $u \neq 0$

$r+s = p/q + t/u = (pu+qt)/qu$

$qu \neq 0$

Hence, $r+s$ is rational (proof succeeded by direct proof)

2) Prove that if n is an integer and n^2 is odd,

Then n is odd.

Here, we first attempt by a direct proof. Suppose n is an integer and n^2 is odd.

$n^2=2k+1$

$n=\pm\sqrt{2k+1}$

The above equation is not useful to show n is odd. Because this attempt to use a direct proof did not bear fruit, we next have a proof by contraposition.

$$\neg q \rightarrow \neg p$$

if n is not odd then n^2 is not odd

if n is even then n^2 is even

$$n=2k$$

$$n^2=(2k)^2=2k(2k) \text{ even}$$

Proof by contradiction

suppose we want to prove that a statement p is true. Suppose that we can find a contradiction q such that $\neg p \rightarrow q$ is true, only when p is false that is p is true.

For $(p \rightarrow q)$ we use $(p \wedge \neg q) \rightarrow F$

1) $\sqrt{2}$ is irrational (prove by proof by contradiction)

Let $\sqrt{2}$ is rational.

Then 2 can be represented as a ratio of integer a and b such that $b \neq 0$. Where a and b have no common factor.

$$\sqrt{2} = a/b$$

$$\sqrt{2} \cdot b = a$$

Squaring both side,

$$2b^2 = a^2$$

a^2 is even number, so a is also even

$$a=2c$$

Again, $2b^2=4c^2$

$$b^2=2c^2$$

b^2 is even, b is even

Both a and b are even, 2 divides both a and b . $\sqrt{2} = a/b$. where a and b have no common factor it means 2 does not divide both a and b . Because $\neg p$ leads to contradiction that 2 divides both a and b . so, $\neg p$ is false. 2 is irrational.

2) If $3n+2$ is odd, then n is odd.

Let us assume that n is even and $3n+2$ is odd.

Then, $n=2k$

$$3n+2=3(2k)+2$$

$$=6k+2$$

$$=2(3k+1)$$

Its even

Hence, our assumption came to wrong.

So, if $3n+2$ is odd then n is odd.

Q) prove if a^2 is even then a is even (by contrapositive)

For $p \rightarrow q$ the contrapositive is $\neg q \rightarrow \neg p$

So, the contrapositive of the given question is

If a is odd then a^2 is odd

$$a=2k+1$$

$a^2=4k^2+4k+1=2(2k^2+2k)+1$ which is odd (see the difference in proof by contrapositive and proof by contradiction)

Hence proved.

If a^2 is even then a is even.

$a=2c$ where c is any integer

$$2b^2=(2c)^2$$

$$b^2=2c^2$$

Vacuous proof

$p \rightarrow q$ is true when p is false. If we can show that p is false then we have a proof, called vacuous proof.

We can quickly prove that a conditional statement $p \rightarrow q$ is true when we know that p is false, because $p \rightarrow q$ must be true when p is false. Consequently, if we can show that p is false, then we have a proof, called a vacuous proof, of the conditional statement $p \rightarrow q$.

Q) Show that the proposition $P(0)$ is true where the domain consists of the integer numbers and $P(n)$ is "If $n \geq 1$ then $n^2 > n$." Note: vacuous proof: when p is false $p \rightarrow q$ is true, regardless of the value of q .

Solution:

Note that $P(0)$ is "If $0 > 1$, then $0^2 > 0$." We can show $P(0)$ using a vacuous proof, because the hypothesis $0 > 1$ is false. Hence $P(0)$ is automatically true.

Trivial proof

A proof of $p \rightarrow q$ that uses the fact that q is true is called trivial proof.

By showing that q is true, it quickly follows that $p \rightarrow q$ must also be true. A proof of $p \rightarrow q$ that uses the fact that q is true is called a trivial proof.

Q) let $P(n)$ is 'If a and b are positive integers with $a \geq b$ then $a^n \geq b^n$ ' where domain consists of all integers. Show that $p(0)$ is true.

The proposition $P(0)$ is "If $a \geq b$, then $a^0 \geq b^0$ " Because $a^0 = b^0 = 1$, the conclusion of the given conditional statement is true.

Thus $p(0)$ is true.

Proof by equivalence

To prove $p \rightarrow q$ show that $p \rightarrow q$ is true and $q \rightarrow p$ is true. The validity of this approach is based on the tautology

$$(p \leftrightarrow q) \leftrightarrow [(p \rightarrow q) \wedge (q \rightarrow p)]$$

Example: Prove that if n is a positive integer, then n is odd if and only if n^2 is odd.

Sometimes, a theorem states that several propositions are equivalent. Such a theorem states that propositions $p_1, p_2, p_3, \dots, p_n$ are equivalent. This can be written as $p_1 \leftrightarrow p_2 \leftrightarrow \dots \leftrightarrow p_n$

which states that all propositions have the same truth values and consequently for all i and j with $1 \leq i \leq n$ and $1 \leq j \leq n$, p_i and p_j are equivalent.

This can be proved as following $\forall (p_1 \leftrightarrow p_2 \leftrightarrow \dots \leftrightarrow p_n) \leftrightarrow (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_3) \wedge \dots \wedge (p_n \rightarrow p_1)$

Q) prove that "If n is an integer, then n is odd iff n^2 is odd".

p : n is odd

q : n^2 is odd

$p \rightarrow q$ [T]

$$n = 2k+1$$

$$n^2 = 4k^2 + 4k + 1$$

$$n^2 = 2(2k^2 + 2k) + 1$$

$q \rightarrow p$

$$n = 2k$$

$$n^2 = (2k)^2 = 4k^2$$

$$n^2 = 2(2k^2)$$

if n is even, then n^2 is even

(proof by contraposition)

Q) show that these statement about integer n are equivalent

p1: n is even

p2: n-1 is odd

p3: n^2 is even

$p1 \rightarrow p2$, $p2 \rightarrow p3$ and $p3 \rightarrow p1$

$n=2k$, $n-1=2k-1=2(k-1)+1$, $n^2=(2k)^2=2(2k^2+4k+2)$

If several proposition are equivalent

$p1 \leftrightarrow p2 \leftrightarrow p3 \leftrightarrow \dots \leftrightarrow pn \leftrightarrow (p1 \rightarrow p2) \wedge (p2 \rightarrow p3) \wedge \dots \wedge (pn \rightarrow p1)$

Proof by cases

Sometimes it is difficult to use a single argument that holds for all cases.

Proof by cases uses the following equivalence:

$[(p1 \vee p2 \vee \dots \vee pn) \rightarrow q] \equiv [(p1 \rightarrow q) \wedge (p2 \rightarrow q) \wedge \dots \wedge (pn \rightarrow q)]$

Example: Prove that if n is integer then $n^2 \geq n$.

We split the proof into three cases.

Case (i) $n = 0$.

In this case, $n^2 = 0^2 = 0 = n$.

Case (ii) $n \geq 1$

In this case, when we multiply both sides of $n \geq 1$ by n we obtain $n \cdot n \geq n \cdot 1$. This simplifies $n^2 \geq n$.

Case (iii) $n \leq -1$.

In this case, $n \leq -1$, but $n^2 \geq 0$. Therefore, $n^2 \geq 0 \geq -1 \geq n$, and so $n^2 \geq n$.

$$[(p_1 \vee p_2 \vee \dots \vee p_n) \rightarrow q] \equiv [(p_1 \rightarrow q) \wedge (p_2 \rightarrow q) \dots \wedge (p_n \rightarrow q)]$$

This shows that the original conditional statement with a hypothesis made up of a disjunction of the propositions $p_1, p_2, \dots, p_n$ can be proved by proving each of the n conditional statement $p_i \rightarrow q$, $i=1, 2, \dots, n$ individually. Such an argument is called a proof by cases.

Exhaustive proof

This is a special form of a proof by cases, when there is a finite and small number of examples for which we need to prove a fact.

Some theorems can be proved by examining a relatively small number of examples. Such proofs are called exhaustive proof, because these proofs proceed by exhausting all possibilities. It is special type of proof by cases.

Q1) Prove that $(n+1)^2 \geq 3^n$ if n is a positive integer with $n \leq 2$.

We use a proof by exhaustion, by examining the cases $n = 1, 2$.

For $n=1, (n+1)^2 = 2^2 = 4 \geq 3^1 = 3$.

For $n=2, (n+1)^2 = 3^2 = 9 \geq 3^2 = 9$.

Q2) Prove that $(n+1)^2 \geq 3^n$ if n is positive integer with $n \leq 4$.

We only need verify the inequality $(n+1)^2 \geq 3^n$

When $n=1, 2, 3$ and 4

For $n=1, (n+1)^2 = 2^2 = 4$ and $3^n = 3^1 = 3$

For $n=2, (n+1)^2 = 3^2 = 9$ and $3^n = 3^2 = 9$

For $n=3, (n+1)^2 = 4^2 = 16$ and $3^n = 3^3 = 27$

For $n=4, (n+1)^2 = 5^2 = 25$ and $3^n = 3^4 = 81$

In each cases, we see that $(n+1)^2 \geq 3^n$

Existence Proofs

Existence proofs prove statements of the form $\exists x P(x)$.

Constructive existence proof: find a such that $P(a)$ is true.

Example: Show that there is a positive integer that can be written as a sum of cubes of positive integers in two different ways.

Proof: $1729=10^3+9^3$ and $1729=12^3+1^3$.

Non-constructive existence proof: show that $\exists x P(x)$ without explicitly giving a for which $P(a)$ is true.

Example: Show that there exist irrational numbers x and y such that xy is rational.

From a previous theorem we know that

$\sqrt{2}$ is irrational.

Consider the $\sqrt{2}^{\sqrt{2}}$ number.

There are two possible cases:

$\sqrt{2}\sqrt{2}$ is rational: In this case, take $x = \sqrt{2}$ and $y = \sqrt{2}$.

$\sqrt{2}\sqrt{2}$ is irrational: In this case, take $x = \sqrt{2}^{\sqrt{2}}$ and $y = \sqrt{2}$. Then

$$x^y = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = (\sqrt{2})^2 = 2.$$

This proof is an example of a non-constructive existence proof because we have not found irrational numbers x and y such that x^y is rational.

Rather, we have shown that either the pair $x=\sqrt{2}$, $y=\sqrt{2}$ or the pair $x = \sqrt{2}^{\sqrt{2}}$, $y = \sqrt{2}$ have the desired property, but we do not know which of these two pairs works!

Mistakes in Proofs

What is the problem with the following proof that $1 = 2$?

Use the following steps, where a and b are two equal positive integers.

1. $a = b$ Given
2. $a^2 = ab$ Multiply both sides of (1) by a
3. $a^2 - b^2 = ab - b^2$ Subtract b^2 from both sides of (2)
4. $(a-b)(a+b) = b(a-b)$ Factoring both sides of (3)
5. $a+b=b$ Divide both sides of (4) by $a-b$
6. $2b=b$ Replace a by b in (5), since $a=b$
7. $2=1$ Divide both sides of (6) by b Therefore $2 = 1$.

Solution: Step 5 where we divided both sides by $a-b$ is invalid. The error is that $a - b$ equals zero; (Divide by Zero)

Mathematical Induction

Principle of mathematical induction

To prove that $p(n)$ is true for all positive integer n where $p(n)$ is a propositional function then we do two steps

Basic step: we verify that $p(0)$ or $p(1)$ is true

Inductive hypothesis : we assume that $p(k)$ is true

Inductive step : we prove $p(k+1)$ is true

we show $p(k) \rightarrow p(k+1)$ is true for all positive integer k , assuming $p(k)$ is true (Inductive hypothesis)

As rule of inference, this can be stated as,

$$[p(1) \wedge \forall k(p(k) \rightarrow p(k+1))] \rightarrow \forall n p(n)$$

Q) show that if n is a positive integer then

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

Let $p(n)$ be the proposition, that sum of the 1st n positive integers is $n(n+1)/2$

Basic step: $p(1)$ is true because

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

Inductive step :

Assume $p(k)$ is true for an arbitrary positive integer k , i.e.

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

Now, we have to show $p(k+1)$ is true, i.e.

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

$$\text{L.H.S.}=1+2+3+\dots+k+(k+1)$$

$$=k(k+1)/2+(k+1)\text{inductive hypothesis)}$$

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

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

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

From the principle of mathematical induction, $p(n)$ is true for all positive integers n .

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

The sum of the first positive odd integers for $n=1,2,3,4,5$ are

$$1=1$$

$$1+3=4$$

$$1+3+5=9$$

$$1+3+5+7=16$$

$$1+3+5+7+9=25$$

Hence the sum of the first n positive odd integers is n^2 i.e.

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

Let $p(n)$ denote the proposition that the sum of the first n positive odd integers is n^2

Basis step : $p(1)$ is true a

$$1=1^2 \text{ (is true)}$$

Inductive step : let $p(k)$ is true, i.e. $1+3+5+\dots+(2k-1)=k^2$ is true.

Now, $p(k+1)$ is the statement

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

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

Here,

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

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

k^2 (inductive hypothesis)

$P(k) \rightarrow P(k+1)$ is true, $P(n)$ is true for all positive integers n

3) *show that*

$1+2+2^2+\dots+2^n=2^{n+1}-1$ for all non-negative integers.

Let $p(n)$ be the proposition

$$1+2+2^2+\dots+2^n=2^{n+1}-1$$

Basis step : $p(0)$ is true as

$$\text{L.H.S.} = 2^0=1 \text{ and } \text{R.H.S.} = 2^{0+1}-1=2-1=1$$

$$\text{L.H.S.} = \text{R.H.S.}$$

Inductive step :

Let $1+2+2^2+\dots+2^k=2^{k+1}-1$ holds for k

Then, $p(k+1)$ is

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

Now,

$$1+2+2^2+\dots+2^k+2^{k+1}=(2^{k+1}-1)+2^{k+1} \text{ (using inductive hypothesis)}$$

$$=2 \cdot 2^{k+1}-1=2^{k+2}-1$$

$P(k) \rightarrow P(k+1)$ is true

$P(n)$ holds for all n (all non-negative integer)

4) $\sum_{i=0}^n ar^i = ar^n = a + ar + ar^2 + \dots + ar^n = \frac{ar^{n+1}-a}{r-1}$ where $r \neq 1$

Basis step : $p(0)$ is true as

$$\frac{ar^{0+1}-a}{r-1} = \frac{ar-a}{r-1} = \frac{a(r-1)}{r-1} = a$$

Inductive step :

let $p(k)$ holds for k , where k is non negative integer. i.e.

$$a+ar+ar^2+\dots+ar^k = \frac{ar^{k+1}-a}{r-1} \text{ is true}$$

Now $p(k+1)$ is the statement

$$a+ar+ar^2+\dots+ar^k+ar^{k+1} = \frac{ar^{(k+1)+1}-a}{r-1} = \frac{ar^{(k+2)}-a}{r-1}$$

$$\text{L.H.S.} = a+ar+ar^2+\dots+ar^k+ar^{k+1}$$

$$= \frac{ar^{(k+1)}-a}{r-1} + ar^{k+1}$$

$$= \frac{ar^{(k+1)}-a+ar^{(k+2)}-ar^{(k+1)}}{r-1}$$

$$= \frac{ar^{(k+2)}-a}{r-1}$$

$p(k) \rightarrow p(k+1)$ is true, $p(n)$ is true for all non-negative integers.

5) *Prove $n < 2^n$ for all positive integers n .*

Let $p(n)$ be the proposition that $n < 2^n$

Basis step : $p(1)$ is true as $1 < 2^1=2$ is true

Inductive step : assume for positive integer k , $p(k)$ is true, i.e. $k < 2^k$

$p(k+1)$ is the statement $(k+1) < 2^{k+1}$

we have, $k < 2^k$

$k+1 < 2^k + 1$ (adding both side 1)

note that $1 < 2^k$

$k+1 < 2^k + 1$

$< 2^k + 2^k$

$< 2 \cdot 2^k$

$< 2^{k+1}$

$k+1 < 2^{k+1}$

Since $p(k) \rightarrow p(k+1)$ is true, $p(n)$ is true for all positive integer n .

6) *Prove that $2^n < n!$ for every positive integer n with $n > 4$*

Let $p(n)$ be the proposition $2^n < n!$

Basis step : $p(4)$ is true as $2^4 = 16 < 4! = 24$

Inductive step : let $p(k)$ be true for positive integer k with $k \geq 4$ i.e. $2^k < k!$

$p(k+1)$ is the statement $2^{k+1} < (k+1)!$

we have, $2^{k+1} = 2^k \cdot 2 = 2 \cdot 2^k < 2 \cdot k!$ (inductive hypothesis)

$< (k+1)k! \quad (2 < (k+1))$

$< (k+1)! \text{ Factorial definition.}$

Hence, $2^{k+1} < (k+1)!$

$P(k) \rightarrow p(k+1)$ is true, $p(n)$ holds for all $n \geq 4$

7) *An inequality for harmonic numbers. The harmonic numbers H_j , $j=1,2,3,\dots$ are defined by $H_j = 1 + 1/2 + 1/3 + \dots + 1/j$*

Show that $H_2^n \geq 1 + n/2$ where n is a non-negative integer.

Let $p(n)$ be the proposition that $H_2^n \geq 1 + n/2$

$H_1 \geq 1$

$1 \geq 1$

Inductive step : assume $p(k)$ is true $H_2^k \geq 1 + k/2$ for a non-negative integer k .

Then $p(k+1)$ is the statement $H_2^{k+1} \geq 1 + (k+1)/2$

Now, from definition of harmonic number

$H_2^{k+1} = 1 + 1/2 + 1/3 + \dots + 1/2^k$

8) *Prove $n^3 - n$ is divisible by 3 wherever n is a positive integer.*

Let $p(n)$ denotes $n^3 - n$ is divisible by 3

Basis step : $p(1)$ is true as $1^3 - 1 = 0$ is divisible by 3.

Inductive step : let $p(k)$ is true, i.e. $k^3 - k$ is divisible by 3 for a positive integer k .

Then, $p(k+1)$ is the statement $(k+1)^3 - (k+1)$ is divisible by 3.

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

$(k^3 - k)$ is divisible by 3 (from hypothesis) and $3(k^2 + k)$ is also divisible by 3 and so their sum is also divisible by 3.

$P(k) \rightarrow p(k+1)$ is true $p(n)$ is true for a positive integer n .

9) If S is a finite set with n element where n is a non-negative integer, then S has 2^n subsets.

Let $p(n)$ be the proposition that a set with n elements has 2^n subsets.

Basis steps : $p(0)$ is true, because $2^0 = 1$ subsets which agrees with the fact that a set with zero element i.e. empty set has only one subset namely itself.

Inductive step : we assume for non-negative integer k that every set with k element has 2^k subsets. $p(k+1)$ is the statement that every set with $k+1$ elements has 2^{k+1} subsets.

Let T be a set with $k+1$ elements, then we can write $T = S \cup \{a\}$ where a is one of the element of T and $S = T - \{a\}$

$(|S|=k)$ subsets of T can be obtained in the following way

For each subsets of there are exactly two subsets of T , namely, X and $X \cup \{a\}$, these constitute all the subsets of T and are all distinct.

But there are 2^k subsets of S , so there are $2 \cdot 2^k = 2^{k+1}$ subsets of T .

$$S = \{a, b\} \quad T = \{a, b, c\}$$

$$X = [\{\}, \{a\}, \{b\}, \{a, b\}] \quad T = \{a, b, c\}$$

$$\text{subset of } T = [\{\}, \{a\}, \{b\}, \{a, b\}, \{\}\cup\{a\}, \{a\}\cup\{c\}, \{b\}\cup\{c\}, \{a, b\}\cup\{c\}]$$

Summarization of proof techniques

S. N.	Proof Technique	Approach to prove $p \rightarrow q$	Remarks
1.	Exhaustive proof	Demonstrate $p \rightarrow q$ for all possibilities	May only be used for finite number of cases
2.	Direct proof	Assume p , deduce q	The standard approach usually the thing to try.
3.	Proof by contraposition	Assume $\neg q$, derive $\neg p$	Use this q if as a hypothesis seems to give more ammunition then q would.

4.	Proof by contradiction	Assume $p \wedge \neg q$, deduce a contradiction	Use this when q says something is not true
5.	Vacuous proof	Show p is false	Give an example to make p is false, in $p \rightarrow q$.
6.	Trivial proof	Show q is true	Give an example to make q is true.
7.	Proof by cases	Demonstrate $p \rightarrow q$ for all cases	May only be used for known cases.
8.	Proof by equivalence	To prove $p \leftrightarrow q$ show $(p \rightarrow q) \wedge (q \rightarrow p)$	$(p_1 \leftrightarrow p_2 \leftrightarrow p_3 \dots \leftrightarrow p_n) \leftrightarrow (p_1 \rightarrow p_2) \wedge (p_2 \rightarrow p_3) \dots \wedge (p_n \rightarrow p_1)$

Valid Formula

Valid formulae are those formulae which are valid under all valuations ($p \vee \neg p$) is valid formula. Valid propositional formulae are called tautologies.

Satisfiable Formula

These can be made true under at least one valuation, though not necessarily all valuations. An example of satisfiable formula is $p \vee q$ which is true at $p=T$ and $q=F$. Every valid formula is satisfiable.

Unsatisfiable Formula

The unsatisfiable formulas come out false under all valuations.

Soundness and Completeness of the system

The axioms are all valid and each inference rule has the following properties : If both premises are valid then the conclusion is also valid. It follows this, that, only valid formulae can be proved in a sound logical system.

If $\emptyset$ can be proved in a sound logical system then $\emptyset$ is valid. If there is a proof of $\emptyset$ then $\emptyset$ is called a theorem.

The soundness of a logical system is expressed as

If $\emptyset$ is a theorem (if $\emptyset$ can be proved), then $\emptyset$ is valid.

The completeness of the system is expressed as

If $\emptyset$ is valid then $\emptyset$ is a theorem ($\emptyset$ can be proved)

Why tableau?

The main problem with the axiomatic system we have described is that although it is relatively easy to check that a proof of a formula is correct, it is much harder to construct a proof. We need intuition to know which axioms to pick and when to apply modus ponens. It is even worse, if you do not know whether a formula can be proved or not.

It is the method to prove the satisfiability of a set of formula.

Two reasons for using tableau

It is quite easy to use (though some may argue, that natural deduction is easier) and

It is very suitable for implementation on a computer.

It breaks each formula into component upto simplest one, where contradiction is easy to spot.

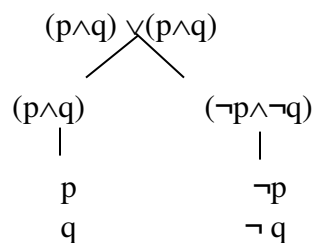
It creates a tree structure called tableau.

Parsing

When we parse a formula, we look at the string of symbols and work out how it was built up from the propositions. In other words, we recover the structure of the formula. The term comes from linguistics where the act of parsing is to determine the grammatical structure of a sentence of natural language.

Example

Let ϕ be the formula $(p \wedge q) \vee (\neg p \wedge \neg q)$ then we can parse as



Tableaux method

Note:

- 1) If only set of formula is given (i.e. conclusion is not given), if any branch is open, then the set is satisfiable sometimes called soundness.
- 2) If the argument (i.e. premises and conclusion) are given, we negate the conclusion. If the all branches of tree (made up of premises and negation of conclusion) then it is valid called completeness.

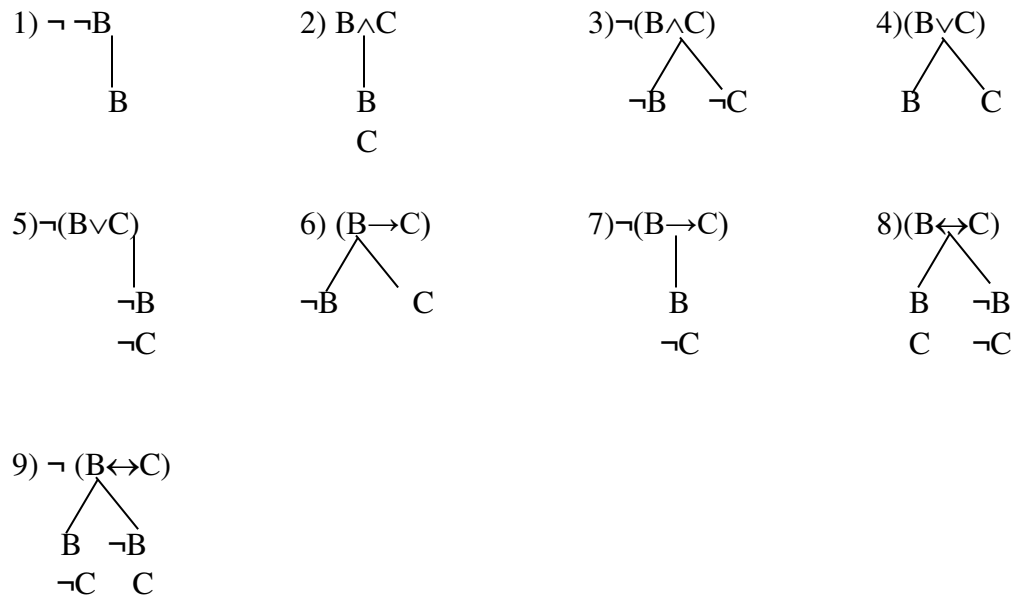
Tableaux construction rules

If a formula ϕ is not a literal, it has one of the following forms.

- 1) $\phi = \neg \neg B$

- 2) $\emptyset = B \wedge C$
- 3) $\emptyset = \neg(B \wedge C)$
- 4) $\emptyset = (B \vee C)$
- 5) $\emptyset = \neg(B \vee C)$
- 6) $\emptyset = (B \rightarrow C)$
- 7) $\emptyset = \neg(B \rightarrow C)$
- 8) $\emptyset = (B \leftrightarrow C)$
- 9) $\emptyset = \neg(B \leftrightarrow C)$

Derivation Rules in Tableaux Form



$$\begin{aligned}
 (B \leftrightarrow C) &\equiv (B \rightarrow C) \wedge (C \rightarrow B) \\
 &\equiv (\neg B \vee C) \wedge (\neg C \vee B) \\
 &\equiv (\neg B \wedge \neg C) \vee (\neg B \wedge B) \vee (C \wedge \neg C) \vee (C \wedge B) \\
 &\equiv (\neg B \wedge \neg C) \vee (C \wedge B) \\
 &\equiv (B \wedge C) \vee (\neg B \wedge \neg C)
 \end{aligned}$$

$$\begin{aligned}
 \neg(B \leftrightarrow C) &\equiv \neg [(B \wedge C) \vee (\neg B \wedge \neg C)] \\
 &\equiv \neg (B \wedge C) \wedge \neg (\neg B \wedge \neg C) \\
 &\equiv (\neg B \vee \neg C) \wedge (B \vee C)
 \end{aligned}$$

$$\begin{aligned} &\equiv (\neg B \wedge B) \vee (\neg B \wedge C) \vee (\neg C \wedge B) \vee (\neg C \wedge C) \\ &\equiv (B \wedge \neg C) \vee (\neg B \wedge C) \end{aligned}$$

Steps in Tableaux method

Given a finite set of formulas $\phi_1, \phi_2, \dots, \phi_m$, a tableau for $\phi_1, \phi_2, \dots, \phi_m$ is a tree of formulas constructed as follows:

- 1) First of all, list the formulas

ϕ_1

ϕ_2

ϕ_3

.

.

ϕ_m

- 2) Starting from any formula ϕ_i , apply the derivation rule, say $\phi_i = (B \rightarrow C)$

ϕ_1

ϕ_i

.

.

ϕ_j

.

ϕ_m

B	$\neg B$
C	$\neg C$

- 3) Now, along each branch of the tree constructed hitherto, we repeat the previous steps, say $\phi_j = (\neg C \wedge D)$

ϕ_1

ϕ_i

.

ϕ_j

.

.

ϕ_m

B	$\neg B$
C	$\neg C$
$\neg C$	$\neg C$
D	D

- 4) After each step, we check for satisfiability along branches. Thus, the left- most branch contains a formula and its negation, namely, C and $\neg C$.

There are, the left-most branch is not satisfiable. So, we can close-off this branch and we will not analyze this further.(indicated by bar)

Note that we will not apply further derivations rules to it.

We apply the previous steps along each branch that is not yet closed off until either

- i) Every branch is closed or
- ii) All non-literals along each unclosed branch is checked.

Then in case (i), the initial set $\phi_1, \phi_2, \dots, \phi_m$ is not satisfiable. On the other hand, if (i) fails that is some branch is not closed off but (ii) holds, then the set of formulas $\phi_1, \phi_2, \dots, \phi_m$ is satisfiable.

- 5) From open branch we can check for validity by valuation. For all literals p we define valuation V as

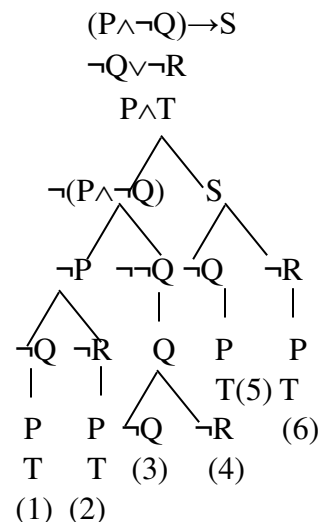
$V(p)=T$ if p appears along the branch

$=F$ if p appears along the branch

$=T$ or F if both p and $\neg p$ not appears in branch

Q) Consider the set $\phi = \{(P \wedge \neg Q) \rightarrow S, \neg Q \vee \neg R, P \wedge T\}$

A tableaux for ϕ



For open branch (4)

$V(T)=T$

$V(P)=T$

$V(R)=F$

$V(S)=T$ or F

$V(Q)=T$

Then

$V((P \wedge \neg Q) \rightarrow S)=T$

$V(\neg Q \vee \neg R)=T$

$$V(P \wedge T) = T$$

Soundness of propositional tableau

When we say that the tableau method is sound we mean that if you can construct a closed tableau for $\neg \phi$ then ϕ must certainly be a valid formula. This means that anything we prove by the tableau method is certainly valid. Soundness is usually the easy part to check for most inference systems.

Completeness

When we say that the tableau method is complete we mean that if ϕ is any valid formula then you can construct a closed tableau for $\neg \phi$. This means that any valid formula can be proved using the tableau method showing completeness is usually the hard part to check for most inference systems.

Corollary:

If a completed tableau for a formula ϕ is open, then ϕ is satisfiable. If a completed tableau for ϕ is closed, then ϕ is unsatisfiable.

By following corollary and the two theorems we know that the tableau method is both sound and complete.

Definition:

An argument

A1

A2

.

.

.

An

B

is said to be valid if the conclusion B is a tautological consequence of the premises A1, A2, ..., An

Now, $\{A1, A2, \dots, An\} \models B$

Q) $P \rightarrow (Q \wedge R)$

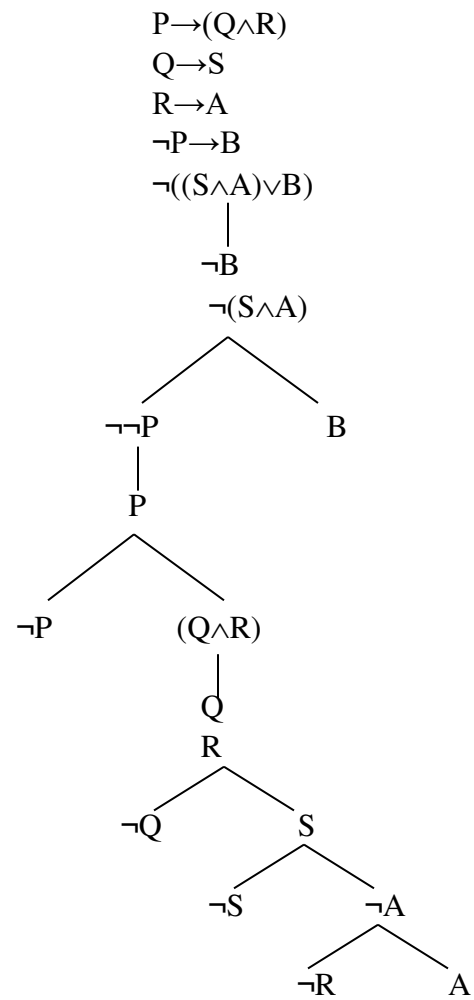
$Q \rightarrow S$

$R \rightarrow A$

$P \rightarrow B$

$(S \wedge A) \vee B$

This argument is valid iff for any valuation 'V' satisfying the hypothesis, V also satisfies the conclusion. Or in other words if the hypothesis plus the negation of the conclusion is not satisfiable. So, we check for satisfiability.



Every branch closes hence the argument is valid.

Chapter 2. Finite State Automata [10 hrs.][16 Marks]

2. Finite State Automata

2.1. Sequential Circuits and Finite State Machine

2.2. Finite state automata

2.3. Language and grammars

2.4. Non-deterministic finite state automata

2.5. Language and automata

2.6. Regular expression and its characteristics

Combinational circuit.

combinational circuits have no memory or in term states, their output depends only on the current values of their inputs.

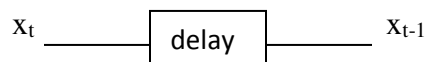
Sequential circuits

It contains memory element or internal states.

The output depends not only on its current inputs but also on the past history of those inputs.

