MFCS :- Dr. Ramalingaswamy cherukuri Sir.

Date: Page No:

Sets, Relations, Functions - Fundamentals of Logic - Quantified Propositions - Mathematical Induction - Combinations and Permutations - Enumeration - Recurrence Relations - Generating Functions - Binary Relations - Lattices - Directed Graphs - Graphs - Spanning trees - Planar Graphs - Euler Circuits - Hamiltonian Graphs.

1. Mathematical Logic : Propositional Calculus & Predicate Cal.
2. (Combinatorics)
3. Recurrence Relations.
4. Graph Theory.
5. Set Theory.
6. Algebraic Structures and
7. Number Theory (if Time permits).

Topic 1 :- Logic and Proofs :- AI, D.B, circuit design
Logic : Propositional logic, first order logic.
Proof : induction, Contradiction.

Topic 2 :- Number Theory :- Cryptography, Coding theory.
Number sequence, Euclidean algo, Prime no., Modular arithmetic.

Topic 3 :- Counting :- Probability, analysis of algorithm.
Sets, Combinations, Permutations, Binomial theorm, functions, Counting by Mapping, Pigeonhole principle, Recursions, generating functions.

Date:_____ Page No:_____

Topic 4: Graph Theory :- Comp. Network, D.S.
• Relations, graphs, Degree sequence, isomorphisms, Eulerian graphs, Trees. → (A).

∴ Propositional logic :-

• Basic blocks of logic - propositions. A proposition is a declarative sentence (that is, sentence that declares a fact) that is either true or false, but not both.
Ex :- Toronto is capital of Canada — f.
1 + 1 = 2 — T
& $x+1=2$ → This is not propositional,

∴ Propositional Connectors :-

1. negation (~) :- If p is case, then ~p means not the case that p.

p	~p
0	1
1	0

2. Conjunction operator (∧) :- (And) :-
The p∧q is true when both p & q are true and is false otherwise.

p	q	p ∧ q
T	T	T
T	F	F

$\Rightarrow$ → Equal to , $\Leftrightarrow$ → not equal to.

Date:_____ Page No:____

F	T	F
F	F	F

(3) Disjunction (inclusive OR) :- true if either of P/q is true.

P	q	PVq
T	T	T
T	F	T
F	T	T
F	F	F

(4) Exclusive OR :- true if either of P or q, but not both P and q will true or false.

P	q	P ⊕ q.
T	T	F
T	F	T
F	T	T
F	F	F

(5) Conditional Statements :- $(P \Rightarrow q)$ it is false when P is true and q is false, and true otherwise.
where, P is hypothesis and q is called conclusion.

(Implication)

P	q	P → q
T	T	T
T	F	F
F	T	T
F	F	T

→ ★ · (10)

(6) Biconditional operation :- $(P \Leftrightarrow q)$:- It is true if P & q both True, & false if P & q both false.

(Equivalence)

Date: Page No:

p	q	$p \Leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

(4.) tautology :- Any possible condition for p & q if output for p & q if output is True for all is called tautology.
If $F_1 \Leftrightarrow F_2$ is tautology then, $F_1 \Leftrightarrow F_2$ (Equal).

(5.) If $p \to q \Leftrightarrow \sim q \to \sim p$ then, $(p \to q) \Leftrightarrow (\sim q \to \sim p)$ Should be tautology.

(6.) Proof Equivalence :-
(1.) Truth table Method.
(2.) $F_1 \to F_2$, if $F_1 \Leftrightarrow F_2$ is tautology.

(3.) Duality law :- just interchange signs, except negation

$\wedge \to \vee$
$\vee \to \wedge$
$T \to F$
$F \to T$

} Don't change negation ($\sim$).

Ex :- $(p \vee q) \wedge R \Rightarrow (p \wedge q) \vee R$
$\sim (p \vee q) \Rightarrow \sim (p \wedge q)$.

) → parenthesis.

Date: Page No:

W.f.f :- well-formed formula :- Should contain propositions, connectors alongwith parenthesis for removing ambiguity.

Ex:- $(P \wedge (Q \vee R)) \Leftrightarrow P \wedge (Q \vee R)$

(well formed formula).

& precedance will be :- $\sim$ (1) $\wedge$ (2) $\vee$ (3)

(5.) Prove the following are tautologies :-

- Conjunction of two tautologies is tautology.
- $P \vee \sim P = T$ (tautology)
- $(R \rightarrow S) \vee \sim (R \rightarrow S) = T$
- If $P \leftrightarrow Q$ is tautology then, $P \Leftrightarrow Q$.

(6.) Logical equivalences :-

(1.) $P \wedge T \equiv P$ - Identity laws

$P \vee F \equiv P$ - Identity laws.

(2.) $P \wedge F \equiv F$ - Domination laws

$P \vee T \equiv T$ - Domination laws.

(3.) $P \wedge P \equiv P$ - Idempotent laws

$P \vee P \equiv P$ - "

(4.) $\sim(\sim P) \equiv P$ - Double Negation law

(5.) $P \wedge q \equiv q \wedge P$ } Commutative law.

$P \vee q = q \vee P$ }

(6.) $(p \wedge q) \wedge R \equiv p \wedge (q \wedge R)$
$(p \vee q) \vee R \equiv p \vee (q \vee R)$
} Associative laws.

(7.) $p \wedge (q \vee R) \equiv (p \wedge q) \vee (p \wedge R)$
$p \vee (q \wedge R) \equiv (p \vee q) \wedge (p \vee r)$
} Distributive law.

(8.) $\sim(p \wedge q) \equiv \sim p \vee \sim q$
$\sim(p \vee q) \equiv \sim p \wedge \sim q$
} De Morgan's law

(9.) $p \wedge (p \vee q) \equiv p$
$p \vee (p \wedge q) \equiv p$
} Absorption law.

(10.) $p \wedge \sim p \equiv F$
$p \vee \sim p \equiv T$
} Negation law.

(11) Logical Equivalences involving conditional Statements :

(11.) $(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$ — Implication
(12.) $p \Leftrightarrow q \Leftrightarrow [(p \Rightarrow q) \wedge (q \Rightarrow p)]$ — Equivalence
(13.) $[(p \wedge q) \Rightarrow R] \Leftrightarrow [p \Rightarrow (q \Rightarrow R)]$ — Exportation
(14.) $[(p \Rightarrow q) \wedge (p \Rightarrow \sim q)] \Leftrightarrow \sim p$ — Absurdity
(15.) $(p \Rightarrow q) \Leftrightarrow (\sim q \Rightarrow \sim p)$ — Contrapositive.

Date:_____ Page No:____

(4.) A tautology is always true, Contradiction or absurdity is always false, & A prepositional form which is neither a tautology nor a Contradiction is called Contigency.

(i) $Q \leftrightarrow (R \wedge \sim P)$

I will go to town if and only if I have time and it is not snowing.

(ii) $(Q \Rightarrow R) \wedge (R \Rightarrow Q)$

I will go to town only if I have time and if I have time I will go to town.

(5.) Predicates and Quantifiers :-

$x > 3$
$x + y = 7$
} Assertion but not prepositions.

(1) Sum $(x, y, z) \Rightarrow x + y = z$
(Predicates)

(2) $M(x, y) \Rightarrow$ x is married to y, (predicates)

(3) if $(x > 3)$ then, condition, (predicates).

$P(x)$ = Unary Predicates
$Q(x, y)$ = Binary Predicates
$P(x_1, x_2, \ldots, x_n)$ = n-ary predicates.

If $P(x_1, x_2 \ldots x_n)$ is true for all values $c_1, c_2 \ldots c_n$ (predicates const.) from universe V then you say $P(x_1, x_2 \ldots x_n)$ is valid in V.

Date: Page No:

(→) If $P(x_1, \ldots x_n)$ is true for some $c_1, \ldots, c_n$ from U
P is satisfiable in U.

(→) If P is not true for any value $c_1, \ldots, c_n$ then P is said to be unsatisfiable in U.

(→) If $P(x,y)$ is there and you gives values for x and y this becomes a proposition.

(→) There are two ways of binding variable
(1.) By giving values.
(2.) By using quantifiers.

(→) Binding by using quantifiers :- If we have, $P(x)$ = (unary predicates depend on x only).

(A) $\boxed{\forall x}\ P(x)$, its read as Universal quantifiers

for all x
for every x , for each x
for any x , for arbitrary x , Some examples are ;-

Eg :-
Suppose, 'I' as set of integers and U is Universe,
(1.) quantifier $\forall x\,[x < x+1]$ → predicates (always true & proposition).

(2.) $\forall x\,[x=3]$ → proposition but false

(3.) $\forall x\,[x=x+1]$ → takes value false.

(B) $\boxed{\exists x}\ P(x)$, its read as Existential quantifiers.

There exists x s.t. $P(x)$ is true. (St. ⇒ Such that).
for some x, $P(x)$ is true

Eg:-
Suppose, 'I' as set of integers and U is universe,

1. $\exists x\ [x < x+1]$, true
2. $\exists x\ [x = 3]$, true
3. $\exists x\ [x = x+1]$, false.

c. $\boxed{\exists! x}\ P(x)$, is read as
There exists a unique x set $P(x)$ is true. Or.
There is one and only one x set $P(x)$ is true.
Eg:- 'I' as set of integers,

1. $\exists! x\ [x < x+1]$, false value.
2. $\exists! x\ [x = 3]$, True
3. $\exists! x\ [x = x+1]$, false.

Hence,
we can say that, $\exists x\ [P(x) \wedge \forall y \{P(y) \Rightarrow x = y\}]$ is equivalent to $\exists! x\ P(x)$.
Similarly,
There exists at most one x.
$\exists x\ \forall y \{P(y) \Rightarrow x = y\}$

3. Consider,
$\forall x\ P(x, y, z)$, then x is bound here & y and z are free.
if we bound, $y = 2$. then,
$\forall x\ P(x, 2, z)$, called a unirary predicate.

'¬' ⇒ negation (~)

Date: Page No:

then again bound z by quantifiers,
$\exists_z \forall_x P(x,2,z)$, called propositions which takes values as True or false

(∵) Scope of Predicates & Quantifiers :- It must be defined as, Suppose, N is non-ve integer,

(1.) $xy=z$ $P(x,y,z)$

(a) If $xy=x$ for all y, then $x=0$

$\forall x [\forall y \; P(x,y,x) \Rightarrow x=0]$

But,
$\forall x \forall y \; P(x,y,z) \Rightarrow x=0$, is not correct because also satisfies by $x=1$.

(b.) If $xy \neq x$ for some y, then $x \neq 0$

$\forall x [\exists y \neg P(x,y,x) \Rightarrow x \neq 0] \quad \neg(x=0)$.

(∵) Well formed formula of I order logic Or predicate logic :- It is valid, if it is true for all universes and all interpretators of predicates variables. Eg :-

$\forall_x P(x) \Rightarrow \exists_x P(x)$, is always true, this is said to be a valid statement in predicates logic.

wff is true for some universe and some interpretation it is said to be satisfiable.

off is not then it is

Date: Page No:

off is not true for any universe & any interpretation then it is said to be unsatisfiable.

∴ for taking negation inside, $\forall x$ change to $\exists x$ & vice-versa.

1. $\neg \forall x P(x) \Leftrightarrow \exists x \neg P(x)$
2. $\neg \exists x P(x) \Leftrightarrow \forall x \neg P(x)$
3. $\neg \forall x \forall y \exists z P(x,y,z) \Leftrightarrow \exists x \exists y \forall z \neg P(x,y,z)$
4. $\exists x (P(x) \Rightarrow Q(x)) \Leftrightarrow \exists x (\neg P(x) \vee Q(x))$

Ques. Prove $\exists x P(x) \Rightarrow \exists x Q(x)$ & $\exists x (P(x) \Rightarrow Q(x))$ are equivalent or not.

Sol:
$\exists x (P(x) \Rightarrow Q(x))$
$\Rightarrow \exists x (\neg P(x) \vee Q(x))$
$\Rightarrow \exists x \neg P(x) \vee \exists x Q(x)$
$\Rightarrow \neg \forall x P(x) \vee \exists x Q(x)$
$\Rightarrow \forall x P(x) \Rightarrow \exists x Q(x)$

, Hence, they are not equivalent

∴ Logical relationships involving quantifiers :-

1) $\forall x P(x) \Rightarrow P(c)$, where c is an arbitrary element of the universe.
2) $P(c) \Rightarrow \exists x P(x)$, where, c - - - - - -
3) $\forall x \neg P(x) \Rightarrow \neg \exists x P(x)$
4) $\forall x P(x) \Rightarrow \exists x P(x)$
5) $\exists x \neg P(x) \Leftrightarrow \neg \forall x P(x)$
6) $[\forall x P(x) \wedge Q] \Leftrightarrow \forall x [P(x) \wedge Q]$
7) $[\forall x P(x) \vee Q] \Leftrightarrow \forall x [P(x) \vee Q]$

(8.) $[\forall x P(x) \wedge \forall x Q(x)] \Leftrightarrow \forall x [P(x) \wedge Q(x)]$

(9.) $[\forall x P(x) \vee \forall x Q(x)] \Leftrightarrow \forall x [P(x) \vee Q(x)]$

(10.) $[\exists x P(x) \wedge Q] \Leftrightarrow \exists x [P(x) \wedge Q]$

(11.) $\exists x [P(x) \wedge Q(x)] \Leftrightarrow [\exists x P(x) \wedge \exists x Q(x)]$

(12.) $[\exists x P(x) \vee \exists x Q(x)] \Leftrightarrow \exists x [P(x) \vee Q(x)]$

(7.)