Finite state machines have internal states, so their output may depend not only on its current inputs but also on the past state or past history of those inputs.

A unit time delay accepts as input a bit x_t at time t and outputs x_{t-1} , the bit received at time $t-1$.



A serial adder accepts as input two binary number

$x = 0x_n x_{n-1} \dots x_0$ and

$y = 0y_n y_{n-1} \dots y_0$ and

Outputs the sum $z_{n+1} z_n \dots z_0$ of x and y .

The numbers x and y are input sequentially in pairs $x_0, y_0, \dots, x_n, y_n$,

The sum output is $z_0, z_1, \dots, z_{n+1}$

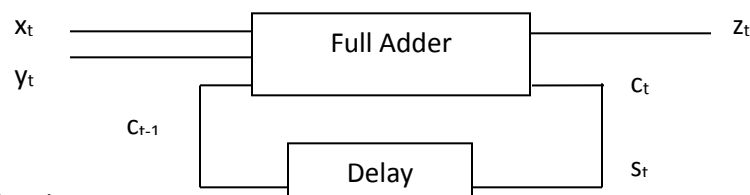
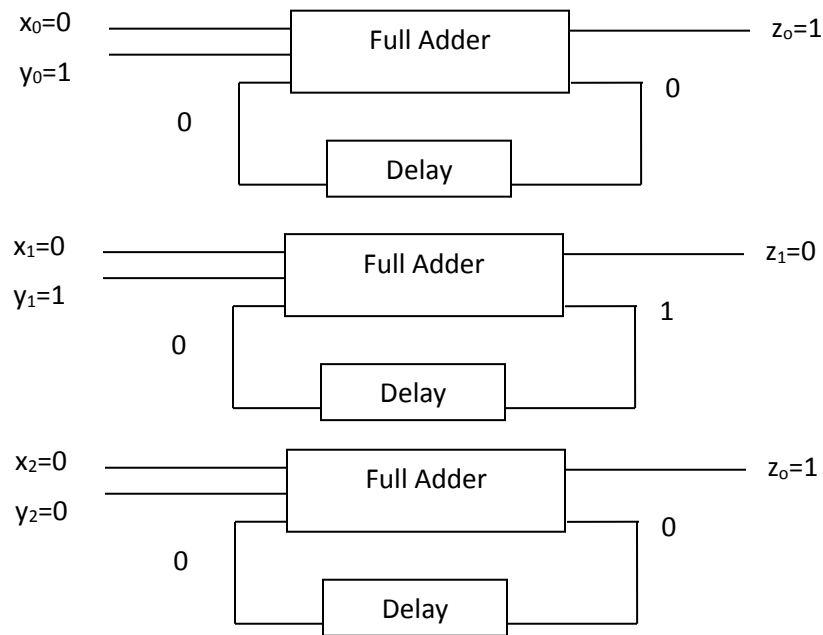


Fig: Serial Adder Circuit

$X=010$ and $y=011$

Let $i=0$ before start (this can be done by first setting $x=y=0$)



Finite State Machine

A finite state machine is an abstract model of a machine with a primitive internal memory.

Definition

A finite state machine M consists of

- A finite set I of input symbols.
- A finite set O of output symbols.
- A finite set S of states.
- A next state function f from $S \times I$ into S .
- An output function g from $S \times I$ into O .
- An initial state

We write

$$M = (I, O, S, f, g, \sigma_0)$$

Example

Let $I = \{a, b\}$, $O = \{0, 1\}$ and $S = \{\sigma_0, \sigma_1\}$

Define the pair of functions

$f: S \times I \rightarrow S$ and $g: S \times I \rightarrow O$ by the rules given below.

	f		g	
S/I	a	b	a	b
σ_0	σ_0	σ_1	0	1
σ_1	σ_1	σ_1	1	0

Then $M=(I, O, S, f, g, \sigma_0)$ is a FSM

$f(\sigma_0, a)=\sigma_0$ $g(\sigma_0, a)=0$

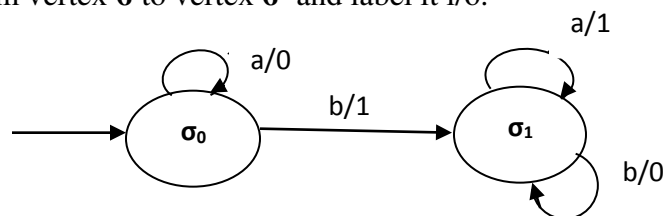
$f(\sigma_0, a)=\sigma_1$ $g(\sigma_0, b)=1$

$f(\sigma_1, a)=\sigma_1$ $g(\sigma_1, a)=1$

$f(\sigma_1, b)=\sigma_1$ $g(\sigma_1, b)=0$

Transition diagram

The transition diagram is a digraph. The vertices are the states. The initial state is indicated by an arrow. If we are in state σ and inputting I causes output O and moves us to state σ' , we draw a directed edge from vertex σ to vertex σ' and label it i/o .



Let $M=\{I,O,S, f, g, \sigma\}$ be a FSM. The transition diagram of M is a digraph G whose vertices are the members S . an arrow designates the initial state. A directed edge (σ_1, σ_2) exist in G . If there exists an input I with $f(\sigma_1,i)=\sigma_2$. In this case, if $g(\sigma_1,i)=0$. The edge (σ_1, σ_2) is labeled i/o .

Definition

Let $M=(I, O, S, f, g, \sigma)$ be a FSM. An I/P string for M is a string over I .

The string

$y_1 \dots\dots\dots y_n$

is the o/p string for M corresponding to the i/p string

$x_1 \dots\dots\dots x_n$

If there exists states $\sigma_0, \dots\dots\dots \sigma_n \in S$

With $\sigma_0 = \sigma$

$\sigma_i = f(\sigma_{i-1}, x_i)$ for $i=1, \dots\dots, n$

$y_i = g(\sigma_{i-1}, x_i)$ for $i=1, \dots\dots, n$

Example

1) Design a FSM that performs serial addition serial adder accepts pair of bits The i/p set will be {00,01,10,11}

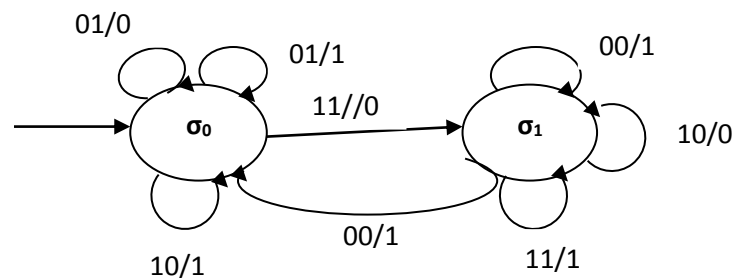
The o/p set is {0,1}

Given an i/p x, y

- i) either add x and y or
- ii) add x, y and 1

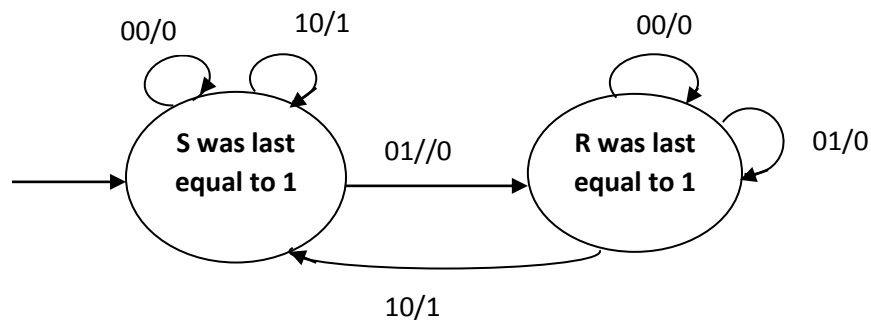
depending on whether the carry bit was 0 or 1. So, we have two states c (carry) and NC (no carry). The initial state is NC.

Transition diagram:



The SR flip flop

S	R	Q
1	1	Not allowed
1	0	1
0	1	0
0	0	1 if s was last equal to 1 0 if R was last equal to 1



Finite State Automata

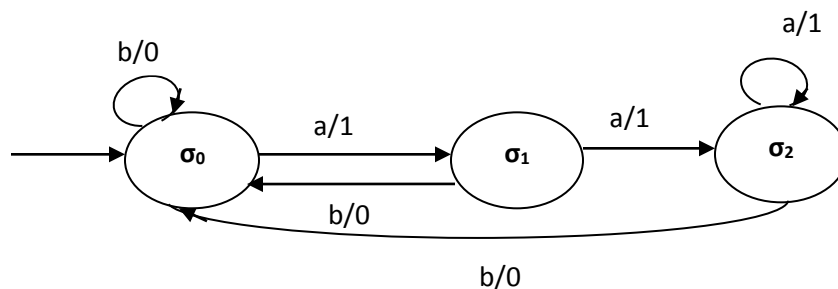
Finite state automata is a special kind of FSM. It has no output and no output function but has accepting or final state.

Definition:

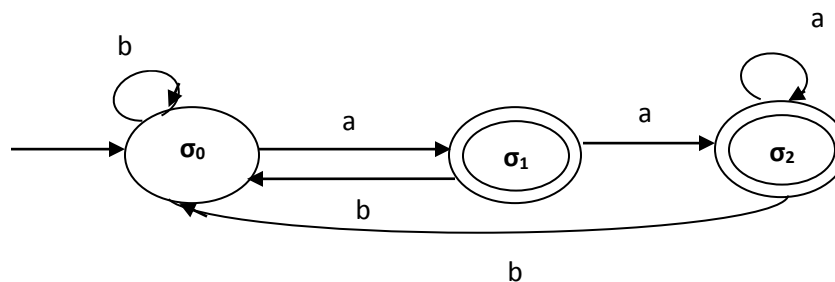
A FSA $A = \{I, O, s, f, g, \sigma\}$ is a FSM in which the set of o/p symbols is $\{0,1\}$ and where the current state determines the last output. Those states for which the last output was 1 are called accepting states.

E.g. The initial state is σ_0

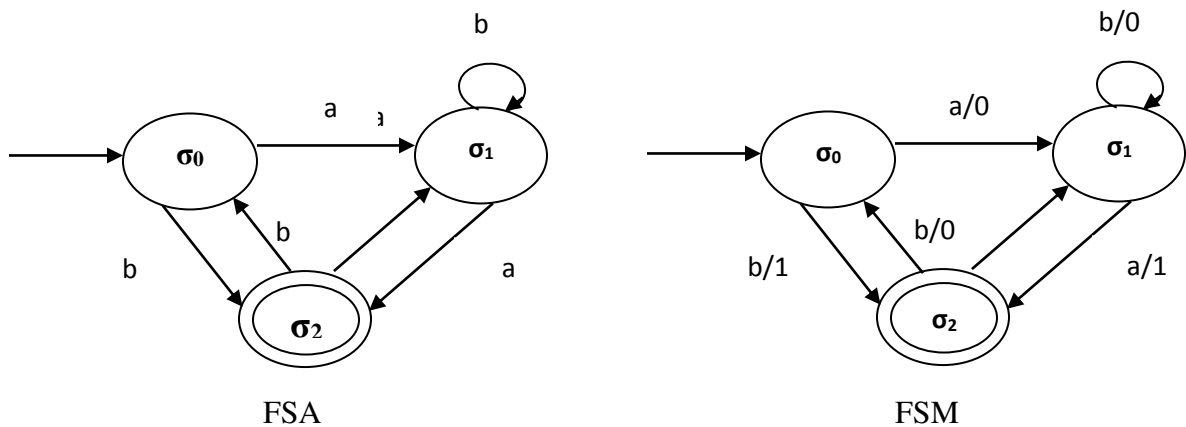
s/I	f		g	
	a	b	a	b
σ_0	σ_1	σ_0	1	0
σ_1	σ_2	σ_0	1	0
σ_2	σ_2	σ_0	1	0



If we are in state σ_0 , the last output was 0. If we are in either state σ_1 or σ_2 , the last o/p was 1. Thus A is FSA. The accepting states are σ_1 and σ_2 .



Q) Draw the transition diagram of the FSA as a transition diagram of a FSM



σ_2 is an accepting states

Its incoming edges with a and b.

The other states σ_0 and σ_1 are not accepting state so that they have incoming along with only a.

Finite State Automata

A finite state automata is similar to finite state machine but with no output, and with a set of states called accepting or final states. FSA can be regarded as.

- 1) A finite set I of input symbols.
- 2) A finite set S of input symbols.
- 3) A next state function f from SXI into S.
- 4) A subset A of S of accepting states
- 5) An initial state $\sigma \in S$

$A = (I, S, f, A, \sigma)$

Example

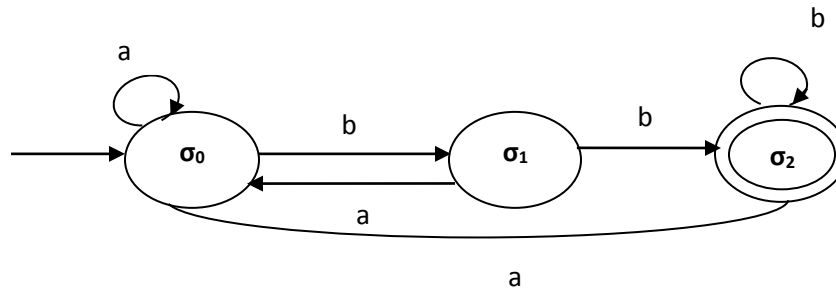
Transition diagram of FSA

$A = (I, S, f, A, \sigma)$ where

$I = \{a, b\}$, $S = \{\sigma_0, \sigma_1, \sigma_2\}$, $A = \{\sigma_2\}$

$\sigma = \sigma_0$ and f is given by table below

	f	
s/I	a	b
σ_0	σ_0	σ_1
σ_1	σ_0	σ_2
σ_2	σ_0	σ_2



Incoming is i/p to a FSA, we will end at either an accepting or a non accepting state. The states of this final state determines whether the string is accepted by FSA.

Definition

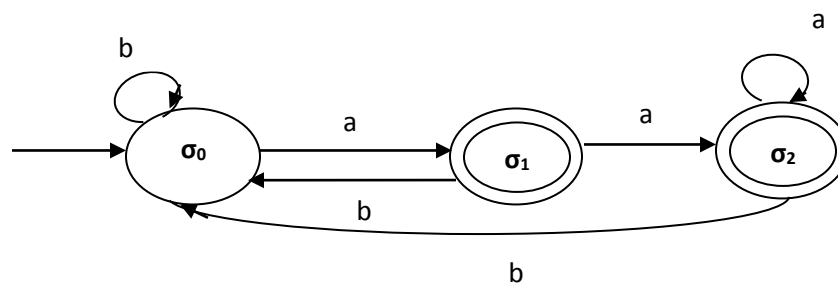
Let $A(I, S, f, A, \sigma)$ be a FSA. Let $\alpha = x_1, \dots, x_n$ be a string over I . If there exists states $\sigma_0, \dots, \sigma_n$ satisfying.

- a) $\sigma_0 = \sigma$
- b) $f(\sigma_{i-1}, x_i) = \sigma_i$ for $i=1, \dots, n$
- c) $\sigma_n \in A$

we say that α is accepted by A . the null string is accepted if $f \sigma \in A$

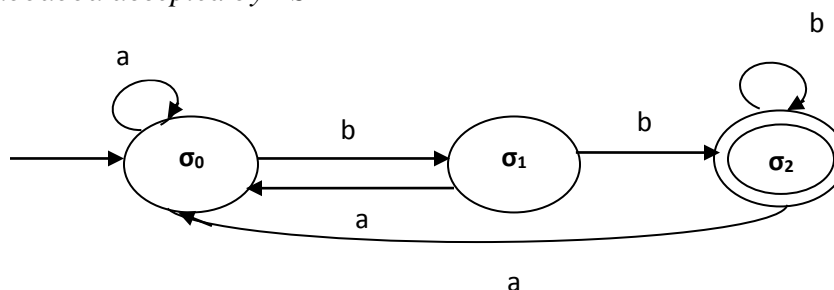
let $Ac(A)$ denoted the set of string accepted by A and we say that A accepted $Ac(A)$.

Is the string abaa accepted by FSA



yes

Is the string abbabba accepted by FSA



No

Q) Design FSA that accepts precisely those strings over $\{a, b\}$ that contain an odd number of a's.

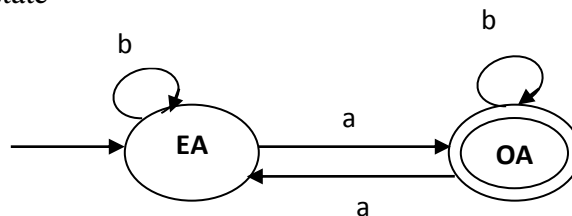
Two states

Odd A = (OA)

Even A = (EA)

OA is accepted state

EA is the initial state



If two FSA accept precisely the same things. We say that the automata are equivalent. FSA A and B are equivalent if $Ac(A) = Ac(B)$

Algorithm

This algorithm determines whether a string over $\{a, b\}$ is accepted by FSA.

Input : n, the length of the string (n=0 designation the null string); $s_1 s_2 \dots s_n$, the string.

Output : 'Accept' if the string is accepted

'Reject' if the string is not accepted procedure fsa (s, n)

State : 'E'

For i=1 to n do

begin

If state = 'E' and $S_i = 'a'$ then

State = 'O'

If state = 'O' and $S_i = 'a'$ then

```

state = 'E'

End

If state = '0' then

Return ("Accept")

Else

Return ("Reject")

End fsa

```

Language and Grammar

Alphabet or vocabulary

Alphabet or vocabulary is a finite non empty set of systems.

It is denoted by Σ or A or v

$V = \{0,1\}$ is a binary alphabet

$V = \{a, b, \dots, z\}$ is a set of lower case letter.

String(w)

A finite sequence of symbols taken from some alphabet is called string.

Word or sentence is string of finite length of element of symbols.

0100 is string from alphabet $v = \{0,1\}$

Concatenation (vw): vw is string v, then string w.

W^n : string w repeated n times

V^* : all possible string using v.

Empty string (λ or ϵ or e)

Zero occurrence of symbols is null or empty string.

Length of string

The number of symbols present in the string.

$$W=0100$$

$$|w|=4$$

$$|\epsilon|=0$$

Language

Language is a body of words and method of combining words used to understand by a considerable community. Such languages are often called natural to distinguish them from formal languages, which are used to model natural languages and to communicate with computers.

The rules of a natural language are very complex and difficult to characterize completely.

It is possible to specify completely the rules by which certain formal languages are constructed.

Definition

The set of all words or string including zero or v is denoted by V^*

Language is set of strings all of which are taken from V^*

A formal language over v is subset of V^*

$$L \subseteq V^*$$

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

$$V^0 = \{ \}$$

$$V^1 = \{0,1\}$$

$$V^2 = VUV$$

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

$$= \{00,01,10,11\}$$

$$V^* = \bigcup_{i=0}^{\infty} V^i$$

$$V^+ = \bigcup_{i=1}^{\infty} V^i$$

V^+ consist of all words formed by concatenating a finite number but not null.

Grammar

A way to determine the structure of a language is with a grammar. In order to define a grammar, we need two kinds of symbols: non terminal, used to represent given subsets of the language, and terminal, the final systems that occur in the strings of the language.

A grammar defining formal language L is a quadruple

$$G = (N, T, R, S)$$

N = set of non terminal

T = set of terminals

R = set of production

S = start symbol

T is alphabet

A grammar is a way to specify the set of all legal sentences of a language.

We write $G = (N, T, P, \sigma)$. A production $(A, B \in P)$ is usually written $A \rightarrow B$

A must include at least one non terminal symbol whereas

B can consists of any combination of non terminal and terminal symbols.

Let,

$$N = \{ \sigma, S \}$$

$$T = \{ a, b \}$$

$$P = \{ \sigma \rightarrow b\sigma, \sigma \rightarrow as, s \rightarrow bs, s \rightarrow b \}$$

Then $G = (N, T, P, \sigma)$ is a grammar.

The language generated by G, written $L(G)$ consists of all strings over T derivable from σ

For above grammar

$$\sigma = b\sigma = bb\sigma = bbas = bbab$$

The only derivations from σ are

$$\sigma = b\sigma \dots = b^n\sigma = b^nas \quad (n \geq 0)$$

$$= b^nab^m \quad (n \geq 0, m \geq 1)$$

Thus, $L(G)$ consists of the strings over $\{a, b\}$ containing precisely one a that end with b.

Types of grammar (Chomsky hierarchy of grammar)

1) Unrestricted grammar (Type 0)

If there is no restriction in the every production of the grammar , it is called unrestricted grammar.

Production rule

$$\alpha \rightarrow \beta$$

Where,

$$\alpha \in (N \cup T)^*$$

$$\beta \in (N \cup T)^*$$

It is recognized by Turing machine

2) Context Sensitive Grammar (Type 1)

If every production is of the form

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

(we may replace A with γ in the context of α and β)

$\alpha, \beta \in (NUT)^*$

$A \in N$

The language generated by this grammar is recognized by Linearly Bounded Automata.

3) Context Free Grammar (Type 2)

If the production is of the form

$A \rightarrow \gamma$

Where $A \in N$

$\gamma \in (NUT)^*$

Left side = single non terminal

Right side = a word in one or more symbols.

$N = \{S, A, B\}$

$T = \{a, b\}$

$P = \{S \rightarrow aA, A \rightarrow aAB, B \rightarrow b, A \rightarrow a\}$

Start symbol S.

The language generated by this grammar is recognized by Push down Automata.

4) Regular Grammar (Type 3)

If every production is of the form

$A \rightarrow a$ or $A \rightarrow aB$ or $A \rightarrow \gamma$

Where, $A, B \in N$

$a \in T$

Left side = single non terminal

Right side = either single terminal or terminal followed by a non terminal.

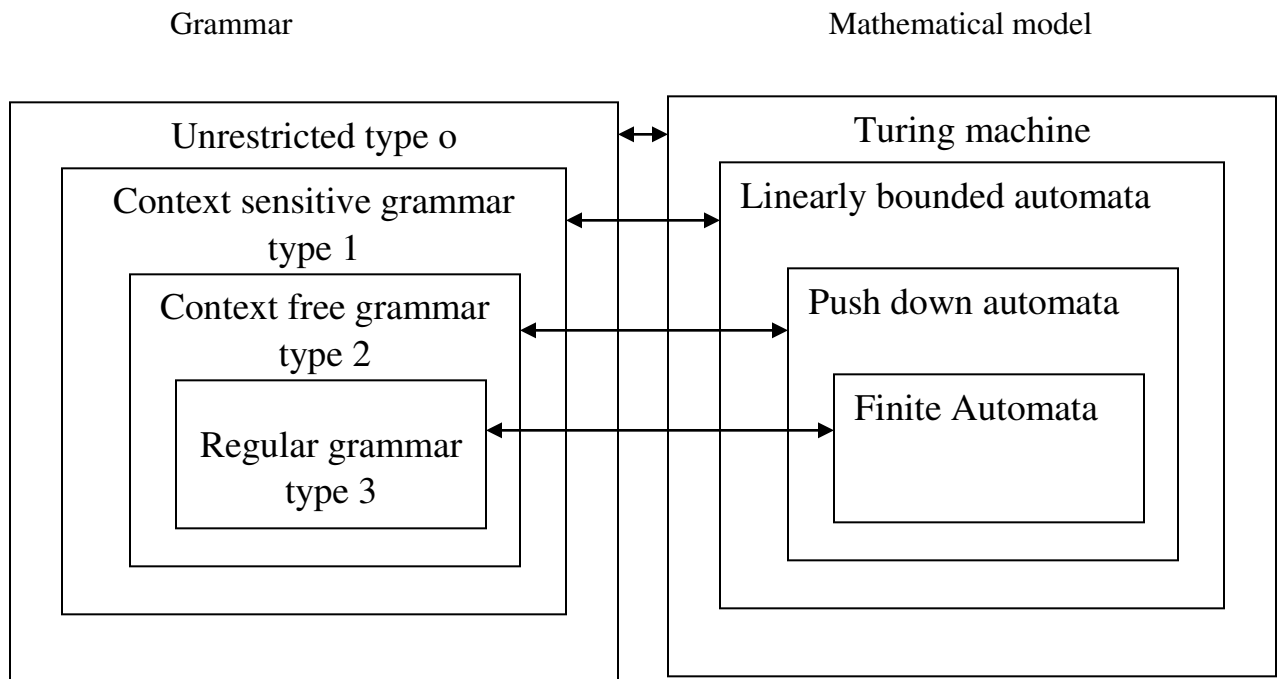
Recognized by Finite State Automata.

$N = \{S, A\}$

$T = \{a, b\}$

$P = \{S \rightarrow bS, S \rightarrow aA, A \rightarrow aS, A \rightarrow bA, A \rightarrow a, S \rightarrow b\}$

Starting symbol S



BNF

An alternative way to state or represent the productions context free grammar is using

- 1) Backus naur form and
- 2) Derivation tree

In BNF the non terminal symbols typically begin with “<” and end with “>” the production $S \rightarrow T$ is written $S::=T$

Production of the form $S::=T_1$, $S::=T_2$, ..., $S::=T_n$ may be combined as $S::=T_1|T_2|.....T_n$

Grammar for integers

$\langle \text{digit} \rangle ::= 0|1|2|3|4|5|6|7|8|9$

$\langle \text{integer} \rangle ::= \langle \text{signed integer} \rangle | \langle \text{unsigned integer} \rangle$

$\langle \text{signed integer} \rangle ::= +\langle \text{digit} \rangle | -\langle \text{unsigned integer} \rangle$

$\langle \text{unsigned integer} \rangle ::= \langle \text{digit} \rangle | \langle \text{digit} \rangle \langle \text{unsigned integer} \rangle$

Derivation of -108

$\langle \text{integer} \rangle ::= \langle \text{signed integer} \rangle ::= \langle \text{unsigned integer} \rangle$

$::= -\langle \text{digit} \rangle \langle \text{unsigned integer} \rangle ::= -\langle \text{digit} \rangle \langle \text{digit} \rangle \langle \text{unsigned integer} \rangle$

$::= -\langle \text{digit} \rangle \langle \text{digit} \rangle \langle \text{digit} \rangle$

$::= -1 \langle \text{digit} \rangle \langle \text{digit} \rangle$

$::= -10 \langle \text{digit} \rangle$

$::= -108$

A regular grammar is a context free grammar and that a context free grammar with no production of the form $A \rightarrow \lambda$ is a context sensitive grammar.

If every production is of the form

$A \rightarrow \alpha$ or $A \rightarrow \alpha B$ or $A \rightarrow \lambda$

Where $A, B \in N$, $\alpha \in T^* - \{ \lambda \}$ then G is also a regular grammar.

Q) The grammar G defined by $T = \{a, b, c\}$,

$N = \{ \sigma, A, B, C, D, E \}$, with productions

$\sigma \rightarrow aAB, \sigma \rightarrow aB, A \rightarrow aAC, A \rightarrow ac, B \rightarrow DC, D \rightarrow b, CD \rightarrow CE, CE \rightarrow DE, DE \rightarrow DC, Cc \rightarrow Dcc$
and starting symbol σ is context sensitive

$CD = CE = DE = DC$ (derivation of dC from CD)

The string $a^3b^3c^3$ is in $L(G)$

$aAB \rightarrow aaACB = aaaCCDc = aaaCDCc$

$= aaaDCCc = aaaDCDcc = aaaDDCccc$

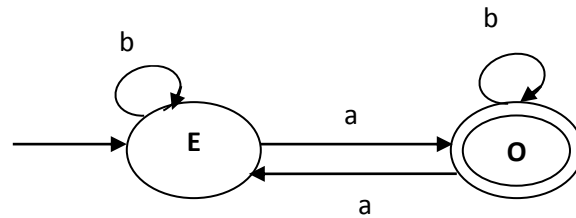
$= aaaDDDccc = aaabbbccc$

The string $a^3b^3c^3$ is in $L(G)$

It can be show that

$L(G) = \{ a^n b^n c^n \mid n = 1, 2, \dots \}$

(Q) Write the regular grammar given by FSA given below



The terminal symbols are the input symbols $\{a, b\}$. The states E and O become the non terminal symbols. The initial state E becomes the starting symbol. The production correspond to the directed edges.

If there is an edge labeled x from S to S',

We write

$S \rightarrow xS'$

In addition, if S is an accepting state we includes the production

$S \rightarrow \lambda$

In case

$E \rightarrow aO, E \rightarrow bE, O \rightarrow aE, O \rightarrow bO$

since O is accepting state we also include $O \rightarrow \lambda$

Then the grammar $G=(N,T,P,E)$ with $N=\{O, E\}$, $T=\{a, b\}$ and P consisting of above productions generate the language $L(G)$ which is the same as the set of string accepted by above FSA.

Theorem

Let A be a FSA given as a transition diagram. Let σ be the initial state. Let T be the set of input symbols and N be the set of states.

Let P be the set of productions.

$S \rightarrow xS'$

If there is an edge labeled x from S to S' and

$S \rightarrow S'$

If S is an accepting state let G be the regular grammar.

$G = (N, T, P, \sigma)$

Then set of strings accepted by A is equal to $L(G)$.

Reverse case

Given a regular grammar G , we want to construct a FSA to that $L(G)$ is precisely the set of string accepted by A .

E.g.

Consider the regular grammar defined by

$T = \{a, b\}$

$N = \{\sigma, c\}$

With productions,

$\sigma \rightarrow b\sigma, \sigma \rightarrow aC, C \rightarrow bC, C \rightarrow b$ and starting symbol σ

The non terminal become state with σ as the initial state. For each production of the form

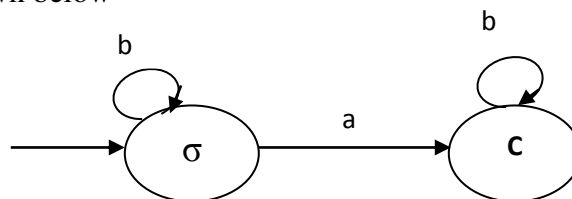
$S \rightarrow xS'$

we draw an edge from state s to S' and label it x .

The productions:

$\sigma \rightarrow b\sigma, \sigma \rightarrow aC, C \rightarrow bC$

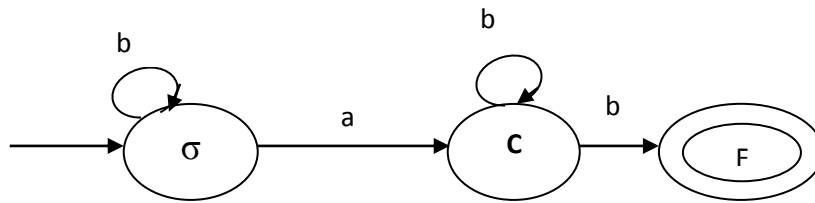
give the graph as shown below



The production $C \rightarrow b$ is equivalent to two productions.

$C \rightarrow bF$ and $F \rightarrow \lambda$

where F is an additional non terminal symbol.



The production $F \rightarrow \lambda$ tells us that F should be an accepting state.

This is not FSA

- (i) vertex C has no outgoing edge label a.
- (ii) vertex F has no outgoing at all
- (iii) vertex C has two outgoing edges labeled b

This is NFA (Non Deterministic Finite Automata)

Definition:

NDFSA 'A' consists of

- a) A finite set I of input symbols.
- b) A finite set S of states
- c) A next-state function f from $S \times I$ into $P(S)$ (subset state S)
- d) A subset A of S accepting states
- e) An initial state $\sigma \in S$

We write $A = (I, S, f, A, \sigma)$

In DFSA, the next-state function takes us to a uniquely defined state, whereas in NDFSA, the next-state function takes us to a set of states.

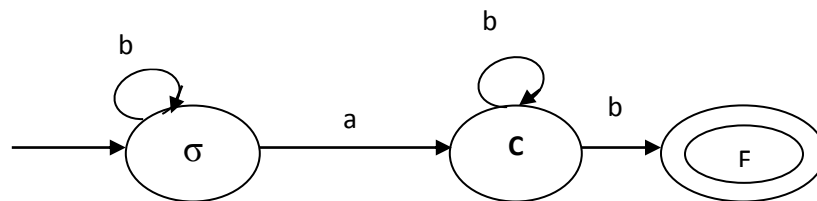
Example

$I = \{a, b\}$

$S = \{\sigma, C, F\}$

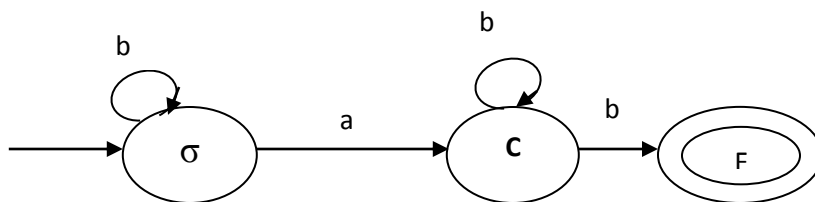
$A = \{F\}$

		f
S \ I	a	b
σ	{c}	{ σ }
C	$\emptyset$	{c,f}
F	$\emptyset$	$\emptyset$

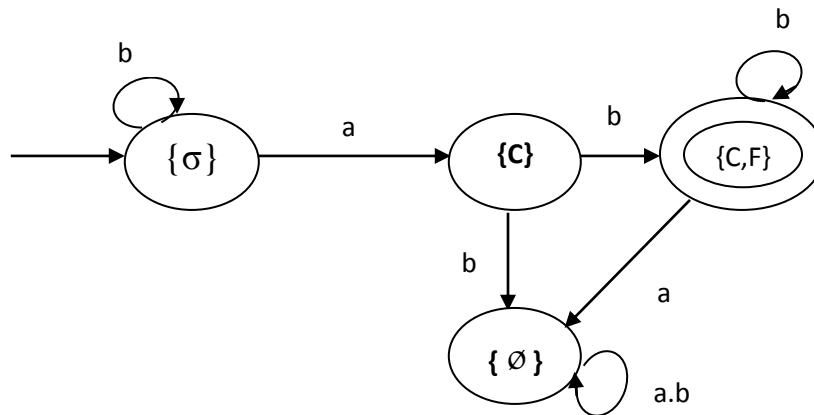


The string $\alpha = \text{bbabb}$ is accepted because there is at least one path representing α that ends at an accepting state.