Logical Inference :-

(∵) Some Rules of inference related to the language of propositions.

Rule of inference	Tautological form	Name
$\frac{P}{\therefore P \vee Q}$	$P \Rightarrow (P \vee Q)$	Addition
$\frac{P \wedge Q}{\therefore P}$	$(P \wedge Q) \Rightarrow P$	Simplification
$\frac{P \quad P \Rightarrow Q}{\therefore Q}$	$[P \wedge (P \Rightarrow Q)] \Rightarrow Q$	Modus ponens (*)
$\frac{\neg Q \quad P \Rightarrow Q}{\therefore \neg P}$	$[\neg Q \wedge (P \Rightarrow Q)] \Rightarrow \neg P$	Modus tollens (*)

Date: Page No:

{ reading as, $\frac{P \wedge Q}{\therefore P}$, if $P \wedge Q$, from this you can conclude P. }

$P \vee Q$ $\neg P$ $\therefore Q$	$[(P \vee Q) \wedge \neg P] \Rightarrow Q$	Disjunctive Syllogism
$P \Rightarrow Q$ $Q \Rightarrow R$ $\therefore P \Rightarrow R$	$[(P \Rightarrow Q) \wedge (Q \Rightarrow R)] \Rightarrow [P \Rightarrow R]$	Hypothetical Syllogism.
P Q $\therefore P \wedge Q$		Conjunction.
$(P \Rightarrow Q) \wedge (R \Rightarrow S)$ $P \vee R$ $\therefore Q \vee S$	$[(P \Rightarrow Q) \wedge (R \Rightarrow S) \wedge (P \vee R)] \Rightarrow [Q \vee S]$	Constructive dilemma.
$(P \Rightarrow Q) \wedge (R \Rightarrow S)$ $\neg Q \vee \neg S$ $\therefore \neg P \vee \neg R$	$[(P \Rightarrow Q) \wedge (R \Rightarrow S) \wedge (\neg Q \vee \neg S)] \Rightarrow [\neg P \vee \neg R]$	Destructive dilemma.

To Solve Some equations (English Sentence).

- If today is Tuesday, then I have a test in C.S or a test in Econ. If my econ. professor is sick, then I will not have a test in econ. Today is Tuesday and my econ. professor is sick. Therefore, I have a test in CS

Date: Page No.

Let, Today is Tuesday = T.
I have test in C.S = CS
I have test in Eco = E
Econ. Prof is Sick = S.

Statement 1 :- $T \Rightarrow (CS \vee E)$
" 2 :- $S \Rightarrow \neg E$
" 3 :- $T \cdot S$
Conclusion $\therefore CS$

Let see it is correct or not,
From (3) by Simplification, (4) T
From (3) by Simplification, (5) S
from (1) & (4) Modus ponens, (6) CSVE
from (2) & (5) Modus ponens, (7) $\neg E$
from (6) & (7) disjunctive Syllogism, CS.

(8) Now, if $\frac{P(c) \text{ for an arbitrary element } c \text{ of } U.}{\therefore \forall x P(x)}$
this is called universal Generalization.

and, from $\frac{\forall x P(x)}{\therefore P(c)}$, c is arbitrary element of U.
this is called universal instantiation

now, if $\frac{P(c) \text{ for some } c \in U.}{\therefore \exists x P(x)}$

This is called existential generalization.

and from, $\frac{\exists x\, P(x)}{\therefore P(c)}$, c is some element of U.

this is called existential instantiation.

Eg :- (1) All men are mortal
Socretes is a man
∴ Socretes is mortal.

Sol:- man (x) x is a man.
mortal (x) x is mortal.

1. $\forall x$ (man (x) ⇒ mortal (x))
2. man (Socretes)
∴ mortal (Socretes)

from (1), using universal Instantiation.
(3) man (Socretes) ⟹ mortal (Socretes)
(3) & (2), modus ponens, mortal (Socretes).

5) Normal Forms :- two types basically,

(1) Conjunctive normal form (CNF)
(2) Disjunctive normal form (DNF)

(-:) Some basic terminologies :-
(1) Literals :- A variable or negation of a variable is called literals. Eg :- P or ¬P.

Date:_____ Page No.

(2) Disjunction of literals is called a Sum.
$P_1 \vee P_2 \vee P_3 \vee \neg P_4$ → Sum.

(3.) Conjunction of literals is called a Product.
$P_1 \wedge P_2 \wedge \neg P_3$ → Product

(4.) A Sum of Products is called DNF :-

$(P_1 \wedge P_2 \wedge \neg P_3) \vee (\neg P_4 \wedge P_5)$

Eg :- (a.) $P \wedge (P \Rightarrow Q)$

Sol :- $P \wedge (\neg P \vee Q)$ [$\because P \Rightarrow Q = \neg P \vee Q$]

$(P \wedge \neg P) \vee (P \wedge Q)$ — Ans.

(5.) A Product of Sums is called CNF :-

$(P_1 \vee P_2) \wedge (P_3 \vee P_4 \vee \neg P_5)$

Eg :- (a.) $P \wedge (P \Rightarrow Q)$

Sol :- $(P) \wedge (\neg P \vee Q)$ — Ans.

MFCS :- (By Sir) After Mid-Sem

Characteristics Equ" Method :-

Solving Recurrence Relations :- (Complex roots)

if roots are Imaginary then we have, Characteristic equation with complex roots.

Solve : $a_n = 2 \cdot a_{n-1} - 2 \cdot a_{n-2}$ (given that $a_0 = 0$ and $a_1 = 2$)

The char. equ^n is :- $r^2 - 2r + 2 = 0$. It has two roots $(1+i)$ and $(1-i)$

The sequence $\{a_n\}$ is a sol^n to this recurrence sol^n iff $a_n = \alpha_1 (1+i)^n + \alpha_2 (1-i)^n$.

∵ Given, $\alpha_1 = -i$ & $\alpha_2 = i$ then by solving,

The final sol^n is $a_n = -i(1+i)^n + i \cdot (1-i)^n$

(2.) $a_n = 2(a_{n-1} - a_{n-2})$, $n \geq 2$, $a_0 = 1$, $a_1 = 2$.

The C.E. is $r^2 - 2r + 2 = 0$ with roots $1 \pm i$.

$\therefore a_n = c_1 (1+i)^n + c_2 (1-i)^n = c_1 \left[\sqrt{2}\left(\cos\frac{\pi}{4} + i \sin\frac{\pi}{4}\right)\right]^n +$

$c_2 \left[\sqrt{2}\left(\cos\frac{\pi}{4} - i \sin\frac{\pi}{4}\right)\right]^n = c_1 (\sqrt{2})^n \left(\cos\frac{n\pi}{4} + i \sin\frac{n\pi}{4}\right) +$

$c_2 (\sqrt{2})^n \left(\cos\frac{n\pi}{4} - i \sin\frac{n\pi}{4}\right) = (\sqrt{2})^n \left[(c_1 + c_2)\cos\frac{n\pi}{4} + i(c_1 - c_2) \sin\frac{n\pi}{4}\right]$.

with $a_0 = 1$ & $a_1 = 2$, we have,

$c_1 + c_2 = 1$ & $c_1 - c_2 = i$, therefore,

$a_n = (\sqrt{2})^n \left(\cos\frac{n\pi}{4} + \sin\frac{n\pi}{4}\right)$. Hence we see

three types of roots :-

1. distinct roots.
2. Repetition roots.
3. Complex roots.

Date: Page No:

(1) Generating functions Method :- $G(x)$ is function generates the series like $a_0, a_1, a_2 \dots a_k$ terms

Or

$$G(x) = a_0 + a_1 x + a_2 x^2 \dots = \sum_{k=0}^{\infty} a_k x^k.$$

$$[a_0, a_1, a_2, a_3 \dots] \leftrightarrow a_0 + a_1 x + a_2 x^2 + a_3 x^3 \dots$$

Let, $G_n = 1 + x + x^2 + x^3 \dots$ — (1)

Multi by x,

$$x G_n = x + x^2 + x^3 \dots$$ — (2)

By (2) − (1),

$$G_n - x G_n = 1$$

$$\boxed{G_n = \frac{1}{1-x}}$$

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 \dots \leftrightarrow [1, 1, 1, \dots \infty]$$

$$\frac{1}{1+x} = 1 - x + x^2 - x^3 \dots \leftrightarrow [1, -1, 1, -1 \dots \infty]$$

$$\frac{1}{1-ax} = 1 + ax + a^2 x^2 + a^3 x^3 \dots \leftrightarrow [1, a, a^2, a^3 \dots \infty]$$

$$\frac{1}{1-x^2} = 1 + x^2 + x^4 \dots \leftrightarrow [1, 0, 1 \dots \infty].$$

Def 1 :- The generating function for the sequence $\{a_n\}$ is the infinite power series.

$$G(x) = a_0 + a_1 x + \dots + a_n x^n + \dots$$
$$= \sum_{k=0}^{\infty} a_k x^k.$$

Ex (2) What will G.F of sequence $\{1, 1, 1, 1, 1, 1, \dots\}$

Date: Page No:

$$G(x) = 1 + x + x^2 + \cdots + x^5$$
$$= \frac{x^6 - 1}{x - 1}$$

(2) Let $m \in Z^+$ and $a_k = \binom{m}{k}$ for $k = 0, 1, \ldots m$
What is G.F for the sequence $a_0, a_1, \ldots, a_m$? $\{m = +ve \text{ int}\}$.

Sol:
$$(1+x)^m = {}^mC_0 x^0 + {}^mC_1 x^1 + {}^mC_2 x^2 + \cdots$$
Binomial Coe.
$$G(x) = (1+x)^m \quad \text{Ans.}$$

(3) Let $u \in R$ and $k \in Z^+ \cup \{0\}$. Then the extended binomial coe. is defined by $\binom{u}{k}$ { where $n = -ve$ int term. } or [fraction 1/2, 2/3]
$$\binom{u}{k} = \begin{cases} \frac{1}{k!}\, u(u-1)(u-2)\cdots(u-k+1), & \text{if } k > 0 \\ 1, & \text{if } k = 0. \end{cases}$$

u = -ve or fractional no.

Ex: (a) Find $\binom{-2}{3}$ and (b) $\binom{1/2}{3}$

(a) $\frac{1}{3!}(-2)(-3)(-4) = -4$ $\left\{\because \frac{1}{n!}(-2)^{(1)}(-2-1)^{(2)}(-2-2)^{(3)}\right\}$.

(b) $\frac{1}{3!}\left[\left(\frac{1}{2}\right)\left(\frac{1}{2}-1\right)\left(\frac{1}{2}-2\right)\right] = \frac{1}{16}$

(4) Using G.F to Solve Recurrence Relations.

(1) $a_k = 3a_{k-1}$ for $k = 1, 2, 3, \ldots$ & $a_0 = 2$.

Sol: $x - 3 = 0 \Rightarrow x = 3$, $a_n = 2 \cdot 3^n$
$\because a_0 = 2 = A$
$\boxed{a_n = 2 \cdot 3^n}$ } By Char. Eqn Method.

By G.F.M,

Let $G(x) = a_0 + a_1 x + a_2 x^2 + \cdots = \sum_{k=0}^{\infty} a_k x^k$ be the G.F.

Multiply by x

$x\, G(x) = a_0 x + a_1 x^2 + a_2 x^3 + \cdots = x \cdot \sum_{k=0}^{\infty} a_k x^k$

$= \sum_{k=0}^{\infty} a_k \cdot x^{k+1}$

$= \sum_{k=1}^{\infty} a_{k-1}\, x^k$ — (1)

$G(x) - 3x\, G(x) = \sum_{k=0}^{\infty} a_k x^k - 3 \sum_{k=0}^{\infty} a_k x^{k+1}$

$= \sum_{k=0}^{\infty} a_k x^k - 3 \sum_{k=1}^{\infty} a_{k-1} x^k$

$= a_0 + \sum_{k=1}^{\infty} a_k x^k - 3 \sum_{k=1}^{\infty} a_{k-1} x^k$

$= a_0 + \sum_{k=1}^{\infty} (\underbrace{a_k - 3a_{k-1}}_{0})\, x^k$

$= a_0 = 2$

$G(x)(1-3x) = 2$

$$\boxed{G(x) = \frac{2}{(1-3x)}} = 2 \cdot \left(\sum_{k=0}^{\infty} (3x)^k\right)$$

$= 2 \sum_{k=0}^{\infty} 3^k \cdot x^k$

$= \sum_{k=0}^{\infty} (2 \cdot 3^k) \cdot x^k$, Coef. of x^k is Seq. of [a_n] or k^{th} term of Seq.

$\therefore$ $\boxed{a_k = 2 \cdot 3^k}$

Or. $\boxed{a_n = 2 \cdot 3^n}$ Ans.

Date: Page No:

Strategy 2 - Divide and Conquer :- Used to design the more effective or efficient algorithm.

Suppose that you wrote a recursive algorithm that divides a problem of size n into
- a subproblems,
- each subproblem is of size n/b.

Additionally, a total of g(n) operations are required to combine the solutions.

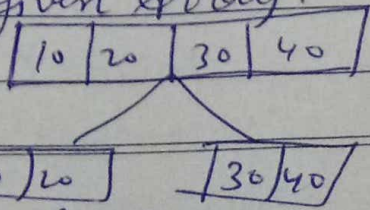
Let f(n) denote the no of operations required to solve a problem of size n. Then,

$$f(n) = a\, f(n/b) + g(n)$$

This is divide conquer recurrence relation.

Steps :- Suppose, we have problem of size n.
1. we divide the n in size like n_1, n_2, n_3, n_4 etc.
2. and solve the sub-problem recursively.
3. then we again divide the n_1 & n_2 --etc. like $n_{11}, n_{12}, n_{13}, n_{21}, n_{22}, n_{23}$ etc. until, we have easiest way to solve all the problem recursively. This is called Divide phase or Divide strategy. Top to bottom
4. Once, we get sol.^{n} we combines all the sol.^{n} for actual sol.^{n}, this is called Conquer phase, Combining phase bottom to top.

Example 1 :- Searching element in the given array.
Divide list in two equal parts.
Example 2:- Binary Search,



for worst case,
$f(n) = 2f(n/2) + 2$

Date: Page No:

Suppose that you have a sorted array with n elements. You want to search for an element within the array. How many comparisons are needed?

Compare with median to find out whether you should search the left $n/2$ or the right $n/2$ elements of the array. Another comparison is needed to find out whether terms of the list remain.

Thus, if $f(n)$ is the no. of comparisons, then

$$f(n) = f(n/2) + 2.$$

Recurrent Algorithms, array, A, find x

```
Binary search (A, lo, hi, x)
  if (lo > hi)
      return false;
  mid ← [(lo + hi) / 2]
  if x = A[mid]
      return TRUE
  if (x < A[mid])
      Binarysearch (A, lo, mid - 1, x)
  if (x > A[mid])
      Binarysearch (A, mid + 1, hi, x)
```

Merge Sort Algo:-

l = left index, r = right index

Example 3:-

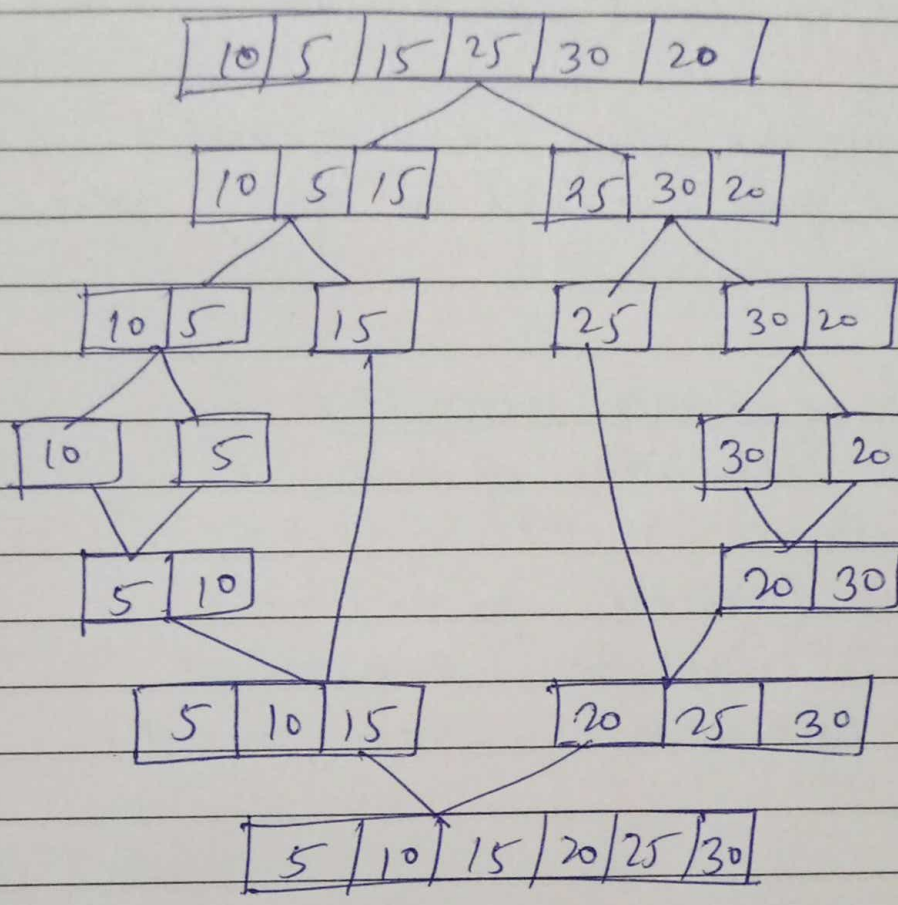
```
void mergesort (int arr[], int l, int r)
{ if (l < r)
  {     // find mid point
        int m = (l + r) / 2;
  /* recursive mergesort first and second
     halves */
  mergesort (arr, l, m);
  mergesort (arr, m+1, r);
  // merge
```

Date: Page No:

merge (arr, l, m, r);
}

Sorting problems :- Merge Sort, Quick Sort.

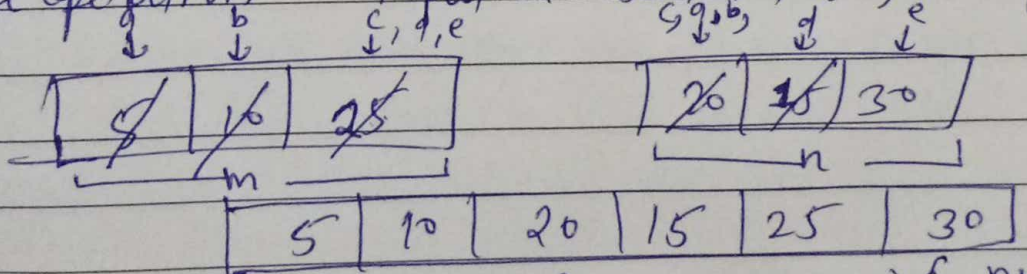
Suppose,



Merge & Sort

Merge operation :- Input :- 2 - Sorted list, Output :- Sorted list.

comp. = (m+n).



merge cost = n → (comparisons.) { n/2 + n/2 = n }.

BiSearch , $f(n) = f(n/2) + 2$

Min-Max , $f(n) = 2f(n/2) + 2$

Merge , $f(n) = 2f(n/2) + n$.

Date: Page No:

⑤ The Graph theory :-

① Graph theory :-

- Graphs used to model pair wise relations between objects.
- Generally, a network can be represented by a graph.
- Many practical problems can be easily Represented in terms of graph theory.

② Basic Concepts of Graph theory :-

A graph is a collection of nodes & edges.
denoted by $G = (V, E)$

V = nodes (vertices, points).

E = edges (links, arcs) between

Graph size parameters : $n = |V|$, $m = |E|$

- Vertex / Node
  - → Basic element.
  - → Drawn as a node or a dot.
  - → Vertex set of G is usually denoted by $V(G)$, or V or V_G

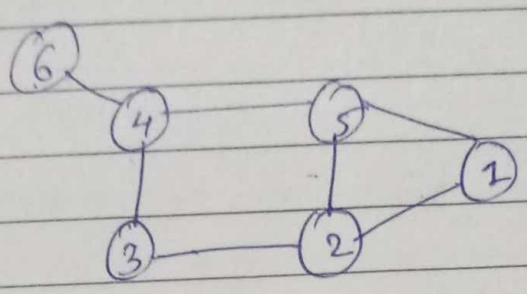
- Edge / Arcs
  - → A set of two elements.
  - → Drawn as a line connecting two vertices, called end vertices, or endpoints.
  - → The edge set of G is usually denoted by $E(G)$, or E or E_G.

Date: Page No:

eighborhood

for any node V, the set of nodes it is connected to via an Edge is called its neighborhood and is represented as N(V).

(4) Example :-



- n := 6 , m := 7
- vertices (V) := {1, 2, 3, 4, 5, 6}
- Edge (E) := {1,2}, {1,5}, {2,3}, {2,5}, {3,4}, {4,5}, {4,6}
- N(4) := Neighborhood (4) = {6, 5, 3}.

(5) Edges types :-

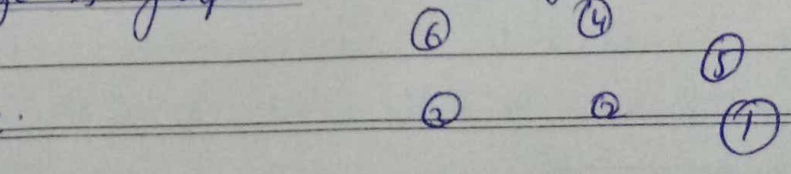
(1) Undirected :- Eg :- distance between two cities, friendships .....

(2) Directed :- Ordered pair of nodes.
Eg :- Directed edges have a source (head, origin) and target (tail, destination) vertices.

(3) Weighted :- Usually weight is associated.

(4) Empty graphs / Edgeless graph :- No edge.

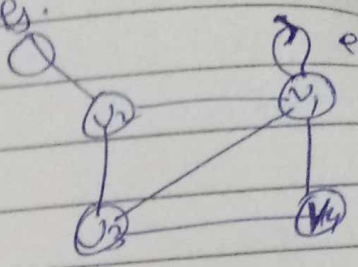
- Null graphs,
→ No nodes & no edge.



Date: Page No:

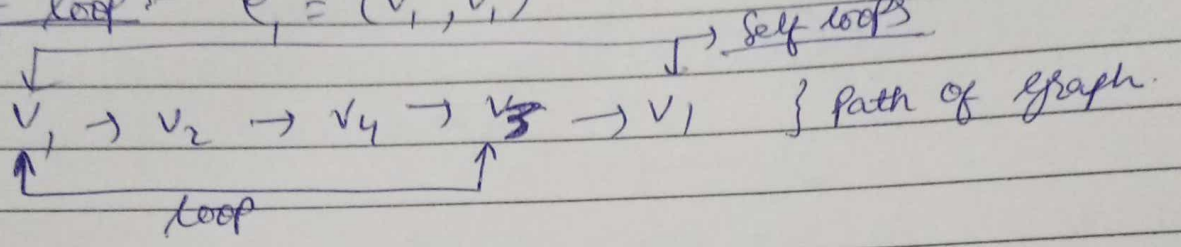
(2) Simple graph (Undirected) :- They are undirected graphs without loop or multiple edges.

- $A = A^T$



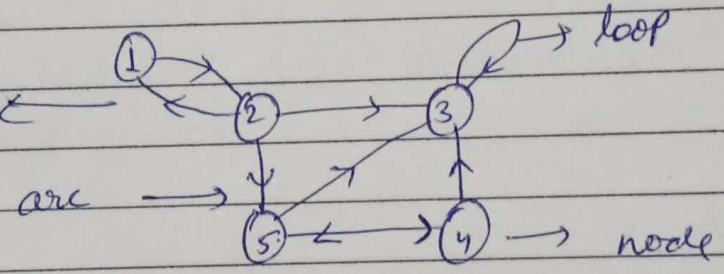
For simple graphs, $\sum_{v_i \in V} \deg(v_i) = 2|E|$

Self loop :- $e_1 = (v_1, v_1)$

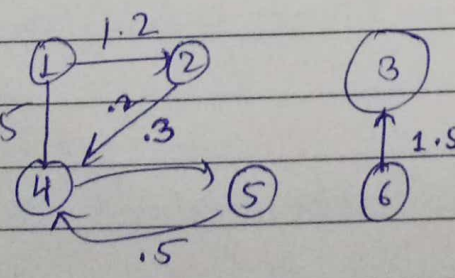


(3) Directed graph : (digraph) :-

- Edges have directions
- $A^T = A^T$



(4) Weighted graph :- It is a graph for which each edge has an associated weight.