(Q) Convert NDFSA to DFSA



S \ I	a	b
{ σ }	{c}	{ σ }
{C}	$\emptyset$	{C, F}
{C,F}	$\emptyset$	{C, F}



$$g(\{\sigma\}, a) = \{C\}$$

$$g(\{\sigma\}, b) = \{\sigma\}$$

$$g(\{C\}, a) = \emptyset$$

$$g(\{C\}, b) = E(C) \cup E(F) = \{C, F\}$$

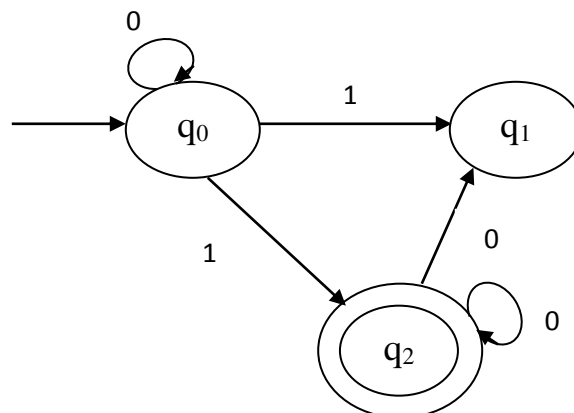
$$g(\{C, F\}, a) = \emptyset$$

$$g(\{C, F\}, b) = E(C) \cup E(F) = \{C, F\}$$

$$g(\{F\}, a) = \emptyset$$

$$g(\{F\}, b) = \emptyset$$

(Q) Change NFA to DFA



Q	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_1, q_2\}$
q_1	$\emptyset$	$\emptyset$
q_2	$\{q_1, q_2\}$	$\emptyset$

$$\delta'(q_0, 0) = \delta(q_0, 0) = \{q_0\}$$

$$\delta'(q_0, 1) = \delta(q_0, 1) = \{q_0, q_1\} \text{ new state}$$

$$\delta'(\{q_1, q_2\}, 0) = \delta(q_1, 0) \cup \delta(q_2, 0)$$

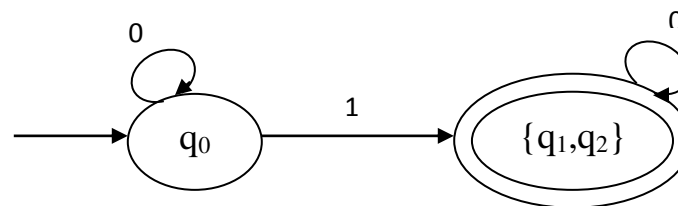
$$= \emptyset \cup \{q_1, q_2\}$$

$$= \{q_1, q_2\} \text{ old state}$$

$$\delta'(\{q_1, q_2\}, 1) = \delta(q_1, 1) \cup \delta(q_2, 1)$$

$$= \emptyset \cup \emptyset$$

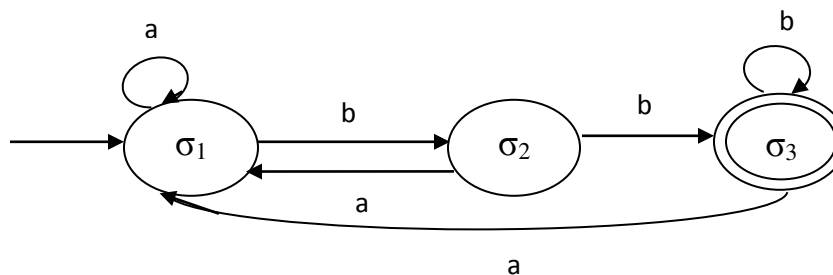
$$= \emptyset$$



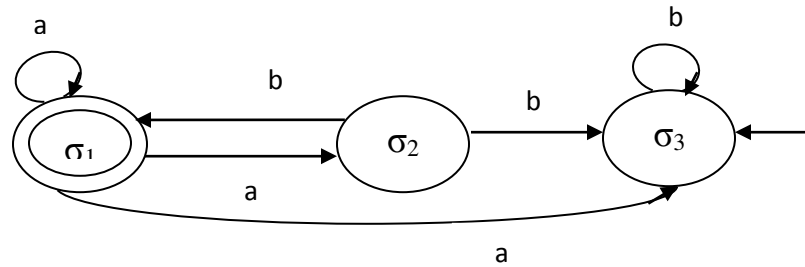
Equivalent DFA

(Q) Let L be the set of strings accepted by the FSA shown below construct a FSA that accepts the strings

$$L^R = \{x_n \dots x_1 \mid x_1 \dots x_n \in L\}$$



Reverse all arrows in above FSA and make σ_3 the starting state and σ_1 the accepting state. The result is a NDFSA that accepts L^R



Change into DFSA

$$E(\sigma_3) = \{\sigma_3\}, E(\sigma_2) = \{\sigma_2\}, E(\sigma_1) = \{\sigma_1\}$$

Transitions

$$(\sigma_3, b, \sigma_3) \quad (\sigma_2, b, \sigma_1) \quad (\sigma_1, a, \sigma_1)$$

$$(\sigma_3, b, \sigma_2) \quad (\sigma_1, a, \sigma_2)$$

$$(\sigma_1, b, \sigma_3)$$

$$g(\{\sigma_3\}, a) = \emptyset$$

$$g(\{\sigma_3\}, b) = E(\sigma_3) \cup E(\sigma_2) = \{\sigma_3, \sigma_2\}$$

$$g(\{\sigma_3, \sigma_2\}, a) = \emptyset$$

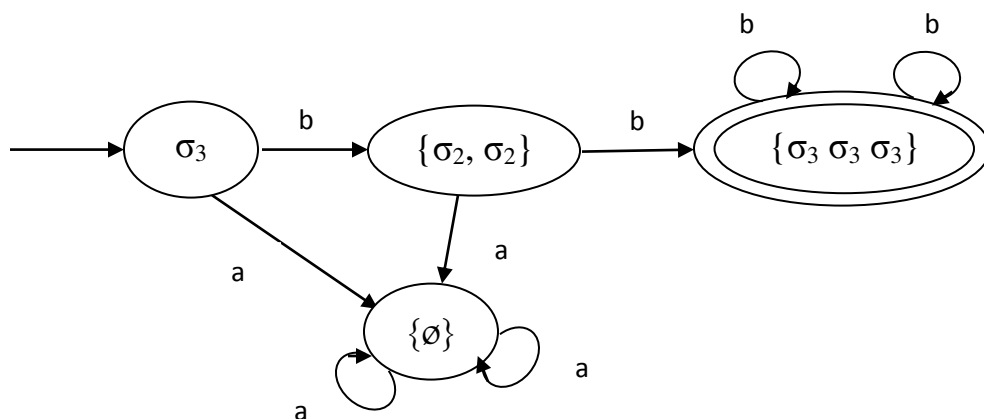
$$g(\{\sigma_3, \sigma_2\}, b) = E(\sigma_3) \cup E(\sigma_2) \cup E(\sigma_1) = \{\sigma_3, \sigma_2, \sigma_1\}$$

$$g(\{\sigma_3, \sigma_2, \sigma_1\}, a) = E(\sigma_1) \cup E(\sigma_2) \cup E(\sigma_3) = \{\sigma_3, \sigma_2, \sigma_1\}$$

$$g(\{\sigma_3, \sigma_2, \sigma_1\}, b) = E(\sigma_1) \cup E(\sigma_2) \cup E(\sigma_3) = \{\sigma_3, \sigma_2, \sigma_1\}$$

$$g(\{\emptyset\}, a) = \{\emptyset\}$$

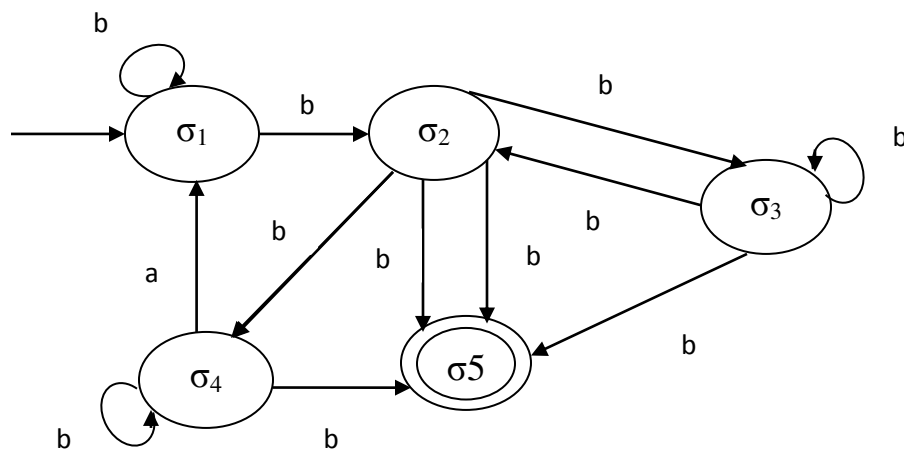
$$g(\{\emptyset\}, b) = \{\emptyset\}$$



Non Deterministic Finite State Automata

A non-deterministic finite automaton is a generalization of a finite state automaton so that at each state there might be several possible choices for the “next state” instead of just one.

Convert NDFSA to DFSA



Regular expressions

Regular expressions are useful for representing certain set of strings in an algebraic manner. Actually this describes the language accepted by finite state automata.

The regular expressions over an alphabet Σ are the strings over the alphabet $\Sigma \cup \{ \epsilon, \cup, \cap, \cdot, *, \emptyset \}$ such that the following hold (Σ consists of alphabet $\Sigma = \{a, b, \dots\}$ I/p symbols)

- 1) The empty set $\emptyset$ and each member of Σ is a regular expression.
- 2) If α and β are regular expressions then their concatenation $\alpha\beta$ is also a regular expression.
- 3) If α and β are regular expressions then their union ($\alpha \cup \beta$) is also a regular expressions.
- 4) If α is a regular expressions then so is the closure (iteration) α^* . If α is R.E (α) is also RE.
- 5) Nothing is a regular expression unless it follows from (1) through (4).

Regular languages can be characterized as languages defined by regular expressions.

The language represented by regular expression is given a function L , such that if α is any regular expressions then $L(\alpha)$ is the language represented by α .

- 1) $L(\emptyset) = \emptyset$, $L(a) = \{a\}$, for each $a \in \Sigma$
- 2) If α and β are regular expressions then $L(\alpha\beta) = L(\alpha) \cdot L(\beta)$
- 3) If α and β are regular expressions

Then

$$L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$$

- 4) If α is a regular expression

Then

$$L(\alpha^*) = L(\alpha)^*$$

What are the strings specified by the regular expression?

Regular expression is **language generator** it generates the set of all strings for finite automata.
(FSA, pumping lemma are **language recognizer**.)

- 1) $10^* = A$ '1' followed by any no. of 0's including non zero.
- 2) $(10)^* =$ Any number of copies of 10 (including the null string)

- 3) $0U01$ = The string 0 or the string 01.
- 4) $0(0U1)^*$ = Any string not beginning with 0.
- 5) $(0^*1)^*$ = Any string not ending with 0`
- 6) $\Sigma^* = \{0,1\}^* = (0+1)^* = \{ \epsilon, 0, 1, 00, 01, 10, 11, \dots \}$
- 7) $0^*+1^* = \{ \epsilon, 0, 00, 000, \dots \}$
 $= \{ \epsilon, 1, 11, 111, \dots \}$ (can it give combination only either 0 or 1.)
- 8) $0^*1^* = \{ \epsilon, 0, 1, 01, 001, 0011, \dots \}$ (zero never come after 1)
- 9) $(01)^* = \{ \epsilon, 01, 0101, 010101, \dots \}$
- 10) 01^*+10^* = either single 0 followed by any number of 1 or vice versa.
- 11) $(0+1)^*.00$

A language is regular iff it is accepted by FSA.

There are three operations on language that the operators of regular expression represent.

These operations are;

1) Union:

The union of two languages L and M denoted by $L+M$ is the set of strings that are in either L or M or both

E.g. $L = \{001, 10, 111\}$ and $M = \{\epsilon, 001\}$

Then,

$L+M = \{\epsilon, 10, 001, 111\}$

2) Concatenation:

The concatenation of language L and M is the set of strings that can be formed by taking any string in L and concatenation it with any string in M and is denoted by $L.M$

E.g. $L = \{001, 10, 111\}$, $M = \{\epsilon, 001\}$

Then,

$L.M = \{001, 10, 111, 001001, 10001, 111001\}$

3) The closure (or star or kleene star) of language

L is denoted by L and represents the set of those strings that can be formed by taking any number of strings from L, possibly with repetitions (i.e. the same string may be selected more than once) and concatenating all of them.

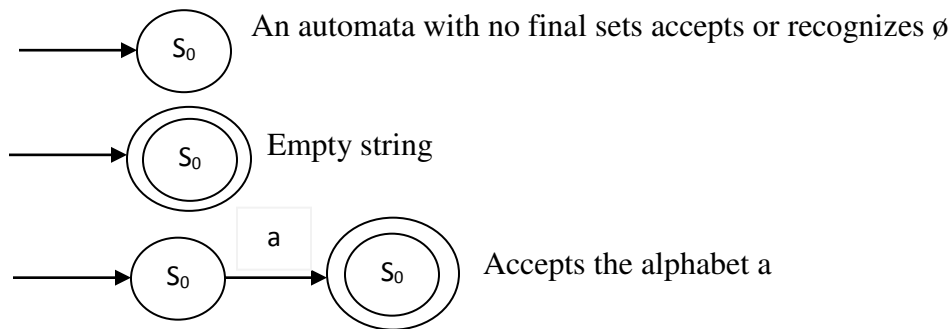
E.g. $L = \{0, 1\}$ then L^* is the set of all the strings of 0's and 1's and is with ϵ or $\wedge$

i.e. $L = \{0, 1\}$

$L^* = \{0, 1\}^* = \{\wedge, 0, 1, 01, 10, 11, 111, \dots\}$

The class of languages accepted by finite state automata (i.e. R.L.) is closed under.

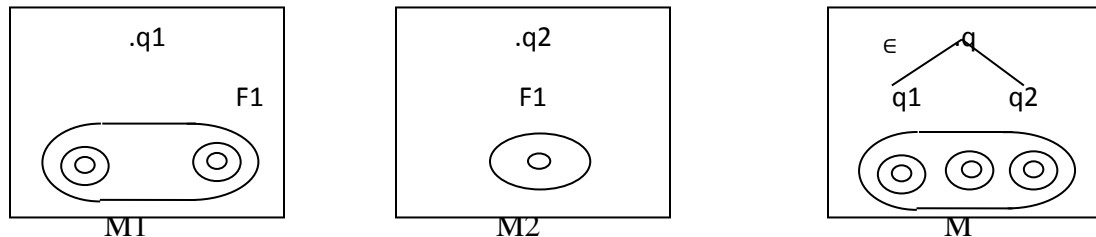
- 1) Union
- 2) Concatenation
- 3) Kleene star
- 4) Complementation
- 5) Intersection



1) Union :

Let L_1 and L_2 be languages accepted by non-deterministic automata M_1 and M_2 respectively. Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$. We assume that Q_1 and Q_2 are disjoint sets.

We construct NDFSA M that accepts $L(M_1) \cup L(M_2)$ as follows



$M = (Q, \Sigma, \delta, q, F)$ where q is new state not in q_1 or Q_2

$Q = Q_1 \cup Q_2 \cup \{q\}$ [Set of states of M]

$F = F_1 \cup F_2$ [Final states of M]

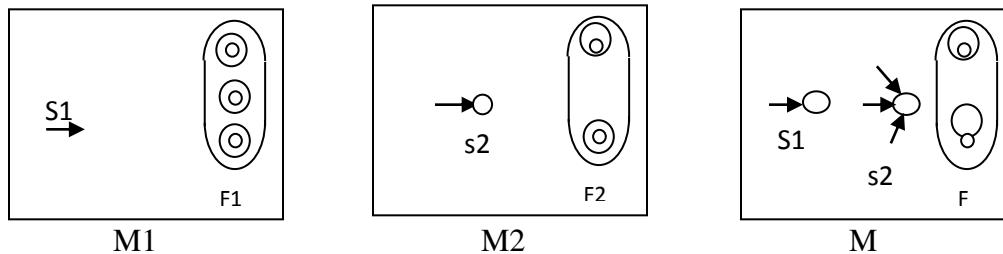
Transition function of $M = \delta_1 \cup \delta_2 \cup \{(q, \epsilon) \rightarrow (q_1, \epsilon), (q, \epsilon) \rightarrow q_2\}$

Or,

$F_1 \cup F_2 \cup \{(s, e, s_1), (s, e, s_2)\}$

2) Concatenation :

Let M_1 and M_2 be NDFA. We construct a NDFA M such that $L(M) = L(M_1) \rightarrow L(M_2)$. The construction is shown in the figure.



M operates by simulating M_1 for a while and then jumping non-deterministically from a final state of M_1 to the initial state of M_2 . Therefore M imitates M_2 .

Formally,

Let,

$M_1 = (S'_1, I, f_1, s_1, F_1)$ or (Q_1, δ, q_1, F_1) or $(Q_1, \Sigma, \delta, q_1, F_1)$

$M_2 = (S'_2, I, f_2, s_2, F_2)$ s_1 and s_2 are disjoint

$L(M) = L(M_1) \cdot L(M_2)$

$M = (S, I, f, \sigma, F)$

Set of states of $M = S = S'_1 \cup S'_2 = \text{union of } M_1 \text{ and } M_2$

Final states (F) = final states of M_2 (F_2)

Transition $S = \text{Union of } M_1 \text{ and } M_2 \text{ plus } (F_1 \times \{e\} \times \{s_2\})$

$F = F_1 \cup F_2 \cup \{(F_1, c) \rightarrow S_2\}$

3) Kleene star :

Let M_1 be a NDFA we construct a NDFA M such that $L(M) = L(M_1)^*$. The construction is shown below.



$$L(M)=L(M_1)$$

States of M =states of M_1 plus s'_1

Final states of M = Final states of M_1 plus s'_1

Transition of M = Transition of M_1 plus $(FX\{e\})X\{s_1\}$

$$F=F_1 \cup F_2 \cup \{(F, e, S_1)\}$$

Pumping lemma (Theorem) for Regular Language

Pumping lemma is a powerful technique for showing certain language to be non-regular.

Statement

Let L be a regular language. Then there exist a constant n (which depend on L) such that for every string w in L , such that $|w| \geq n$.

We can break w into three sub strings

- 1) $y \neq \epsilon$
- 2) $|xy| \leq n$
- 3) For all $i \geq 0$, string xy^iz is also in L

Technique to identify the class of language for finite automata is pumping lemma.

Show that the language

$$L=\{a^n b^n \mid n > 0\} \text{ is not regular}$$

$$w \in L$$

$$w = a^p b^p$$

According to the pumping lemma

$$x = a^q$$

$$y = a^r$$

$$z = a^{p-(q+r)} b^p$$

Since,

$$xy \in L$$

Now,

$$xy^2z$$

$$=a^q(a^r)^2(a^{p-(q+r)}b^p)$$

$$=a^qa^{2r}a^{p-(q+r)}b^p$$

$$=a^{p+r}b^p \notin L$$

Since xy^2z is not the form $a^p b^p$ therefore $xy^2z \notin L$. It is contradiction. Hence L is not regular.

Closure properties of Regular Language

If L and M be regular language. Then following languages are all regular.

- 1) Union : $L \cup M, L+M$
- 2) Intersection : $L \cap M$
- 3) Complement : $\bar{L}, \bar{L}$ and $\bar{M}, \bar{L} = \Sigma^* \setminus L$
- 4) Difference : $L \setminus M = L \cap \bar{M}$
- 5) Reversal $L^R = W^R : w \in L$
- 6) Closure: L^*
- 7) Concatenation : LM
- 8) Homomorphism
 $h(L) = \{h(w) \mid w \in L, h \text{ is homomorphism}\}$
- 9) Inverse homomorphism
 $h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L, h: \Sigma^* \rightarrow \Delta^* \text{ is a homomorphism}\}$

homomorphism

$h: \Sigma^* \rightarrow \Delta^*$ (alphabet Σ, Δ)

$w: a_1 a_2 \dots a_n \in \Sigma^*$

$h(w) = h(a_1) h(a_2) \dots h(a_n)$

$h(L) = \{h(w) \mid w \in L\}$

example

let $h: \{0,1\}^* \rightarrow \{a,b\}^*$ be defined by

$h(0) = a$ and $h(1) = b$

now,

$h(0011) = abab$

$h(L(10^*1)) = L((ab)^*)$

Chapter 3. Recurrence Relation [8 hrs.][8 Marks]

3. Recurrence Relation

3.1. Recursive Definition Sequences

3.2. Solution of Linear recurrence relations

3.3. Solution of Nonlinear recurrence relations

3.4. Application to algorithm analysis

Recurrence Relations

A sequence can be defined by giving a general formula for its n^{th} term by writing a few of its terms. An alternative approach is to represent the sequence by finding a relationship among its terms. Such relations are referred as recurrences.

Recursively defined sequences are often referred to as recurrence relations.

A recurrence relation for the sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more of the previous terms of the sequence namely $a_0, a_1, a_2, \dots, a_{n-1}$ for all integers n with $n \geq n_0$ where n_0 is a non-negative integer.

A sequence is called a solution of a recurrence relation if its term satisfy the recurrence relation.

Example

Consider the following instructions for generating a sequence

1) start with 3

2) Given any term, add 3 to get the next term.

The term of sequence obtained is

3, 6, 9, 12, 15, 18, ...

If we denote the sequence as $a_1, a_2, \dots$

$a_1 = 3$ (1)

Instruction (2) can be rephrased as

$a_n = a_{n-1} + 3 \quad n \geq 2$ (A)

By using a_1 and a_n , we can compute any term in the sequence.

Equation (A) is an example of recurrence relation (1)

Example

2) Let $\{a_n\}$ be a sequence that satisfies the recurrence relation. $a_n = a_{n-1} - a_{n-2}$ for $n=2,3,4$ and suppose that $a_0=3$ and $a_1=5$ what are a_2 and a_3 ?

From the recurrence relation.

$$a_n = a_{n-1} - a_{n-2}$$

$$a_2 = a_1 - a_0 = 5 - 3 = 2$$

$$a_3 = a_2 - a_1 = 2 - 5 = -3$$

Similarly a_4 , a_5 and also each successive terms. The sequence forms in such a way is $3, 5, 2, -3, \dots$

3) Determine whether the sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = 2a_{n-1} - a_{n-2}$ for $n=2,3,4, \dots$ where $a_n = 3n$ for every non negative integer n . Answer the same question where $a_n = 2^n$ and where $a_n = 5$.

Suppose that $a_n = 3n$ for every n . therefore $n \geq 2$, $2a_{n-1} - a_{n-2}$

$$= 2(3(n-1)) - 3(n-2)$$

$$= 6n - 6 - 3n + 6$$

$$= 3n$$

$$= a_n$$

Since $\{a_n\}$, where $a_n = 3n$, is a solution of the recurrence relation.

Suppose $a_n = 2^n$ for every n . note $a_0 = 2^0 = 1$

And $a_1 = 2^1 = 2$ and $a_2 = 2^2 = 4$

$$2a_{n-1} - a_{n-2}, \quad (n=2)$$

$$= 2a_1 - a_0$$

$$= 2 \cdot 2 - 1 = 3 \neq a_2, \text{ we see that } \{a_n\}, \text{ where } a_n = 2^n \text{ is not a solution of recurrence relation. } a_n = 5$$

4) A customer deposits 10,000 in a bank and he gets 11% interest per annual and the interest amount is accumulated every year. Find the recurrence relation to compute total amount customer receive after 10 year. And also calculate amount that customer receive after 30 years.

$$p_0 = 10,000$$

$$\text{Interest} = 11\%$$

$$p_1 = p_0 + 0.11 * p_0$$

$$= 1.11p_0$$

$$= 11100$$

$$p_2 = p_1 * 0.11 + p_1$$

$$=1.11p_0 * 0.11 + p_0 * 1.11$$

$$=1.11p_0(0.11+1)$$

$$=(1.11)p_0 * 1.11$$

$$=(1.11)^2 p_0$$

$$p_3 = 1.11p_2 = (1.11)^3 p_0$$

$$p_n = (1.11)p_{n-1} = (1.11)^n p_0$$

$$p_n = (1.11)^n 10,000.$$

Solution of Recurrence Relation

A linear homogeneous recurrence relation of degree k with constant coefficient is a recurrence relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

Where $c_1, c_2, \dots, c_k$ are real numbers and $c_k \neq 0$.

The recurrence relation in the definition is linear since R.H.S is a sum of multiples of the previous terms of the sequence.

The recurrence relation is homogeneous since no term occur that are not multiple of the a_j s.

The degree is k because a_n is expressed in terms of the previous k term of the sequence.

-If $f(n) \neq 0$ for the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k} + f(n)$$

Then it is called non-homogeneous recurrence relation.

If $f(n) = 0$ then it is called linear homogeneous recurrence relation.

Solving linear homogeneous recurrence relation

Method of characteristic equation

We know that the linear homogeneous recurrence relation of degree k with constant coefficients is of the form.

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

Let $a_n = r^n$ is the solution of this equation then

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \dots + c_k r^{n-k}$$

Dividing both by r^{n-k}

$$r^k = c_1 r^{k-1} + c_2 r^{k-2} + \dots + c_k$$

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \dots - c_k = 0$$

This is the characteristic equation of degree k of the recurrence relation.

Let us consider the linear homogeneous recurrence relations of degree 2.

i.e.

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

Its characteristic equation is of the form

$$r^2 - c_1 r - c_2 = 0$$

The roots of this equation can be

1) Distinct roots, say r_1 and r_2 (i.e. $r_1 \neq r_2$)

If roots are distinct then $\{a_n\}$ is the solution of the given linear homogeneous recurrence relation if and only if a_n is of the form

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n \text{ where } \alpha_1 \text{ and } \alpha_2 \text{ are constant}$$

1) Repeated roots, say r_0 (i.e. $r_1 = r_2 = r_0$)

If roots are repeated root, say r_0 then $\{a_n\}$ is the solution of the given linear homogeneous recurrence relation if and only if a_n is of the form

$$a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$$

Theorem:

Let c_1 and c_2 be real numbers suppose that $r^2 - c_1 r - c_2 = 0$ has two distinct roots r_1 and r_2 , then the sequence $\{a_n\}$ is a solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ iff $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ for $n = 0, 1, 2, \dots$ where α_1 and α_2 are constant. To find α_1 and α_2 initial condition is used.

1) What is the solution of recurrence relation

$$a_n = a_{n-1} + 2a_{n-2} \text{ with } a_0 = 2 \text{ and } a_1 = 7$$

$$\text{Let } a_n = r^n$$

$$\text{Substituting in the equation } a_n = a_{n-1} + 2a_{n-2}$$

we get,

$$r^n = r^{n-1} + 2r^{n-2} \dots \dots \dots (i)$$

Dividing equation (i) by r^{n-2} , we get

$$r^2 - r - 2 = 0$$

$$r^2 - (2-1)r - 2 = 0$$

$$r^2 - 2r + r - 2 = 0$$

$$r(r-2) + 1(r-2) = 0$$

$$(r-2)(r+1) = 0$$

Its roots are $r=2$ and $r=-1$. Hence the sequence $\{a_n\}$ is a solution to the recurrence relation iff

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$$

$$a_n = \alpha_1 2^n + \alpha_2 (-1)^n$$

For some constants α_1 and α_2 . From initial conditions it follows that,

$$a_0 = \alpha_1 + \alpha_2 = 2$$

$$a_1 = \alpha_1 2 + \alpha_2 (-1)$$

$$7 = \alpha_1 2 + \alpha_2 (-1)$$

$$\text{i.e. } \alpha_1 + \alpha_2 = 2 \quad \dots\dots\dots (ii)$$

$$2\alpha_1 - \alpha_2 = 7 \quad \dots\dots\dots (iii)$$

Solving these two equation

$$\alpha_1 = 3 \text{ and } \alpha_2 = 1$$

Hence the solution to the recurrence relation and initial conditions is the sequence $\{a_n\}$ with

$$a_n = 3 \cdot 2^n - (-1)^n$$

2) Find the explicit formula for the fibonacci numbers.

OR

Find the solution of recurrence relation of fibonacci series.

$$a_0 = 0$$

$$a_1 = 1$$

$$a_n = a_{n-1} + a_{n-2}$$

The characteristic equation of $a_n = a_{n-1} + a_{n-2}$ is

$$r^n = r^{n-1} + r^{n-2}$$

(since assume $a_n = r^n$ then $r_n = r^{n-1} + r^{n-2}$)

$$r^n / r^{n-2} = r^{n-1} / r^{n-2} + r^{n-2} / r^{n-2}$$

$$r^2 - r - 1 = 0$$

$$r = \frac{+1 \pm \sqrt{(-1)^2 - 4(1)(-1)}}{2 \cdot 1}$$

$$r = \frac{1 \pm \sqrt{5}}{2}$$

$$r_1 = \frac{1 + \sqrt{5}}{2}$$

$$r_2 = \frac{1 - \sqrt{5}}{2}$$

Then fibonacci numbers are given by

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$$

$$= \alpha_1 \left(\frac{1 + \sqrt{5}}{2} \right)^n + \alpha_2 \left(\frac{1 - \sqrt{5}}{2} \right)^n$$

For some constants α_1 and α_2 , using $a_0=0$ and $a_1=1$

When $n=0$, $a_0=0$

$$a_0 = \alpha_1 + \alpha_2$$

$$\alpha_1 = -\alpha_2$$

When $n=1$, $a_1=1$

$$a_1 = \alpha_1 \left(\frac{1+\sqrt{5}}{2} \right)^1 + \alpha_2 \left(\frac{1-\sqrt{5}}{2} \right)^1$$

$$1 = -\alpha_2 \left(\frac{1+\sqrt{5}}{2} \right) + \alpha_2 \left(\frac{1-\sqrt{5}}{2} \right)$$

$$= \alpha_2 \left(-\frac{1}{2} - \frac{\sqrt{5}}{2} + \frac{1}{2} - \frac{\sqrt{5}}{2} \right)$$

$$= \alpha_2 \left(-\sqrt{5} \right)$$

$$\alpha_2 = -1/\sqrt{5}$$

$$\alpha_1 = 1/\sqrt{5}$$

$$\text{Hence, } a_n = 1/\sqrt{5} \left(\frac{1+\sqrt{5}}{2} \right)^n + (-1/\sqrt{5}) \left(\frac{1-\sqrt{5}}{2} \right)^n$$

3) What is the solution of recurrence relation

$$a_n = 6a_{n-1} + 9a_{n-2} \text{ with } a_0=3 \text{ and } a_1=7$$

Solution,

$$a_n = 6a_{n-1} + 9a_{n-2}$$

Let $a_n = r^n$

we get,

$$r^n = 6r^{n-1} + 9r^{n-2} \dots\dots\dots (1)$$

Dividing equation (1) by r^{n-2} , we get

$$r^2 - 6r - 9 = 0$$

Solving,

We get....

....

Theorem :

Let c_1 and c_2 be real number with $c_2 \neq 0$. Suppose that $r^2 - c_1r - c_2 = 0$ has only one root r_0 . A sequence $\{a_n\}$ is a solution of recurrence relation $a_n = c_1a_{n-1} + c_2a_{n-2}$ iff $a_n = \alpha_1 r_0^n + \alpha_2 n r_0^n$, for $n=0,1,2,\dots$. Where α_1 and α_2 are constant.

Theorem :

Let $c_1, c_2, \dots, c_k$ be real numbers. Suppose that the characteristic equation.

$$r^k - c_1 r^{k-1} - \dots - c_k = 0$$

has k distinct roots $r_1, r_2, \dots, r_k$ then sequence $\{a_n\}$ is a solution of recurrence relation.

$$a_n = c_1(a_{n-1}) + c_2a_{n-2} + \dots + c_ka_{n-k} \text{ iff}$$

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \dots + \alpha_k r_k^n \text{ for } n=0,1,2,\dots \text{ where } \alpha_1, \alpha_2, \alpha_3, \dots, \alpha_k \text{ are constants.}$$

Theorem :

Let $c_1, c_2, \dots, c_k$ be real numbers, suppose that the characteristic equation

$$r^k - c_1 r^{k-1} - \dots - c_k = 0$$

has t distinct roots $r_1, r_2, \dots, r_k$ with multiplicity $m_1, m_2, m_3, \dots, m_t$ respectively so that $m_i \geq 1$ for $1, 2, \dots, t$ and $m_1 + m_2 + \dots + m_t = k$. Then a sequence $\{a_n\}$ is a solution of the recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k} \text{ iff}$$

$$a_n = (\alpha_{1,0} + \alpha_{1,1}n + \dots + \alpha_{1,m_1-1}n^{m_1-1})r_1^n + (\alpha_{2,0} + \alpha_{2,1}n + \dots + \alpha_{2,m_2-1}n^{m_2-1})r_2^n + \dots + (\alpha_{t,0} + \alpha_{t,1}n + \dots + \alpha_{t,m_t-1}n^{m_t-1})r_t^n$$

For $n=0, 1, 2, \dots$. Where $\alpha_{i,j}$ are constants for $1 \leq i \leq t$ and $0 \leq j \leq m_i - 1$

Suppose that roots of the characteristic equation of a linear homogeneous recurrence relation are 2, 2, 2, 5, 5 and 9 (that is there are three roots, the root 2 with multiplicity two three, the root 5 with multiplicity two and the root 9 with multiplicity one.

What is the form of general solution?

$$(\alpha_{1,0} + \alpha_{1,1}n + \alpha_{1,2}n^2)2^n + (\alpha_{2,0} + \alpha_{2,1}n)5^n + \alpha_{3,0}9^n$$

1) $a_n = 6a_{n-1} - 9a_{n-2}$ with $a_0 = 1, a_1 = 6$

Find the solution of given recurrence relation.

Solution:

$$a_n = 6a_{n-1} - 9a_{n-2}$$

Let $a_n = r^n$, we get

$$r^n = 6r^{n-1} - 9r^{n-2}, \text{ we get,}$$

$$r^2 = 6r - 9$$

$$r^2 - 6r + 9 = 0 \dots\dots\dots(1)$$

This is a characteristic equation,

Solving the equation (1)

$$r = 3, 3$$

Hence,

$$a_n = \alpha_1 3^n + \alpha_2 n 3^n \text{ for some constants, applying initial condition.}$$

Here, $a_0 = 1$

$$a_0 = \alpha_1$$

$$\alpha_1 = 1$$

Here $a_1 = 6$

$$a_1 = \alpha_1 3 + \alpha_2 3$$

$$6 = 3 + \alpha_2 3$$

$$\alpha_2 = 1$$

Hence the solution is $a_n = 3^n + n 3^n$

(2) Find the solution to the recurrence relation $a_n = 6a_{n-1} - 11a_{n-2} + 6a_{n-3}$

With the initial conditions $a_0 = 2, a_1 = 5$ and $a_2 = 15$.

Solution:

The characteristic equation is given by

$$r^3 - 6r^2 + 11r - 6 = 0$$

$$\text{Or, } r^3 - r^2 - 5r^2 + 5r + 6r - 6 = 0$$

$$\text{Or, } r^2(r-1) - 5r(r-1) + 6(r-1) = 0$$

$$\text{Or, } (r-1)(r^2 - 5r + 6) = 0$$

$$\text{Or, } (r-1)(r^2 - 3r - 2r + 6) = 0$$

$$\text{Or, } (r-1)[r(r-3) - 2(r-3)] = 0$$

$$\text{Or, } (r-1)(r-2)(r-3) = 0$$

$$r_1 = 1, r_2 = 2, r_3 = 3.$$

The solution is then given by

$$a_n = \alpha_1 \cdot 1^n + \alpha_2 \cdot 2^n + \alpha_3 \cdot 3^n$$

For some constants α_1, α_2 , and α_3

Finding α_1, α_2 and α_3 using initial conditions

$$a_0 = 2 = \alpha_1 + \alpha_2 + \alpha_3 \dots \dots \dots (1)$$

$$a_1 = 5 = \alpha_1 + \alpha_2 \cdot 2 + \alpha_3 \cdot 3 \dots \dots \dots (2)$$

$$a_2 = 15 = \alpha_1 + \alpha_2 \cdot 4 + \alpha_3 \cdot 9 \dots \dots \dots (3)$$

Solving equation (1), (2), and (3), we get

$$\alpha_1 = 1, \alpha_2 = -1, \alpha_3 = 2$$

Hence the solution is given by

$$a_n = 1 - 2^n + 2 \cdot 3^n$$

3) Find the solution to the recurrence relation $a_n = -3a_{n-1} = 3a_{n-2} - a_{n-3}$ with initial conditions $a_0 = 1$, $a_1 = -2$ and $a_2 = -1$

The characteristic of this recurrence relation is

$$r^3 + 3r^2 + 3r + 1 = 0$$

$$(r+1)^3 = 0$$

$r_1 = r_2 = r_3 = -1$ i.e. there is a single root $r = -1$ of multiplicity three of characteristic equation.

The solution is given by,

$$\begin{aligned} a_n &= (\alpha_{1,0} + \alpha_{1,1}n + \alpha_{1,2}n^2)(-1)^n \\ &= (\alpha_{1,0}(-1)^n + \alpha_{1,1}n(-1)^n + \alpha_{1,2}n^2(-1)^n) \end{aligned}$$

To find the constants $\alpha_{1,0}, \alpha_{1,1}$ and $\alpha_{1,2}$ use the initial condition. This gives

$$a_0 = 1 = \alpha_{1,0} \dots \dots \dots (1)$$

$$a_1 = -2 = \alpha_{1,0}(-1) + \alpha_{1,1}(-1) + \alpha_{1,2}(-1)$$

$$a_2 = -1 = -\alpha_{1,0} - \alpha_{1,1} - \alpha_{1,2} \dots \dots \dots (2)$$

$$a_2 = -1 = (\alpha_{1,0}(-1)^2 + \alpha_{1,1}2(-1)^2 + \alpha_{1,2}2^2(-1)^2)$$

$$a_2 = -1 = \alpha_{1,0} + 2\alpha_{1,1} + 4\alpha_{1,2} \dots \dots \dots (3)$$

Solving (1),(2) and (3), we get,

$$\alpha_{1,0}=1, \alpha_{1,1}=3, \alpha_{1,2}=-2$$

Hence, the solution is given by,

$$a_n=1(-1)^n+3n(-1)^n+(-2)n^2(-1)^n$$

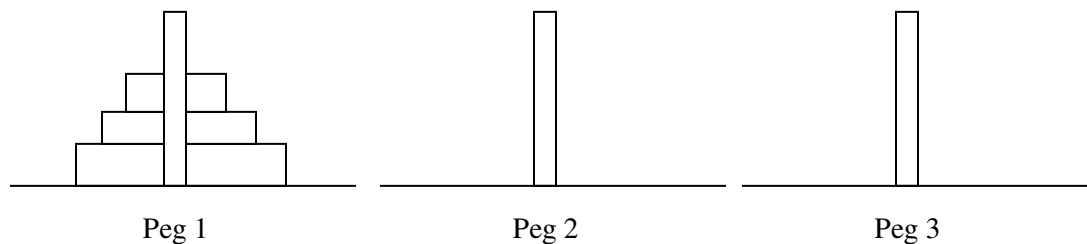
$$a_n=(1+3n-2n^2)(-1)^n$$

Tower of Hanoi Puzzle

There are 3 pegs and n circular disks of increasing on one peg, with the largest on the bottom. These disks are to be transferred one at a time onto the another peg, with the provision that larger disk is not allowed to place on a smaller disk. The problem is to determine the number of moves necessary for the transfer.

Solution:

Let h_n be the number of moves required to transfer n disk. We verify that $h_0=0$, $h_1=1$ and $h_2=3$. We have to find a recurrence relation that is satisfied by h_n .



To transfer n disks to another peg, we first transfer the top $n-1$ disks to a peg, transfer the largest disk to the vacant peg and then transfer all $n-1$ disks to the peg, which now contain the largest disk, this h_n satisfies $h_n=2h_{n-1}+1$ ($n \geq 1$) where $h_0=0$

This is a recurrence relation of order 1 with constant coefficients, but it is not homogeneous, because of the presence of the quantity.

1. To find h_n , we iterate the above equation.

$$\begin{aligned} h_n &= 2h_{n-1} + 1 \\ &= 2(2h_{n-2} + 1) + 1 \\ &= 2^2h_{n-2} + 1 \\ &= 2^2(2h_{n-3} + 1) + 2 + 1 \\ &= 2^{n-1}(2h_0 + 1) + 2^{n-2} + \dots + 2^2 + 2 + 1 \\ &= 2^{n-1} + 2^{n-2} + \dots + 2^2 + 2 + 1 \\ &= \frac{1(2^n - 1)}{2 - 1} \quad (\text{since } s_n = \frac{a(r^n - 1)}{r - 1}) \\ &= 2^n - 1 \end{aligned}$$

Therefore, the no. h_n are the partial sums of the geometric sequence $1, 2, 2^2, \dots, 2^n$ and hence satisfy.

$$\begin{aligned} h_n &= \frac{2^n - 1}{2 - 1} = 2^n - 1 \\ &= 2^n - 1 \end{aligned}$$

We have to verify this formula for h_n and it can be easily verified by mathematical induction.

Since $h_0=0$, it holds for $n=0$

Assume that it holds for n . we then show it holds with n replaced by $n+1$

$$\text{i.e. } h_{n+1}=2h_n+1$$

$$=2(2^n-1)+1$$

$$=2^{n+1}-1$$

Since it also holds for $n+1$, hence the formula is verified.

Linear non-homogeneous recurrence relations

To solve the non-homogeneous recurrence relation with constant coefficient, we need to solve its associated linear homogeneous equation and to find a particular solution for the given non-homogeneous equation.

A recurrence relation of the form

$$a_n=c_1a_{n-1}+c_2a_{n-2}+\dots+c_ka_{n-k}+F(n)$$

Where $c_1, c_2, \dots, c_k$ are real numbers and $F(n)$ is a function not identically zero depending only on n is linear non-homogeneous recurrence relation. The recurrence relation

$$a_n=c_1a_{n-1}+c_2a_{n-2}+\dots+c_ka_{n-k}$$

is called the associated homogeneous recurrence relation.

For example,

$$1) a_n=a_{n-1}+2^n$$

$$2) a_n=a_{n-1}+a_{n-2}+n^2+n+1$$

$$3) a_n=3a_{n-1}+n3^n$$

$$4) a_n=a_{n-1}+a_{n-2}+a_{n-3}+n$$

These are examples of linear non-homogeneous recurrence relation with constant coefficients corresponding associated linear homogeneous recurrence relation are

$$1) a_n=a_{n-1}$$

$$2) a_n=a_{n-1}+a_{n-2}$$

$$3) a_n=3a_{n-1}$$

$$4) a_n=a_{n-1}+a_{n-2}+a_{n-3}$$

Theorem :

If $\{a_n^{(p)}\}$ is a particular solution of the non-homogeneous linear recurrence relation with constant coefficients.

$$a_n=c_1a_{n-1}+c_2a_{n-2}+\dots+c_ka_{n-k}+f(n)$$

Then every solution is of the form $\{a_n^{(p)}+a_n^{(h)}\}$ where $a_n^{(h)}$ is a solution of the associated homogeneous recurrence relation.

$$a_n=c_1a_{n-1}+c_2a_{n-2}+\dots+c_ka_{n-k}$$

Theorem:

Suppose $\{a_n\}$ satisfies the linear recurrence relation.

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k} + f(n)$$

Where $c_1, c_2, \dots, c_k$ are real numbers and

$$f(n) = (b_t n^t + b_{t-1} n^{t-1} + \dots + b_1 n + b_0) s^n$$

Where $b_0, b_1, \dots, b_t$ and s are real numbers. When s is not a root of the characteristics equation of the associated linear homogeneous recurrence relation there is a particular solution of the form.

$$(p_t n^t + p_{t-1} n^{t-1} + \dots + p_1 n + p_0) s^n$$

When s is a root of this characteristic equation and its multiplicity is m , there is a particular solution is the form

$$n^m (p_t n^t + p_{t-1} n^{t-1} + \dots + p_1 n + p_0) s^n$$

1) what form does a particular solution of the linear non-homogeneous recurrence relation

$$a_n = 6a_{n-1} - 9a_{n-2} + f(n).$$

have when $f(n) = 3^n$

$$f(n) = n 3^n$$

$$f(n) = n^2 2^n$$

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

Solution,

The associated linear homogeneous recurrence relation is $a_n = 6a_{n-1} - 9a_{n-2}$. Its characteristics equation, $r^2 - 6r + 9 = (r-3)^2 = 0$ has a single root 3, of multiplicity two.

For $f(n) = 3^n$

Because $s=3$ is a root with multiplicity 2, particular solution has the form $p_0 n^2 3^n$ if $f(n) = 3^n$

For $f(n) = n 3^n$

The form is $n^2 (p_1 n + p_0) 3^n$ if $f(n) = n 3^n$

For $f(n) = n^2 2^n$

The form is $(p_2 n^2 + p_1 n + p_0) 2^n$ if $f(n) = n^2 2^n$

For $f(n) = (n^2 + 1) 3^n$

The form is $n^2 (p_2 n^2 + p_1 n + p_0) 3^n$ if $f(n) = (n^2 + 1) 3^n$

2) Find all solutions of the recurrence relation $a_n = 3a_{n-1} + 2n$. what is the solution with $a_1 = 3$?

Solution,

The associated linear homogeneous equation is $a_n = 3a_{n-1}$

Its solution are $a_n^{(h)} = \alpha 3^n$

Where α is constant

Now, finding particular solution

$f(n) = 2n$ is a polynomial of degree one

Comparing with $f(n) = p(n) s^n$

we get, $s=1$

The root of the characteristic equation is 3

Here, s is not the root of characteristic equation of the associated linear homogeneous recurrence relation, there is a particular solution of the form

$$(p_0 + p_1 n + \dots + p_{t-1} n^{t-1} + p_t n^t) s^n$$

$$= (p_0 + p_1 n) s^n$$

$$= (p_0 + p_1 n)$$

New, equation $a_n = 3a_{n-1} + 2n$ becomes

$$(p_0 + p_1 n) = 3(p_0 + p_1(n-1)) + 2n$$

$$\text{or, } (p_0 + p_1 n) = 3p_0 + 3p_1 n - 3p_1 + 2n$$

$$\text{or, } n(p_1 - 3p_1 - 2) = 2p_0 - 3p_1$$

$$\text{or, } n(-2p_1 - 2) = 2p_0 - 3p_1$$

$$\text{or, } n(2p_1 + 2) + 2p_0 - 3p_1 = 0$$

It follows that $cn+d$ is a solution if and only if $2+2p_1=0$ and $2p_0-3p_1=0$

$$\therefore p_1 = -1, p_0 = -3/2$$

$$a_n^{(p)} = -n - 3/2$$

$$a_n = a_n^{(p)} + a_n^{(h)}$$

$$= -n - 3/2 + \alpha \cdot 3^n$$

To find solution with $a_1=3$

$$n=1, a_1=3$$

$$3 = -1 - 3/2 + 3\alpha$$

$$6 = -2 - 3 + 6\alpha$$

$$\alpha = 11/6$$

$$a_n = -n - 3/2 + (11/6)3^n$$

3) Find the solution of recurrence relation $a_n = 5a_{n-1} - 6a_{n-2} + 7^n$

Solution,

The solution of its associated homogeneous recurrence relation

$$a_n = 5a_{n-1} - 6a_{n-2}$$

Here,

$$f(n) = 7^n$$

Comparing with $f(n) = p(n) s^n$

$$s=7$$

Here s is not the root of characteristic equation, there is particular solution in the form

$$p_0 = 7^n$$

Now, equation $a_n = 5a_{n-1} - 6a_{n-2} + 7^n$ becomes

$$p_0 7^n = 5 \cdot p_0 7^{n-1} - 6 \cdot p_0 7^{n-2} + 7^n$$

$$p_0 = 49/20$$

$a_n^{(p)} = (49/20) \cdot 7^n$ is a particular solution

$$a_n = a_n^{(p)} + a_n^{(h)}$$

$$= (49/20) \cdot 7^n + \alpha_1 3^n + \alpha_2 2^n$$

4) Let a_n be the sum of the first n positive integer so that

$$a_n = \sum_{k=1}^n k$$

a_n satisfies the linear non-homogeneous recurrence relation

$$a_n = a_{n-1} + n \dots\dots\dots (1)$$

The associated linear homogeneous recurrence relation are given by

$$a_n^{(h)} = c(1)^n = c,$$

Where c is constant

Here,

We have,

$$f(n) = n$$

Comparing with $f(n) = p(n)s^n$

$$s = 1$$

Here, s is the root of characteristic equation. The particular solution is in the form

$$n(p_1 n + p_0)$$

$$= p_1 n^2 + p_0 n \quad (\text{here } m=1)$$

Now,

Equation (1) becomes

$$p_1 n^2 + p_0 n = p_1 (n-1)^2 + p_0 (n-1) + n$$

Simplifying we see,

$$n(2p_1 - 1) + (p_0 - p_1) = 0 \text{ which means}$$

$$2p_1 - 1 = 0 \text{ and } p_0 - p_1 = 0$$

$$p_1 = p_0 = 1/2$$

Hence,

$$a_n^{(p)} = n^2/2 + n/2 = n(n+1)/2 \text{ is a particular solution}$$

$$a_n = a_n^{(p)} + a_n^{(h)}$$

$$a_n = n(n+1)/2 + c$$

$$\text{We have, } a_1 = 1, 1 = a_1 = c + (1/2) \cdot 2 = c + 1$$

$$c = 0$$

$$a_n = n(n+1)/2$$

Solving recurrence relation using substitution

Solve recurrence relation

$$a_n^2 - 2a_{n-1}^2 = 1 \text{ for } n \geq 1 \text{ where } a_0 = 1$$

This non-linear equation can be converted into linear equation by using substitution $b_n = a_n^2$

Now equation become $b_n - 2b_{n-1} = 1$ with $b_0 = a_0^2 = 1^2 = 1$

Solving $b_n - 2b_{n-1} = 1$ with $b_0 = 1$, the homogeneous solution is $\alpha \cdot 2^n$ and particular solution is c .

So, $b_n = a \cdot 2^n + c$ where a and c are constants.

Using fact $b_0 = 1$ and $b_1 = 3$ we get $a + c = 1$ and $2a + c = 3$

$$\therefore a = 2 \text{ and } b = -1.$$

Hence the solution is $b_n = 2 \cdot 2^n - 1$

$$= 2 \cdot 2^n - 1$$

$$= 2^{n+1} - 1$$

Substituting back $a_n^2 = 2^{n+1} - 1$

$$a_n = \sqrt{2^{n+1} - 1}$$

Divide and conquer recurrence relations

Divide and conquer relation is of the form $a_n = c \cdot a_{n/d} + f(n)$. Hence this equation can be looked at as $b_k = c b_{k-1} + f(d^k)$ using $n = d^k$ and solved. Initial condition have to be changed appropriately. For substituting back, we use $k = \log_d n$.

Q.1) Solve the divide and conquer relation $a_n = 3a_{n/2} + n$

Where, $n = 2^k$ for $k \geq 1$ and $a_1 = 1$

Using change of variables the equation becomes

$$b_k = 3b_{k-1} + 2^k$$

The homogeneous solution $b_k^{(h)} = A3^k$ and the particular solution is $b_k^{(p)} = B2^k$

We get,

$$b_k = A3^k + B2^k$$

Since,

$$a_1 = 1, b_0 = 1. \text{ So } A + B = 1$$

$$a_2 = b_1 = 5,$$

We get,

$$A = 3 \text{ and } B = -2$$

So, solution is,

$$b_k = 3 \cdot 3^k + (-2) \cdot 2^k$$

Substituting back,

$$a_n = 3 \cdot 3^{\log_2 n} - 2 \cdot 2^{\log_2 n}$$

$$= 3 \cdot 3^{\log_2 n} - 2n$$

But,

$$3^{\log_2 n} = n^{\log_2 3}$$

So,

$$a_n = 3n^{\log_2 3} - 2n$$

Analysis of algorithm

Binary search:

Binary search looks for a value in a sorted sequence and returns the index of the value if it is found or 0 if it is not found.

Input : A sequence $s_i, s_{i+1}, \dots, s_j, i \geq 1$ sorted in increasing order, a value key, i and j.

Output : The output is an index k for which $s_k = \text{key}$ or if key is not in the sequence the output is the value 0.

- 1) Procedure binary search (s, i, j, key)
- 2) If $i > j$ then return(0) //not found.
- 3) $K = (i+j)/2$
- 4) Key = s_k then return (k) //found
- 5) If key $< s_k$ then $j = k-1$ //search left half
- 6) Else $i = k+1$ //search right half
- 7) (repeat the process) return (binary_search (s, i, j, key))
- 8) End binary search

This algorithm uses the divide and conquer approach. The sequence is divided into two nearly equal parts. (line 3)

- If the item is found at the dividing point (line 4), the algorithm terminates.
- If the item is not found, because the sequence is sorted, an additional comparison will locate the half of the sequence in which the item appears if it is present.
- We then recursively invoke binary search (line 7) to continue the search.

Example

Input

$s_1=2, s_2=4, s_3=5, s_4=6$ and key 6

- $i=1, j=4, k=\text{index}$
- Since $(i > j)$ ($1 > 4$) is false
- $k=2$
- Since key(6) is not equal to $s_4(4)$
- Key $(6) < s_k$ ($6 < 4$) is false
- We set, $i=k+1 = 3$
 $i=3, j=4$
 $s_3=5, s_4=6$

$i > j$ ($3 > 4$) is false

$k = (3+4)/2 = 3$

- Since key (6) is not equal to $s_3(5)$

- Key (6) $< s_3$ ($6 < 5$) is false

We set

$i = k + 1 = 3 + 1 = 4$

$i = 4, j = 4, s_4 = 6$

- $i > j$ ($4 > 4$) is false

- Set $k = (4+4)/2 = 4$

Key (6) is equal to s_4 (6). So we return 4, the index of key in the sequence s.

Thus, binary search reduces the search of an element in a search sequence of size n to the binary search sequence of the element in a search sequence of size $n/2$ when n is even. Hence the problem of size n has reduced to one problem of size $n/2$. Two comparisons are needed to implement this reduction (one to determine which half of the list to use and other to determine whether any term of the list remains). Hence if $f(n)$ is number of comparisons required to search for an element in a search sequence of size n, then

$f(n) = f(n/2) + 2$

Selection sort

This algorithm sorts the sequence

$s_1, s_2, \dots, s_n$

in increasing order by first selecting the largest item and placing it last and then recursively sorting the remaining elements.

Input : $s_1, s_2, \dots, s_n$ and length n of the sequence

Output ; $s_1, s_2, \dots, s_n$ arranged in increasing order

1) Procedure selection sort (s, n)

2) Base case, if $n=1$ then return

3) // find largest value

$\text{max_index} = 1$ // assume

4) For $i=2$ to n do

 If $s_i > s_{\text{max_index}}$ then // found larger so update $\text{max_index} = i$

- 5) // more largest to end
Swap (s_n , $s_{\max_index}$)
- 6) Call selection_sort (s , $n-1$)
- 7) End selection_sort
 $b_n = n-1 + b_{n-1}$

Chapter 4. Graph Theory [15 hrs.][32 Marks]

4. Graph Theory

- 4.1. Undirected and directed graphs
- 4.2. Walk path, circuits, components
- 4.3. Connectedness algorithm
- 4.4. Shortest path algorithm
- 4.5. Bipartite graphs, planar graphs, regular graphs
- 4.6. Planarity testing algorithms
- 4.7. Eulerian graph
- 4.8. Hamiltonian graph
- 4.9. Tree as a directed graph
- 4.10. Binary tree, spanning tree
- 4.11. Cutsets and cutvertices
- 4.12. Network flows, maxflow and mincut theorem
- 4.13. Data structures representing trees and graphs in computer
- 4.14. Network application of trees and graphs
- 4.15. Concept of graph coloring

Graph Theory

Graph is a discrete structure consisting of vertices and edges that connect these vertices.

A graph $G = (V, E)$ consists of two things.

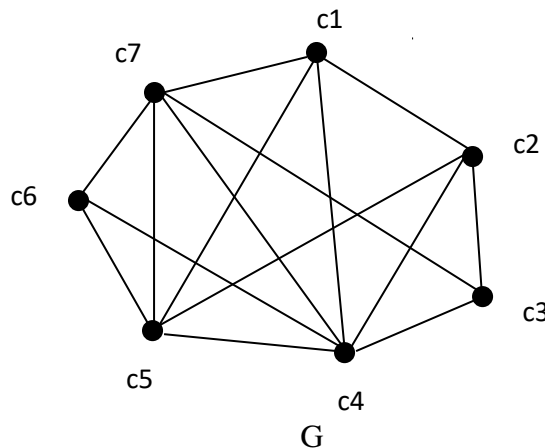
- i. A set $V=V(G)$ whose elements are called vertices or nodes of G .
- ii. A set $E=E(G)$ of unordered pairs of distinct vertices called edges of G .

Use of graph (example)

A major publishing company has ten editors in the scientific, technical and computing areas. These ten editors have a standard meeting time during the first Friday of every month and have divided themselves into seven committees to meet later in the day to discuss specific topics of interest of company. Namely, advertising, recurring reviewers, contacting new potential authors, finances, used copies and new editors, competing text books and text book representatives.

The 10 editors have decided on seven committees $C_1 = \{1, 2, 3\}$, $C_2 = \{1, 3, 4, 5\}$, $C_3 = \{2, 5, 6, 7\}$, $C_4 = \{4, 7, 8, 9\}$, $C_5 = \{2, 6, 7\}$, $C_6 = \{8, 9, 10\}$ $C_7 = \{1, 3, 9, 10\}$

They have set aside three time periods for the seven committees to meet on those Fridays when a ten editors are present. Some committees can not meet during the same period because one or two of the editors are on both committees.



In this figure, there are seven small circles representing the seven committees and a straight line segment is drawn between two circles if the committees they represent have at least one committee members in common. In other words this means that two committee should not be scheduled to meet at a same time.

A graph consist of a finite non empty set V of objects called vertices and a set E of element subsets of V called edges. So, graph is a pair of two sets V and E , vertex set and edge set respectively.

Vertices are also called points or nodes. Edges are sometimes called lines or arcs.

The vertex of set G is $V(G) = \{ c_1, c_2, \dots, c_7 \}$ and the edge set $E(G) = \{ \{ c_1, c_2 \}, \{ c_1, c_3 \}, \{ c_1, c_5 \}, \{ c_1, c_7 \}, \{ c_2, c_3 \}, \{ c_2, c_4 \}, \{ c_2, c_7 \}, \{ c_3, c_4 \}, \{ c_3, c_5 \}, \{ c_4, c_5 \}, \{ c_4, c_6 \}, \{ c_5, c_6 \}, \{ c_6, c_7 \} \}$

- The number of vertices in G is often called the order of G while the number of edges is its size. We use n and m for order and size respectively.
- Since, the vertex set of every graph is non-empty, the order of every graph is at least 1.
- A graph with exactly one vertex is called a trivial graph.

Types of graphs

1. Labeled graph
2. Unlabeled graph

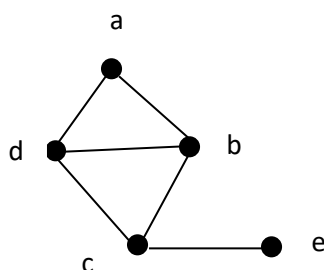


Fig: Labeled graph

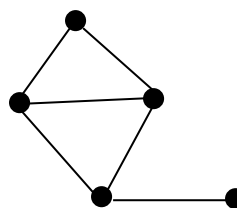


Fig: Unlabeled graph

Simple Graph:

Simple graph is 2-tuple consists of non-empty set of vertices V , a set of unordered pairs of distinct elements of vertices called edges. We represent this graph as $G=(V,E)$. This kind of graph has undirected edges, no loops and no multiple edges.

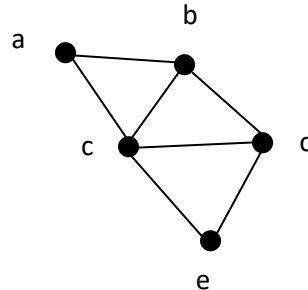


Fig : Simple graph

In above graph $G=(V, E)$, set of vertices.

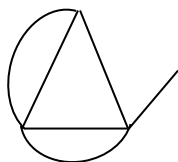
$V(G)$ or $V = \{a, b, c, d, e\}$ and set of edges.

$E(G)$ or $E= \{ \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{c, d\}, \{c, e\}, \{d, e\} \}$

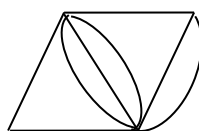
Multi graph

In a graph, two vertices are either adjacent or they are not, that is , two vertices are joined by one edge or no edges. A multigraph M consist of a finite non empty set V of vertices and a set of E of edges where every two vertices of M are joined by finite number of edges.

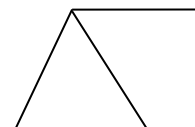
- If two or more edges join the same pair of vertices, then these edges are called parallel edges.
- In multi graph multiple edge occurs.
- In a pseudograph, an edge is also permitted to join a vertex to itself. Such an edge is called a loop.



M_1



M_2

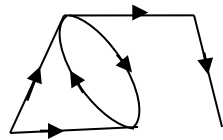


M_3

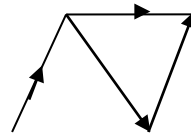
Every graph is a multigraph

Directed graph (Digraph)

A directed graph (V, E) consists of set of V of vertices, a set E of edges that are ordered pairs of elements of V . In this graph, loop is allowed but no two vertices can have multiple edges in same direction



D_1



D_2

D_2 is oriented graph

Directed Mutligraph

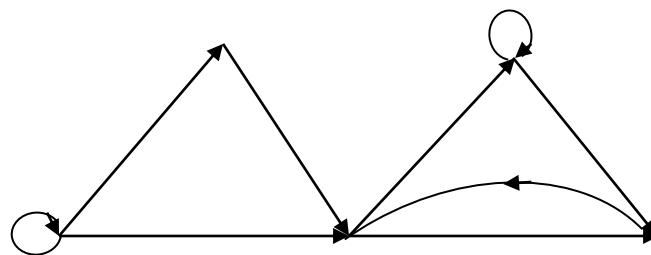


Fig: Directed Mutligraph

Graph Terminologies

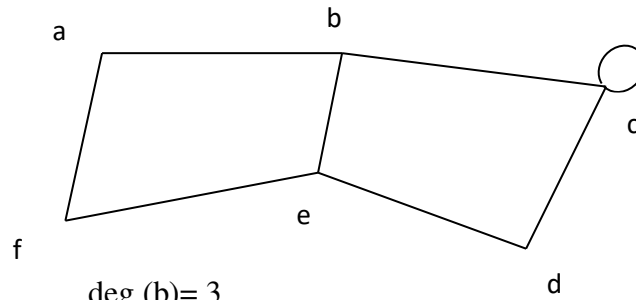
Adjacent : Two vertices u, v in an undirected graph G are called adjacent or neighbor if $\{u, v\}$ is an edge.

The edge e is called incident with the vertices u and v if $e = \{u, v\}$

Degree of a vertex :

The degree of a vertex in an undirected graph is a number of edges incident with it, except a loop at a vertex. Degree vertex v is denoted by $\deg(v)$. A vertex of degree zero is called isolated vertex and a vertex with degree one is called pendant vertex.

Loop in a vertex counts twice to the degree. e.g.: find the degrees.



$$\deg(a)=2$$

$$\deg(b)=3$$

$$\deg(c)=4$$

$$\deg(d)=2$$

$$\deg(e)=3$$

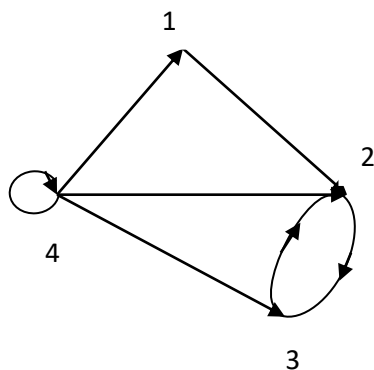
$$\deg(f)=2$$

Let (u, v) be an edge representing edge of a directed graph G , u is called adjacent to v and v is called adjacent from u . The vertex u is called initial vertex and the vertex v is called terminal or end vertex. Loop has same initial and terminal vertex.

In directed graph the in-degree of a vertex v , denoted by $\deg^-(v)$, is a number of edges, that have v as their terminal vertex. The out-degree of a vertex v , denoted by $\deg^+(v)$, is the number of edges that have v as their initial vertex.

Loop at a vertex adds up both in-degree and out-degree to one or more than calculated in-degree and out-degree.

- Find in-degree and out-degree of each vertex



In-degrees:

$$\deg^-(1) = 2$$

$$\deg^-(2) = 3$$

$$\deg^-(3) = 2$$

$$\deg^-(4) = 1$$

Out-degrees:

$$\deg^+(1) = 1$$

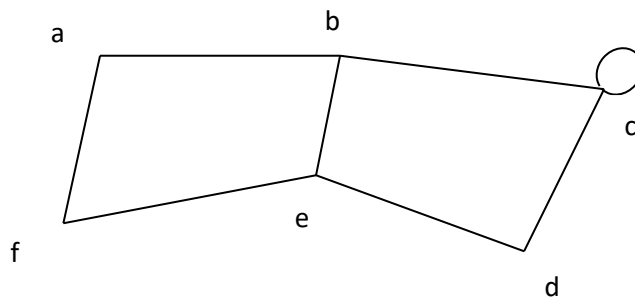
$$\deg^+(2) = 1$$

$$\deg^+(3) = 1$$

$$\deg^+(4) = 4$$

Theorem 1 : The Handshaking Theorem

Let $G=(V,E)$ be an undirected graph with e edges Then $2e = \sum \deg(v)$



Proof :

$$\deg(a) = \deg(f) = \deg(d) = 2$$

$$\deg(b) = \deg(e) = 3$$

$$\deg(c) = 4$$

We have

$$\text{LHS} = 2 * e$$

$$= 2 \times 8$$

$$= 16$$

$$\text{RHS} = \sum_{v \in V} \deg(v)$$

$$= 2 + 2 + 2 + 3 + 3 + 4$$

$$= 16$$

Therefore , LHS= RHS

Theorem 2: An undirected graph has an even number of vertices of odd degree.