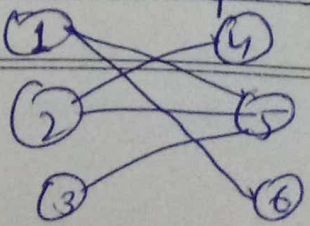


Edge list

12	1.2
24	0.2
41	.5

(5) Bipartite graph :- V can be partitioned into 2 sets V_1 and V_2 such that $(u,v) \in E$ implies either $u \in V_1$ and $v \in V_2$

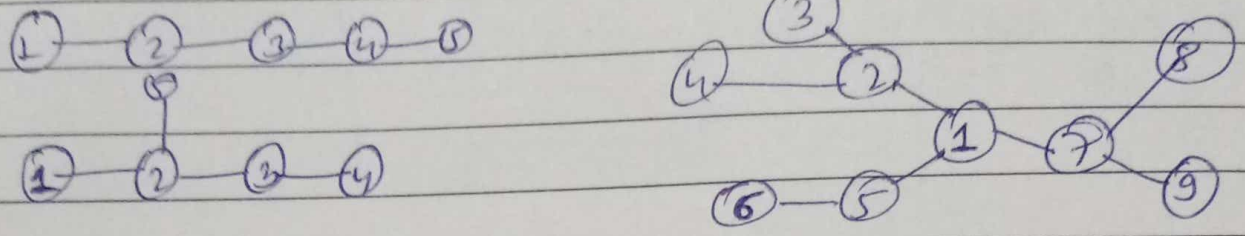
OR $v \in V_1$ and $u \in V_2$.



Trees :-

An undirected graph is a tree if it is connected and does not contain a cycle (Connected Acyclic Graph)

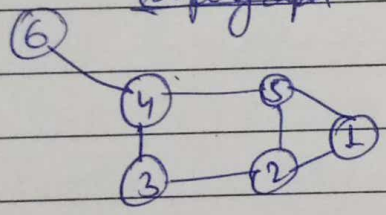
- Two nodes have exactly one path between them.



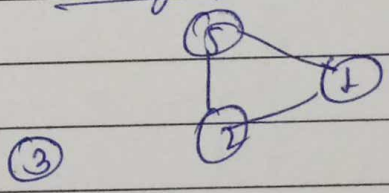
(4) Subgraph :-

- vertex and edge sets are subsets of those of G.
  - a supergraph of a graph G is a graph that contains G as a subgraph.

Supergraph

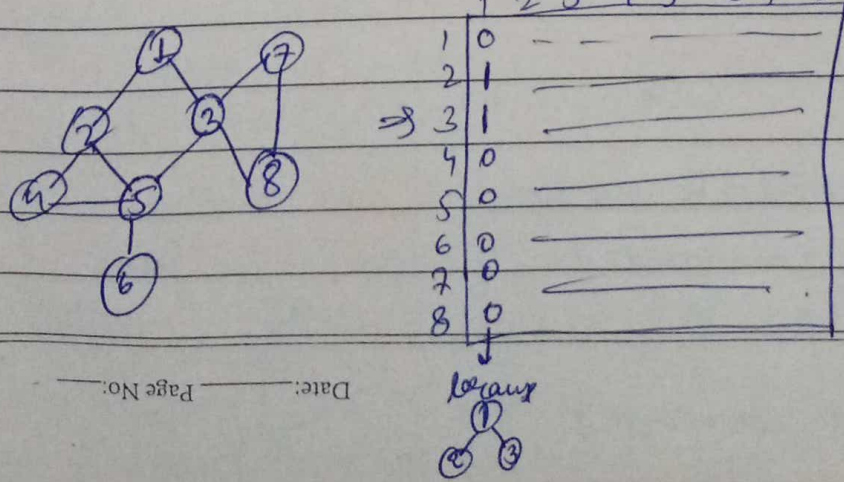


Subgraph



(1) Adjacency Matrix :-

- n-by-n matrix with $A_{uv} = 1$ if (u, v) is an edge.
  - Diagonal Entries are self-links or loops.
  - Symmetric matrix for undirected graphs. ($A^T = A$).

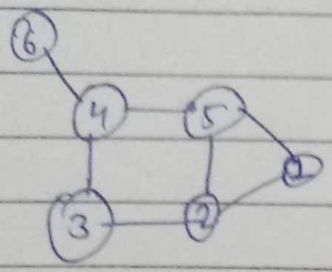


	1	2	3	4	5	6	7	8
1	0							
2	1							
3	1							
4	0							
5	0							
6	0							
7	0							
8	0							

Date: Page No:

(2) Incidence Matrix :-

- V X E
- [vertex, Edges] contains the edge's data

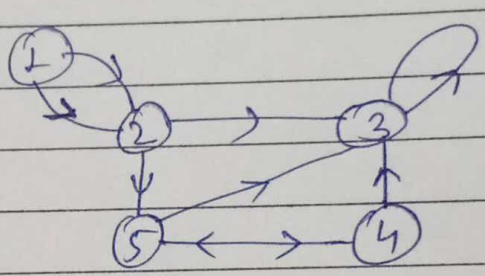


	1,2	1,5	2,3	2,5	5,4	4,5	4,6
1	1	1	0	0	0	0	0
2	1	0	1	1	0	0	0
3							
4							
5							
6							

(3) Adjacency list :-

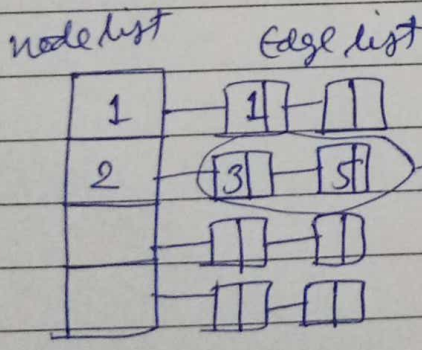
- Edge list

Edge List
1 2
1 2
2 3
2 5
3 3
4 3
4 5
5 3
5 4



- Adjacency List (node list)

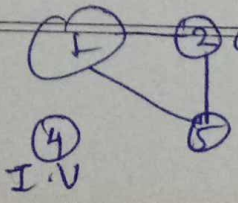
Node List
1 2 2
2 3 5
3 3
4 3 5
5 3 4



Adjacent Neighbours. (link list).

(4) Connected and Isolated Vertex :-

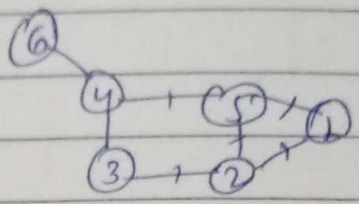
- Two vertices are connected if there is path between them.
- Isolated vertex – not connected.



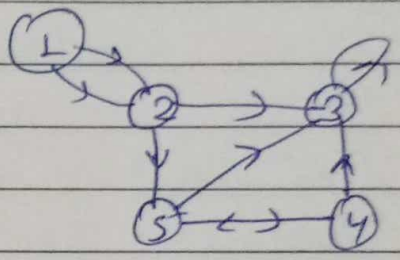
(3) Adjacent nodes :- Two nodes are adjacent if they are connected via an edge.
- if edge $e = \{u, v\} \in E(G)$, who say that u and v are neighbours or adjacent.

(4) Degree of Undirected Graph :- No. of edges incident on a node.

degree of 1 = 2
degree of 2 = 3
degree of 5 = 3



(4) Degree of Directed Graphs :-
- In-degree :- No. of edges entering.
- out-degree :- No. of edges leaving.
- Degree = indeg + out deg.

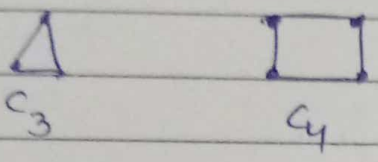


$D(1) = \text{outdeg}(1) + \text{Indeg}(1) = 2+0 = 2$
$D(2) = \text{outdeg}(2) + \text{Indeg}(2) = 2+2 = 4$
$D(3) = \text{outdeg}(3) + \text{outdeg}(3) = 1+4 = 5$

(4) Some other things :-
- Trail :- no edge can be repeat
  a-b-c-d-e-b-d.
  (a,b) (b,c) (c,d) (d,e) (e,b) (b,d)
- walk :- a path in which edges/nodes can be repeated.
  a-b-d-a-b-c.
  (a,b) (b,d), (d,a) (a,b), (b,c).
  - A walk is closed is a = c.
- Paths :- is a sequence P of nodes $v_1, v_2, \cdots, v_{k-1}, v_k$.
  → No vertex can be repeated.
  → A closed path is called a cycle.

→ A closed path is called a cycle.
→ The length of a path or cycle is no. of edges visited in the path or cycle.

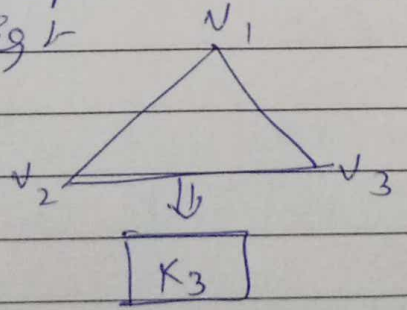
- Cycle :- closed path, denoted by C_k, k is no of nodes in the cycle, cycle (a-b-d-a), closed if $x = y$.

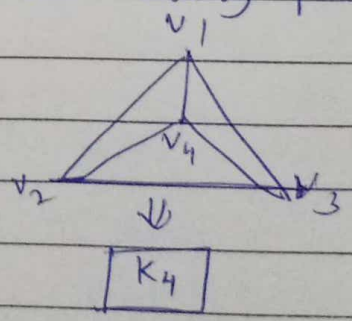


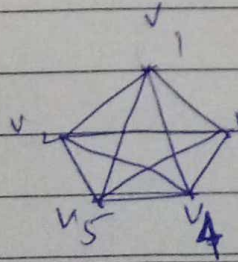
- Shortest path :- It is the path between two nodes that has the shortest length.
  - Length :- no. of edges.
  - Diameter :- of a graph is the length of the longest shortest path between any pairs of nodes in the graph.

(3.) Some important graphs :-

(1.) Complete graph :- direct edge bt^n every pair of vertices. Eg :-







⇒ K_5, Hence, total no of edges in K_n.

Complete graph :-

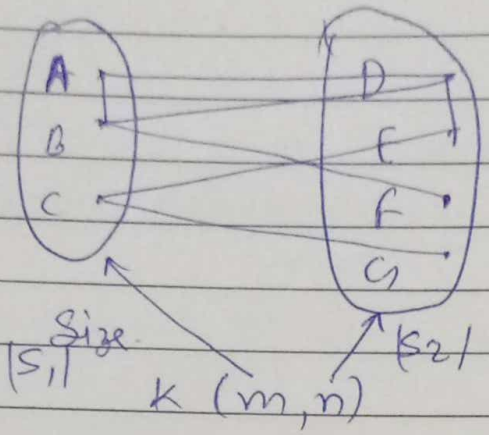
$$= \frac{n(n-1)}{2}.$$

Date: Page No:

(2.) Bi-partite Graph :- there exists two vertexes which are disjoint i.e., there intersection is ϕ.

$S_1 \cap S_2 = \phi$.

there should not be any common element.



A → B
↓ ↓
$S_1 \times S_1$
↳ not acceptable

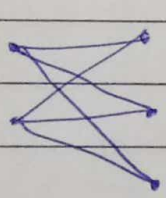
(3.) Complete Bi-partite graphs :-

Total edge in CBP graph $K_{m,n} = m \times n$.

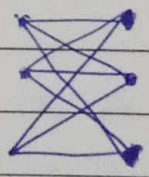
$|E| = m \times n$

$|V| = m + n$

Eg :- $K_{(2,3)}$



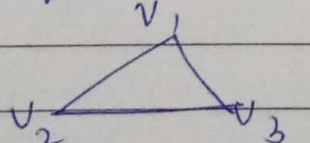
$K_{(3,3)}$



(4.) Cycle graphs :- In this graph, $|V| = |E|$

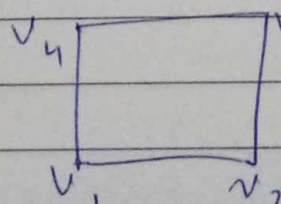
no. of vertex = no. of Edges.

Eg :- (1)



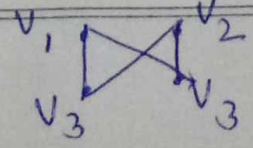
$|V| = |E| = 3$.