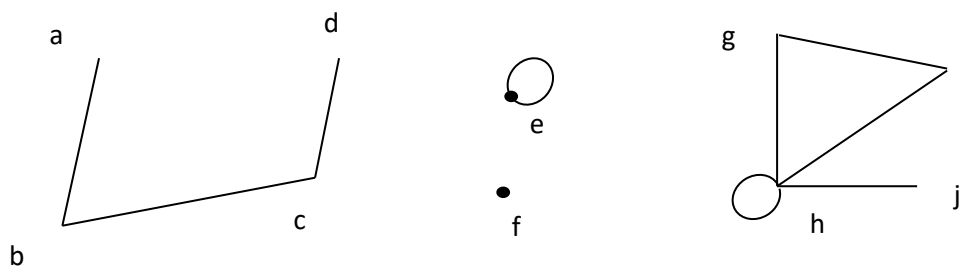
- Let V_1 and V_2 be set of vertices of even and odd degrees respectively. In an undirected graph $G = (V, E)$

We have,

$$2e = \sum_{v \in V} \deg(v) = \sum_{v \in V_1} \deg(v) + \sum_{v \in V_2} \deg(v)$$

From equality above, we can say the left part is even i.e. $2e$ is even, the sum of $\deg(v)$ for $v \in V_1$ is even since every vertex has even degree. So for the left hand to be even sum of $\deg(v)$ for $v \in V_2$ must be even. Since all vertices in the set V_2 have odd degree the number of such vertices must be even for the sum to be even.

Example: Which vertices in following graph are isolated which are pendant and what is maximum degree? What type of graph is it?



- Vertex f is isolated, vertices a, d and j are pendant. Maximum degree is $\deg(h) = 5$.

This graph is pseudograph (undirected loops)

Example : Determine the number of its edges and sum of the degrees of all its vertices.

- There are 9 edges and sum of degrees is 18.

Theorem 3:

The sum of the in-degrees of all vertices in a digraph = The sum of the out-degrees = The number of edges.

Let $G = (V, E)$ be a graph with directed edges,

Then,

$$\sum_{v \in V} (\deg^-(v)) = \sum_{v \in V} (\deg^+(v)) = |E|$$

Complete graph

- The complete graph of n vertices is the simple graph that contains exactly one edge between each pair of distinct vertices.

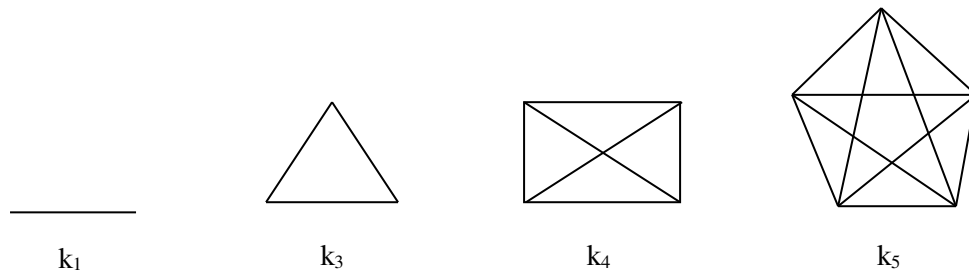


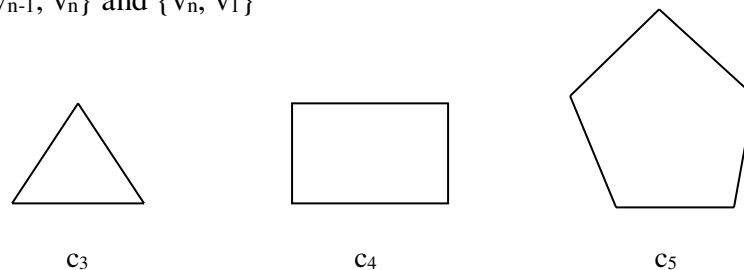
Fig: Complete graph

- A complete graph of order n is denoted by k_n . Since every distinct vertices of k_n are joined by an edge, the number of pairs of vertices in k_n is $\binom{n}{2}$ the size of k_n is

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

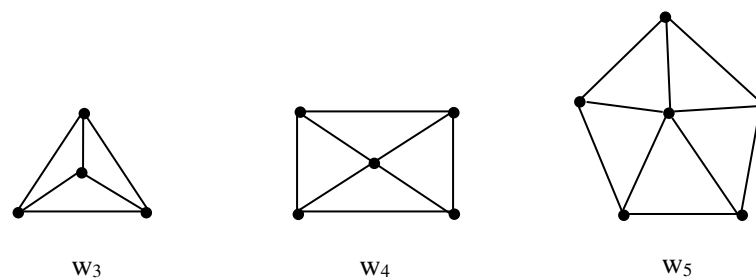
Cycle:

The cycle C_n , $n \geq 3$, consists of n vertices $v_1, v_2, \dots, v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{n-1}, v_n\}$ and $\{v_n, v_1\}$



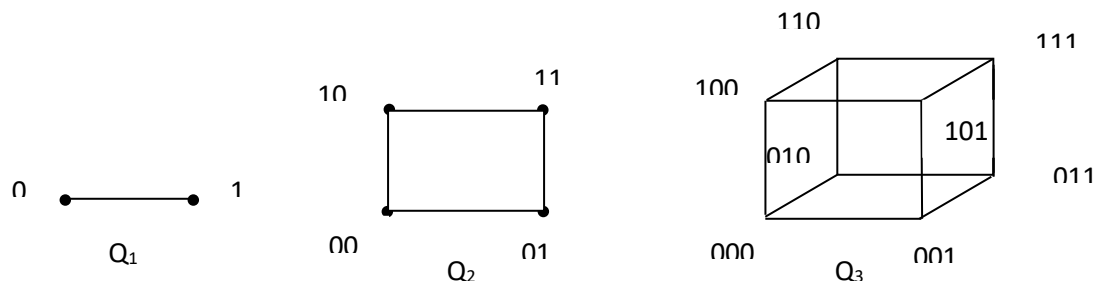
Wheel :

When a new vertex is added to a cycle C_n and this new vertex is connected to each of the n vertices in C_n , we obtain a wheel W_n .



n-cubes

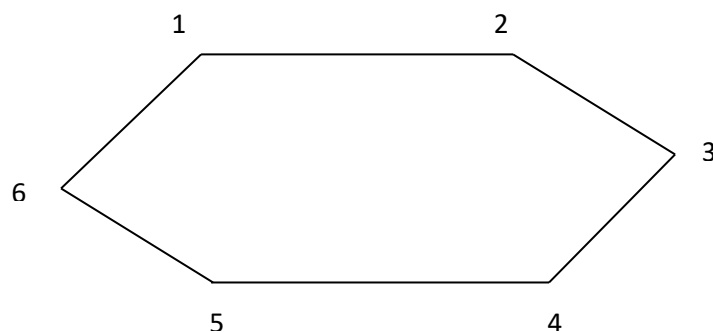
The n-cube, denoted by Q_n , is the graph that has vertices representing the 2^n bit string of length of n. Two vertices are adjacent if and only if the bit strings that they represent differ in exactly one bit position.



Bipartite Graph

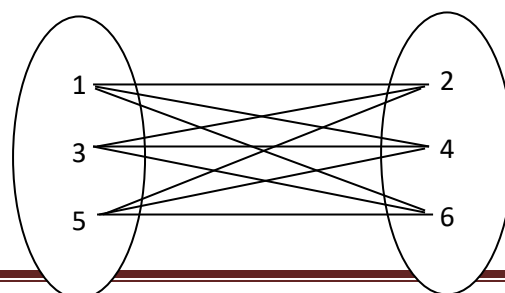
A simple graph G is bipartite if its vertex set can be partitioned into two disjoint subsets V_1 and V_2 such that every edge in the graph connects a vertex from the set V_1 to the vertex of set V_2 . No two vertices of the same set are connected by an edge

E.g.: Is C_6 Bipartite.

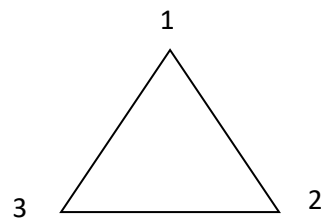


Its vertex set can be partitioned into two sets $V_1 = \{1, 3, 5\}$ and $V_2 = \{2, 4, 6\}$

every edge of C_6 connects a vertex in V_1 with a vertex in V_2 .



Q) Is K_3 Bipartite?

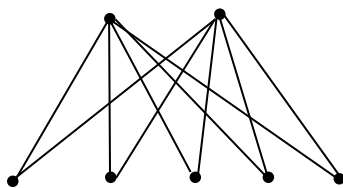


=No

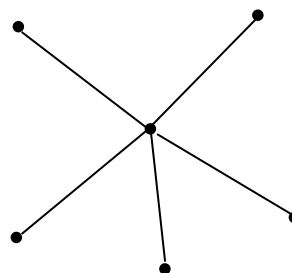
Complete Bipartite Graph

The complete bipartite graph $K_{m,n}$ is the graph where the vertex set is partitioned into two subsets of m and n vertices, respectively. In this graph, there is an edge between two vertices if and only if two vertices are in different subsets of vertices.

E.g. : $K_{2,5}$



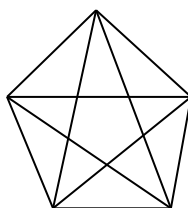
E.g. : $K_{1,5}$



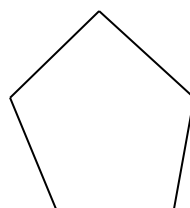
Operations on Graphs

– Sub graph

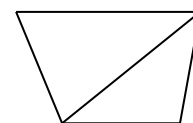
A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$, where $w \in V$ and $F \subseteq E$.



K_5



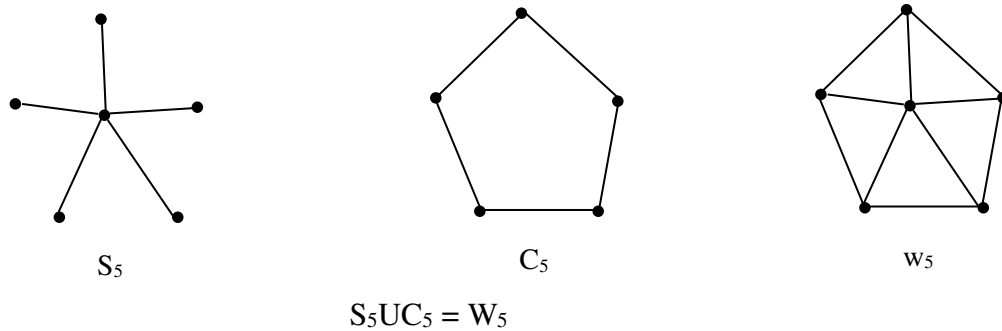
C_5



sub graph of K_5

Union

The union of two simple graphs $G_1=(V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V=(V_1 \cup V_2)$ and set $E=(E_1 \cup E_2)$. The union is denoted by $G_1 \cup G_2$



Representing graphs

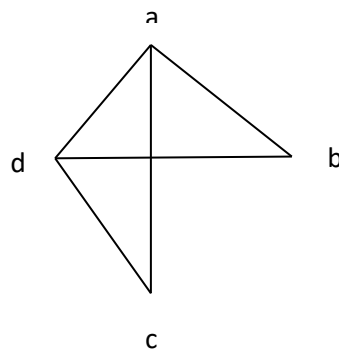


Fig : 1

Edge list for simple graph

Vertex	Adjacent vertices
a	b, c, d
b	a, d
c	a, d
d	a, b, c

Edge list for Directed graph

Initial Vertex	End vertex
a	b
b	c
c	b
d	a, b, c, d

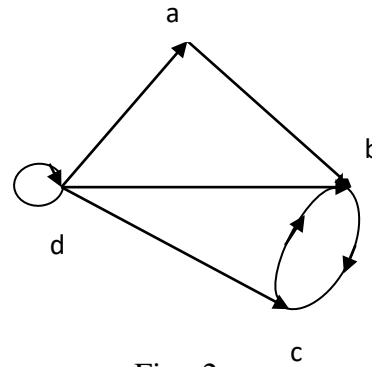


Fig : 2

Adjacency Matrix:

Given a simple graph $G=(V, E)$ with $|V|=n$ of the vertices of the graph are listed in some arbitrary order like $V_1, V_2, \dots, V_n$, the adjacency matrix A of G , with respect to the order of vertices is n -by- n zero-one matrix ($A=[a_{ij}]$) with the condition

$$a_{ij} = \begin{cases} 1 & \text{if } (V_i, V_j) \text{ is an edge of } G \\ 0 & \text{otherwise} \end{cases}$$

Adjacency matrix for undirected graph is symmetric, in case of pseudograph or multigraph the representation is similar but the matrix here is not zero-one matrix rather the $(i, j)^{\text{th}}$ entry of matrix contains the no. of edges appearing between the pair of vertices.

In case of directed graph, we can extend the same concept as in undirected graph as below

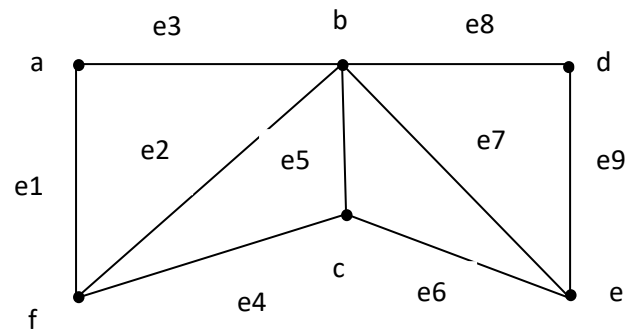
$$a_{ij} = \begin{cases} 1 & \text{if } (V_i, V_j) \text{ is an edge of } G \\ 0 & \text{otherwise} \end{cases}$$

Incidence Matrix

Given an undirected graph $G = (V, E)$. Assume that vertices of the graph $V_1, V_2, \dots, V_n$ and the edges of graph are $e_1, e_2, \dots, e_m$. The incidence matrix of a graph with respect to the above ordering of V and E is n by m matrix. Where

$$m_{ij} = \begin{cases} 1 & \text{when edge, is incident with } V_1 \\ 0 & \text{otherwise} \end{cases}$$

When the graph is not simple then graph can be represented by using incidence matrix where multiple edges corresponds to two different columns with exactly same entries. Loops are represented with column with only one entry.



The order of vertices be a, b, c, d, e, f, and edges order be e1, e2, e3, e4, e5, e6, e7, e8, e9

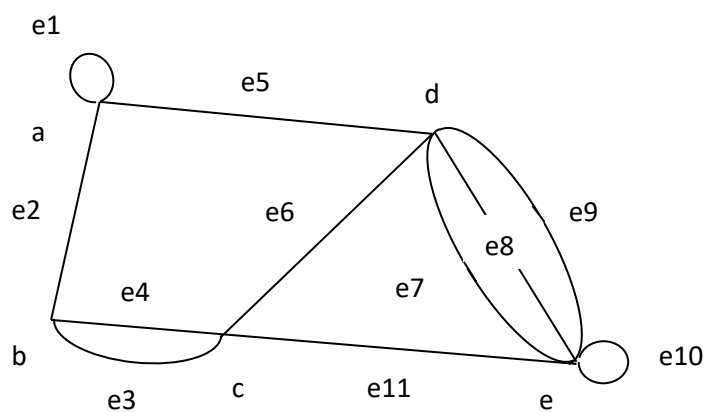
Adjacency Matrix:

	a	b	c	d	e	f
a	0	1	0	0	0	1
b	1	0	1	1	1	1
c	0	1	0	0	1	1
d	0	1	0	0	1	0
e	0	1	1	1	0	0
f	1	1	1	0	0	0

Incidence matrix

	e1	e2	e3	e4	e5	e6	e7	e8	e9
a	1	0	1	0	0	0	0	0	0
b	0	1	1	0	1	0	1	1	0
c	0	0	0	1	1	1	0	0	0
d	0	0	0	0	0	0	0	1	1
e	0	0	0	0	0	1	1	0	1
f	1	1	0	1	0	0	0	0	0

2

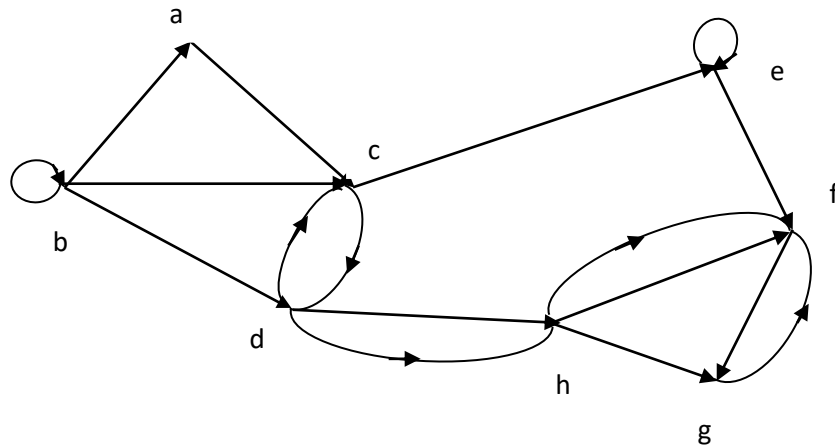


Adjacency matrix

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 3 & 1 \end{pmatrix}$$

Incidence matrix

	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11
a	1	1	0	0	1	0	0	0	0	0	0
b	0	1	1	1	0	0	0	0	0	0	0
c	0	0	1	1	0	1	0	0	0	0	1
d	0	0	0	0	1	1	1	1	1	0	0
e	0	0	0	0	0	0	1	1	1	1	1



Adjacency matrix

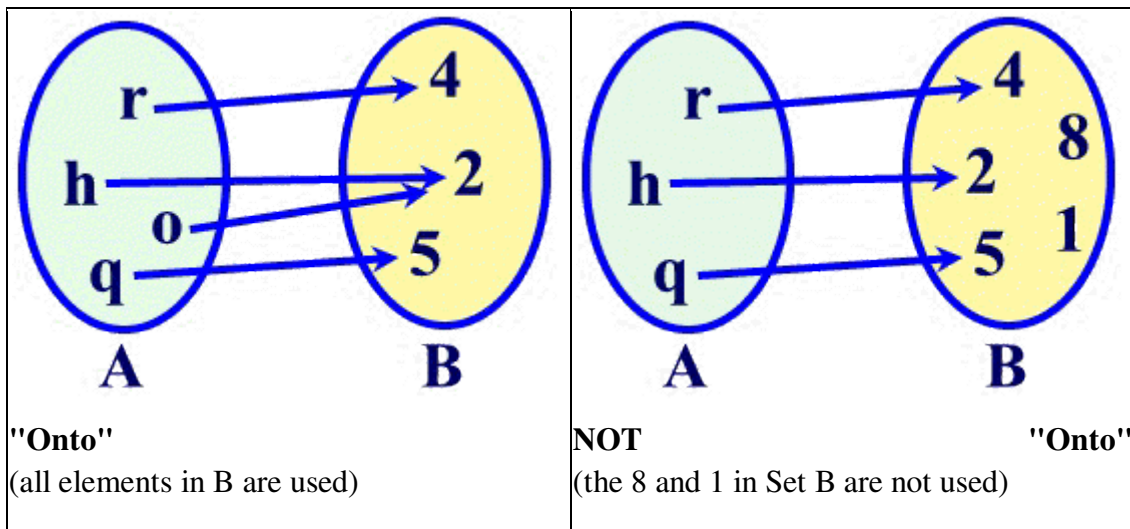
	a	b	c	d	e	f	g	h
a	0	0	1	0	0	0	0	0
b	1	1	1	1	0	0	0	0
c	0	0	0	1	1	0	0	0
d	0	0	0	1	1	0	0	2
e	0	0	0	0	1	1	0	0
f	0	0	0	0	0	0	1	0
g	0	0	0	0	0	1	0	0
h	0	0	0	0	0	2	1	0

Isomorphism of Graphs

Two simple graphs $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ are isomorphic if there is a **one-to-one and onto function** f from V_1 to V_2 with the property that a and b are adjacent in G_1 if and only if $f(a)$ and $f(b)$ are adjacent in G_2 , for all a and b in V_1 . Such a function is called isomorphism.

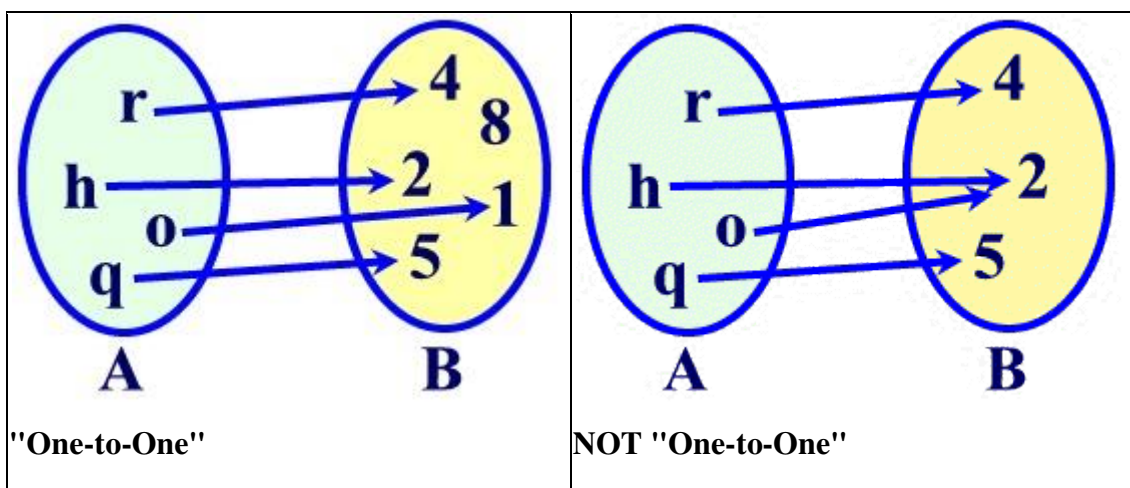
A function f from $\mathbf{A}$ to $\mathbf{B}$ is called **one-to-one** (or 1-1) if whenever $f(a) = f(b)$ then $a = b$. No element of $\mathbf{B}$ is the image of more than one element in $\mathbf{A}$.

In a one-to-one function, given any y there is only one x that can be paired with the given y . Such functions are referred to as injective.

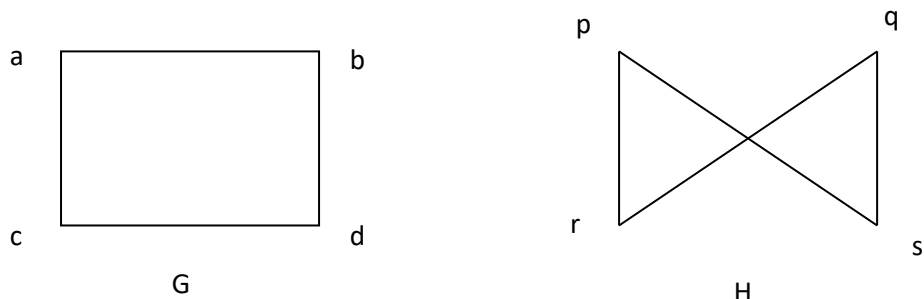


A function f from A to B is called **onto** if for all b in B there is an a in A such that $f(a) = b$. All elements in B are used.

Such functions are referred to as **surjective**.



E.g. : $G = (V, E)$ and $H(W, F)$

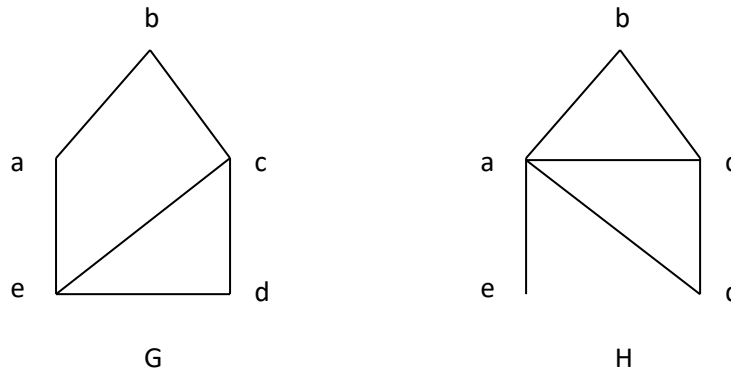


The function f with $f(a) = p$, $f(b) = s$, $f(c) = r$, $f(d) = q$ is a **one to one** correspondence between V and W .

We can show two simple graphs are not isomorphic by showing they don't share a property is called invariant. These properties are described as follows:

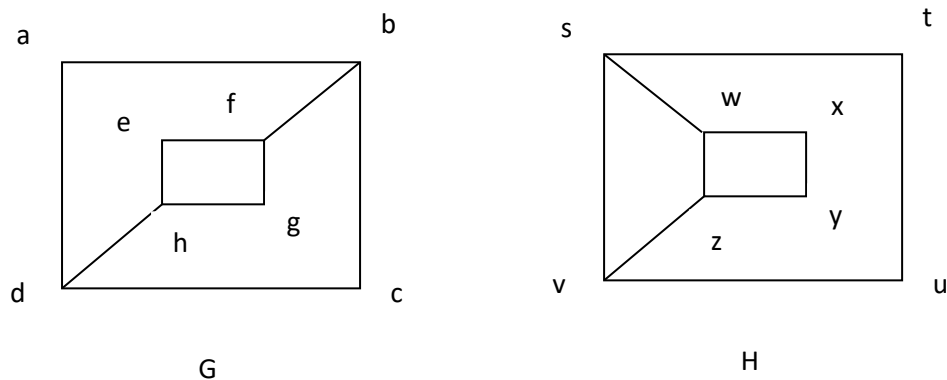
- 1) Isomorphic simple graphs must have the same number of vertices.
- 2) Isomorphic simple graphs must have same number of edges.
- 3) The degree of vertices must be same because the number of edges from the vertex is determined by degree.

Example 1 : show that the graphs displayed below are not isomorphic

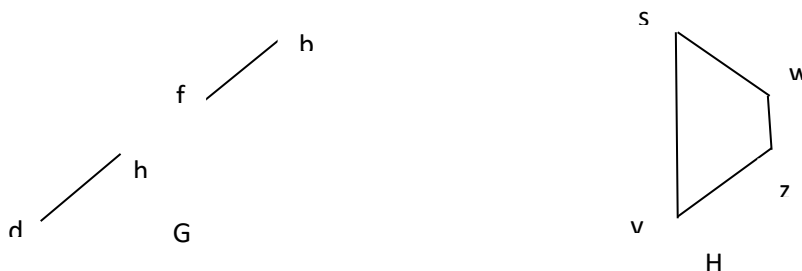


- Both graphs have 5 vertices and six edges. But H has a vertex of degree one, namely e, whereas G has no vertices of degree one. Hence G and H are not isomorphic.

Example 2 :

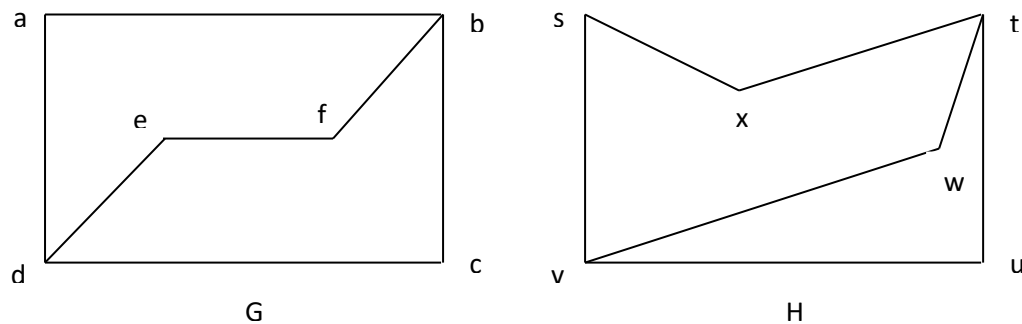


- Both graphs have eight vertices and ten edges. Also, they both have four vertices of degree two and four vertices of degree three. However, G and H are not isomorphic. Since, $\deg(a)=2$ in G, a must correspond to either t, u, x or y in H because these are the vertices of degree two in H. Here, each of these four vertices in H is adjacent to another vertex of degree two in H, which is not true for a in G.
- Another way to see that two graphs are not isomorphic is to note that subgraphs formed by connecting the edges from the vertex with same degree in both the graphs are not isomorphic. For e.g.: subgraphs of G and H made up of vertices of degree three and the edges connecting them are not isomorphic.



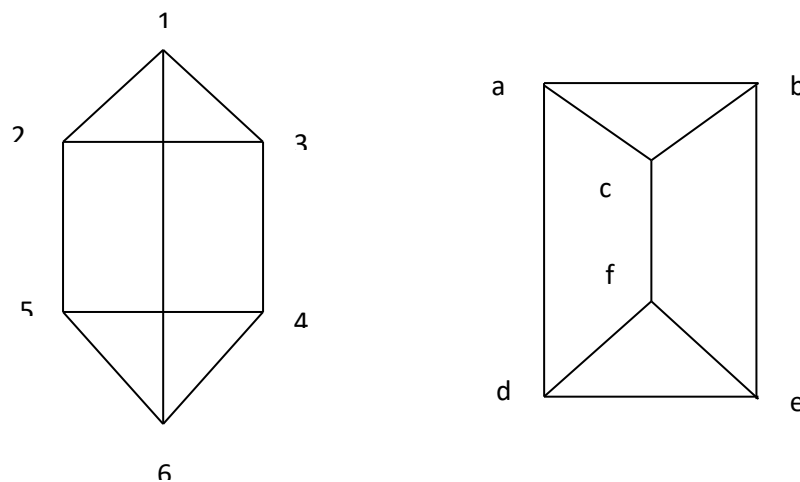
- To show isomorphism of graphs, we can also use adjacency matrix. For this, adjacency matrices of two graphs are same.

Example : 3 Determine whether the graphs are isomorphic



- Both graphs have six vertices and seven edges. Both have four vertices of degree two and two vertices of degree three. Also, Subgraphs are isomorphic. Here, we cannot say these graphs are isomorphic or not. For this, we can use adjacency matrix with the order of vertices a, b, c, d, e, f and w, t, u, v, s, x. If we can see similar matrices after drawing adjacency matrices. Hence, these graphs are isomorphic.

Example 4 : Determine whether given two graphs are isomorphic or not?

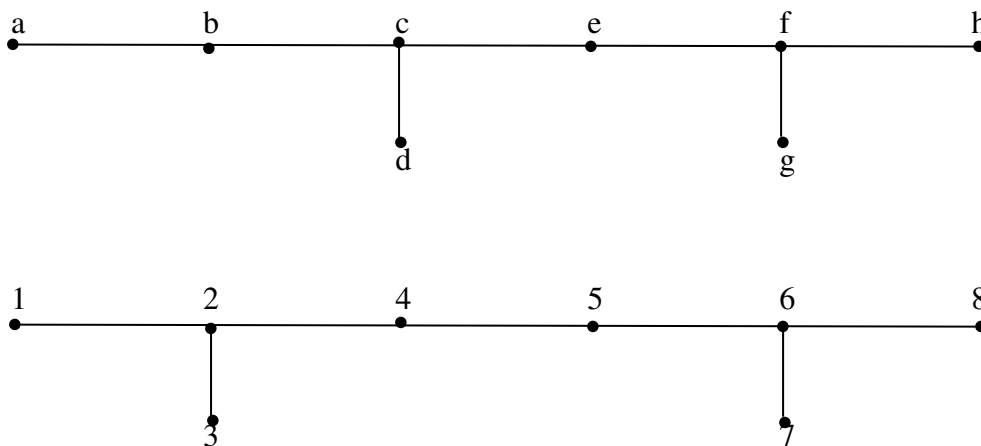


- In above graphs, no. of vertices is same, no. of edges are same and all vertices in both the graphs have degree 3.

Since the invariants agree on both the graphs, we can try out to find the function that is isomorphism. Take the sequence of vertices from the first graph as 1, 2, 3, 4, 5 and 6. Now, define $f(1) = c$, $f(2) = a$ here, there is adjacency preservation since we have $\{1, 2\}$ as an edge in the first graph where as $\{f(1), f(2)\} = \{c, a\}$ is an edge in second graph. Similarly, we can assign $f(3) = b$, $f(4) = e$, $f(5) = d$, $f(6) = f$. Since we found one to one correspondence between vertices of two graphs preserving the adjacency, above graphs are isomorphic. Here, the

adjacency matrices of two graphs in which vertices are ordered in terms of function i.e, 1, 2, 3, 4, 5 and 6 for first graph and c, a, b, e, d and f in the second graph are same.

one to one correspondence between vertices of two graphs preserving the adjacency, above graphs are isomorphic. Here, the adjacency matrices of two graphs in which vertices are ordered in terms of function i.e, 1, 2, 3, 4, 5 and 6 for first graph and c, a, b, e, d and f in the second graph are same. **Example 5 :**



- Both graphs have 8 vertices, 7 edges in both graphs two vertices have degree 3, 4 vertices have degree 1 and remaining 2 vertices have degree 2. Since, the invariants agree in both the graphs, we can continue to get the function such that it is isomorphism.

However, in case of first graph, the subgraph containing vertex with degree 3 is not isomorphic with any graph formed by connecting edges with vertex 2 or 6 (both of degree 3). Hence, two graphs are not isomorphic.

Walk, Paths, Circuit

Walk:

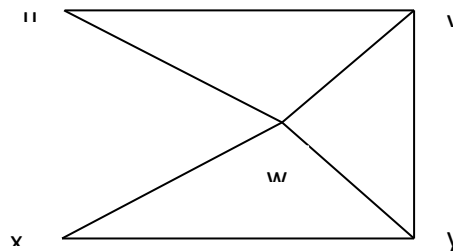
A walk is defined as a finite alternative sequence of vertices and edges of the form.

$$v_i e_j, v_{i+1} e_{j+1}, v_{i+2} e_{j+2} \dots \dots \dots e_k v_m.$$

Which begins and ends with vertices, such that

i) each edge in the sequence is incident on the vertices preceding and following it in the sequence.

A walk that begins and ends at the same vertex is called a closed walk . A walk that is not closed is an open walk. If u be initial vertex, v be terminal vertex. If $u=v$, then walk w is closed and $u \neq v$ it is open . As we move from one vertex of w to next, we are actually encountering or traversing edges of G , Possibly traversing some edges of G more than once. The number of edges encountered in a walk (including multiple occurrence of an edge) is called the length of the walk.



$w: x, y, w, y, v, w$ is therefore a walk. Length of walk is the no. of edges. Here 5 is the length.

A walk of length 0 is a trivial walk.

- We define a u - v **trail** in a graph G to be a u - v walk in which **no edge** is traversal more than once.

Since $w: x, y, w, y, v, w$ is not an x - w trail as the edge wy is repeated.

$T: u, w, y, x, w, v$ is a u - v trail.

Trail T repeats the vertex w .

- A u - v walk is a graph in which **no vertices** are repeated in u - v **path**.

$P: u, w, y, v$ is a u - v **path**.

If no vertex in a walk is repeated then no edge is repeated either. Hence, every path is a trail.

- A circuit in graph G is a closed trail of length 3 or more. Hence, a circuit begins and ends at the same vertex but repeats no edges. In a circuit, vertices can be repeated in addition to the first and last.

For example:

$c : y, w, u, v, w, x, y$

$e: x, y, w, u, v, w, x$

$c: w, x, y, w, u, v, w$

is a circuit.

Cycle

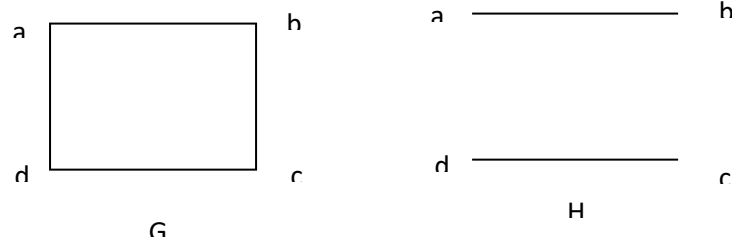
A circuit that repeats no vertex except for the first and last is a cycle. A K -cycle is a cycle of length K . A 3-cycle is also referred to as a triangle. A cycle of odd length is called odd cycle, a cycle of even length is called as even cycle.

For example:

$C' = x, y, v, w, x$ is a cycle.

Connectedness

- An undirected graph is called connected if there is a path between every pair of distinct vertices of the graph.
- There is a simple path between every pair of distinct vertices of a connected undirected graph.



Graph G is connected but graph H is not connected.

- A graph that is not connected is the union of more than one connected graphs that do not share the common vertex. These disjoint connected sub graphs are called connected components of a graph.

For example:

What are the connected components of the following graph?



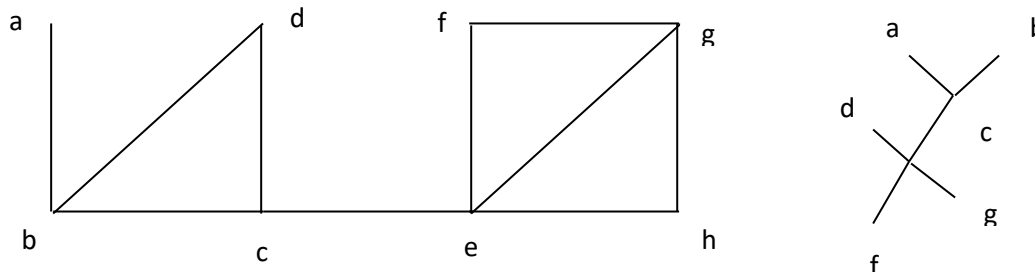
connected components are {a, b, c}, {d, e}, {f, g, h}

Cut Edges and Vertices

If one can remove a vertex (and all incident edges) and produce a graph with more connected components, the vertex is called a cut vertex.

If removal of an edge creates more connected component, the edge is called a cut edge or bridge.

Q. Find the cut vertices and cut edges in the following graph:



Cut vertices : c and e

Cut edge ; {c, e}

Vertex V is cut vertex of G iff $\deg v > 2$

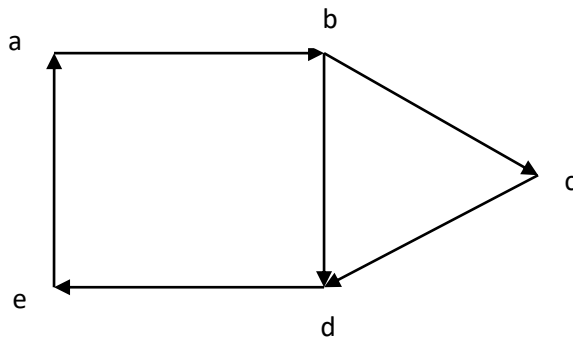
Connectedness in Directed graphs

A directed graph is strongly connected if there is a directed path between every pair of vertices.

A directed graph is weakly connected if there is a path between every pair of vertices in the underlying undirected graph.

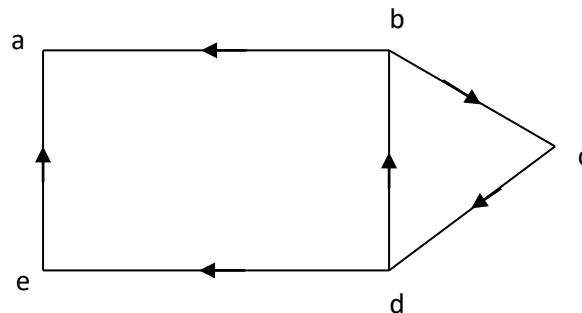
For example:

Q.N. Is the following graph strongly connected?



The graph is strongly connected because there is a directed path between every pair of vertices. If a directed graph is strongly connected, then it must also be weakly connected.

Q2. Is the following graph strongly connected?



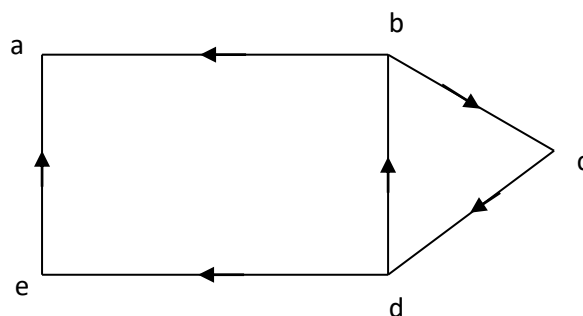
This graph is not strongly connected because there is no directed path between a and b, a and e etc.

However, it is weakly connected (imagine the graph as an undirected graph)

The subgraphs of a directed graph of G that are strongly connected but not contained in larger strongly connected subgraphs (the maximal strongly connected subgraphs) are called the strongly connected components or strong components of G .

For example:

What are strongly connected components of following graphs:



This graph has three strongly connected components

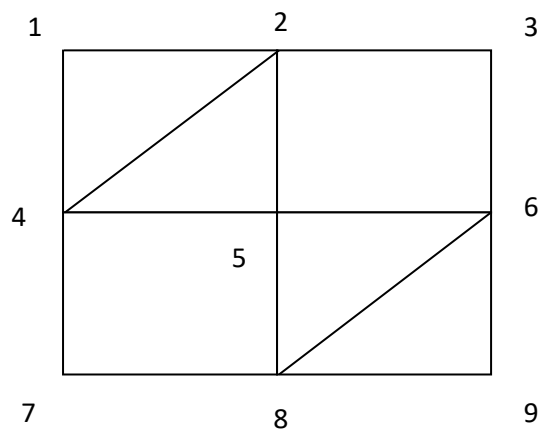
- The vertex a
- The vertex e
- The graph consists of
 $V = \{b, c, d\}$ and
 $E = \{\{b, c\}, \{c, d\}, \{d, b\}\}$

Euler path and circuits

A circuit C in a graph G is called Eulerian circuits if C contains every edge of G . Since no edge is repeated in a circuit, every edge appears exactly once in Eulerian Circuits.

A connected graph that contains as Eulerian circuit is called Eulerian graph.

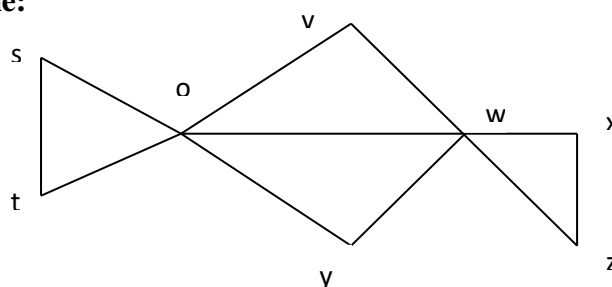
For example:



Euler circuit is 1, 2, 3, 6, 9, 8, 7, 4, 5, 8, 6, 5, 2

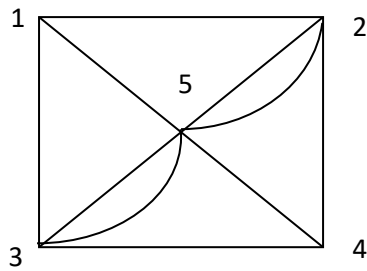
For connected graph G , we refer to an open trail that contains every edge of G as an Eulerian trail

For example:



$T = u, s, t, u, v, w, y, u, w, x, z, w$

An euler path is a path using every edge the graph exactly once.



No Euler circuit exist

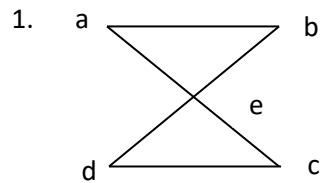
Necessary and sufficient conditions

A connected multigraph has a euler circuit iff each of its vertices has an even degree.

A connected multigraph has a eulerian trail iff exactly two vertices of G have odd degree .

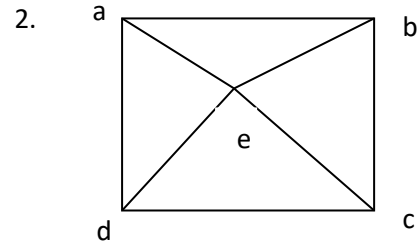
For example:

Q. Which of the following graphs is a eularian circuit?

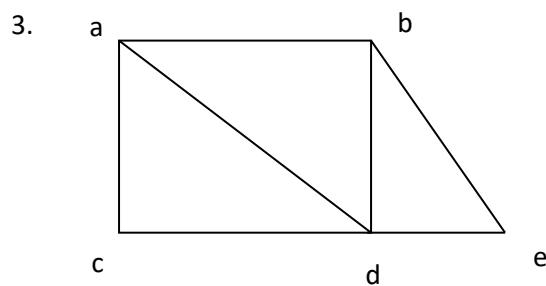


Yes

(a, e, c, d, e, b, a)



No

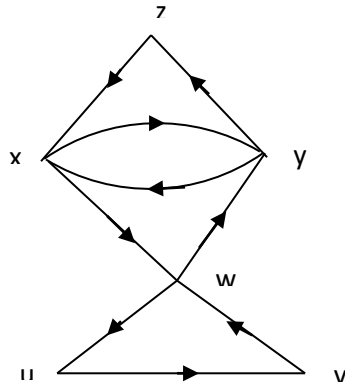


No

Euler trail : (a, c, d, e, b, d, a, b)

Eulerian for Directed graph

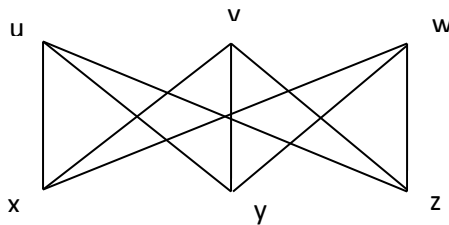
A non trivial connected digraph D is eulerian if and only if $\text{od } v = \text{id } v$ for every vertex



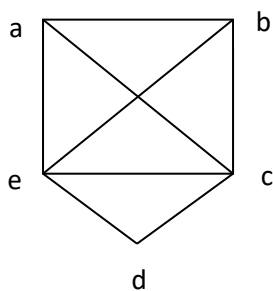
$C: u, v, w, y, z, x, y, x, w, u$, is eulerian circuit.

Hamilton Paths and Circuits

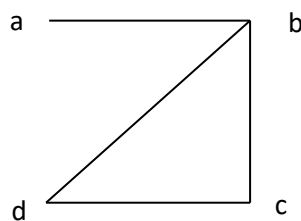
A cycle in graph G that contains every vertex of G is called a Hamiltonian cycle. Certainly a graph G ($n \geq 3$) is Hamiltonian path in G . If a graph contains Hamiltonian cycle, it contains Hamiltonian path.



$c: u, x, v, y, w, z, u$ is a Hamiltonian cycle of G



(a, b, c, d, e, a)



(a, b, c, d)

Hamilton circuit

For Hamilton circuit

No vertex of degree 1

If a node has degree 2, then both edges incident to it must be in any Hamilton circuit incident to it must be in any Hamilton circuit.

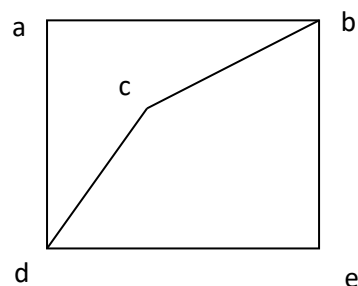
No smaller circuits contained in any Hamilton circuit (the start/end point of any smaller circuit would have to be visited twice).

A sufficient condition

Let G be a connected simple graph with n vertices with $n \geq 3$.

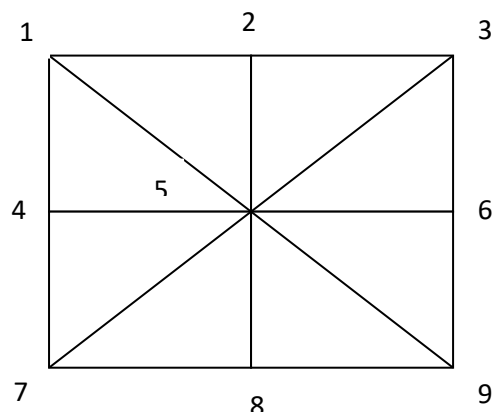
G has Hamilton circuit if the degree of each vertex is $\geq n/2$.

For example 1:



In this graph, there is no hamilton circuit since node has degree 2 and both edges from it must be in Hamilton circuit which is not possible one of the hamilton path is a, b, c, d, e.

Example 2 :



In graph, we can have hamilton circuit, the circuit can be 1, 2, 3, 5, 6, 9, 8, 7, 4, 1. Since, there is circuit. We can have path also.

Dirac's theorem:

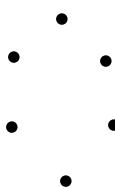
If G is a simple graph with n vertices with $n \geq 3$ such that the degree of every vertex in G is at least $\frac{n}{2}$, then G has a hamilton circuit.

Ore's Theorem:

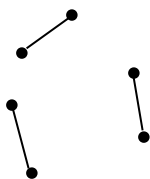
If G is a simple graph with n vertices with $n \geq 3$ such that $\deg(u) + \deg(v) \geq n$ for every pair of non adjacent vertices u and v in G , then G has a Hamilton path.

Regular Graphs:

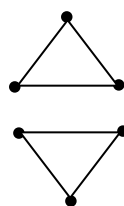
In graph theory, a regular graph is a graph where each vertex has the same number of neighbors i.e., every vertex has the same degree. A regular graph with vertices of degree k is called a k -regular graph or regular graph of degree k . Regular graph of degree at most 2 are easy to classify. A 0-regular graph consists of disconnected vertices a 1-regular graph consists of disconnected edges, and a 2-regular graph consists of disconnected cycles. A 3-regular graph is known as cubic graph.



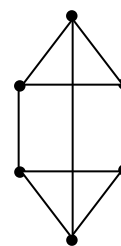
0-regular graph



1- regular graph



2-regular graph



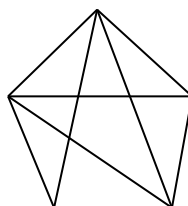
3-regular graph

Planner graphs

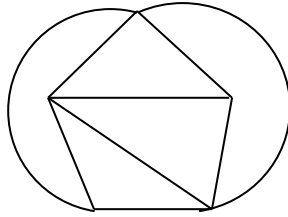
A graph is called a planar if it can be drawn in the plane without any edges crossing such drawing is called a planar representation of the graph.

For example:

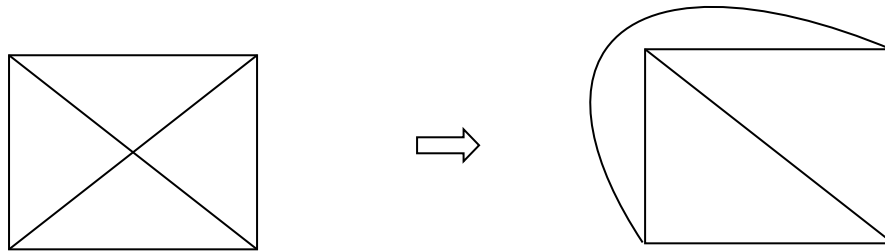
Draw the graph below as planar representation of the graph.



Solution,



2.

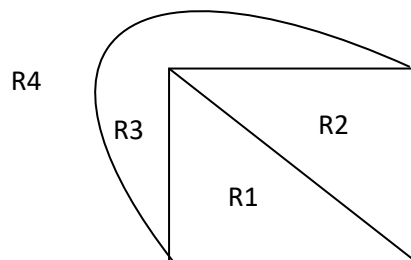


Not all graphs are planar

It may be difficult to show that graph is non-planar. We would have to show that there is no way to draw the graph without any edges crossing.

Regions

Euler showed that all planar representations of a graph split the plane into the same number of regions, including on unbounded region.



Any planar representation of $K_{3,3}$ V_1 and V_2 connected to V_4 and V_5 .

The four edges $\{V_1, V_4\}, \{V_4, V_2\}, \{V_2, V_5\}, \{V_5, V_1\}$ form a closed curve that splits the plane into two regions R_1 and R_2 .

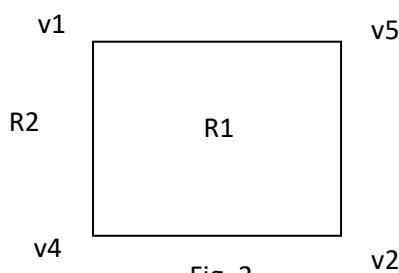


Fig. 2

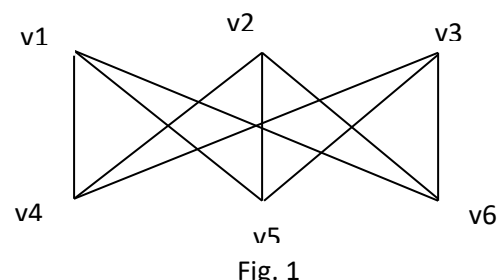
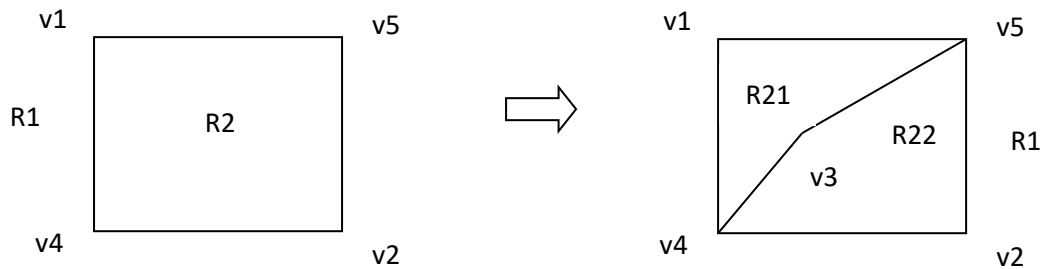


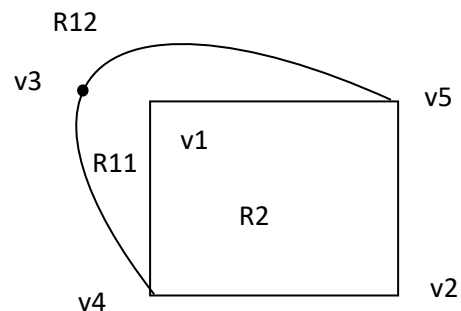
Fig. 1

Next, V_3 must be either in R_1 or R_2 . Assume V_3 is in R_2 . Then edges (V_3, V_4) and (V_3, V_5) separate R_2 into two sub regions R_{21} and R_{22} .



Now, there is no way to place vertex V_6 without forcing a crossing.

- If V_6 is in R_1 , then (V_6, V_3) must cross an edge.
- If V_6 is in R_{21} , then $\{V_6, V_2\}$ must cross an edge.
- If V_6 is in R_{22} , then $\{V_6, V_1\}$ must cross an edge.
- Alternatively assume V_3 in R_1 . Then the edge (V_3, V_4) and (V_3, V_5) separate R_1 into two sub-regions R_{11} and R_{12} .



Now there is no way to place vertex V_6 without forcing a crossing.

- If V_6 is in R_2 , then $\{V_6, V_3\}$ must cross an edge.
- If V_6 is in R_{11} , then $\{V_6, V_2\}$ must cross an edge.
- If V_6 is in R_{12} , then $\{V_6, V_1\}$ must cross an edge.

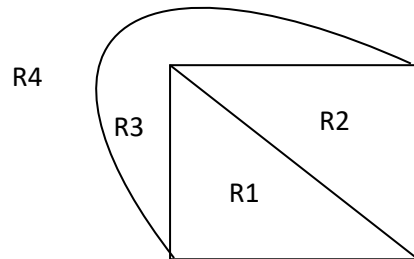
Consequently, the graph $K_{3,3}$ must be non-planar.

Euler devised a formula for expressing the relationship between the number of vertices, edges and regions of a planar graph.

- These may help to determine if a graph can be planar or not.

Euler's Formula:

→ Let G be a connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planar representation of G . Then $r = e - v + 2$



no. of edge, $e = 6$

no. of vertices, $v = 4$

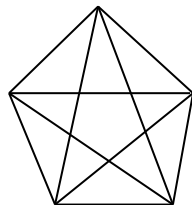
no. of regions, $r = e - v + 2$

$$= 4$$

Corollary 1: If G is a connected planar simple graph with e edges and v vertices where $v \geq 3$, then $e \leq 3v - 6$

For example:

Is K_5 planar?



K_5

⇒ K_5 has 5 vertices and 10 edges.

Here, $V \geq 3$

So, if K_5 is planar, it must be true that

$$e \leq 3v - 6$$

$$3v - 6 = 3 \times 5 - 6 = 15 - 6 = 9$$

Thus, e must be less than or equal to 9

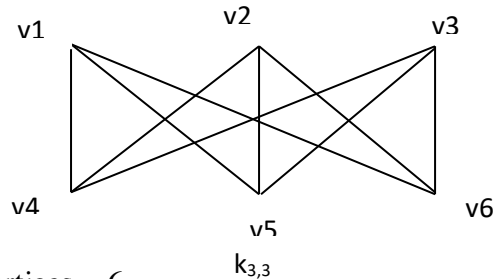
But $e = 10$

Therefore K_5 is non-planar.

Corollary 2 : If G is a connected planar simple graph, the G has a vertex of degree not exceeding 5.

Corollary 3: If a connected planar simple graph has e edges and v vertices with $v \geq 3$ and no circuits of length 3, then $e \leq 2V - 4$

For example: Is $K_{3,3}$ planar?



$K_{3,3}$ has vertices = 6

$K_{3,3}$ has edges = 9

Here, $v \geq 3$ and there is no circuits of length 3, If $K_{3,3}$ were planar, then $e \leq 2v - 4$ would be true.

$$2V - 4$$

$$= 2 \times 6 - 4$$

$$= 8$$

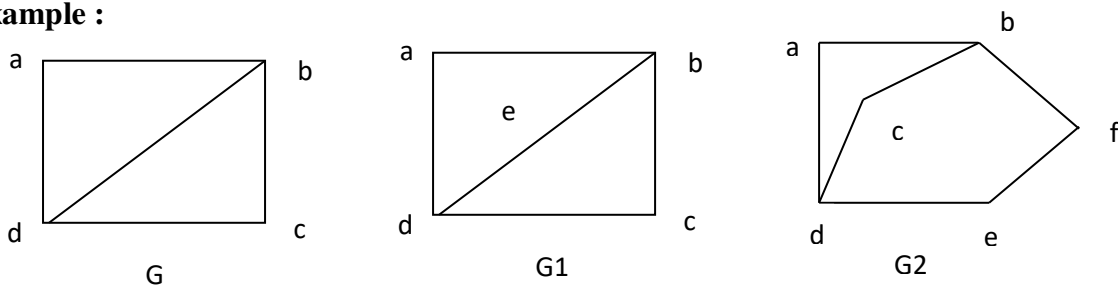
So, e must be ≤ 8

But $e = 9$

Therefore $K_{3,3}$ is a non-planar.

$\Rightarrow$ If a graph is planar, so will be any graph obtained by removing an edge $\{u,v\}$ and adding a new vertex w together with edges $\{u,w\}$ and $\{w,v\}$. such an operation is called an elementary sub-division. The graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called homeomorphic if they can be obtained from the same graph by a sequence of elementary sub division.

For example :

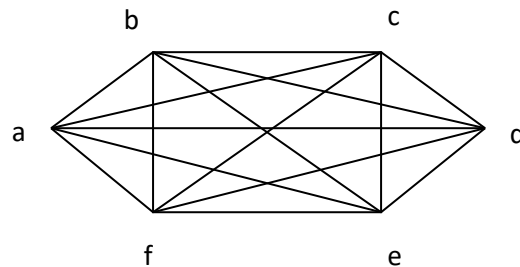


Graphs G_1 and G_2 are homeomorphic to the graph

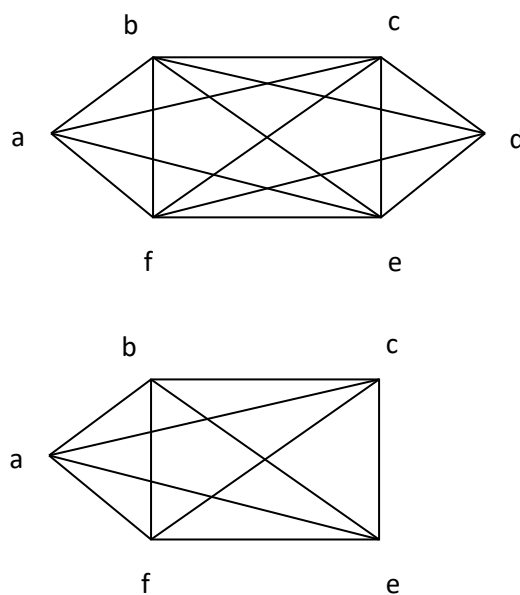
Kuratowski's Theorem (Planarity Testing Algorithm)

A graph is non-planar if and only if it contains a sub graph homeomorphic to $K_{3,3}$ or K_5 .

For example:



Solution



Above graph is homeomorphic to K_5 , the given graph is not planar.

Weighted Graphs:

Graphs that have a number assigned to each edge (u, v) has a weight (u, v) . Each weight is a real number.

Weight can represent distance, cost, time, capacity etc.

The length of a path in a weighted graph is the sum of the weights on the edges.

Dijkstra's Algorithm finds the shortest path between two vertices.

A shortest –path Algorithm

There are several algorithm that find a shortest path between two vertices in a weighted graph.

Dijkstra's Algorithm

Procedure Dijkstra (G: weighted connected simple graph with all weights positive)

G has vertices $a=v_0, v_1, \dots, v_n=2$ and weighs (v_i, v_j) is not an edge in G.

For $i=1$ to x

$L(v_i) = \infty$

$L(a) = 0$

$s ::= \emptyset$

{ the labels are now initialized so that the label of a is 0 and all other labels are ∞ and s is an empty set }

While $z \notin s$

Begin

u: a vertex not in s with $L(u)$ minima

$s: s \cup \{u\}$

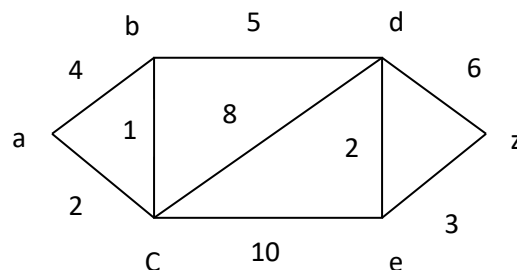
for all vertices u not in S.

if $L(u) + w(u, v) < L(v)$ then $L(v) := L(u) + w(u, v)$

{ this adds a vertex to s with minimal label and updates the labels of vertices not in s }

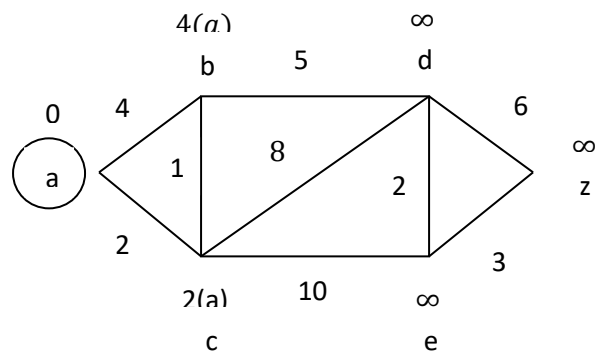
End { $L(z) = \{ \text{length of a shorter path from a to z} \}$ }

QN.1. Use Dijkstra's algorithm to find the shortest path from the vertices a to z in the following weighted graph given below.