(2)



$|V| = |E| = 4$.

Or,

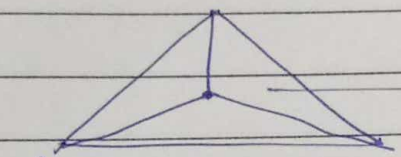


$|V| = |E| = 4$.

Date: Page No:

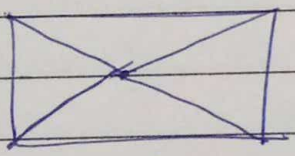
(5.) Wheel Graph :- A wheel graph of n vertices can be obtained by inserting an extra edge in extra vertex of cycle graph of C_{n-1} and find the adjacent to the all the vertex of C_{n-1}.

Eg :- (1) W_4 :- $C_3 + (1)$



$W_4 = C_3 + (1)$

(2) W_5 :- $C_4 + (1)$



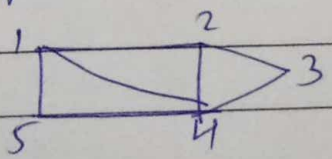
Total edge in wheel of W_n = edge in C_{n-1}

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

$$W_n = 2(n-1).$$

in • wheel graph of size n,

(6.) Cyclic Graph :- atleast one cycle is there i.e., there exists a path which looks cyclic.



path :-

1-2-3-4-5-1

if there is no cycle in graph called Acyclic Graph.

Date: Page No:

Tree ⟷ Connected
Acyclic

Hence, Tree is a special type of graph which is connected and Acyclic Graph.

(2) Sum of degrees theorm:- Let $G(|V|, |E|)$, undirected graph then,

$$\sum_{i=1}^{|V|} \log(v_i) = \underbrace{2|E|}_{\text{even}}$$

Sum of degrees

Suppose, for directed graph, Indegree (+) outdegree (−)

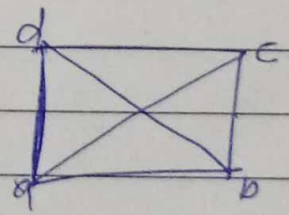
$$\sum_{i=1}^{|V|} \text{Indegree}(v_i) = |E|$$

$$\sum_{i=1}^{|V|} \text{Outdegree}(v_i) = |E|$$

Proof :- $G(V, E)$ is a graph and degree of each vertex is 'k', then,

$$|k| + |k| + |k| \cdots\cdots$$

$$k \cdot |V| = 2|E|.$$



degree(a) + degree(b) + 'c' + 'd' = 2|E|

$3 + 3 + 3 + 3 = 2 \times 6$

$12 = 12$ — Ans.

- If in a graph 'G' degree of each vertex is atleast 'k', then,

$$\deg(v) \geq k.$$

$$k + \cdots$$

$$\boxed{k \cdot |V| \leq 2|E|}$$

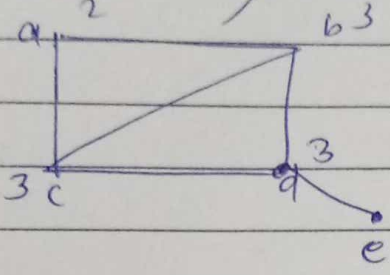
∵ for atmost K,

$$\boxed{k|V| \geq 2|E|}$$

Date: Page No:

(4) Degree Sequence :-

Arrange the degrees of all vertices in non-Increasing order i.e., decreasing order.

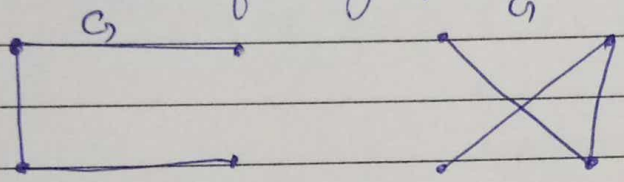


2 3 3 3 1

→ D.S :- 3, 3, 3, 2, 1

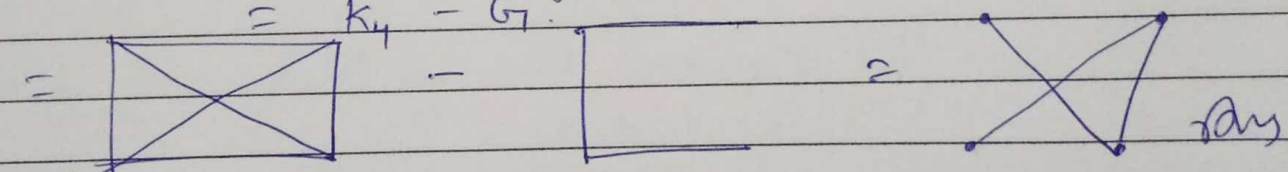
(5) Compliment of a graph :-

$\bar{G}$



$\bar{G}$ = Complete graph (G) − G

= K_4 − G.



Ans

edge in ($\bar{G}$)

$= \frac{n(n-1)}{2} - G$

$$\boxed{\bar{G} = \frac{n(n-1)}{2} - |E|}$$ — Ans.

Q.1. G − $|V| = 10$ & $|E| = 31$, find edge in ($\bar{G}$) = ?

Sol :- $\bar{G} = K_{10} - |E|$

$= \frac{10 \times 9}{2} - 31$

$= 45 - 31 = 14$. Ans

for no. of vertex,

$$\boxed{\bar{G} + G = \frac{n(n-1)}{2}}$$

Date: Page No:

(7) Graph Isomorphism :- The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there is a one-to-one 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.

In other words, G_1 and G_2 are isomorphic if their vertices can be ordered in such a way that the adjacency matrices M_{G_1} and M_{G_2} are identical.

for a visual standpoint,

G_1 and G_2 are isomorphic if they can be arranged in such a way that their displays are identical (of course without changing adjacency).

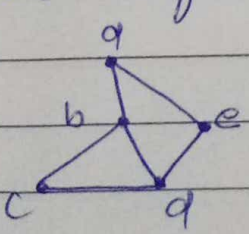
But, for two simple graphs, each with n vertices, there are "n! possible isomorphisms" that we have to check in order to show that these graphs are isomorphic.

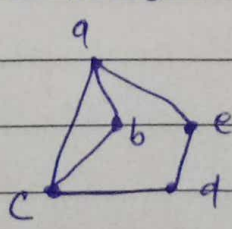
However, showing that two graphs are not isomorphic can be easy.

they must have, to check.

(1) Same no. of vertices.
(2) Same no. of edges.
(3) Same degrees of vertices.

Eg :- (1)





Date: Page No:

We make Matrixes for both are,

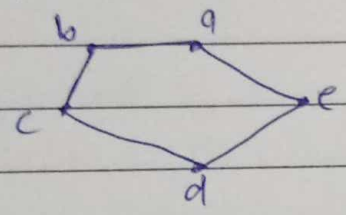
	a	b	c	d	e
a	0	1	1	0	0
b	1	0	1	1	0
c	0	1	0	1	0
d	0	1	1	0	1
e	1	0	0	1	0

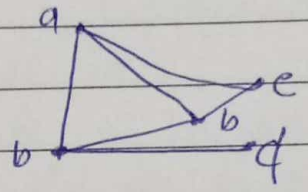
	a_1	b_1	c_1	d_1	e_1
a_1	0	1	1	0	0
b_1	1	0	1	1	0
c_1	a	1	a	1	0
d_1	0	1	1	b	1
e_1	1	0	0	1	0

edges as, $(a,b)=1$, $(b,c)=1$

if both adjacent matrix is same then called Isomorphic.

(2.)





(a.) $|V|=5$ $|V|=5$ ✓

(b.) $|E|=6$ $|E|=6$ ✓

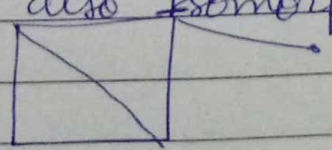
(c.) 3, 3, 2, 2, 2 3, 3, 3, 2, 1 ✗ d.s.

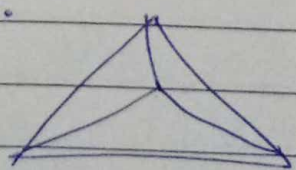
So, not isomorphic.

∴ Properties :- (1) if G_1 & G_2 are isomorphic, then, $\bar{G}_1$ & $\bar{G}_2$ are also isomorphic,

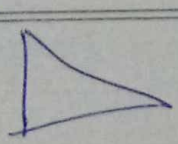
(2.) if $G_1 \cong G_2$, then the corresponding Sub graphs (by deleting an vertex in G_1 & corresponding vertex in G_2) Subgraphs also Isomorphic.

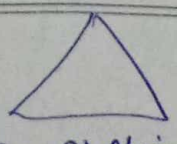
Eg :- (1)





Subgraphs





also isomorphic.

Date: Page No:

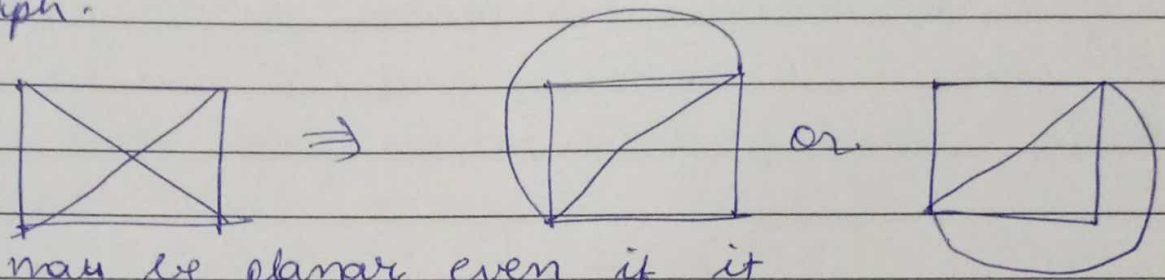
(4) Planarity and Kuratowski's Theorem :-

(A) **Planar Graphs :-** A graph is called planar if it can be drawn in the plane without any edges crossing.

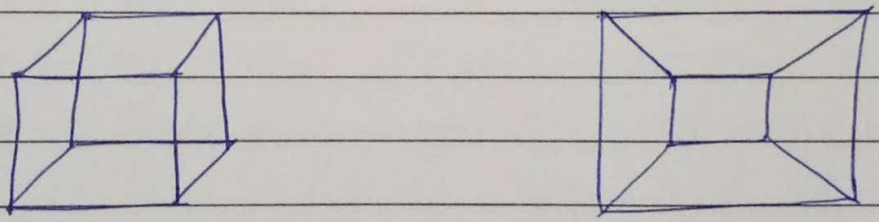
A crossing of edges is the intersection of the lines or arcs representing them at a point other than their common end-point.

Such a drawing is called planar representation of the graph.

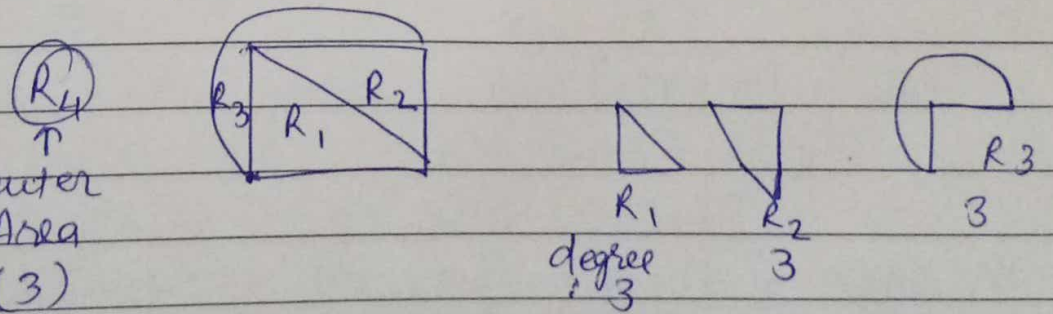
Eg :-



A graph may be planar even if it represents a 3-D object.



It is easy to prove a graph is planar & difficult to prove non-planar



Degree of interior region = No. of edges enclosing that region
Degree of exterior region = No. of edges exposed to that region

(A) Properties :-

(1.) In any planar graph, Sum of degrees of all the vertices = 2 x Total no. of edges in graph.

i.e., $\sum_{i=1}^{n} \deg(v_i) = 2|E|$

(2.) In any planar graph, Sum of degrees of all the regions = 2 x Total no. of edges in graph.

i.e., $\sum_{i=1}^{n} \deg(r_i) = 2|E|$

(3.) If G is a connected planar simple graph with 'e' edges, 'v' vertices and 'r' number of regions in the planar representation of G, then -

$$\boxed{r = e - v + 2}$$, Euler's formula, and remains same in all the planar representations of the graph.

Q.1 :- G has 20 vertices & deg. of vertex is 3. Find no. of regions in G.