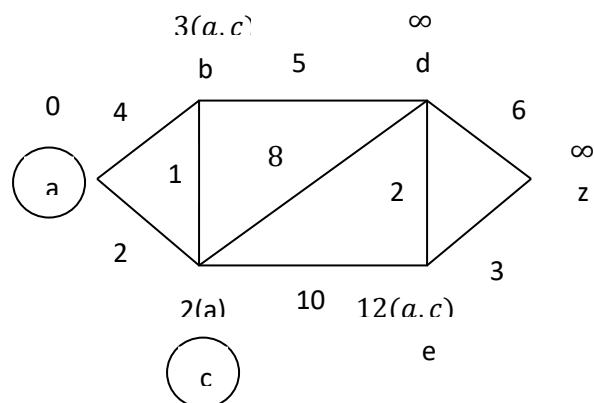


1)

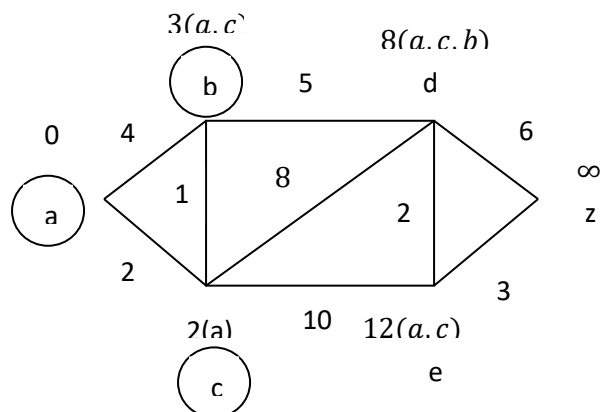
2)



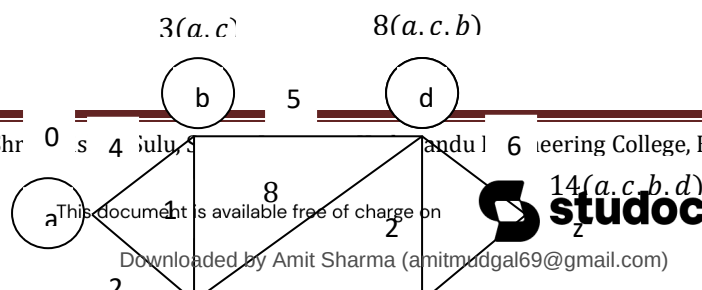
3)



4)

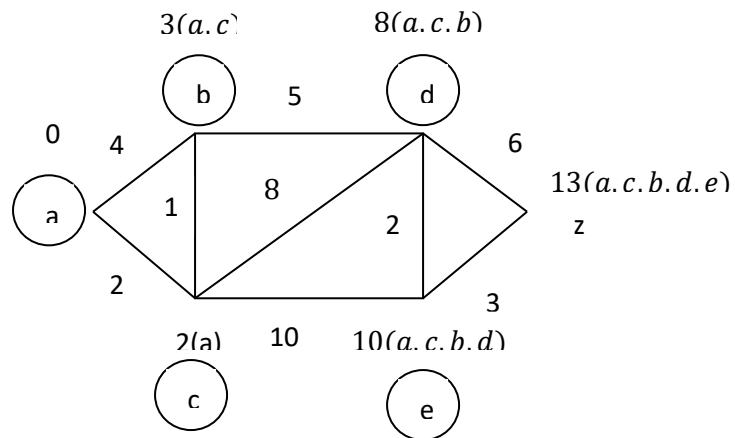


5)

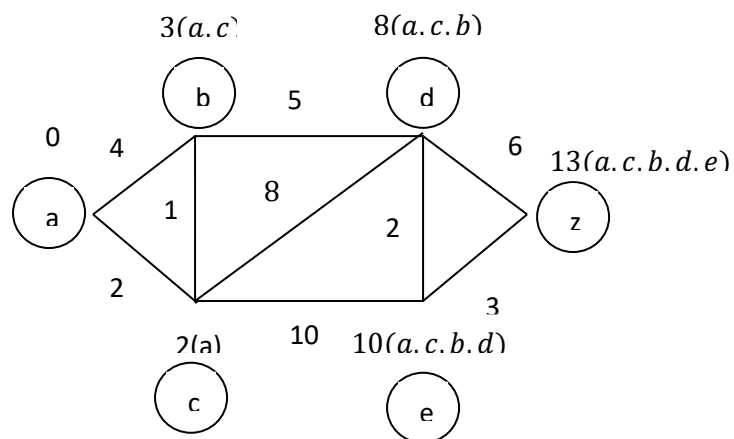


10

6)



7)



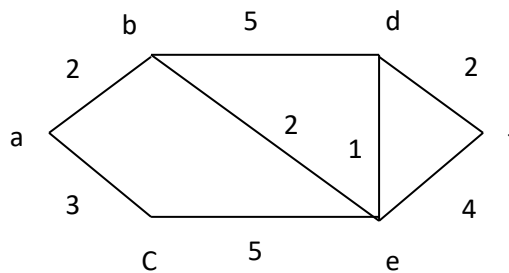
$w(u,v)$ a b c d e z

a	0	4	2	∞	∞	∞
b	4	0	1	5	∞	∞
c	2	1	0	8	10	∞
d	∞	5	8	0	2	6
e	∞	∞	10	2	0	3
2	∞	∞	∞	6	3	0

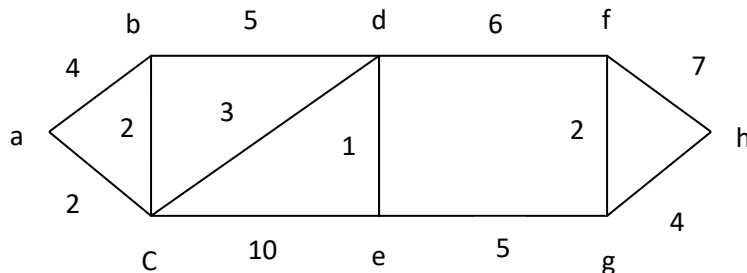
s	$\emptyset$	{a}	{a,c}	{a,c,b}	{a,c,b,d}	{a,c,b,d,e}	{a,c,b,d,e,z}
L(a)	0	0	0	0	0	0	0
L(b)	∞	4	3	3	3	3	3
L(c)	∞	2	2	2	2	2	2
L(d)	∞	∞	10	8	8	8	8
L(e)	∞	∞	12	12	10	10	10
L(f)	∞	∞	∞	∞	14	13	13

Hence, the shortest path from a to z has length $L(z)=13$

Q 2)



Q 3)



The travelling salesman problem

The travelling salesman is one of the classical problem in computer science.

A travelling salesman wants to visit a number of cities and then return to his strarring point of course he wants to save time and energy. So he wants to determine the shortest cycle for his trip.

We can represent the cities and the distances between them by a weighted, complete, undirected graph.

The problem then is to find the shortest cycle of minimal total weight that visits each vertex exactly one.

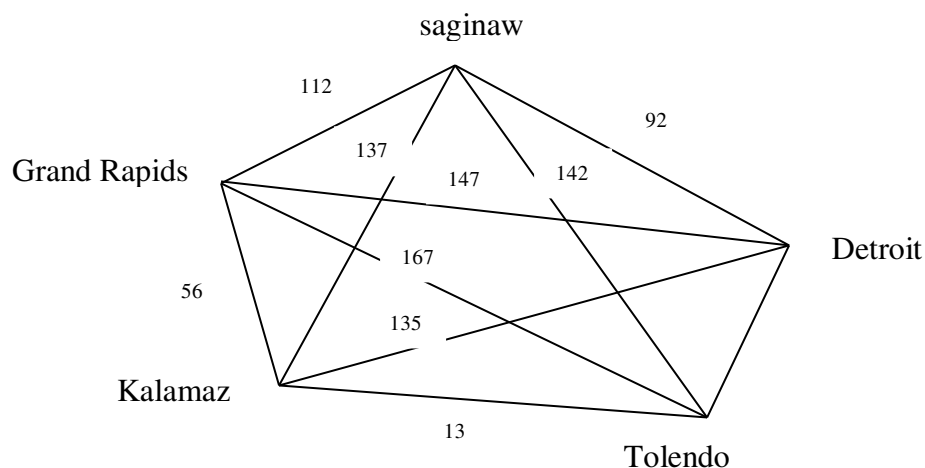
Importance

Variety of scheduling application can be solved as a traveling salesman problem.

Examples:

Ordering drill position on a drill press.

School bus routing



Route

- Detroit, Toledo, Grand- Rapids-saginaw-kala-mazoo- Detroit 610
- Detroit Toledo-Grand Rapids-kalamazoo-saginar-Detroit 546
- Detroit Toledo-Grand Rapids-kalamazoo-saginaw-Grand-Raphids-Detroit 588
- Detroit-Toledo-kalamazoo-Grand Rapids-saginaw-Detroit 458
- Detroit-Toledo-saginaw-kalamazoo-Grand Rapids-Detroit 540
- Detroit-Toledo-saginaw-Grand Rapids-kalamazoo-Detroit- 504
- Detroit-saginaw-Tolede-Grand Rapids-kalamazoo-Detroit-598
- Detroit-saginaw-Toledo-kalamazoo-Grand Rapids-Detroit-576
- Detroit-saginaw-kalamazoo-Toledo-Grand Rapids-Detroit-682
- Detroit-saginaw-Grand Rapids-Toledo-kalamazoo-Detroit-646

- Detroit-GrandRapids-saginaw-Toledo-kalamazoo-Detroit-670
- Detroit-GrandRapids-Toledo-saginaw-kalamazoo-Detroit- 728

Tree

A tree is a connected undirected simple graph with no simple circuit. A tree is a particular type of graph.

Properties

1. There is a unique simple path between any 2 of its vertices.
2. No loops
3. No multiple edges

For example:

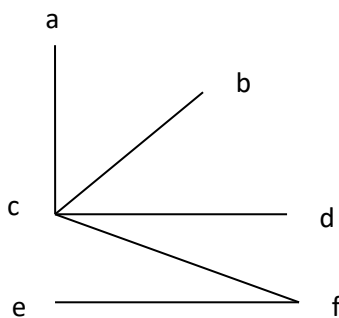


Fig: G1

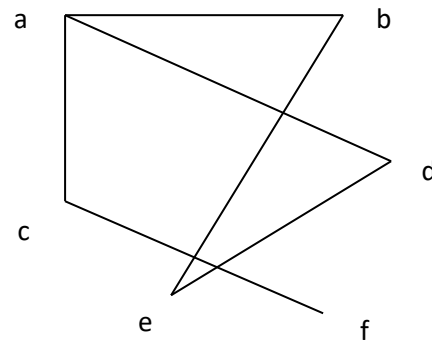


Fig: G2

G1: It is tree because it is connected graph with no simple circuit.

G2: It is not a free because there is a cycle a, b, e, d, a

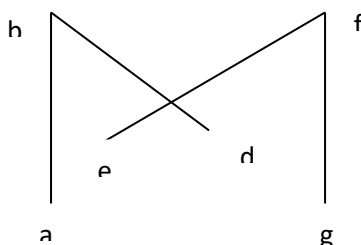


Fig : G3

An undirected graph having no simple circuit and is not connected is called forest. The forest has each of its connected components an tree G3 displays a forest.

Theorem1:

An undirected is a tree if and only if there is a unique simple path between any two of its vertices.

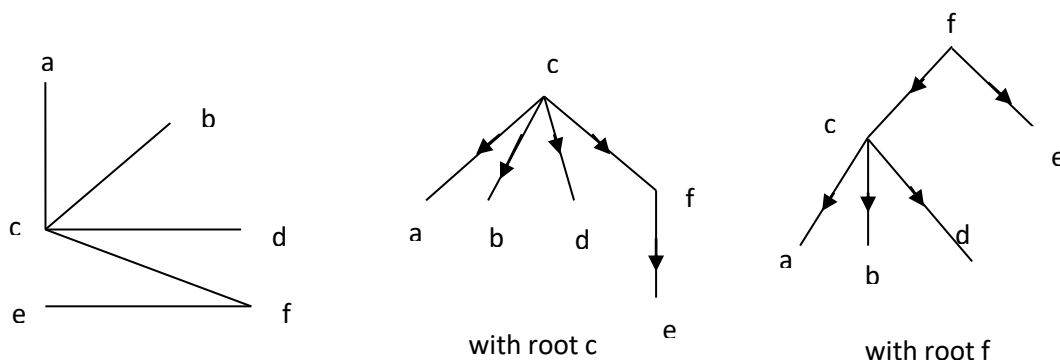
Proof:

Assume that T is a tree. Since T is a tree. It is a connected simple graph with no simple circuits. Let x and y be two vertices of T . We know that every connected graph has a simple path between every pair of vertices. So, there is a simple path from x to y . This path must be unique because, if the path between x and y is not unique then there is another path between x and y that uses edges different from the path, then reversing the path i.e, going to from x to y from first path and going from y to x through second path forms a circuit. This is a contradiction that is a tree. Hence, there is a unique simple path between any two vertices of a tree.

Rooted (Directed) Tree

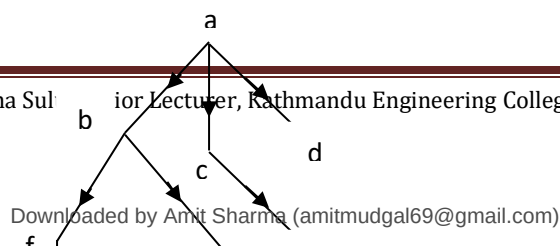
In many applications of trees, a particular vertex of a tree is designated as the root. A rooted tree is a tree in which one vertex has been designated as the root and every edge is directed away from the root.

We can change an unrooted tree into a rooted by choosing any vertex as the root. The tree in which root is defined produces a directed graph.



- Root :Vertex with in degree 0.
- In a rooted tree, if V is the vertex in T other than root, then the **Parent** of V is a vertex of u in T such that there is directed edge from u to v .
- In this scenario v is called **child** of u .
- Vertices with same parents are called **Siblings**
- All the vertices that appear in the path from root to some vertex v in T , including root are called **ancestors** of V .
- The **descendants** of a vertex V are those vertices that have V as their ancestor.
- All the vertices that have children are called internal vertices (root is also an internal vertex if the tree has more than one vertices).

For example:



Parent : b is parent of f and g.

Child : g and f are children of b

Siblings: f and g are siblings.

Ancestors : Ancestors of g are b, a.

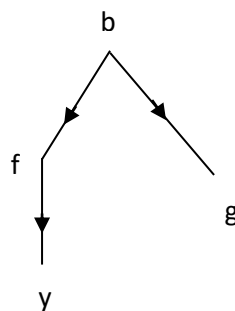
Descendents : Descendents of b : f, g, y

Leaf : Vertices with no children (y, g, e, d)

Internal vertices : a, b, c, f

Sub tree : Subgraphs consisting of V and its descendants and their incident edges

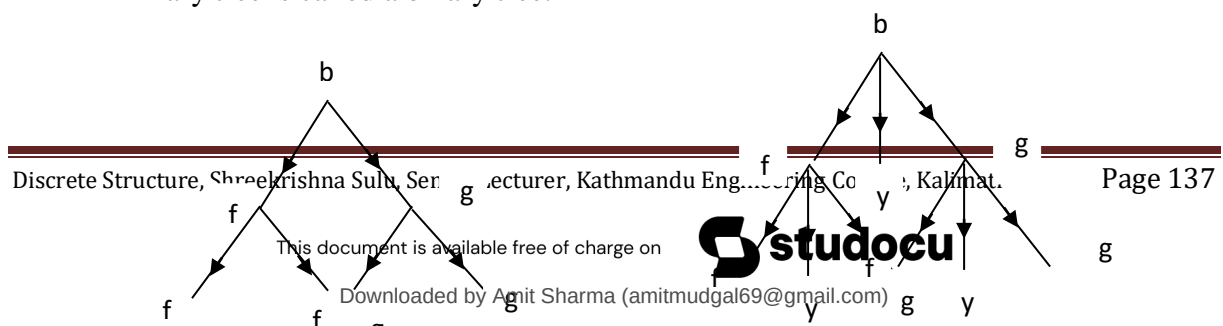
subtree rooted at b:



- Level (of v) is length of unique path from root to v (level of root = 0, level of b =1, level of g=2)
- Height is the maximum of vertices level (height : 3)

m-ary trees

- A rooted tree is called m-ary if every internal vertex has no more than m children.
- It is full m-ary if every internal vertex has exactly m children.
- A 2-ary tree is called a binary tree.



Full binary tree

Full 3-ary tree

Ordered Rooted Tree

- An rooted tree where the children of each internal node are ordered.
- In ordered binary tree, we can define :
 - left child, right child
 - left subtree, right subtree
- For m-ary trees with $m > 2$, we can use terms like "left most", "right most" etc.

Properties of trees:

Theorems 1 : A tree with n vertices has $n-1$ edges.

Proof :

Basis step: When $n=1$, a tree with $n=1$ vertex has no edge. It follows that theorem is true for $n=1$.

Inductive hypothesis : Assume that the tree with K vertices has $K-1$ edges, where k is positive integer.

Inductive steps: Suppose that a tree T has $K+1$ vertices and that V is a leaf of T . Removing vertex V and the associated edge from T produces a tree T_1 with K vertices, since the resulting graph is still connected and has no simple circuit. By the induction hypothesis, T_1 has $k-1$ edges. Hence T has K edges since it has one more edge than the edge connecting V to its parent.

Theorem 2 : A full m-ary tree with i -internal vertices contains $n=mi + 1$ vertices.

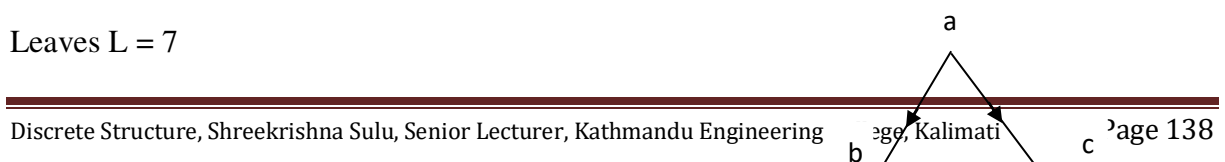
A level(depth) of a vertex v in a rooted tree is the length of the unique path from the root to vertex. The level of the root is zero. The height of the rooted tree is the length of the longest path from the root to any vertex. A rooted m-ary tree of height h is balanced if all leaves are at levels h or $h-1$.

For example:

Vertices = 13

Internal vertices (i) = 6

Leaves $L = 7$



$$13 = 6 \times 2 + 1$$

3) A full m-ary tree with

i) n vertices has $I = (n-1)/m$ internal vertices and

$$L = ((m-1) * n + 1)/m \text{ leaves}$$

ii) I internal vertices has $n = m * I + 1$ vertices

$$\text{and } L = (m-1) * I + 1 \text{ leaves}$$

iii) L leaves has $n = (m * L - 1)/(m-1)$ vertices and

$$I = (L-1)/(m-1) \text{ internal vertices.}$$

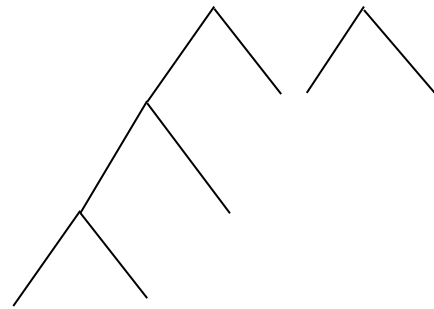


Fig: I

For example:

$m=2, n=13, I=6, L=7$ from fig (I)

$$\text{i. } I = (n-1)/m = (13-1)/2 = 6$$

$$\text{and } L = ((m-1) * n + 1)/m = ((2-1) * 13 + 1)/2 = 7$$

$$\text{ii. } n = m * I + 1 = 2 * 6 + 1 = 13$$

$$\text{and } L = (m-1) * I + 1 = (2-1) * 6 + 1 = 7$$

$$\text{iii. } n = (m * L - 1)/(m-1) = (2 * 7 - 1)/(2-1) = 13$$

$$\text{and } I = (L-1)/(m-1) = (7-1)/(2-1) = 6$$

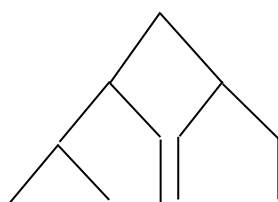
The level of a vertex in a rooted tree is the length of the path from the root to the vertex (level of the root is 0)

The height of the rooted tree is the maximum of the levels of vertices (length of the longest path from the root to any vertex).

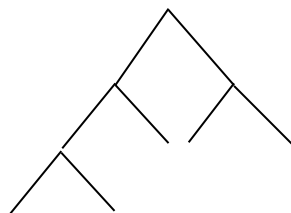
Balanced Tree:

A rooted m-ary tree of height h is balanced if all leaves are at levels h or h-1

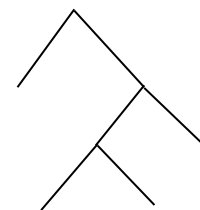
For example



Balanced



Balanced



Not Balanced

Applications of trees:

- How should items in a list be stored so that an item can be easily located? For this, we use the concept of binary search trees.
- What series of decisions should be made to find an object with a certain property in a collection of objects of a certain type? For this, we use the concepts of decision trees.
- How should a set of characters be efficiently coded by bit. Strings? Here, we use the concept of the prefix codes.

Binary Search Trees:

Searching for items in a list is one of the most important task that arises in computer science. Our primary goal is to implement a searching algorithm that finds items efficiently when the items are totally ordered. This can be accomplished through the used binary search tree.

A binary search tree (BST) is a binary tree in which each child of a vertex is designated as a right or left child, no vertex has more than one right child or left child and each vertex is labeled with key, which is one of the items. Further more vertices are assigned keys so that the key of a vertex is both larger than the keys of all vertices in its left sub tree and smaller than keys of all vertices in its right subtree.

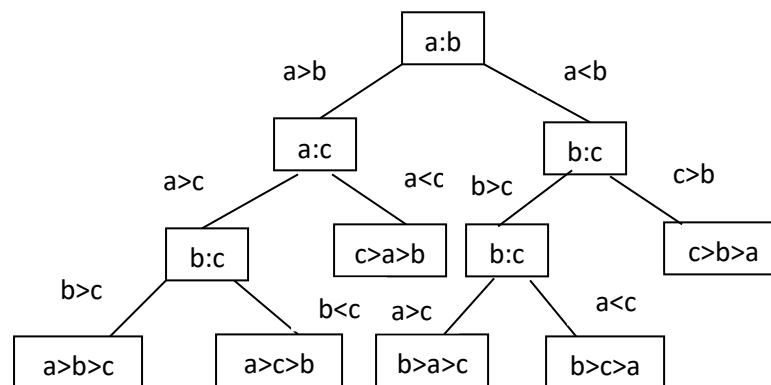
We use recursive procedure to form a binary search tree for a list of items. We start with a tree containing just one vertex, namely, the root. The first item in the list is assigned as the key of the root. To add a new item, first compare it with the keys of vertices already in the tree, starting at the root and moving to the left if the item is less than the key of respective vertex if this vertex has a left child or moving to the right if the item is greater than the key of the respective vertex if this vertex has a right child. When the item is less than respective vertex and this vertex has no left child, then a new vertex with this item as its key is inserted as a new left child. Similarly, when the item is greater than the respective vertex and this vertex has no right child, then a new vertex with this item as its key is inserted as a new right child.

For example:

Form a BST for the words mathematics, physics, geography, zoology, metrology, geology, psychology and chemistry.

Decision Trees:

Rooted trees can be used to model problems in which a series of decision leads to a solution. For instance, a binary search tree can be used to locate items based on a series of comparisons, where each comparison tell us whether we have located the item or whether we should go left or right in a substance. A rooted tree in which each internal vertex corresponds to a decision with a subtree at these vertices for each possible outcome of these vertices for each possible outcome of the decision is called a decision tree. The possible solution of the problem correspond to the paths to the leaves of the rooted tree.



A decision tree for sorting three distinct elements.

Prefix codes:

Consider a problem using bit string to encode the letters of the English alphabet. We can represent each letter with a bit string of length five as there are only 26 letters and there are 32 bit strings of length five.

Consider using bit strings of different length to encode letters. Letters that occur more frequently should be encoded using short bit strings. When letters are encoded using varying number of bits, some method must be used to determine where the bits of a each character start and end.

For instance, if we were encoded e with 0, a with 1 and t with 01, then the bit string 0110 could represent to eat or tae or bit string 0101 could correspond to eat, tea, eaea or tt.

One way to ensure that no bit string corresponds to more than one sequence of letters is to encode letters so that the bit string for a letter never occurs as the first part of the bit string for another letter codes with this property are called prefix codes.

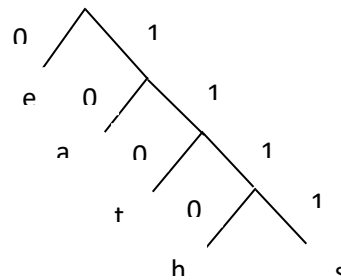
For example:

Encoding of e as 0, a as 10 and t as 11 is a prefix code. A word can be recovered from unique bit string that encodes its letters.

For example: 10110 is the encoding of ate.

A prefix code can be represented using binary tree, where the characters are the labels of the leaves of tree. The edge of tree are labeled so that an edge leading to a left child is assigned a 0 and an edge leading to a right child is assigned a 1.

We construct prefix code from any binary tree where the left edge at each internal vertex is labeled by 0 and right edge by 1 and leaves are labeled by characters.



The tree in the figure represents encoding of e by 0., a by 10 t by 110, m by 1110 and s by 1111.

For example: 11111011100

s a n e

Huffman coding

Here is an algorithm that takes input the frequencies of symbols in a string and produces as output a prefix code that encodes the string using fewest possible bits. This algorithm is called Huffman coding. It is used in data compression. The subject devoted to reducing no. of bits required to represent information.

Procedure

F: Forest of n rooted trees.

While F is not a tree.

begin:

Replace the rooted trees T and T' of least weight from F with $W(T) \geq W(T')$ with a tree having a new root that has T as its left subtree and T' as its right subtree. Label the new edge to T with 0 and new edge to T' with 1.

Again $W(T) + W(T')$ as weight of new tree end.

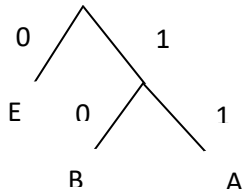
Use Huffman coding to encode following symbols with the frequencies listed what is the average no. of bits used to encode a character.

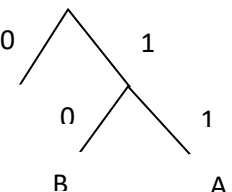
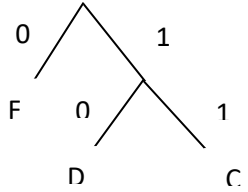
A=0.08, B=0.10, C=0.12, D=0.15, E=0.20, F=0.35

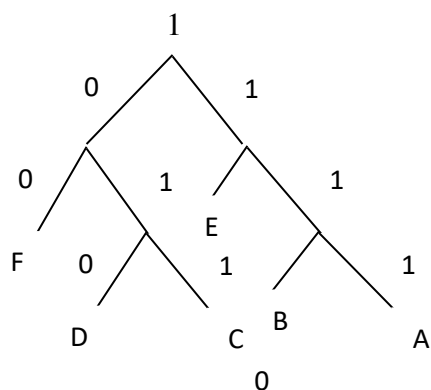
0.08	0.10	0.12	0.15	0.20	0.35
A	B	C	D	E	F initial Forest

0.12	0.15	0.18	0.20	0.35	
C	D		E	F	Step 1

0.18	0.20	0.27	0.35	
	E		F	Step 2

0.27	0.35	0.38	
	F		Step 3

0.38	0.62	
		Step 4



From the graph, we can encode A by 111, B by 110, C by 011, D by 010, E by 10 , and F by 00.

The average number of bits used to encode a symbol using this encoding is

$P_k l_k$ (P_k : Probability, l_k : length)

$$= 0.08 \times 3 + 0.10 \times 3 + 0.12 \times 3 + 0.15 \times 3 + 0.2 \times 2 + 0.35 \times 2$$

$$= 2.45$$

Spanning Trees:

Let G be a simple graph. A spanning tree is a sub graph of G that is a tree containing every vertex of G .

Consider a system of roads in Maine. The only way the roads can be kept open in the winter is by frequently plowing them. The highway department wants to plow the fewest roads so that there will always be cleared roads connecting any two towns. At least 5 roads must be plowed to ensure that there is a path between any two towns. Fig (b) represents one such set of roads. Note that the subgraph representing these roads is a tree because it is connected and contains six vertices and five edges.

This problem was solved with a connected subgraph with the minimum number of edges containing all vertices of the original simple graph such a graph must be a tree.

Example: 1

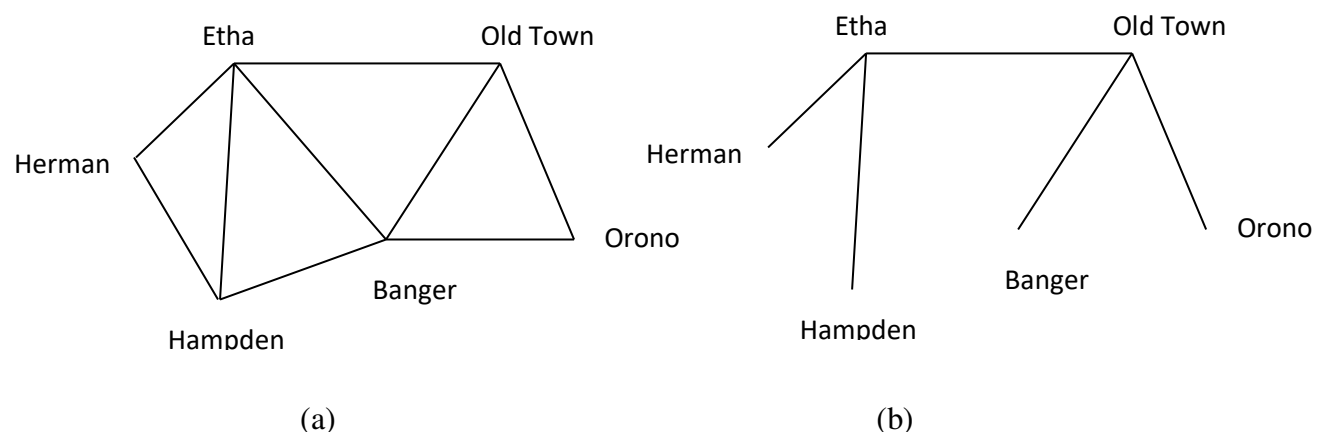


Fig (a) : Road system and (b) sets of roads to plow

Example: 2

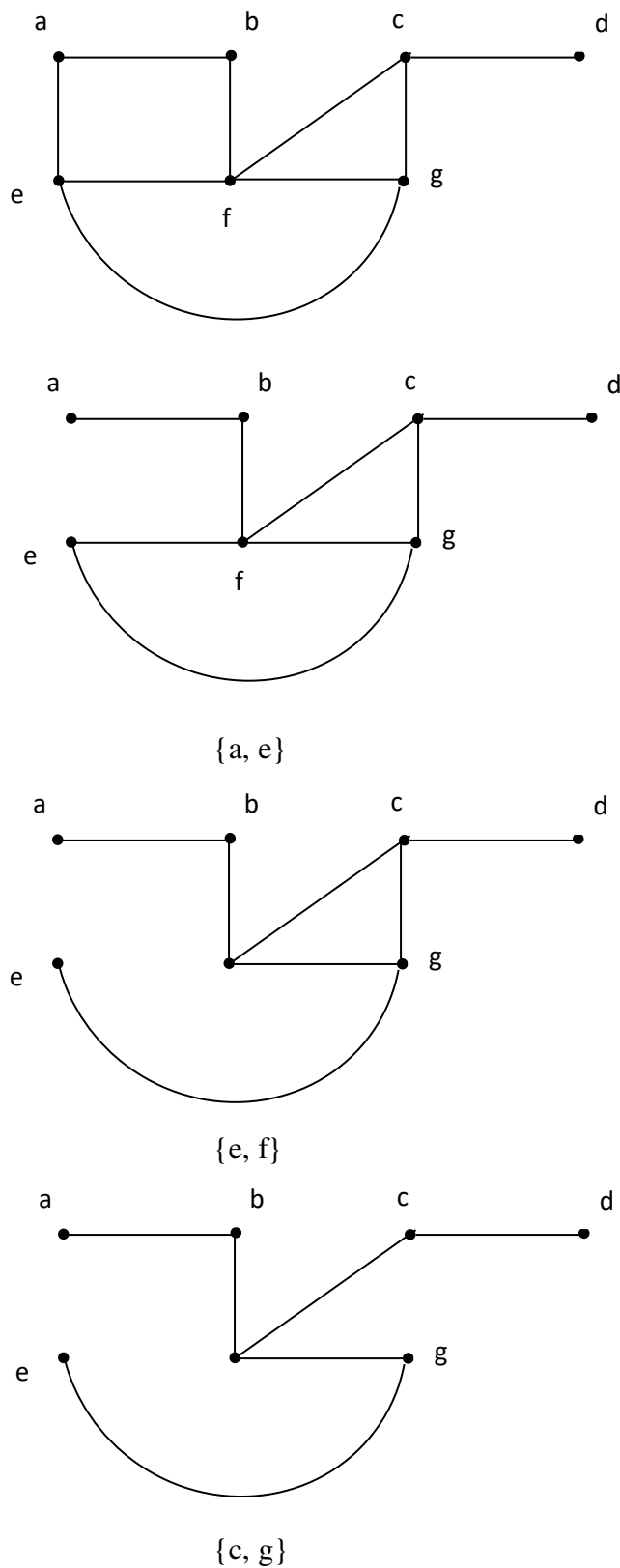
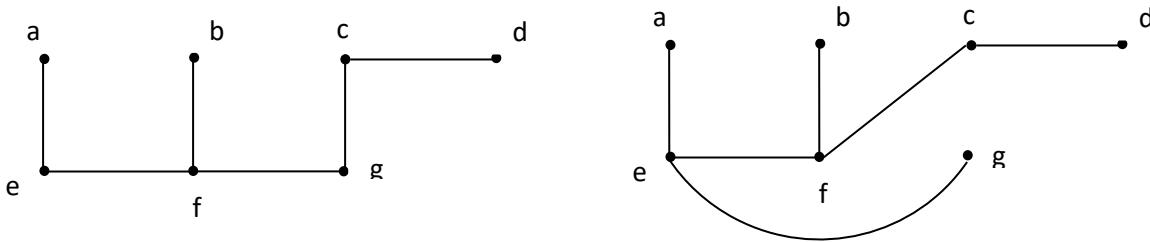


Fig : Producing spanning tree for G by removing edges that form simple circuits.

Theorem 1: A simple graph is connected if and only if it has a spanning tree.

Proof:

Firstly let a simple graph G has a spanning tree T . T contains every vertex of G . Therefore, there is a path T between any two of its vertices. Because T is a subgraph of G , there is a path in G between any two of its vertices. Hence, G is connected. Now suppose G is connected, If G is not a tree, it must contain a simple circuit. Remove an edge from one of those simple circuits. The resulting graph has one fewer edge but still contains all the vertices of G and is connected.



spanning trees are important in data networking such as IP multicasting.

The proof by theorem 1 gives an algorithm to finding spanning trees by removing edges from simple circuits. This algorithm is insufficient because it requires that simple circuits be identified.

We can build spanning tree for a connected simple graph using depth first search.

For this,

Firstly, We will arbitrarily choose a vertex of the graph as a root.

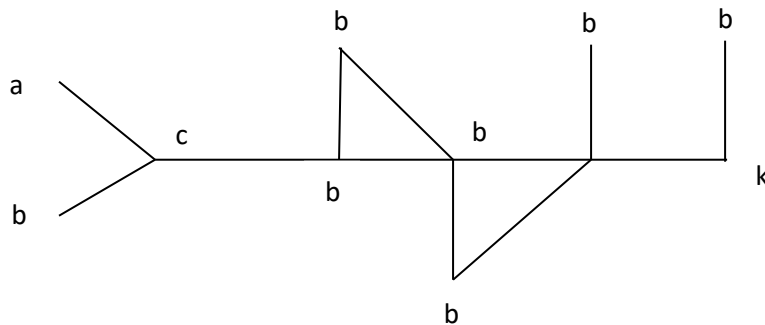
Form a path starting at this vertex by successively adding vertices and edges, where each new edges is incident with last vertex in path and a vertex not already in the path.

If path goes through all vertices of graph, the tree consisting of this path is spanning tree.

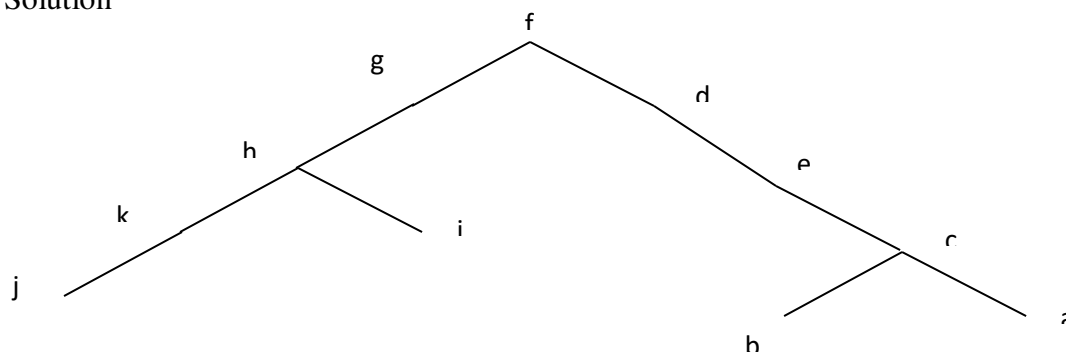
If path does not go through all vertices, more vertices and edges must be added.

Move back to the next to last vertex in the path and if possible, form a new path starting all this vertex passing through vertices that were not already visited. If this can not be done, move back another vertex in the path, that is two vertices back in the path and try again.

Example:

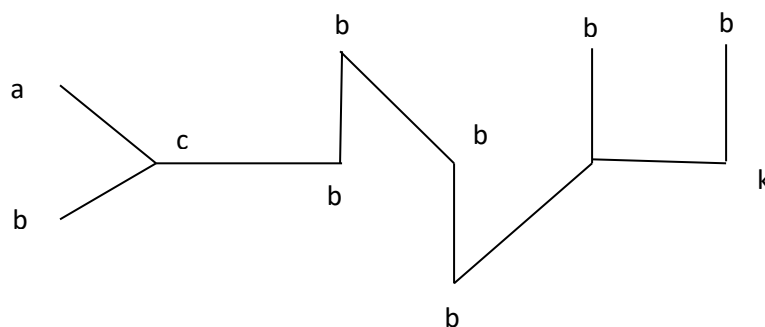


Solution



The edges selected by depth first search of a graph are called free edges.

All other edges of the graph must connect a vertex to an ancestor or descendant of this vertex in the tree. These edges are called back edges.



Depth first search is also called back tracking Algorithm

Procedure DFS (G: connected graph with vertices $V_1, V_2, \dots, V_n$)

T: tree consisting only of the vertex V_1 visit (V_1)

Procedure visit (Vertex of G)

for each vertex w adjacent to and not yet in T .

begin,

add vertex w and edge $\{v, w\}$ to T visit (w)

end.

Depth first search can be used to find paths and circuits in a graph, can be used to determine the connected components of a graph and to find cut vertices of a connected graph.

Breadth-First Search:

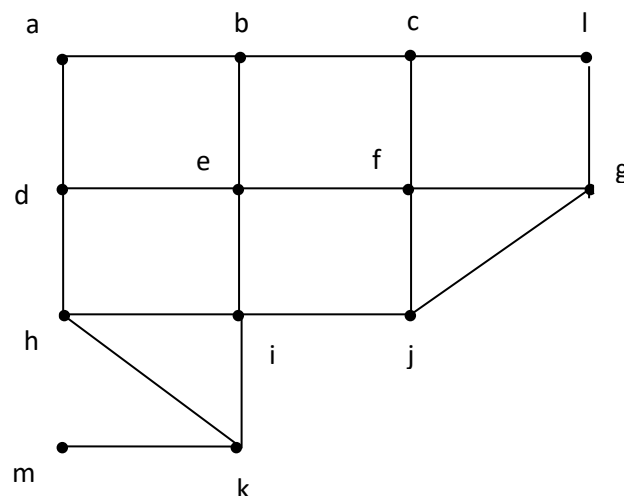
Choose a root to the vertex from the vertices of the graph.

Then, add all edges incident to this vertex. The new vertices added at this stage become the vertices at level 1 in a spanning tree.

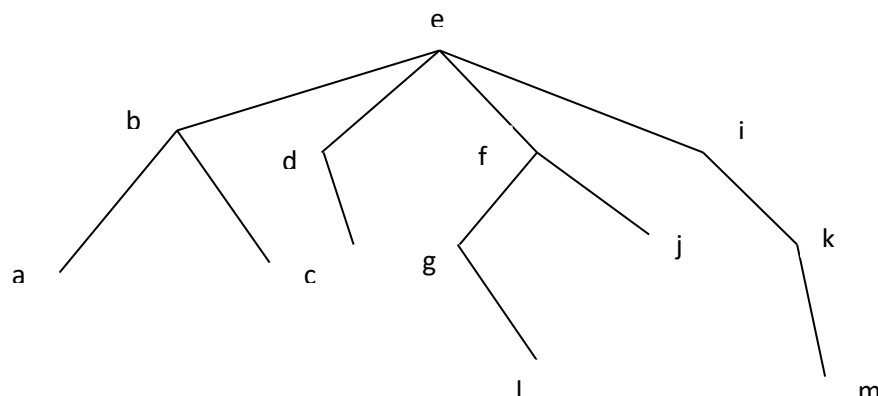
Arbitrarily order them.

For each vertex at level 1, visited in order, add each edge incident to this vertex to the tree as long as it doesnot produce a simple circuits.

Example:



Solution



$\{\text{Chicago, Atlanta}\} = \$700$
 $\{\text{Atlanta, New York}\} = \$800$
 $\{\text{Chicago, San Francisco}\} = \$1200$
 $\{\text{San Francisco, Denver}\} = \$900$
 Total \$3600

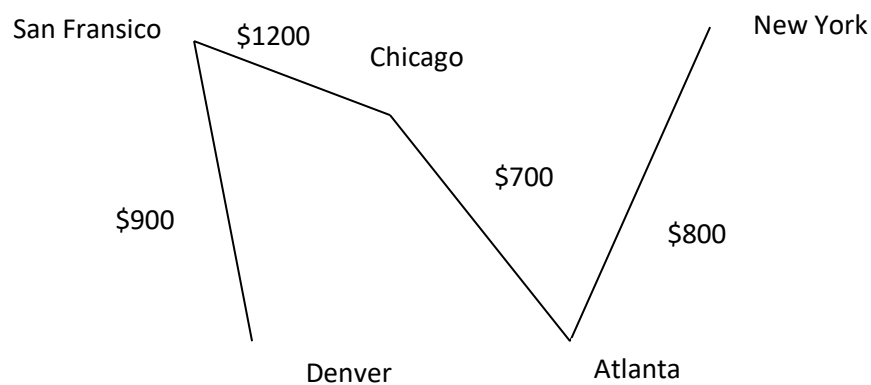
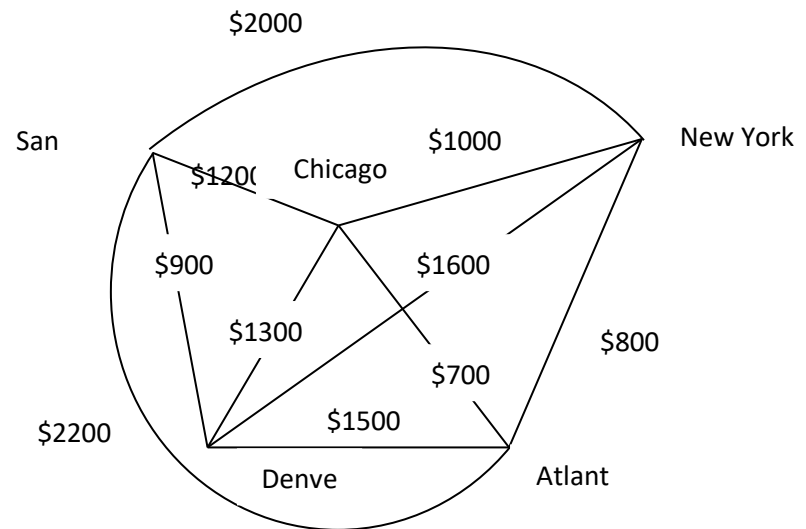


Fig : Minimum Spanning tree

Example

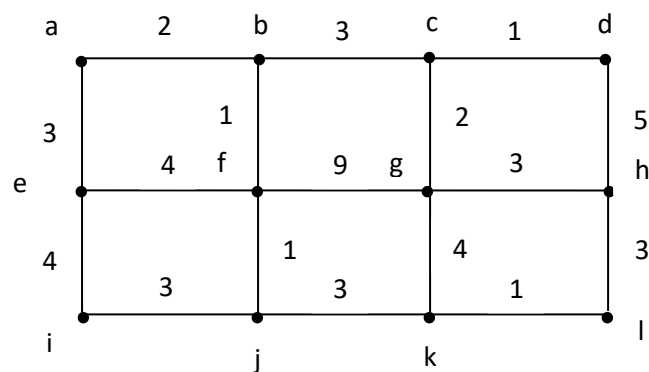


Fig : II

$\{b, f\} = 1$

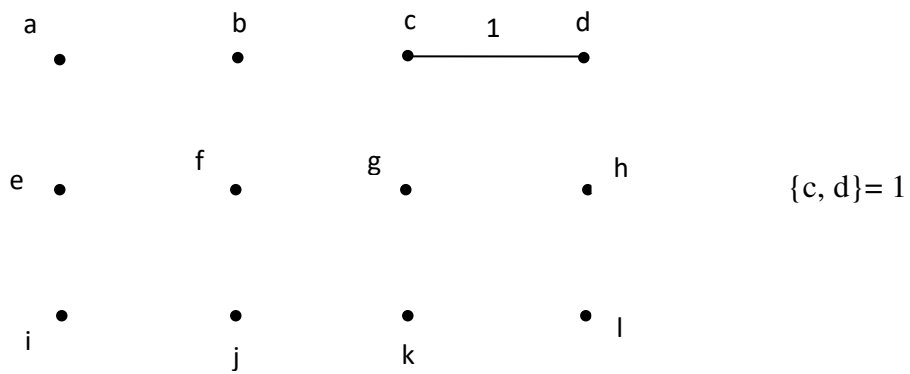
$\{f, i\} = 2$

$\{a, b\} = 2$

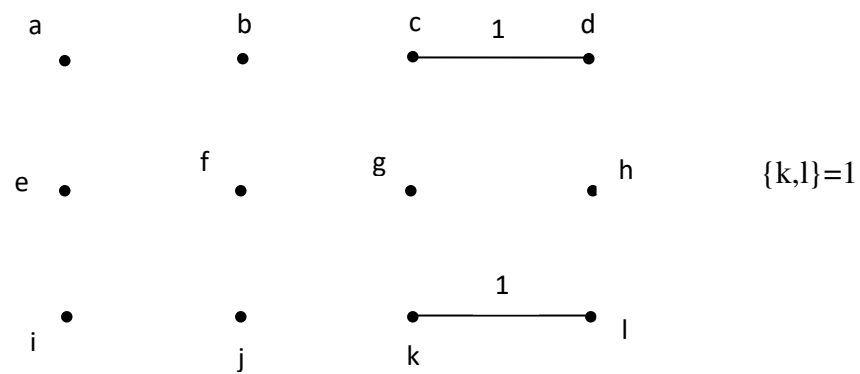
$\{a, e\} = 3$

Solution

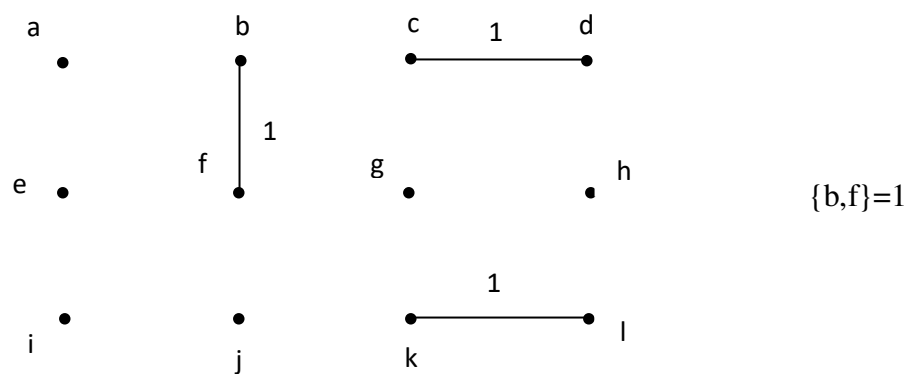
(a)



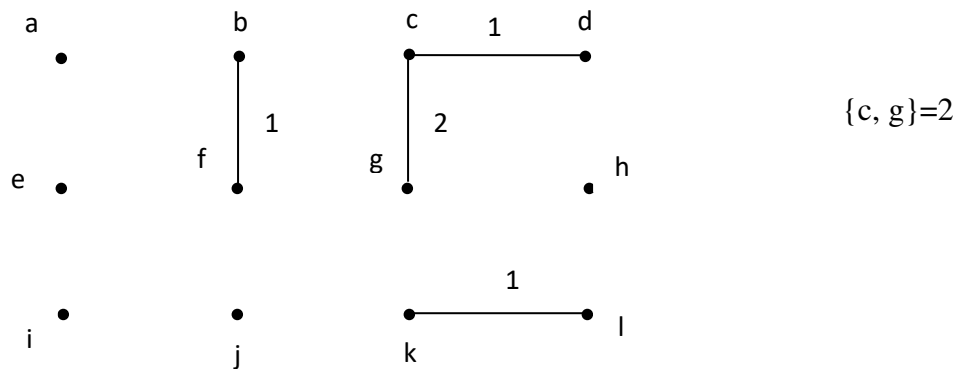
(b)



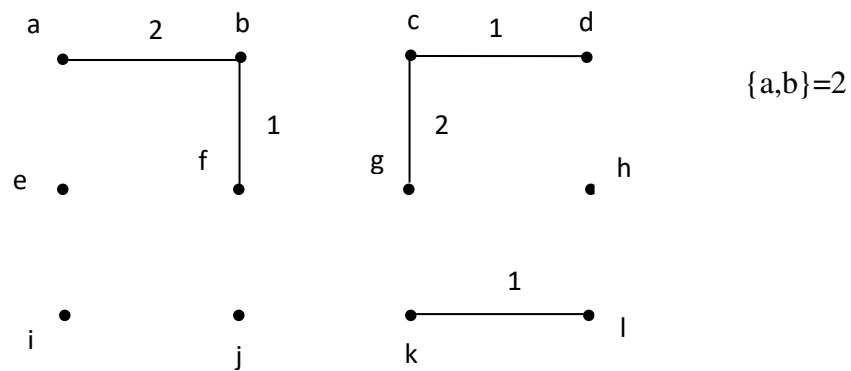
(c)



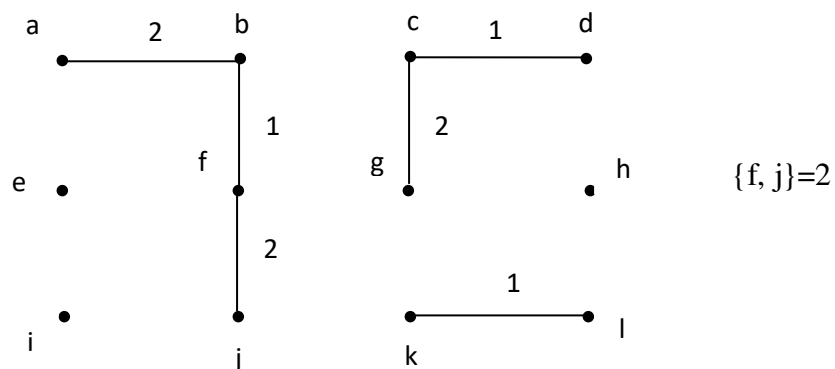
(d)



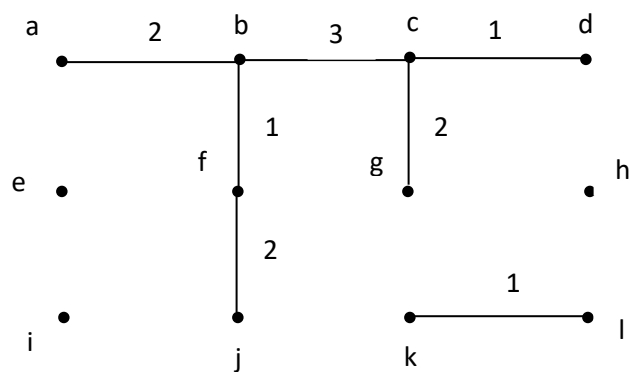
(e)



(f)

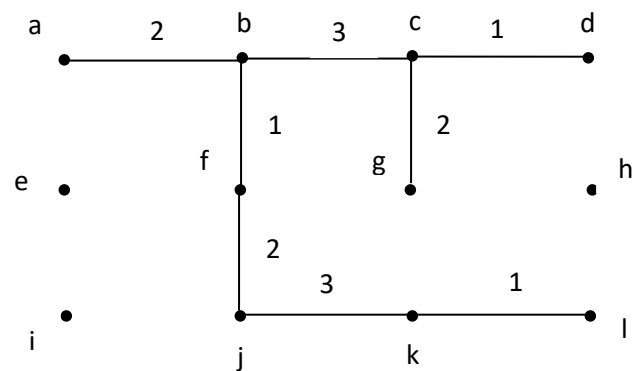


(f)



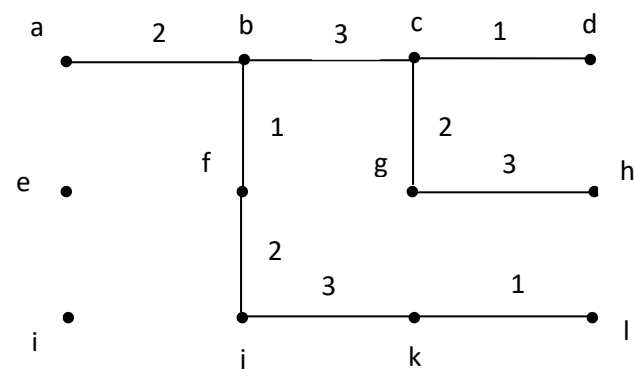
$$\{b, c\} = 3$$

(g)



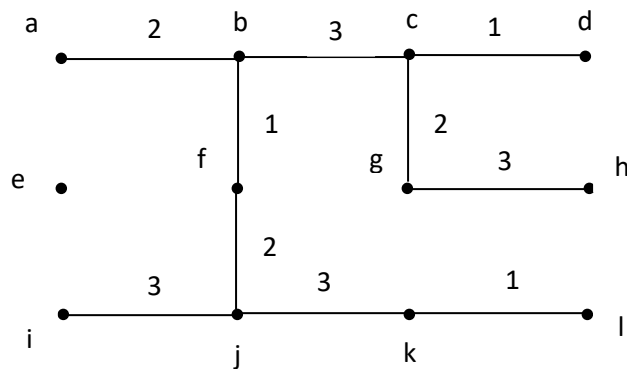
$$\{j, k\} = 3$$

(h)



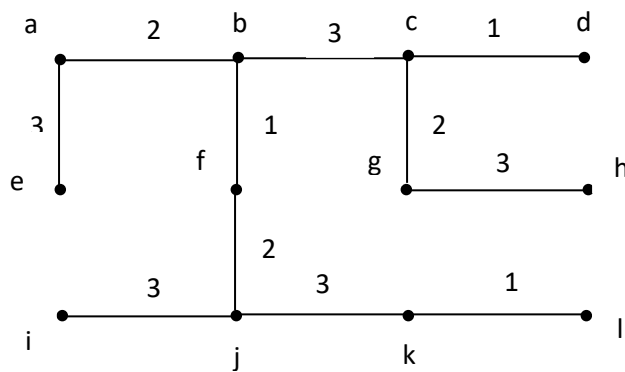
$$\{g, h\} = 3$$

(i)



$$\{i,j\}=3$$

(j)

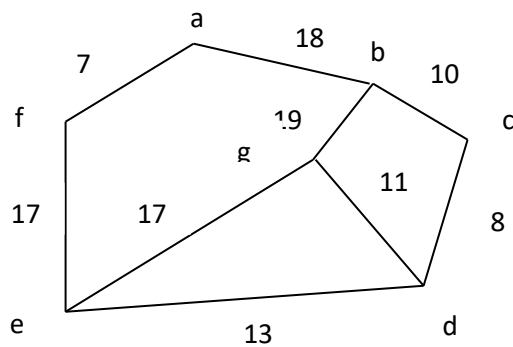


$$\{a,e\}=3$$

$$\text{Total}=24$$

minimum spanning tree

Q. Graph given below, find the minimum spanning tree



Graph coloring

A coloring of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.

Smallest number of colors must be used instead of just assigning every region its own color.

A chromatic number of a graph is the least no. of colour needed for coloring graph.

4 color Map Theorem:

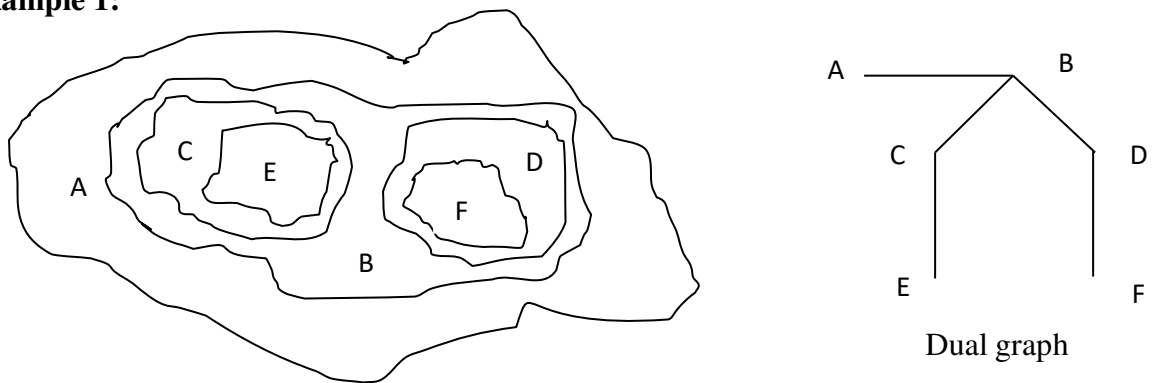
The chromatic number of a planar graph is no greater than four.

It can be shown that any two dimensional map can be painted using four color in such a way that adjacent regions are different colors.

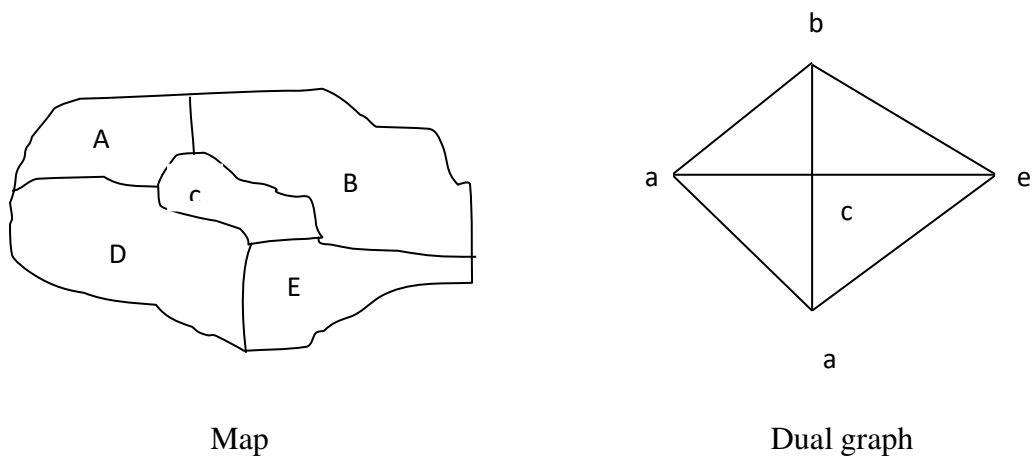
Dual graph

Each map in a plane can be represented by a graph .Each region is represented by a vertex. Edges connect to vertices if the regions represented by these vertices have a common border. Two regions that touch at only one point are not considered adjacent.

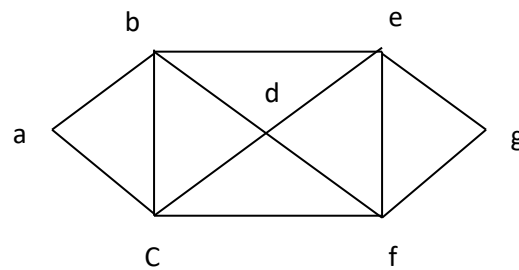
Example 1:



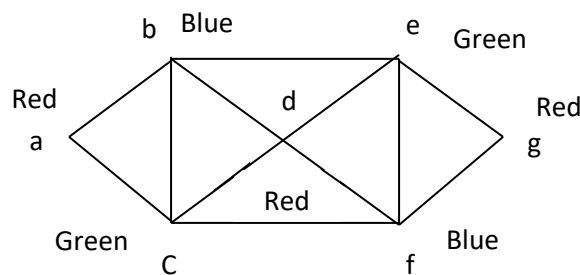
Example 2:



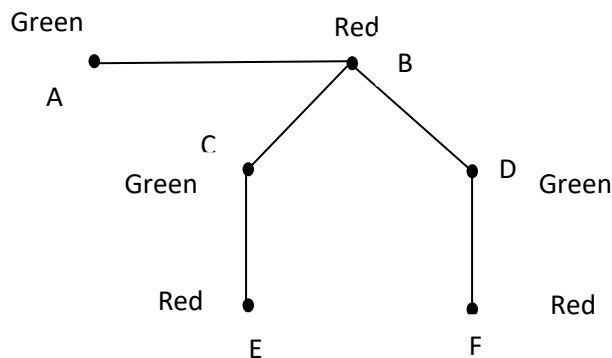
Q1. What is the chromatic no. of graph.



Chromatic no. must be at least 3 since a, b, c must be assigned different colors

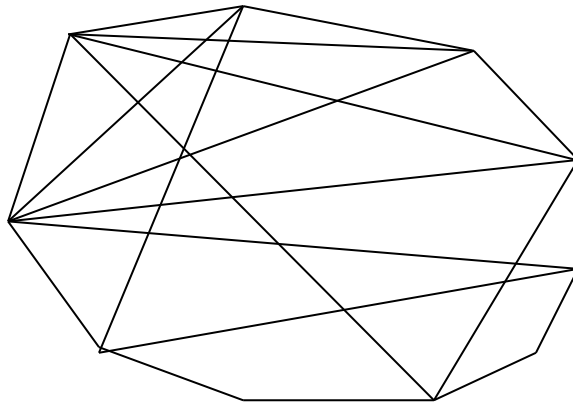


QN.2 What is the chromatic no. of the fig(A)



We can color the graph with at most 2 colors as shown in the graph red and Green.

Q.N.3. Find the chromatic no. of a graph.



Max flow and Min cut

- In network or transport network the flow is the amount of commodity (number of car in a road, gallon of oil in pipe, bits of formation in channel etc.) transported from one place to another for an instant of time.
- The maximum amount of flow is called the **capacity** of the line arc or edge.
- The flow $f(e)=f(v,w)$ and the capacity $c(e)=c(v,w)$ are non-negative real numbers.
- $f(e) \leq c(e)$.
- The unique starting vertex that has in degree=0 is called **source** $s \in V$
- The unique ending vertex that has out degree=0 is called **sink**, $t \in V$
- In a network there is one source and one sink. Any other vertex : called intermediate vertex.
- **Flow conservation** : For any intermediate vertex, total flow into x equals to the flow out of x .
$$\sum_{w \in V} f(w, x) = \sum_{v \in V} f(x, v)$$
- Sum of flow from the source = Sum of flow to the sink.
- **Saturated flow**: The flow along an edge $e(v, w)$ is said to be saturated if $f(e)=c(e)$.
- If $f(e) < c(e)$, then the flow is unsaturated, then residual capacity or slack is $s(e)=c(e)-f(e)$.

Flow augmenting path

- The edge e_i is called forward edge if: is directed to x_j to x_{i+1} and transportation flow from x_i to x_{i+1} , it is backward edge if transportation flow from x_{i+1} to x_i .
- If a flow f is given by path $p:s=x_1,x_2 \dots x_{k-1},x_k=t$ is called a flow augmenting path
(i) every forward edge of path has excess capacity that is $f(e) < c(e)$. (ii) every backward edge has $f(e) > 0$.

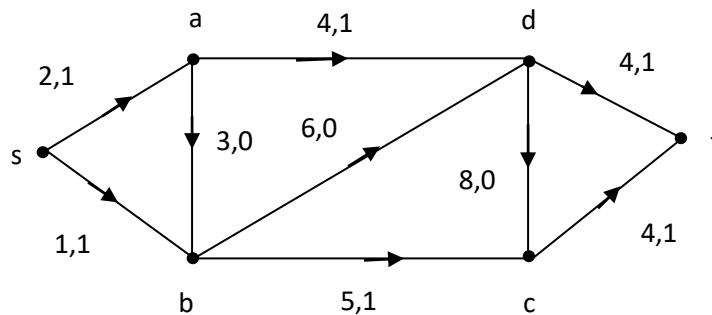
Max flow

- The flow that achieves the largest possible value is called the maximum flow or maximal flow in a network.
- To calculate max flow calculate the unused capacity $c(e)-f(e)$ for each forward arc and $f(e)$ in each backward arc.
- In a f- augmenting path from source to sink, for the forward edge increase the flow with the minimum slack(unused capacity) followed in the path.
- For backward edge in the f- augmenting path from source to sink, decrease the flow with the minimum slack.

$$\Delta = \min(c(e)-f(e))$$

$$f^* = f + \Delta \text{ (for forward edge)}$$

$$f^* = f - \Delta \text{ (for backward edge)}$$



Max flow

Here the unused capacity=slack in f- augmenting path s, a, d, t are

$$S(s,a)=2-1=1$$

$$S(a,d)=4-1=3$$

$$S(d,t)=4-1=3$$

The minimum value is 1 in above slack value. So, now add the value 1 in the flow of s, a, d, t.

$$f(s,a)=1+1=2$$

$$f(a,d)=1+1=2$$

$$f(d,t)=1+1=2$$

Now $s(s,a)=2-2=0$. So, there is not necessity for further calculation of slack f-augmenting path containing s(s,a) edge.(remember for f-augmenting path $(c(e)-f(e))>0$).

- Again, in $s(s,b)=1-1=0$, So, there is no necessity for further calculation of slack in the augmented path containing edge (s,b)
- The max flow = flow from source $s=2+1=3$

Min cut

- In min cut we calculate the capacity by separating the vertices into two subsets p and p'
- p contains s only or s and other vertices which are not in p'.

- p' contains t only or t and other vertices which are not in p .
- If there are 4 vertices other than source and sink, then there are $2^4=16$ s-t cuts.
- The capacity of a cut denoted by $c(p, p')$ is defined to be the s-m of the capacities of those edges directed from the vertices in set p to the vertices in p' .
- If the edge is backward edge put the value of the backward edge=0.
- A cut is called minimum cut if its capacity not exceed the capacity of any other cut of the network.

In above network,

If $p=\{s\}$ and $p'=\{a,b,c,d,t\}$ then $c(p,p')=c(s,a)+c(s,b)=2+1=3$.

If $p=\{s,a\}$ and $p'=\{b,c,d,t\}$ then $c(p,p')=c(s,b)+c(a,b)+c(a,d)=1+3+4=8$.

Similarly,

Table for possible s-t cut

p	p'	Capacity $c(p,p')$
$\{s\}$	$\{a,b,c,d,t\}$	3
$\{s,a\}$	$\{b,c,d,t\}$	8
$\{s,b\}$	$\{a,c,d,t\}$	13
$\{s,c\}$	$\{a,b,d,t\}$	7
$\{s,d\}$	$\{a,b,c,t\}$	15
$\{s,a,b\}$	$\{c,d,t\}$	15
$\{s,a,c\}$	$\{b,d,t\}$	20
$\{s,a,d\}$	$\{b,c,t\}$	16
$\{s,b,c\}$	$\{a,d,t\}$	12
$\{s,b,d\}$	$\{a,c,t\}$	19
$\{s,c,d\}$	$\{a,b,t\}$	11
$\{s,a,b,c\}$	$\{d,t\}$	10
$\{s,a,b,d\}$	$\{c,t\}$	17
$\{s,a,c,d\}$	$\{b,t\}$	12
$\{s,b,c,d\}$	$\{a,t\}$	10
$\{s,a,b,c,d\}$	$\{t\}$	8

Min cut =3

Hence, in the above example max flow=min cut=3.