Sol :- $r = e - v + 2$

$r = e - 20 + 2$

$r = e - 18$

$= 30 - 18$

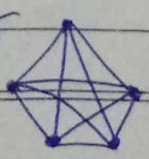
$\boxed{r = 12}$

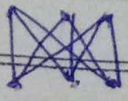
, $K \cdot |V| = 2(e)$

$e = \frac{K}{2} \cdot |V|$

$e = \frac{3}{2} \times 20^{10} = 30.$

(B.) Non-Planar Graphs :- A graph which is not possible to crossing is called non-Planar graphs. Eg :- K_5 , $K_{3,3}$





Date: _____ Page No: _____

(5) Kuratowski's theorem :- It states that a graph is planar if and only if it does not contain either K_5 or $K_{3,3}$ or a subdivision or of either K_5 or $K_{3,3}$ as a subgraph.

- Testing for Planarity :-

- Planarity Algorithms :- - Kuratowski's theorem

(a) Hamiltonian Cycle :- A Ham. Cycle is a closed path which visits each vertex once and only once.

(b) Eulerian Cycle :- A Eul. Cycle is a closed path that travels along every edge once.

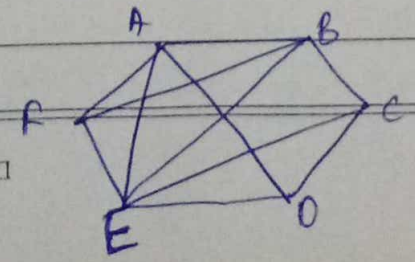
- (A.)

The Algo. States that :-

1. Find a Ham. Cycle containing all vertices
2. Re-draw the graph with this H.C. as an outer polygon with all other edges inside.
3. Select any inside edge and assign this to set P
4. Assign any edge that crosses the first edge to Set Q.
5. Assign any edge that any edge in Q crosses to Set P, etc.
6. Re-draw the graph with edges in set P drawn inside the cycle and edges in set Q drawn outside (or vice versa).

if it is impossible to allocate all edges to two distinct sets, the graph must not be planar.

Eg :- (1)

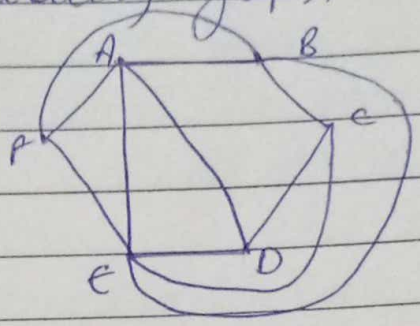


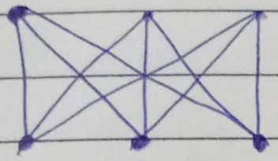
Set P	Set Q
AD	BF
	BE
	CE

Date: Page No:

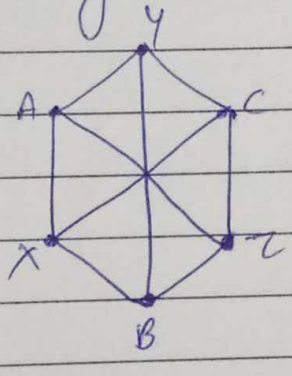
Set P	Set Q
A E	B F

redrawing graphs,



(2.)  $K_{3,3}$.

Hence, Another form of $K_{3,3}$ (remember it)



Set P	Set Q
A Z	
	B Y

We cannot add CX as, it crosses two edges but we only add if it crosses the one edges,

Thus, $K_{3,3}$ is non-planar.

(B) K.T :- States below the Page, above planarity Algorithm.

(:) Some properties :-

Theorem 1 :- (Euler's formula)

if G is a connected plane graph with p vertices, q edges, and r regions, then

$p - q + r = 2$.

if G is a tree, then G has p vertices, p-1 edges and 1 region.

$\Rightarrow \quad p - q + r = p - (p-1) + 1 = 2$

if G is not a tree, then some edge e of G is on a cycle.

Hence $G - e$ is a connected plane graph having order p and size $q-1$, and $r-1$ regions.

$\Rightarrow \quad p - (q-1) + (r-1) = 2 \quad$ (by assumption)

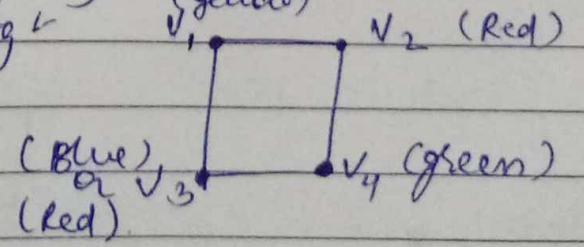
$\Rightarrow \quad p - q + r = 2$

∴ Corollary 1 :- If G is a connected planar graph with $e > 1$, then $e \leq 3v - 6$.

∴ Corollary 2 :- Every planar graph contains a vertex of degree 5 or less.

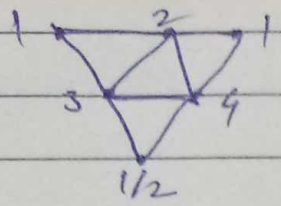
Note :- K_5 & $K_{3,3}$ are two important non-planar graphs.

→ Graph colouring :- Any adjacent nodes must not be of same colour. Eg :-



then min. no. of color required is 2, also called chromatic no. of the graph, $\chi(K_1) = 2$.

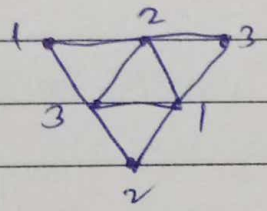
Date: Page No:



min. no. of color required = 4.

two adjacent vertices get different colours.

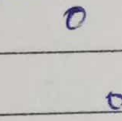
Objective :- use min. no. of colours.



min. no. = 3.

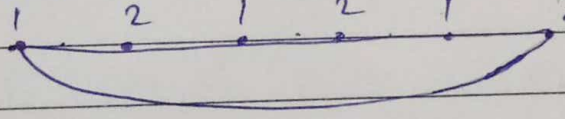
min. # Colors of G is chromatic no., $\chi(G)$.

① graphs have chromatic number one?



⇒ empty graph.

② graphs have chromatic no. 2? Conditions :-

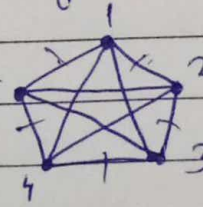


① length of graph = 6 (even)

② Cyclic graph.

then, the chromatic no. will always be 2, if satisfies both condition.

③ Complete graphs :-

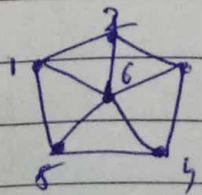


⇒ K_n then, chromatic no. are $\chi(K_n) = n$.

$= 5$

because every node is adjacent to every other nodes.

④ wheels graph :- W_5

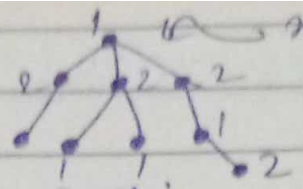


⇒ 6 edges (even)

$\chi(W_{odd}) = 3$

$\chi(W_{even}) = 4$

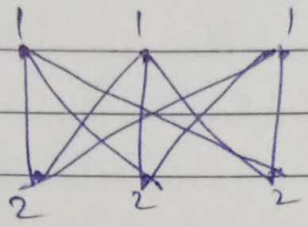
Date: Page No:

(iv) Trees :- 

χ (Trees) = 2. (always)

Pick any vertex as "root", if (unique) path from root is,
even length : 1 (red).
odd length : 2 (green)

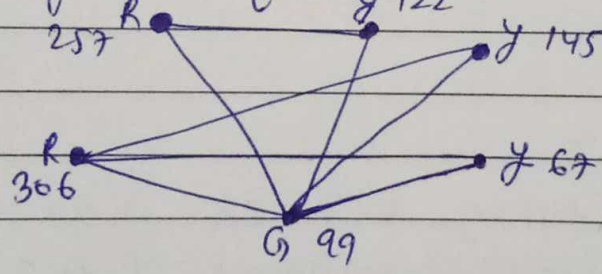
(v) $K_{3,3}$:- $\chi(K_{3,3}) = 2$ for,



$\chi(K_{2,3}) = 2$

if a bi-parted graph, then, $\chi(K_{m,n}) = 2$.

(vi) Conflict graph of Aeroplanes :-



$\chi(A) = 3$

min. 3 gates are required.

(vii) Register Allocation :- no. of register required for an algorithm we use colouring algorithm to solve it.

Step : Inputs : a, b

1 $c = a + b$

2 $d = a * c$

3 $e = c + 3$

4 $f = c - e$

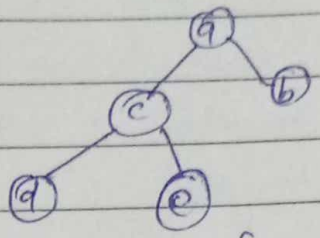
5 $g = a + f$ etc. ----

6. outputs : d g h.

Date: Page No:

(A) Graph Traversal :- Techniques :-

Basically two approaches,



Suppose, we have to traverse e,

(1) Lev 0 → lev 1 → lev 2
a → c → e (luckily or randomly found)

(2) lev 0 → lev 1 → lev 2 → lev 1 → lev 2
a → c → d → e (e = e, ✓)
Back

(B) Depth First Search (DFS) :-

(a.) Select any vertices., Mark it as visited.
(b.) Find its one of the Neighbours (unvisited).
(c.) Mark it as visited.

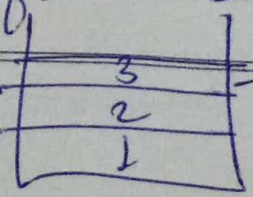
(i) Algorithm of DFS :-

(1) Start with any vertex & mark it as visited.
(2) find unvisited neighbours steps.1.
(3.) Select one of the unvisited vertex and repeat (1) (2) (3) again.
(4.) if there is no unvisited vertex. stop, {∅}

(C) Stack Data Structure :- LIFO,

recursion will takes place with the help of Stack D.S only. Eg :- 1, 2, 3

Push (1) ++
Push (2) ++
Push (3) ++

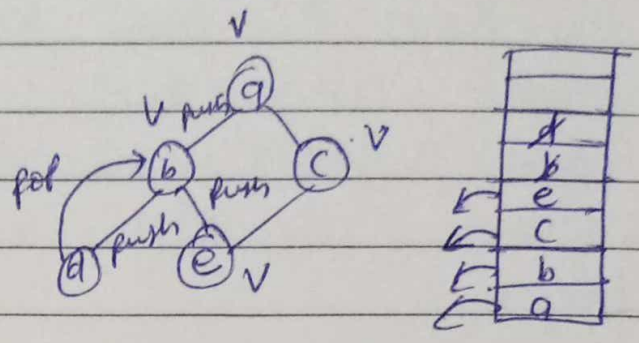


Date: Page No:

pop (3) --- , pop (2) --- , pop (1) --- .

push = Inserting data , pop = Removing data.

(iv) Stack of DFS :- Let, G, when stack becomes empty then it became a successful DFS.



(v) Pseudo Code for DFS :-

```
DFS (input : Graph G) {
    Stack S ; Integer x , t ;
while ( G has an unvisited node x )
    visit x ; push (x, S) ;
    while ( S is not empty ) {
        t := peek (S) ;
        if ( t has an unvisited neighbour y)
            { visit (y) ; push (y , S) ; }
        else
            pop (S) ;
}}}
```

max. Size of your stack that can grow = 'n'

(vi) Breadth-First Search (BFS) :-

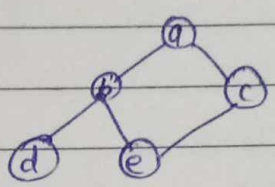
(1.) Select an universal node x , visit it , have it as the root in a BFS tree being formed . Its level is called the current level.

Date: _____ Page No: _____

(2.) From each node z in the current level, in the order in which the level nodes were visited, visit all the unvisited neighbors of z. The newly visited nodes from this level form a new level that becomes the next current level.

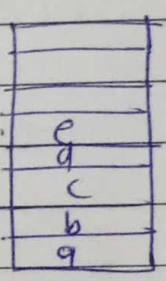
(3.) Repeat 2 step until no more nodes can be visited.

(4.) If there are still unvisited nodes, Repeat from Step 1.



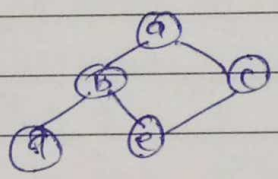
a → b → c → e → d
or. d → e.

Max. height of graph = no. of push operations.
Implement using Queue Data Structure.



Stack V/S queue.

(5.) DFS V/S BFS :-

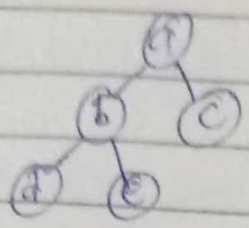


a → b → d → e → c
a → b → c → d → e.

(7.) Pseudo Code for BFS :-

```
BFS (input : graph G) {
    Queue Q ; integer x, z, y;
    while (G has an unvisited node x) {
        visit (x) ; Enqueue (x, Q);
        while (Q is not empty) {
            z := Dequeue (Q);
            for all (unvisited neighbor y of z) {
                visit (y); Enqueue (y, Q);
            }
}}}
```

Date: Page No:



		e	d	c	b	a

Queue.

Un. V. N of (a) = {b, c}.

Un. V. N of (b) = {d, e}.

Un. V. N of (c) = {ϕ}.

Un. V. N of (d) = ϕ & e = ϕ. } Queue is empty means

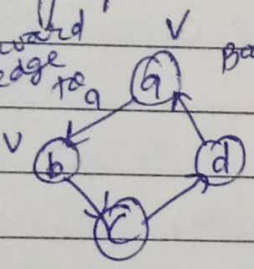
BFS is successfully traverse.

⇒ Cycle Detection using DFS / BFS :-

(1.) In case undirected graph — # no. of cycles

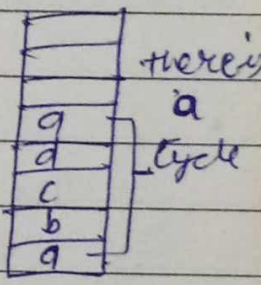
(2) directed graph → BFS / DFS — Cycle detection & # Cycles.

Eg:- (a.)



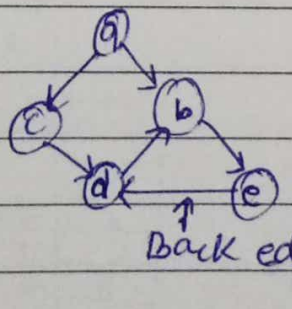
Let us take DFS,

Un. V. N of a = {b} only.

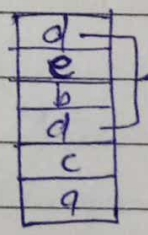


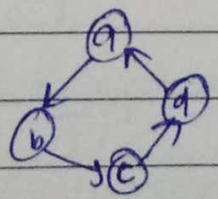
(I) whenever you add any vertex in Stack, if "v" is already in the stack means there is a cycle.

(b.)



DFS





BFS

	a	d	c	b	a

Queue list = {a, b, c, d}

Cycle is detected.

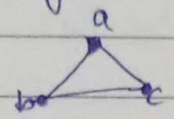
# of Cycle is detected by visiting every vertices.

Date: _____ Page No: _____

(1) Strongly Connected Components ~ App of DFS/BFS

- Connected graph, • path between every pair of vertices.

Eg :- K_3

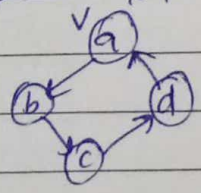


a - b
a - c
b - a
b - c
c - a
c - b

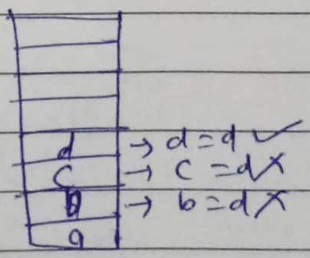
- If there is a graph and there exists a sub-graph and this graph should be connected graph its means there exists connected subgraph, so this connected subgraph is called connected component.
- A graph may contained multiple connected components.
- A strongly connected component is a subset of the vertices with paths between every pair of vertices. ∈ (directed graph).

(A) How to check path b/t {a, d} ?

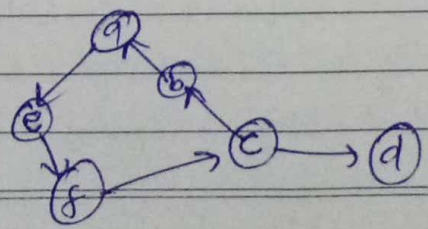
Eg :- Let, (a.)



By DFS,
Source = a
destination = d.
(d = d ✓), there is a path.



(b.)



Source v. = a
d. v = d.

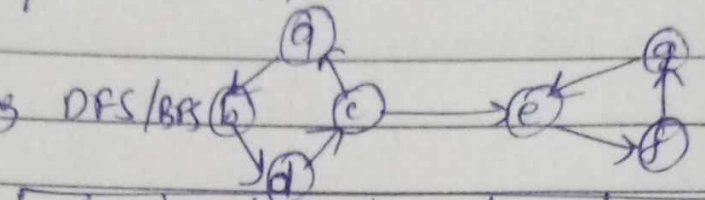
Date: Page No:

DFS, there is a path between a to d.

d	d=d ✓
b	b=d ✗
c	c=d ✗
f	f=d ✗
e	e=d ✗
a	

- DFS/BFS can be used to find path b/n two vertices.

(7) Directed graph —, Let, S.C.C = ?



path — using DFS/BFS

a	b	c	d	e	f	g.

a→b path =? — using DFS/BFS
a→c path =? — using DFS/BFS
a→d " — using DFS/BFS
a→e " — using DFS/BFS
a→f " — using DFS/BFS
a→g " — using DFS/BFS.

} Run everytime.

S.C.C. | a | b | c | d | e | f | g | ⇒ what are all vertices are reachable from 'a'.

① Initially copied all vertices, search paths, those having no path should be removed.

② its only demonstrate the paths from, a→b not b→a ?, a→c not c→a ? this is one of the connected component. , now, we can implement the method for above graph.

List | a | b | c | d | e | f | g |

scc | a | b | c | d | e | f | g |

for a only:
a-b ~ | b-a ✓
a-c ~ | c-a ✓
a-d ~ | d-a ✓
a-e ~ | e-a ✗
a-f ~ | f-a ✗

for b only:
b-c ~ | c-b ✓
b-d ~ | d-b ✓
b-e ~ | e-b ✗
b-f ~ | f-b ✗
b-g ~ | g-b ✗

Same for all remaining vertices.

At the end, S.C.C. | a | b | c | d |
for other strongly C.C remove above SCC.
S.C.C | e | f | g |.
Here, there are two SCC. using DFS / BFS.

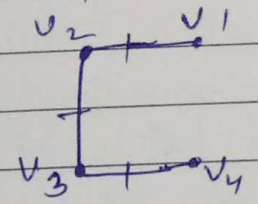
(3) MST :- Minimum Spanning tree :-

(a) Spanning tree :-

Graph → Tree ↗ Connected Graph
↘ Acyclic (not cycle)

Spanning tree :- (1) have properties of trees
(2.) # of edges in S.T is equal to (n-1) or (V-1).
where, V = no. of vertices.

Eg :-



No. of vertices = V-1 = 3.

there exists multiple Spanning tree for a given Graph.
and Counting no. of S.T are,
if "G" is Complete graph,
S.T = n^{n-2}.
if "G" is not Complete Graph,
then we have to Kirchoff's theorem applicable on it.

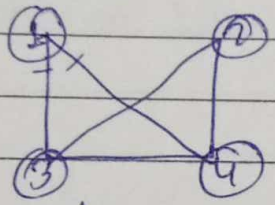
(b) Kirchoff's theorem :-
Step 1 :- Create Adjacency Matrix for the given graph.
Step 2 :- Replace all the Diagonal elements with the

Date: Page No:

degree of nodes. for Eg:- element at (1,1) position of adjacency matrix will be replaced by the degree of node 1, element at (2,2) position of adjacency matrix will be replaced by the degree of node 2, and so on.

Step 3 :- Replace all non-diagonal 1's with -1.
Step 4 :- Calculate Co-factor for any element
Step 5 :- The Cofactor that you get is the total no. of Spanning tree for that graph.

Eg :-



$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

$$M = \left[\begin{array}{c|ccc} 2 & 0 & -1 & -1 \\ \hline 0 & 2 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{array}\right]$$

Co-factor of (2) = 8.

(:) How to get Spanning tree from given graph :-
Can we get S.T using DFS / BFS.
We paying attentions towards the cycles in graphs means we ignore edges which is forming a cycle

G → B.T → ⇆ #S.T

(:) MST → weighted graph, every edge associated with weights

G (weights)

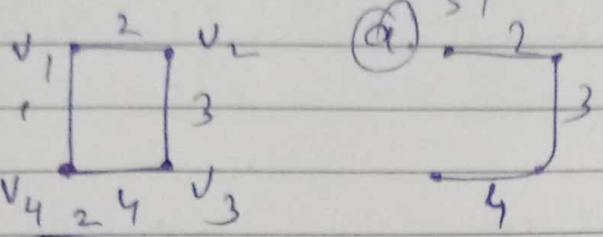
Cost 1 — ST1 | ST2 — Cost 2 | STn — Cost n

Cost of ST1 = sum of Cost of all the edges in S.T.

Date: Page No:

Hence, the Spanning tree gives minimum cost is called M.S.T.

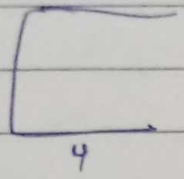
Eg:-



(a) ST

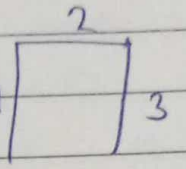
Cost = 2+3+4 = 9.

(b)



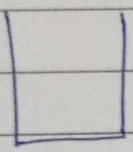
1+2+4 = 7

(c)



= 1+2+3 = 6.

(d.)



1+4+3 = 8

Hence, (c) one is MST as total cost is 6.

(2) How to get MST from given Graph 'G' (weighted) ?

- Kruskal Algo
- Prims algo

} M.S.T

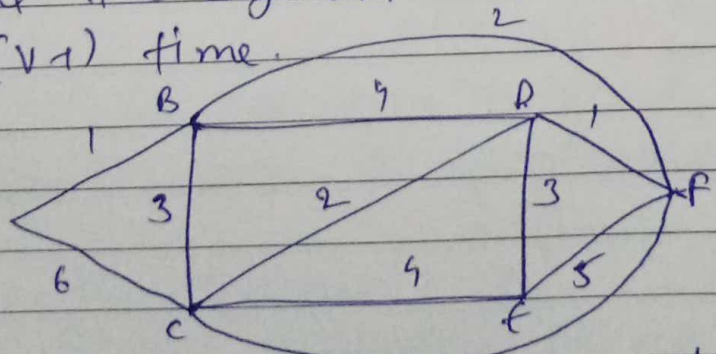
(3) Kruskal's Algorithm :-

(1) Sort the edges in a graph in ascending order.

(2) every time pick up the least cost edge and add it to M.S.T so that no cycle is formed.

(3) update the edge list and continue step (2) & (3) for (V-1) time.

Let,

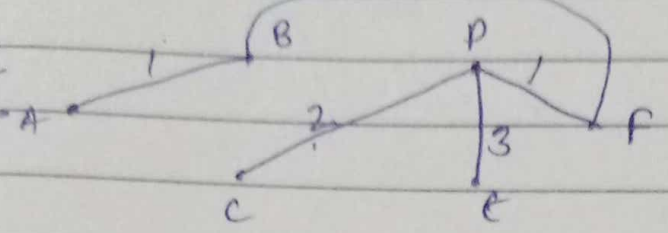


(1) (A,B), (D,F), (B,E), (C,D), (D,E), (B,C), (B,D)

edge Cost 1 1 2 2 3 3 4

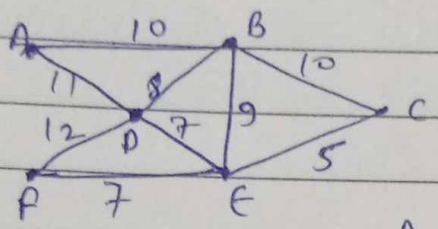
(C,E) , (E,B) , (A,C) , (C,F) , V=6.
4 5 2 6 7
upto (V-1)=5 edges.

M.S.T

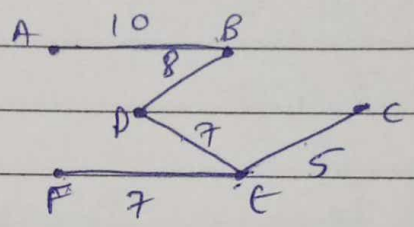


1+2+2+3+1=9 Cost.

(B.)



V=6.
upto (V-1)=5



10+5+7+7+8 = M.S.T

We should take care of not forming cycle & loop runs until (V-1) edges are formed.

(V) Prims Algorithm:-

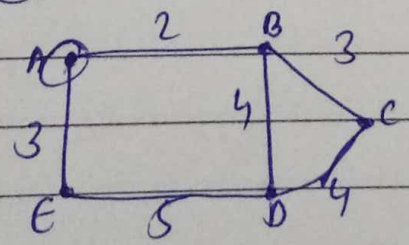
(I) Select any vertex Randomly.

(II) Among all the edges not in "S.T" that are incident with any vertex in "S.T". And not form the cycle. is added.

(III) Repeat (II) until S.T Contains exactly (V-1) edges.

Let

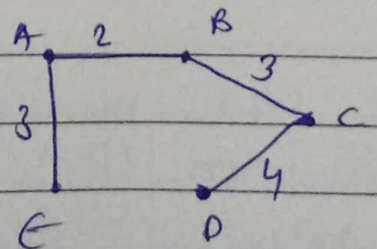
(A.)



S.T

{(A,B)} } { (B,C) (C,D) (B,D) (E,D) } edge in G

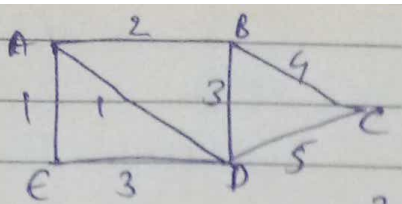
(A,E)



2+3+3+4=12

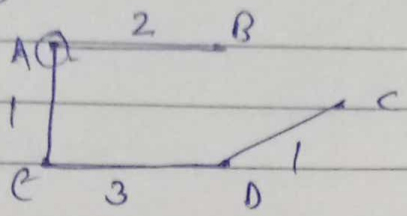
Date: Page No:

(B.)



V=5

(A,B) (B,C) (C,D)
(C,D) (D,B) -- (A,E)



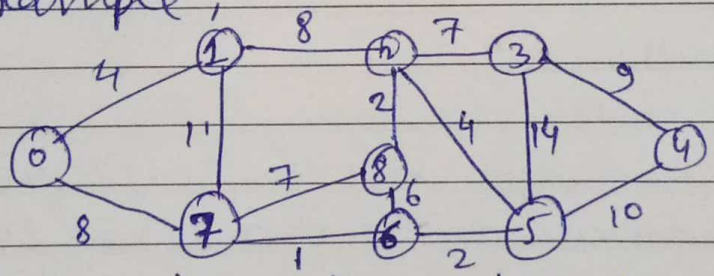
(D,C) will not add because it is not an incident edge of either of A or E [not connected by line].
(B,D) not taken as it formed cycle.

1+1+2+3 = 7 cost

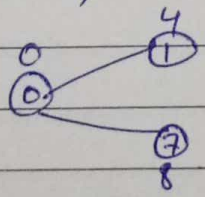
(b) ~~Shortest path tree :- Dijakstras Algo :-~~

Step 1:- Make a Set and assign 0 to all and proceed as below example,

Suppose,

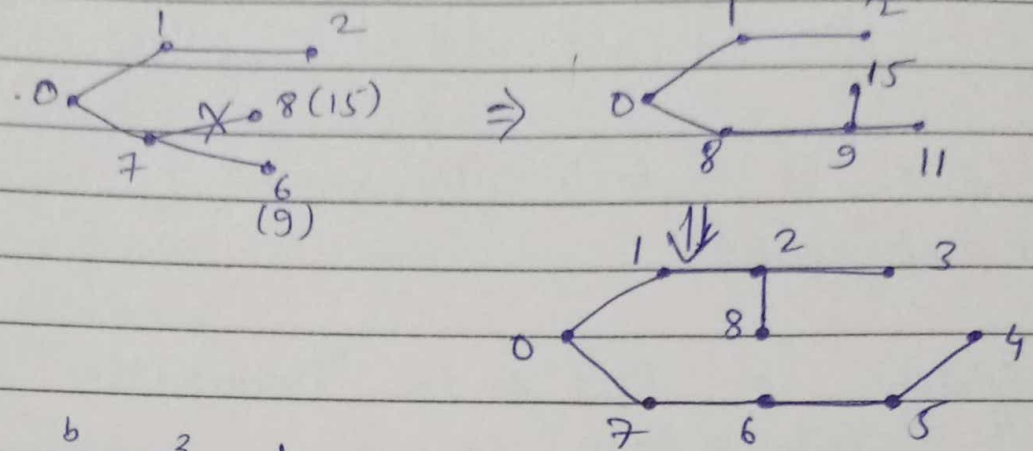


Let, The vertex 0 is picked, include it in set, takes its adjacent vertices i.e, 1 & 7, and are updated as 4, & 8.

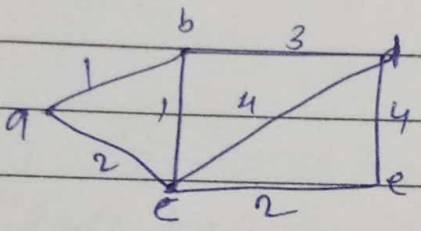


Pick the vertex with minimum distance value and not already included in SPT. The vertex 1 is picked and added to SPT. now SPT becomes {0,1}. The distance value of vertex 2 becomes 12.

Again pick min. distance as $8 < 12$ then pick 7. now set becomes {0, 1, 7}. Update vertices of 7 and so on.



(B.)

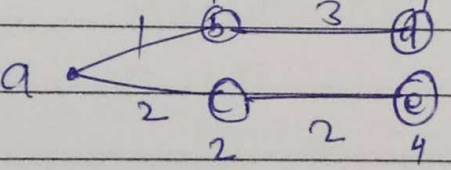


a – b
a – c
a – d
a – e

Source vertex : a

{a, b, d} {c, e . . .}

SPT



SPT

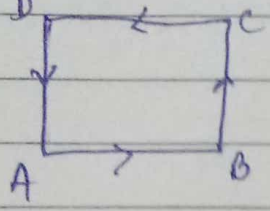
a – b = 1
a – c = 2
a – d = 4
a – e = 4

- The only difference in this and prims algo is that we add cumulative distance as (1+3 = 4).
- This is called Single source SPT and for every source we called it All-pairs SPT.
- The graph contain any Negative weight or cycle then this algo will fail.

(7.) **Euler's Path** :- If there is a 'G' graph, there exists a path such that which contains each

Date: Page No:

"edge" of the graph exactly 'once'.



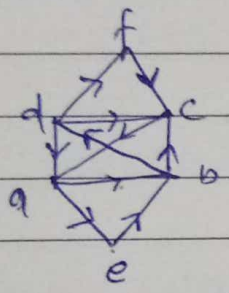
$A \to B \to C \to D \to A$

$\{(A,B)\ (B,C)\ (C,D)\ (D,A)\}$

(1) Euler's path ⇒ Euler's Circuit

(2) closed Euler's path ⇗

☺ A graph 'G' said to be "Traversable" iff there exist "Euler's path" in G.



(a) → b → c → d → a → c → b → d → f → (a)

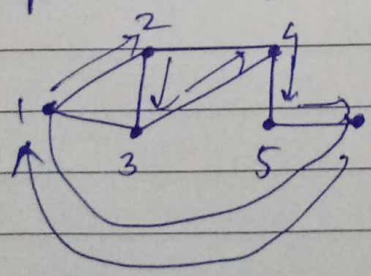
10 edges, Euler's path ⇓ Euler's Circuit ⇙

Graph is called traversable

⑦ Hamiltonian Graphs :-

① Hamiltonian Cycle :- in a path every "vertex" of the graph exactly 'once'.

Eg,



1-2-3-4-5-6→1

if a graph contains Hamiltonian Cycle then this graph is called Hamiltonian Graphs.

Graph theory End.

Date: Page No:

Combinatorics :-

Topics :-

1. P & C without repetition.
2. P & C with repetition.
3. Binomial theorm & Coefficient.
4. P & C Algorithms.

(i) Permutations :-

Theorm 1 :- if n is a positive and r is an Integer with $1 \le r \le n$, then there are,

$$P(n,r) = n(n-1)(n-2)\cdots(n-r+1)$$

r- permutations of a set with n distinct elements.

Col 1 :- $$P(n,r) = \frac{n!}{(n-r)!}, \quad 0 \le r \le n.$$

(ii) Combinations :-

$$C(n,r) = \frac{n!}{r!\,(n-r)!}, \quad 0 \le r \le n.$$

1. $C(n,r) = C(n,n-r)$.
2. $P(n,r) = r!\, C(n,r)$.
3. $\sum_{k=0}^{n} {}^nC_k = 2^n$
4. $\sum_{k=0}^{n} (-1)^k \cdot {}^nC_k = 0$

Date: _____ Page No: _____

(5.) $^nC_0 + {}^nC_2 + {}^nC_4 + \cdots = {}^nC_1 + {}^nC_3 + {}^nC_5 + \cdots$

(6.) Pascal's Identity :- Let n and k be positive integers with $n \geq k$, Then,

$$^{n+1}C_k = {}^nC_{k-1} + {}^nC_k.$$

(7.) Permutations with Repetition :-

Theorm 1 :- The no. of r-permutations of a set of n objects with Repetition allowed is n^r.

Theorm 2 :- The no. of different of n objects, where there are n_1 indistinguishable objects of type 1, n_2 indistinguishable objects of type 2, ..., and n_k indistinguishable objects of type k, is

$$\frac{n!}{n_1!\, n_2! \cdots n_k!}$$

P & C with and without Repetition.

Type	Repetition Allowed	Formula
r-Perm.	NO	$n! / (n-r)!$
r-Comb.	NO	$n! / r!\,(n-r)!$
r-perm.	Yes	n^r
r-Comb.	yes	$\frac{(n+r-1)!}{r!\,(n-1)!}$

Date: Page No:

① Steps to next larger permutation in increasing order.

1. Given permutation $a_1, a_2 \cdots a_n$
2. First, find the integers a_j and a_{j+1} with $a_j < a_{j+1}$ and $a_{j+1} > a_{j+2} > \cdots > a_n$.
   i.e., the last pair of adjacent integers in the permutation where the first integer in the pair is smaller than the second.

3. Then, the next larger permutation in lexicographic order is obtained by putting in j^{th} position the least integer among $a_{j+1}, a_{j+2}, \cdots$, and an that is greater than a_j and listing in increasing order the rest of the integers $a_j, a_{j+1}, \cdots a_n$ in positions $j+1$ to n.

4. It is easy to see that there is no other permutation larger than the permutation $a_1 a_2 \cdots a_n$ but smaller than the new permutation produced.

② Algorithm 1 Generating the next permutation in lexicographic order.

procedure next permutation ($a_1 a_2 \cdots a_n$: permutation of $\{1, 2, \cdots n\}$ not equal to $n\ n-1 \cdots 2\ 1$).

```
j := n-1
while a_j > a_{j+1}
    j := j-1
{ j is the largest subscript with a_j < a_{j+1} }
k := n
while a_j > a_k
```

Date: Page No:

$k := k+1$

{ q_k is the smallest integer greater than q_j to the right of q_j }.

interchange q_j and q_k.

$r := n$

$s := j+1$

while $r > s$

{interchange q_r and q_s.

$r := r-1$

$s := s+1$

{ this puts the tail end of the permutation after the j^{th} position in increasing order }

{ $a_1 a_2 \ldots a_n$ is now the next permutation }

(5.) Generating the next larger Bit String :- Locating the first position from the right that is 0 (making this 1), then changing all the 1s to the right of this position to 0s. Eg:- 10 0010 0111

→ 10 0010 1000

Algorithm 2 Generating the next larger Bit String :-

procedure next bit string ($b_{n-1}, b_{n-2} \ldots b_1, b_0$: bit string not equal to $11\ldots11$)

$i := 0$

while $b_i = 1$

$b_i := 0$

$i := i+1$

Date: Page No:

$b_i := 1$

{ $b_{n-1}, b_{n-2} \cdots b_1, b_0$ is now the next bit string }.

(6) Algorithm for Generating the r-Combinations of the set $\{1, 2, 3 \cdots n\}$:-

Hint : First, locate the last element a_i in the sequence such that $a_i \neq n-r+i$. Then, replace a_i with a_i+1 and a_j with $a_i + j - i + 1$, for $j = i+1, i+2, \cdots, r$.

(7) Procedure next r-comb{--}: (proper subset of $\{1, 2, \cdots n\}$ not equal to $\{n-r+1, \cdots, n\}$ with $a_1 < a_2 < \cdots < a_r$)

$i := r$

while $a_i = n - r + i$

$i := i - 1$

$a_i := a_i + 1$

for $j := i + 1$ to r

$a_j := a_i + j - i$

{ $a_1, a_2, \cdots, a_r$ } is now the next combination }

(8) Topics :-

1. Basics of Counting :-
2. Pigeonhole principle
3. Permutations & Combinations
4. Binomial & Multinomial Coefficients
5. Advanced Counting Techniques

Rest in PPT Slide.

Date: Page No: