MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

Autonomous Institution – UGC, Govt. of India

Department of COMPUTATIONAL INTELLIGENCE (CSE-AIML)

B. TECH (R-22 Regulation)
(II YEAR – II SEM)

2024-25
DISCRETE MATHEMATICS
(R22A0028)

LECTURE NOTES

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Department of Computational Intelligence
CSE (Artificial Intelligence and Machine Learning)

Vision

To be a premier centre for academic excellence and research through innovative interdisciplinary collaborations and making significant contributions to the community, organizations, and society as a whole.

Mission

- ❖ To impart cutting-edge Artificial Intelligence technology in accordance with industry norms.
- ❖ To instill in students a desire to conduct research in order to tackle challenging technical problems for industry.
- ❖ To develop effective graduates who are responsible for their professional growth, leadership qualities and are committed to lifelong learning.

QUALITY POLICY

- ❖ To provide sophisticated technical infrastructure and to inspire students to reach their full potential.
- ❖ To provide students with a solid academic and research environment for a comprehensive learning experience.
- ❖ To provide research development, consulting, testing, and customized training to satisfy specific industrial demands, thereby encouraging self-employment and entrepreneurship among students.

For more information: www.mrcet.ac.in

Dept. of CSE-AIML

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

B. TECH- II- YEAR- II-SEM   L T/P/D   C

                                                3   1/-/-   4

(R22A0028)DISCRETE MATHEMATICS

Course Objectives:
● Introduces elementary discrete mathematics for computer science and engineering.
● Topics include formal logic notation, methods of proof, induction, sets, relations, algebraic structures, elementary graph theory, permutations and combinations, counting principles; recurrence relations and generating functions.

UNIT - I
Mathematical logic: Introduction, Statements and Notation, Connectives, Truth tables, Well formed formulas, Tautology, Contradiction, Contingency, Logical equivalence, Normal Forms, Theory of Inference for the Statement Calculus, The Predicate Calculus, Inference Theory of the Predicate Calculus.

UNIT - II
Set theory: Introduction, Basic Concepts of Set Theory, Representation of Discrete Structures, Relations, Types of relations, Partial order relation, POSET, External elements in POSET, Lattices, Functions, Types of functions, inverse of functions, invertible functions and Composition of functions

UNIT - III
Algebraic Structures: Introduction, Algebraic Systems, Semi groups, Monoids, Group, Abelian group, Some particular Groups-Klein 4- group, Additive group of modulo n, Multiplicative group of modulo p. Homomorphism and isomorphism in Groups.

UNIT - IV
Combinatorics: . Permutation and combination, Principle of Inclusion and Exclusion. Recurrence relations, Generating functions,Solving of recurrence relations of first,second and higher order Homogeneous and Non homogeneous using characteristic equation and generating functions.

UNIT-V
Graph Theory: Basic Concepts, Types of graphs, Sub graphs, Isomorphism, Walk, Trail and Path, Hamiltonial and Eulerian Graphs, Planar Graphs, Euler's Formula, Multi-graph, Chromatic Numbers,The Four-Color Problem. Trees and their Properties, Spanning Trees, Directed Trees, BinaryTrees.

TEXT BOOKS:
1. Discrete Mathematical Structures with Applications to Computer Science: J.P. Tremblay, R. Manohar, McGraw-Hill, 1st ed.
2. Discrete Mathematics for Computer Scientists & Mathematicians: Joe l. Mott, Abraham

Kandel, Teodore P. Baker, Prentis Hall of India, 2nd ed.

REFERENCE BOOKS:
1. Discrete and Combinatorial Mathematics - an applied introduction: Ralph.P. Grimald, Pearson education, 5th edition.
2. Discrete Mathematical Structures: Thomas Kosy, Tata McGraw Hill publishing co.

Course Outcomes:
- Understand and construct precise mathematical proofs
- Apply logic and set theory to formulate precise statements
- Analyze and solve counting problems on finite and discrete structures
- Describe and manipulate sequences
- Apply graph theory in solving computing problems

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Computational Intelligence

CSE (Artificial Intelligence and Machine Learning)

INDEX

UNIT-I

Mathematical Logic

Statements and notations:

A proposition or statement is a declarative sentence that is either true or false (but not both).

For instance, the following are propositions:

- Paris is in France (true),
- London is in Denmark(false),
- —2 < 4 (true),
- 4 = 7 (false)

However the following are not propositions:

- what is your name? (this is a question),
- do your homework (this is a command),
- this sentence is false (neither true nor false),
- x is an even number (it depends on what x represents),
- Socrates (it is not even a sentence).

The truth or falsehood of a proposition is called its truth value.

Connectives:

Connectives are used for making compound propositions. The main ones are the following (p and q represent given propositions):

- not
- conjunction
- disjunction
- conditional
- biconditional

Truth Tables:

Logical negation

Logical negation is an operation on one logical value, typically the value of a proposition that produces a value of true if its operand is false and a value of false if its operand is true.

The truth table for **NOT p** (also written as **¬p** or **~p**) is as follows:

p	¬p
T	F

F	T

Logical conjunction

Logical conjunction is an operation on two logical values, typically the values of two propositions, that produces a value of true if both of its operands are true.

The truth table for **p AND q** (also written as **p ∧ q**, **p & q**, or **p q**) is as follows:

p	q	p ∧ q
T	T	T
T	F	F
F	T	F
F	F	F

In ordinary language terms, if both p and q are true, then the conjunction p ∧ q is true. For all other assignments of logical values to p and to q the conjunction p ∧ q is false.

It can also be said that if p, then p ∧ q is q, otherwise p ∧ q is p.

Logical disjunction

Logical disjunction is an operation on two logical values, typically the values of two propositions, that produces a value of true if at least one of its operands is true.

The truth table for **p OR q** (also written as **p ∨ q**, **p || q**, or **p + q**) is as follows:

p	q	p Vq
T	T	T
T	F	T
F	T	T
F	F	F

Logical implication

Logical implication and the material conditional are both associated with an operation on two logical values, typically the values of two propositions, that produces a value of false just in the singular case the first operand is true and the second operand is false.The truth table associated with the material conditional **if p then q** (symbolized as **p → q**) and the logical implication **p implies q** (symbolized as **p ⇒ q**) is as follows:

p	q	p → q
T	T	T
T	F	F
F	T	T

F	F	T

Logical equality

Logical equality (also known as biconditional) is an operation on two logical values, typically the values of two propositions, that produces a value of true if both operands are false or both operands are true.The truth table for **p XNOR q** (also written as **p ↔ q** ,**p = q**, or **p ≡ q**) is as follows:

p	**q**	**p ≡ q**
T	T	T
T	F	F
F	T	F
F	F	T

Well formed formulas(wff):

Not all strings can represent propositions of the predicate logic. Those which produce a proposition when their symbols are interpreted must follow the rules given below, and they are called wffs(well-formed formulas) of the first order predicate logic.

Rules for constructing Wffs

A predicate name followed by a list of variables such as P(x, y), where P ispredicate name, and x and y are variables, is called an atomic formula.

A well formed formula of predicate calculus is obtained by using the following rules.

• An atomic formula is a wff.

• If A is a wff, then 7A is also a wff.

• If A and B are wffs, then (A V B), (A ∧ B), (A → B) and (A D B).

• If A is a wff and x is a any variable, then (x)A and ($x)A are wffs.

• Only those formulas obtained by using (1) to (4) are wffs.

Since we will be concerned with only wffs, we shall use the term formulas for wff. We shall follow the same conventions regarding the use of parentheses as was done in the case of statement formulas.

Wffs are constructed using the following rules:

• True and False are wffs.

• Each propositional constant (i.e. specific proposition), and each propositional variable (i.e. a variable representing propositions) are wffs.

• Each atomic formula (i.e. a specific predicate with variables) is a wff.

• If A, B, and C are wffs, then so areA, (A ∧ B), (A ∨ B), (A->B),and (A<->B)

•   If x is a variable (representing objects of the universe of discourse), and A is a wff, then so are ∀x A and ∃x A .

The strings that cannot be constructed by using those rules are not wffs. For example, ∀ xB(x)R(x), and B(∃x) are NOT wffs, NOR are B(R(x)), and B(∃x R(x)) . More examples: To express the fact that Tom is taller than John, we can use the atomic formula taller(Tom, John), which is a wff. This wff can also be part of some compound statement such as taller(Tom, John) ∧taller(John, Tom), which is also a wff. If x is a variable representing people in the world, then taller(x,Tom), ∀x taller(x,Tom), ∃x taller(x,Tom), ∃ x ∀y taller(x,y) are all wffs among others. However, taller(∃ x,John) and taller(Tom∧Mary, Jim), for example, are NOT wffs.

Tautology, Contradiction, Contingency:

A proposition is said to be a tautology if its truth value is T for any assignment of truth values to its components. Example: The proposition p ∨ ¬p is a tautology.

A proposition is said to be a contradiction if its truth value is F for any assignment of truth values to its components. Example: The proposition p ∧ ¬p is a contradiction.

A proposition that is neither a tautology nor a contradiction is called a contingency.

p	¬p	p ∨ ¬p	p ∧ ¬p
T	F	T	F
T	F	T	F
F	T	T	F
F	T	T	F

Equivalence Implication:

We say that the statements r and s are logically equivalent if their truth tables are identical.

For example the truth table of

p	q	$\neg p \lor q$
T	T	T
T	F	F
F	T	T
F	F	T

$\neg p \lor q$ is equivalent to $p \to q$. It is easily shown that the statements r and s are equivalent if and only if (r<->s) is a tautology.

Normal forms:

Let A(P1, P2, P3, …, Pn) be a statement formula where P1, P2, P3, …, Pn are the atomic variables. If A has truth value T for all possible assignments of the truth values to the

variables P1, P2, P3, …, Pn , then A is said to be a tautology. If A has truth value F, then A is said to be identically false or a contradiction.

Disjunctive Normal Forms

A product of the variables and their negations in a formula is called an elementary product. A sum of the variables and their negations is called an elementary sum. That is, a sum of elementary products is called a disjunctive normal form of the given formula.

Example:

A (1)

$(A \wedge B) \vee (! A \wedge C)$ (2)

$(A \wedge B \wedge ! A) \vee (C \wedge ! B) \vee (A \wedge ! C)$ (3)

$A \wedge B$ (4)

$A \vee (B \wedge C),$ (5)

Conjunctive Normal Forms

A formula which is equivalent to a given formula and which consists of a product of elementary sums is called a conjunctive normal form of a given formula.

Example:

A (1)

$(A \vee B) \wedge (! A \vee C)$ (2)

$A \vee B$ (3)

$A \wedge (B \vee C),$ (4)

PDNF:

It stands for Principal Disjunctive Normal Form. It refers to the Sum of Products, i.e., SOP. For eg. : If P, Q, R are the variables then $(P . Q' . R) + (P' . Q . R) + (P . Q . R')$ is an example of an expression in PDNF. Here '+' i.e. sum is the main operator.

PCNF:

It stands for Principal Conjunctive Normal Form. It refers to the Product of Sums, i.e., POS. For eg. : If P, Q, R are the variables then $(P + Q'+ R).(P'+ Q + R).(P + Q + R')$ is an example of an expression in PCNF. Here '.' i.e. product is the main operator.

Properties of PCNF and PDNF:

- Every PDNF or PCNF corresponds to a unique Boolean Expression and vice versa.
- If X and Y are two Boolean expressions then, X is equivalent to Y if and only if PDNF(X) = PDNF(Y) or PCNF(X) = PCNF(Y).
- For a Boolean Expression, if PCNF has m terms and PDNF has n terms, then the number of variables in such a Boolean expression = $\log_{2} (m + n)$.

Predicates

Predicative logic:

A predicate or propositional function is a statement containing variables. For instance

- x + 2 = 7,
- X is American,
- x < y, —p is a prime number are predicates.

The truth value of the predicate depends on the value assigned to its variables. For instance if we replace x with 1 in the predicate x + 2 = 7 we obtain 1 + 2 = 7, which is false, but if we replace it with 5 we get 5+ 2 = 7, which is true. We represent a predicate by a letter followed by the variables enclosed between parenthesis: P (x), Q(x, y), etc. An example for P (x) is a value of x for which P (x) is true. A counterexample is a value of x for which P (x) is false. So, 5 is an example for x + 2 = 7, while 1 is a counterexample. Each variable in a predicate is assumed to belong to a universe (or domain) of discourse, for instance in the predicate n is an odd integer 'n' represents an integer, so the universe of discourse of n is the set of all integers. In X is American we may assume that X is a human being, so in this case the universe of discourse is the set of all human beings.

Qantifiers:

In predicate logic, predicates are used alongside quantifiers to express the extent to which a predicate is true over a range of elements. Using quantifiers to create such propositions is called quantification.

There are two types of quantification-

1. Universal Quantification- Mathematical statements sometimes assert that a property is true for all the values of a variable in a particular domain, called the domain of discourse. Such a statement is expressed using universal quantification.

The universal quantification of P(x) for a particular domain is the proposition that asserts that P(x) is true for all values of x in this domain. The domain is very important here since it decides the possible values of x. The meaning of the universal quantification of P(x) changes when the domain is changed. The domain must be specified when a universal quantification is used, as without it, it has no meaning.

Formally,

The universal quantification of P(x) is the statement

"P(x) for all values of x in the domain"

The notation ∀P(x) denotes the universal quantification of P(x).

Here ∀is called the universal quantifier.

∀P(x) is read as "for all x P(x)".

Example 1: Let P(x) be the statement "x + 2 > x". What is the truth value of the statement ∀ xP(x)?

Solution: As x+2 is greater than x for any real number, so P(x) \equiv T for all x or ∀xP(x) \equiv T.

2. Existential Quantification- Some mathematical statements assert that there is an element with a certain property. Such statements are expressed by existential quantification. Existential quantification can be used to form a proposition that is true if and only if P(x) is true for at least one value of x in the domain.

Formally,

The existential quantification of P(x) is the statement

"There exists an element x in the domain such that P(x)"

The notation ∃P(x) denotes the existential quantification of P(x).

Here ∃ s called the existential quantifier.

∃P(x) is read as "There is atleast one such x such that P(x)".

Example : Let P(x) be the statement "x > 5″. What is the truth value of the statement ∃xP(x) ?

Solution: P(x) is true for all real numbers greater than 5 and false for all real numbers less than 5. So ∃ xP(x) \equiv T.

Free & Bound variables:

Let's now turn to a rather important topic: the distinction between free variables and bound variables.

Have a look at the following formula:

$\neg(\text{THERAPIST}(x) \lor \forall x(\text{MORON}(x) \land \forall y \text{PERSON}(y)))$

The first occurrence of x is free, whereas the second and third occurrences of x are bound, namely by the first occurrence of the quantifier . The first and second occurrences of the variable y are also bound, namely by the second occurrence of the quantifier .

Informally, the concept of a bound variable can be explained as follows: Recall that quantifications are generally of the form:

$\forall x \varphi$ or $\exists x \varphi$

where x may be any variable. Generally, all occurrences of this variable within the quantification are bound.

Here's a full formal simultaneous definition of free and bound:

1. Any occurrence of any variable is free in any atomic formula.

2. No occurrence of any variable is bound in any atomic formula.

If a formula contains no occurrences of free variables we call it a **sentence**.

Rules of inference:

The two rules of inference are called rules P and T.

Rule P: A premise may be introduced at any point in the derivation.

Rule T: A formula S may be introduced in a derivation if s is tautologically implied by any one or more of the preceding formulas in the derivation.

Before proceeding the actual process of derivation, some important list of implications and equivalences are given in the following tables.

Implications

I1 $P \land Q \Rightarrow P$ } Simplification

I2 $PQ \land \Rightarrow Q$
I3 $P \Rightarrow P \lor Q$ } Addition
I4 $Q \Rightarrow P \lor Q$
I5 $7P \Rightarrow P \rightarrow Q$
I6 $Q \Rightarrow P \rightarrow Q$
I7 $7(P \rightarrow Q) \Rightarrow P$
I8 $7(P \rightarrow Q) > Q$
I9 $P, Q \Rightarrow P \land Q$
I10 $7P, P \lor Q \Rightarrow Q$ (disjunctive syllogism)

I11 $P, P \rightarrow Q \Rightarrow Q$ (modus ponens)

I12 $7Q, P \rightarrow Q \Rightarrow 7P$ (modus tollens)

I13 $P \rightarrow Q, Q \rightarrow R \Rightarrow P \rightarrow R$ (hypothetical syllogism)

I14 $P \lor Q, P \rightarrow Q, Q \rightarrow R \Rightarrow R$ (dilemma)

Equivalences

E1 $77P \Leftrightarrow P$
E2 $P \land Q \Leftrightarrow Q \land P$ } Commutative laws
E3 $P \lor Q \Leftrightarrow Q \lor P$
E4 $(P \land Q) \land R \Leftrightarrow P \land (Q \land R)$ } Associative laws
E5 $(P \lor Q) \lor R \Leftrightarrow P \lor (Q \lor R)$
E6 $P \land (Q \lor R) \Leftrightarrow (P \land Q) \lor (P \land R)$ }   Distributive laws
E7 $P \lor (Q \land R) \Leftrightarrow (P \lor Q) \land (P \lor R)$
E8 $7(P \land Q) \Leftrightarrow 7P \lor 7Q$

E9 $7(P \lor Q) \Leftrightarrow 7P \land 7Q$ }   De Morgan's laws

Example 1.Show that R is logically derived from $P \rightarrow Q, Q \rightarrow R$, and P

Solution.   {1}   (1) $P \rightarrow Q$ Rule P

            {2}   (2)   P   Rule P

{1, 2}	(3)	Q	Rule (1), (2) and I11
{4}	(4)	Q → R	Rule P
{1, 2, 4}	(5)	R	Rule (3), (4) and I11.

Example 2. Show that S V R tautologically implied by (P V Q) ∧ (P → R) ∧ (Q → S).

Solution .

{1}	(1)	P V Q	Rule P
{1}	(2)	7P → Q	T, (1), E1 and E16
{3}	(3)	Q → S	P
{1, 3}	(4)	7P → S	T, (2), (3), and I13
{1, 3}	(5)	7S → P	T, (4), E13 and E1
{6}	(6)	P → R	P
{1, 3, 6}	(7)	7S → R	T, (5), (6), and I13
{1, 3, 6}	(8)	S V R	T, (7), E16 and E1

Example 3 . Prove that R ∧ (P V Q) is a valid conclusion from the premises PVQ , Q → R, P → M and 7M.

Solution .

{1}	(1)	P → M	P
{2}	(2)	7M	P
{1, 2}	(3)	7P	T, (1), (2), and I12
{4}	(4)	P V Q	P
{1, 2 , 4}	(5)	Q	T, (3), (4), and I10.
{6}	(6)	Q → R	P
{1, 2, 4, 6}	(7)	R	T, (5), (6) and I11
{1, 2, 4, 6}	(8)	R ∧ (PVQ)	T, (4), (7), and I9.

There is a third inference rule, known as rule CP or rule of conditional proof.

Rule CP: If we can derives s from R and a set of premises , then we can derive R → S from the set of premises alone.

• Let P denote the conjunction of the set of premises and let R be any formula The above equivalence states that if R is included as an additional premise and S is derived from P ∧ R then R → S can be derived from the premises P alone.

• Rule CP is also called the deduction theorem and is generally used if the conclusion is of the form R → S. In such cases, R is taken as an additional premise and S is derived from the given premises and R.

Example .Show that R → S can be derived from the premises P → (Q → S), 7R V P , and Q.

Solution.

{1}	(1)	7R V P	P
{2}	(2)	R	P, assumed premise
{1, 2}	(3)	P	T, (1), (2), and I10
{4}	(4)	P → (Q → S)	P
{1, 2, 4}	(5)	Q → S	T, (3), (4), and I11
{6}	(6)	Q	P
{1, 2, 4, 6}	(7)	S	T, (5), (6), and I11
{1, 4, 6}	(8)	R → S	CP.

Example Show that P → S can be derived from the premises, 7P V Q, 7Q V R, and R → S .

Solution.

{1}	(1)	7P V Q	P
{2}	(2)	P	P, assumed premise
{1, 2}	(3)	Q	T, (1), (2) and I11
{4}	(4)	7Q V R	P
{1, 2, 4}	(5)	R	T, (3), (4) and I11
{6}	(6)	R → S	P
{1, 2, 4, 6}	(7)	S	T, (5), (6) and I11
{2, 7}	(8)	P → S	CP

Consistency of premises:

Consistency

A set of formulas H1, H2, …, Hm is said to be consistent if their conjunction has the truth value T for some assignment of the truth values to be atomic appearing in H1, H2, …, Hm.

Inconsistency

If for every assignment of the truth values to the atomic variables, at least one of the formulas H1, H2, … Hm is false, so that their conjunction is identically false, then the formulas H1, H2, …, Hm are called inconsistent.

A set of formulas H1, H2, …, Hm is inconsistent, if their conjunction implies a contradiction, that is H1∧ H2∧ … ∧ Hm => R ∧ 7R Where R is any formula. Note that R ∧ 7R is a contradiction and it is necessary and sufficient that H1, H2, …,Hm are inconsistent the formula.

Indirect method of proof

In order to show that a conclusion C follows logically from the premises H1, H2,…, Hm, we assume that C is false and consider 7C as an additional premise. If the new set of premises is

inconsistent, so that they imply a contradiction, then the assumption that 7C is true does not hold simultaneously with H1∧ H2∧ ….. ∧ Hm being true. Therefore, C is true whenever H1∧ H2∧ …..∧ Hm is true. Thus, C follows logically from the premises H1, H2 ….., Hm.

Example Show that 7(P ∧ Q) follows from 7P∧ 7Q.

Solution.

We introduce 77 (P∧ Q) as an additional premise and show that this additional premise leads to a contradiction.

{1}	(1)	77(P∧ Q)	P assumed premise
{1}	(2)	P∧ Q	T, (1) and E1
{1}	(3)	P	T, (2) and I1
{1}	{4} 7P∧7Q		P
{4}	(5)	7P	T, (4) and I1
{1, 4}	(6)	P∧ 7P	T, (3), (5) and I9

Example Show that the following premises are inconsistent.

•   If Jack misses many classes through illness, then he fails high school.

•   If Jack fails high school, then he is uneducated.

•   If Jack reads a lot of books, then he is not uneducated.

•   Jack misses many classes through illness and reads a lot of books.

Solution.

P: Jack misses many classes. Q: Jack fails high school.

R: Jack reads a lot of books. S: Jack is uneducated.

The premises are P→ Q, Q → S, R→   7S and P∧ R

{1}	(1)	P→Q	P
{2}	(2)	Q→ S	P
{1, 2}	(3)	P → S	T, (1), (2) and I13
{4}	(4)	R→ 7S	P
{4}	(5)	S → 7R	T, (4), and E18
{1, 2, 4}	(6)	P→7R	T, (3), (5) and I13
{1, 2, 4}	(7)	7PV7R	T, (6) and E16
{1, 2, 4}	(8)	7(P∧R)	T, (7) and E8
{9}	(9)	P∧ R	P
{1, 2, 4, 9}	(10)	(P∧ R) ∧ 7(P∧ R)	T, (8), (9) and I9

The rules above can be summed up in the following table. The "Tautology" column shows how to interpret the notation of a given rule.

Rule of inference   Tautology   Name

$$\therefore \frac{p \wedge q}{p} \qquad (p \wedge q) \to p$$

                                     Simplification

$$\therefore \frac{p}{p \vee q} \qquad p \to (p \vee q)$$

                                     Addition

$$\begin{array}{l} p \\ q \\ \hline \therefore p \wedge q \end{array} \qquad ((p) \wedge (q)) \to (p \wedge q)$$

                                     Conjunction

$$\begin{array}{l} p \\ p \to q \\ \hline \therefore q \end{array} \qquad ((p \wedge (p \to q)) \to q$$

                                     Modus ponens

$$\begin{array}{l} \neg q \\ p \to q \\ \hline \therefore \neg p \end{array} \qquad ((\neg q \wedge (p \to q)) \to \neg p$$

                                     Modus tollens

$$\begin{array}{l} p \to q \\ q \to r \\ \hline \therefore p \to r \end{array} \qquad ((p \to q) \wedge (q \to r)) \to (p \to r)$$

                                     Hypothetical syllogism

$$\begin{array}{l} p \vee q \\ \neg p \\ \hline \therefore q \end{array} \qquad ((p \vee q) \wedge \neg p) \to q$$

                                   Disjunctive syllogism

$$\begin{array}{l} p \vee q \\ \neg p \vee r \\ \hline \therefore q \vee r \end{array} \qquad ((p \vee q) \wedge (\neg p \vee r)) \to (q \vee r)$$

                                   Resolution

Example 1

Let us consider the following assumptions: "If it rains today, then we will not go on a canoe today. If we do not go on a canoe trip today, then we will go on a canoe trip tomorrow. Therefore (Mathematical symbol for "therefore" is), if it rains today, we will go on a canoe trip tomorrow. To make use of the rules of inference in the above table we let p be the proposition "If it rains today", q be " We will not go on a canoe today" and let r be "We will go on a canoe trip tomorrow". Then this argument is of the form:

$$p \rightarrow q$$
$$q \rightarrow r$$
$$\therefore \overline{p \rightarrow r}$$

Proof of contradiction:

The "Proof by Contradiction" is also known as reductio ad absurdum, which is probably Latin for "reduce it to something absurd".

Here's the idea:

•   Assume that a given proposition is untrue.

•   Based on that assumption reach two conclusions that contradict each other.

This is based on a classical formal logic construction known as Modus Tollens: If P implies Q and Q is false, then P is false. In this case, Q is a proposition of the form (R and not R) which is always false. P is the negation of the fact that we are trying to prove and if the negation is not true then the original proposition must have been true. If computers are not "not stupid" then they are stupid. (I hear that "stupid computer!" phrase a lot around here.)

Example:

Lets prove that there is no largest prime number (this is the idea of Euclid's original proof). Prime numbers are integers with no exact integer divisors except 1 and themselves.

•   To prove: "There is no largest prime number" by contradiction.

•   Assume: There is a largest prime number, call it p.

•   Consider the number N that is one larger than the product of all of the primes smaller than or equal to p. N=1*2*3*5*7*11...*p + 1. Is it prime?

•   N is at least as big as p+1 and so is larger than p and so, by Step 2, cannot be prime.

•   On the other hand, N has no prime factors between 1 and p because they would all leave

a remainder of 1. It has no prime factors larger than p because Step 2 says that there are no primes larger than p. So N has no prime factors and therefore must itself be prime (see note below).

We have reached a contradiction (N is not prime by Step 4, and N is prime by Step 5) and therefore our original assumption that there is a largest prime must be false.

Note: The conclusion in Step 5 makes implicit use of one other important theorem: The Fundamental Theorem of Arithmetic: Every integer can be uniquely represented as the product of primes. So if N had a composite (i.e. non-prime) factor, that factor would itself have prime factors which would also be factors of N.

Automatic Theorem Proving:

Automatic Theorem Proving (ATP) deals with the development of computer programs that show that some statement (the conjecture) is a logical consequence of a set of statements (the axioms and hypotheses). ATP systems are used in a wide variety of domains. For examples, a mathematician might prove the conjecture that groups of order two are commutative, from the axioms of group theory; a management consultant might formulate axioms that describe how organizations grow and interact, and from those axioms prove that organizational death rates decrease with age; a hardware developer might validate the design of a circuit by proving a conjecture that describes a circuit's performance, given axioms that describe the circuit itself; or a frustrated teenager might formulate the jumbled faces of a Rubik's cube as a conjecture and prove, from axioms that describe legal changes to the cube's configuration, that the cube can be rearranged to the solution state. All of these are tasks that can be performed by an ATP system, given an appropriate formulation of the problem as axioms, hypotheses, and a conjecture.

The **language** in which the conjecture, hypotheses, and axioms (generically known as formulae) are written is a logic, often classical 1st order logic, but possibly a non-classical logic and possibly a higher order logic. These languages allow a precise formal statement of the necessary information, which can then be manipulated by an ATP system. This formality is the underlying strength of ATP: there is no ambiguity in the statement of the problem, as is often the case when using a natural language such as English. Users have to describe the problem at hand precisely and accurately, and this process in itself can lead to a clearer understanding of the problem domain. This in turn allows the user to formulate their problem appropriately for submission to an ATP system.

The **proofs** produced by ATP systems describe how and why the conjecture follows from the axioms and hypotheses, in a manner that can be understood and agreed upon by everyone, even other computer programs. The proof output may not only be a convincing argument that the conjecture is a logical consequence of the axioms and hypotheses, it often also describes a process that may be implemented to solve some problem. For example, in the Rubik's cube

example mentioned above, the proof would describe the sequence of moves that need to be made in order to solve the puzzle.

ATP systems are enormously powerful computer programs, capable of solving immensely difficult problems. Because of this extreme capability, their application and operation sometimes needs to be guided by an expert in the domain of application, in order to solve problems in a reasonable amount of time. Thus ATP systems, despite the name, are often used by domain experts in an interactive way. The interaction may be at a very detailed level, where the user guides the inferences made by the system, or at a much higher level where the user determines intermediate lemmas to be proved on the way to the proof of a conjecture. There is often a synergetic relationship between ATP system users and the systems themselves:

- The system needs a precise description of the problem written in some logical form,
- the user is forced to think carefully about the problem in order to produce an appropriate formulation and hence acquires a deeper understanding of the problem,
- the system attempts to solve the problem, if successful the proof is a useful output,
- if unsuccessful the user can provide guidance, or try to prove some intermediate result, or examine the formulae to ensure that the problem is correctly described,
- and so the process iterates.

ATP is thus a **technology** very suited to situations where a clear thinking domain expert can interact with a powerful tool, to solve interesting and deep problems. Potential ATP users need not be concerned that they need to write an ATP system themselves; there are many ATP systems readily available for use.

RELATIONS

Introduction

The elements of a set may be related to one another. For example, the set of natural numbers.

The elements of one set may also be related to the elements another set.

Binary Relation

A binary relation between two sets A and B is a rule R which decides, for any elements, whether a is in relation R to b. If so, we write a R b. If a is not in relation R to b, then we shall write aRb.

We can also consider a R b as the ordered pair (a, b) in which case we can define a binary relation from A to B as a subset of A X B. This subset is denoted by the relation R.

In general, any set of ordered pairs defines a binary relation.

For example, the relation of father to his child is F = {(a, b) / a is the father of b} In this relation F, the first member is the name of the father and the second is the name of the child.

The definition of relation permits any set of ordered pairs to define a relation.

For example, the set S given by

S = {(1, 2), (3, a), (b, a) ,(b, Joe)}

Definition

The domain D of a binary relation S is the set of all first elements of the ordered pairs in the relation.(i.e) D(S)= {a / $ b for which (a, b) Є S}

The range R of a binary relation S is the set of all second elements of the ordered pairs in the relation.(i.e) R(S) = {b / $ a for which (a, b) Є S}

For example

For the relation S = {(1, 2), (3, a), (b, a) ,(b, Joe)} D(S) = {1, 3, b, b} and R(S) = {2, a, a, Joe}

Let X and Y be any two sets. A subset of the Cartesian product X * Y defines a relation, say C. For any such relation C, we have D(C) Í X and R(C) Í Y, and the relation C is said to from X to Y. If Y = X, then C is said to be a relation form X to X. In such case, c is called a relation in X. Thus any relation in X is a subset of X * X . The set X * X is called a universal relation in X, while the empty set which is also a subset of X * X is called a void relation in X.

For example

Let L denote the relation ―less than or equal to and D denote the relation ―divides where x D y means ― x divides y. Both L and D are defined on the set {1, 2, 3, 4}

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

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

L Ç D = {(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (3, 3),(4, 4)} = D

Properties of Binary Relations:

Definition: A binary relation R in a set X is reflexive if, for every x Є X, x R x, That is (x, x)

Є R, or R is reflexive in X ó (x) (x Є X ® x R x).

For example

- The relation £ is reflexive in the set of real numbers.
- The set inclusion is reflexive in the family of all subsets of a universal set.
- The relation equality of set is also reflexive.
- The relation is parallel in the set lines in a plane.
- The relation of similarity in the set of triangles in a plane isreflexive.

Definition: A relation R in a set X is symmetric if for every x and y in X, whenever x R y,

then y R x.(i.e) R is symmetric in X ó (x) (y) (x Є X ∧ y Є X ∧ x R y ® y R x}

For example

- The relation equality of set is symmetric.
- The relation of similarity in the set of triangles in a plane is symmetric.
- The relation of being a sister is not symmetric in the set of all people.
- However, in the set females it is symmetric.

Definition: A relation R in a set X is whenever x R y and y R z , then x R z. (i.e) transitive if,

for every x, y, and z are in X, R is transitive in X ó (x) (y) (z) (x Є X∧ y Є X∧ z

ЄX∧xRy∧yRz®xRz)

For example

- The relations <, £, >, ³ and = are transitive in the set of real numbers
- The relations Í, Ì, Ê, É and equality are also transitive in the family of sets.
- The relation of similarity in the set of triangles in a plane is transitive.

Definition: A relation R in a set X is irreflexive if, for every x Є X , (x, x)ÏX.

For example

- The relation < is irreflexive in the set of all real numbers.
- The relation proper inclusion is irreflexive in the set of all nonempty subsets of a universal set.

Let X = {1, 2, 3} and S = {(1, 1), (1, 2), (3, 2), (2, 3), (3, 3)} is neither irreflexive nor reflexive.

Definition: A relation R in a set x is anti symmetric if , for every x and y in X, whenever x R y and y R x, Then x = y.

Symbolically,(x) (y) (x Є X ∧ y Є X ∧ x R y ∧ y R x ® x = y)

For example

•   The relations £ , ³ and = are anti symmetric

•   The relation Í is anti symmetric in set of subsets.

•   The relation ―divides‖ is anti symmetric in set of real numbers.

•   Consider the relation ―is a son of‖ on the male children in a family.Evidentlythe relation is not symmetric, transitive and reflexive.

•   The relation ― is a divisor of ― is reflexive and transitive but not symmetric on the set of natural numbers.

•   Consider the set H of all human beings. Let r be a relation ― is married to ―

R is symmetric.

•   Let I be the set of integers. R on I is defined as a R b if a – b is an even number.R is an reflexive, symmetric and transitive.

Equivalence Relation:

Definition:A relation R in a set A is called an equivalence relation if

•   a R a for every i.e. R is reflexive

•   a R b => b R a for every a, b Є A i.e. R is symmetric

•   a R b and b R c => a R c for every a, b, c Є A, i.e. R is transitive.

For example

•   The relation equality of numbers on set of realnumbers.

The relation being parallel on a set of lines in a plane.

Problem 1:Let us consider the set T of triangles in a plane. let us define a relation R in T as R={(a,b)/(a,bЄT and a is similar to b}. we have to show that relation R is an equivalence relation.

Solution :

•   A triangle a is similar to itself. a R a

•   If the triangle a is similar to the triangle b, then triangle b is similar to the triangle a then a R b => b R a

•   If a is similar to b and b is similar to c, then a is similar to c (i.e) aRb and b R c => a R c.

Hence R is an equivalence relation.

Problem 2: Let x = {1, 2, 3, … 7} and R = {(x, y) / x – y is divisible by 3} Show that R is an equivalence relation.

Solution: For any a Є X, a- a is divisible by 3, Hence a R a, R is reflexive

For any a, b Є X, if a – b is divisible by 3, then b – a is also divisible by 3, R is symmetric.

For any a, b, c Є, if a R b and b R c, then a – b is divisible by 3 and b–c is divisible by 3.

So that (a – b) + (b – c) is also divisible by 3, hence a – c is also divisible by 3. Thus R is transitive.

Hence R is equivalence.

Equivalence Classes:

Let R be an equivalence relation on a set A. For any a ЄA, the equivalence class generated by a is the set of all elements b Є A such a R b and is denoted [a]. It is also called the R – equivalence class and denoted by a Є A. i.e., [a] = {b Є A / b R a}

Let Z be the set of integer and R be the relation called ―congruence modulo 3‖ defined by R = {(x, y)/ xÎ Z Ù yÎZ Ù (x-y) is divisible by 3}

Then the equivalence classes are

[0] = {… -6, -3, 0, 3, 6, …}

[1] = {…, -5, -2, 1, 4, 7, …}

[2] = {…, -4, -1, 2, 5, 8, …}

Composition of binary relations:

Definition: Let R be a relation from X to Y and S be a relation from Y to Z. Then the relation given by R o S = {(x, z) / xÎX Ù z Î Z Ù y Î Y such that (x, y) Î R Ù (y, z) Î S)} is called the composite relation of R and S.

The operation of obtaining R o S is called the **composition of relations**.

Example: Let R = {(1, 2), (3, 4), (2, 2)} and

S = {(4, 2), (2, 5), (3, 1),(1,3)}

Then R o S = {(1, 5), (3, 2), (2, 5)} and S o R = {(4, 2), (3, 2), (1, 4)}

It is to be noted that R o S ≠ S o R.

Also Ro(S o T) = (R o S) o T = R o S o T

Note: We write R o R as R2; R o R o R as R3 and so on.

Definition

Let R be a relation from X to Y, a relation R from Y to X is called the converse of R, where the ordered pairs of Ř are obtained by interchanging the numbers in each of the ordered pairs of R. This means for x Î X and y Î Y, that x R y ó y Ř x.

Then the relation Ř is given by R = {(x, y) / (y, x) Î R} is called the converse

of R Example:

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

Then Ř = {(2, 1),(4, 3),(2, 2)}

Note: If R is an equivalence relation, then Ř is also an equivalence relation.

Definition   Let X be any finite set and R be a relation in X. The relation

R+ = R U R2 U R3…in X. is called the transitive closure of R in X

Example: Let R = {(a, b), (b, c), (c, a)}.

Now R2 = R o R = {(a, c), (b, a), (c, b)}

R3 = R2 o R = {(a, a), (b, b), (c, c)}

R4 = R3 o R = {(a, b), (b, c), (c, a)} = R

R5= R3o R2 = R2 and so on.

Thus, R+ = R U R2 U R3 U R4 U…

= R U R2 U R3.

={(a, b),(b, c),(c, a),(a, c),(b, a),(c ,b),(a, b),(b, b),(c, c)}

We see that R+ is a transitive relation containing R. In fact, it is the smallest transitive relation containing R.

Partial Ordering Relations:

Definition

A binary relation R in a set P is called partial order relation or partial ordering in P iff R is reflexive, anti symmetric, and transitive.

A partial order relation is denoted by the symbol £., If £ is a partial ordering on P, then the ordered pair (P, £) is called a partially ordered set or a poset.

•   Let X be a set and r(X) be its power set. The relation subset, Í on X is partial ordering.

•   Let Sn be the set of divisors of n. The relation D means ―divides‖ on Sn ,is partial ordering on Sn .

In a partially ordered set (P, £) , an element y Î P is said to cover an element x Î P if x <y and if there does not exist any element z Î P such that x £ z and z £ y; that is, y covers x Û (x < y Ù (x £ z £ y Þ x = z Ú z = y)) A partial order relation £ on a set P can be represented by means of a diagram known as Hasse diagram or partial order set diagram of (P, £). In such a diagram, each element is represented by a small circle or a dot. The circle for x Î P is drawn below the circle for y Î P if x< y, and a line is drawn between x and y if y covers x.

If x < y but y does not cover x, then x and y are not connected directly by a single line.However, they are connected through one or more elements of P.
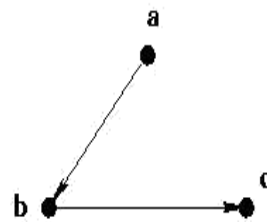
Hasse Diagram:

A Hasse diagram is a digraph for a poset which does not have loops and arcs implied by the transitivity.

Example 10: For the relation $\{< a, a >, < a, b >, < a, c >, < b, b >, < b, c >, < c, c >\}$ on set $\{a, b,c\}$, the Hasse diagram has the arcs $\{< a, b >, < b, c >\}$ as shown below.
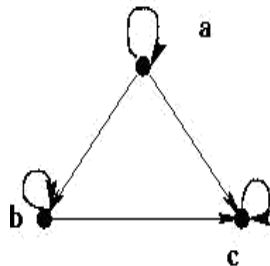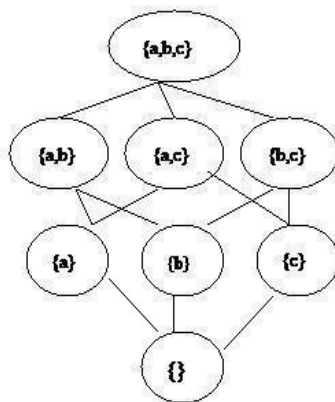

Digraph for Partial Order   Hasse Diagram

Ex: Let A be a given finite set and r(A) its power set. Let Í be the subset relation on the elements of r(A). Draw Hasse diagram of (r(A), Í) for A = {a, b, c}

Functions

Introduction

A function is a special type of relation. It may be considered as a relation in which each element of the domain belongs to only one ordered pair in the relation. Thus a function from A to B is a subset of A X B having the property that for each a ЄA, there is one and only one b Є B such that (a, b) Î G.

Definition: Let A and B be any two sets. A relation f from A to B is called a function if for every a Є A there is a unique b Є B such that (a, b) Є f .

Note that the definition of function requires that a relation must satisfy two additional conditions in order to qualify as a function. The first condition is that every a Є A must be related to some b Є B, (i.e) the domain of f must be A and not merely subset of A. The second requirement of uniqueness can be expressed as (a, b) Є f ∧ (b, c) Є f => b = c

Intuitively, a function from a set A to a set B is a rule which assigns to every element of A, a unique element of B. If a ЄA, then the unique element of B assigned to a under f is denoted by f(a).

The usual notation for a function f from A to B is f: A® B defined by a ® f (a) where a Є A, f(a) is called the image of a under f and a is called pre image of f(a).

•   Let X = Y = R and f(x) = x2 + 2. Df = R and Rf Í R.

•   Let X be the set of all statements in logic and let Y = {True, False}. A mapping f: X®Y is a function.

•   A program written in high level language is mapped into a machine language by a compiler. Similarly, the output from a compiler is a function of its input.

•   Let X = Y = R and f(x) = x2 is a function from X ® Y,and g(x2) = x is not a function from X ® Y.

A mapping f: A ® B is called one-to-one (injective or 1 –1) if distinct elements of A are mapped into distinct elements of B. (i.e) f is one-to-one if

a1 = a2 => f (a1) = f(a2) or equivalently f(a1) ¹ f(a2) => a1 ¹ a2

For example, f: N ® N given by f(x) = x is 1-1 where N is the set of a natural numbers.

A mapping f: A® B is called onto (surjective) if for every b Є B there is an a Є A such that f (a) = B. i.e. if every element of B has a pre-image in A. Otherwise it is called into.

For example, f: Z®Z given by f(x) = x + 1 is an onto mapping. A mapping is both 1-1 and onto is called bijective

For example f: R®R given by f(x) = X + 1 is bijective.

Definition: A mapping f: R® b is called a constant mapping if, for all aÎA, f (a) = b, a fixed element.

For example f: Z®Z given by f(x) = 0, for all x ÎZ is a constant mapping.

Definition: A mapping f: A®A is called the identity mapping of A if f (a) = a, for all aÎA. Usually it is denoted by IA or simply I.

Composition of functions:

If f: A®B and g: B®C are two functions, then the composition of functions f and g, denoted by g o f, is the function is given by g o f : A®C and is given by

g o f = {(a, c) / a Є A ∧ c Є C ∧ \$bÎ B ': f(a)= b ∧ g(b)= c} and (g of)(a) = ((f(a))

Example 1: Consider the sets A = {1, 2, 3},B={a, b} and C = {x, y}. Let f: A® B be defined by f (1) = a ; f(2) = b and f(3)=b and Let g: B® C be defined by g(a) = x and g(b) = y (i.e) f = {(1, a), (2, b), (3, b)} and g = {(a, x), (b, y)}. Then g o f: A®C is defined by (g of) (1) = g (f(1)) = g(a) = x

(g o f) (2) = g (f(2)) = g(b) = y

(g o f) (3) = g (f(3)) = g(b) = y

i.e., g o f = {(1, x), (2, y),(3, y)}

If f: A® A and g: A®A, where A= {1, 2, 3}, are given by

f = {(1, 2), (2, 3), (3, 1)} and g = {(1, 3), (2, 2), (3, 1)}

Then g of = {(1, 2), (2, 1), (3, 3)}, fog= {(1, 1), (2, 3), (3, 2)}

f of = {(1, 3), (2, 1), (3, 2)} and gog= {(1, 1), (2, 2), (3, 3)}

Example 2: Let f(x) = x+2, g(x) = x − 2 and h(x) = 3x for x Î R, where R is the set of real numbers.

Then f o f = {(x, x+4)/xÎ R} f o g = {(x, x)/ x Î X} g

o f = {(x, x)/ xÎ X}

g o g = {(x, x-4)/x Î X}

h o g = {(x,3x-6)/ x Î X} h o f = {(x, 3x+6)/ x Î X}

Inverse functions:

Let f: A® B be a one-to-one and onto mapping. Then, its inverse, denoted by f -1 is given by

f - 1 = {(b, a) / (a, b) Î f} Clearly f-1: B® A is one-to-one and onto.

Also we observe that f o f -1 = IB and f -1o f = IA. If f -1 exists then f is called **invertible**.

example: Let f: R ®R be defined by f(x) = x + 2 Then f -1: R® R is defined by f -1(x) = x - 2

Theorem: Let f: X ®Y and g: Y ® Z be two one to one and onto functions. Then gof is also one to one and onto function.

Proof

Let f:X ® Y g : Y ® Z be two one to one and onto functions. Let x1, x2 Î X

•   g o f (x1) = g o f(x2),

•   g (f(x1)) = g(f(x2)),

•   g(x1) = g(x2) since [f is 1-1]

x1 = x2 since [g is 1-1}so that gof is 1-1.

By the definition of composition, gof : X ® Z is a function.

We have to prove that every element of z Î Z an image element for some x Î X

under gof. Since g is onto $ y ÎY ': g(y) = z and f is onto from X to Y, $ x ÎX ': f(x) = y.

Now, gof (x) = g (f (x)) = g(y) [since f(x) = y] = z [since g(y) = z] which shows that gof is onto.

 Theorem (g o f) $^{-1}$ = f $^{-1}$ o g $^{-1}$

(i.e) the inverse of a composite function can be expressed in terms of the composition of the inverses in the reverse order.

Proof.

f: A ® B is one to one and onto. g: B ® C is one to one and onto.

gof: A ® C is also one to one and onto. Þ (gof) -1: C ® A is one to one and onto.

Let a Î A, then there exists an element b Î b such that f (a) = b Þ a = f-1

(b). Now b Î B Þ there exists an element c Î C such that g (b) = c Þ b = g - 1(c). Then (gof)(a)

= g[f(a)] = g(b) = c Þ a = (gof) -1(c). **(1)**

(f -1 o g-1) (c) = f -1(g -1 (c)) = f -1(b) = a Þ a = (f -1 o g **-1**)(c) ….(2)

Combining (1) and (2), we have (gof) -1 = f -1 o g -1

Theorem: If f: A ® B is an invertible mapping , then f o f -1 = I B and f-1 o f = IA

Proof: f is invertible, then f -1 is defined by f(a) = b ó f-1(b)= a where a Î A and bÎ B .

Now we have to prove that f of -1 = IB

 Let bÎ B and f -1(b) = a, a Î A then fof-1(b) = f(f-1(b))= f(a) = b

therefore f o f -1 (b) = b " b Î B => f o f -1 = IB Now f -1 o f(a) = f -1 (f(a)) = f -1 (b) = a

therefore f -1 o f(a) = a " a Î A => f -1 o f = IA. Hence the theorem.

Recursive Functions:

The term "recursive function" is often used informally to describe any function that is defined with recursion.

Kleene (1952) defines a "partial recursive function" of nonnegative integers to be any function f that is defined by a noncontradictory system of equations whose left and right sides are composed from function symbols (for example, f, g, h, etc.), (2) variables for nonnegative integers (for example, x, y, z, etc.), (3) the constant 0, and (4) the successor function $S(x) = x + 1$.

For example,

$$f(x, 0) = 0 \quad (1)$$

$$f(x, S(y)) = g(f(x, y), x) \quad (2)$$

$$g(x, 0) = x \quad (3)$$

$$g(x, S(y)) = S(g(x, y)) \quad (4)$$

defines $f(x, y)$ to be the function $x\,y$ that computes the product of x and y.

Note that the equations might not uniquely determine the value of f for every possible input, and in that sense the definition is "partial." If the system of equations determines the value of f for every input, then the definition is said to be "total." When the term "recursive function" is used alone, it is usually implicit that "total recursive function" is intended. Note that some

authors use the term "general recursive function to mean partial recursive function, although others use it to mean "total recursive function."

The set of functions that can be defined recursively in this manner is known to be equivalent to the set of functions computed by Turing machines and by the lambda calculus.

Lattice and its Properties:

Introduction:

A lattice is partially ordered set (L, £) in which every pair of elements a, b ÎL has a greatest lower bound and a least upper bound.The glb of a subset, {a, b} Í L will be denoted by a * b and the lub by a Å b.

Usually, for any pair a, b Î L, GLB {a, b} = a * b, is called the meet or product and LUB{a, b} = a Å b, is called the join or sum of a and b.

Example1 Consider a non-empty set S and let P(S) be its power set. The relation Í contained in is a partial ordering on P(S). For any two subsets A, BÎ P(S)

GLB {A, B} and LUB {A, B} are evidently A Ç B and A È B respectively.

Example: Let I+ be the set of positive integers, and D denote the relation of ―division in I+ such that for any a, b Î I+ , a D b iff a divides b. Then (I+, D) is a lattice in which the join of a and b is given by the least common multiple(LCM) of a and b, that is,a Å b = LCM of a and b, and the meet of a and b, that is , a * b is the greatest common divisor (GCD) of a and b.

A lattice can be conveniently represented by a diagram.

For example, let Sn be the set of all divisors of n, where n is a positive integer. Let D denote the relation ―division such that for any a, b Î Sn, a D b iff a divides b. Then (Sn, D) is a lattice with a * b = gcd(a, b) and a Å b = lcm(a, b).

Two lattices can have the same diagram but the nodes are differently labeled . We observe that for any partial ordering relation £ on a set S the converse relation is also partial ordering relation on S. If (S, £) is a lattice With meet a * b and join a Å b , then (S, ³) is the also a lattice with meet a Å b and join a * b i.e., the GLB and LUB get interchanged . Thus we have the principle of duality of lattice as follows.

Any statement about lattices involving the operations ^ and V and the relations £ and ³ remains true if ^, V, ³ and £ are replaced by V, ^, £ and ³ respectively.

The operation ^ and V are called duals of each other as are the relations £ and ³.. Also, the lattice (L, £) and (L, ³) are called the duals of each other.

Properties of lattices:

Let (L, £) be a lattice with the binary operations * and Å then for any a, b, c Î L,

- $a * a = a$ $a \text{ Å } a = a$ (Idempotent)

- $a * b = b * a$, $a \text{ Å } b = b \text{ Å } a$ (Commutative)
- $(a * b) * c = a * (b * c)$, $(a \text{ Å }) \text{ Å } c = a \text{ Å } (b \text{ Å } c)$ (Associative)
- $a * (a \text{ Å } b) = a$ $a \text{ Å } (a * b) = a$ (absorption)

For any a Î L, a £ a, a £ LUB {a, b} => a £ a * (a Å b). On the other hand, GLB {a, a Å b} £ a

i.e., (a Å b) Å a, hence a * (a Å b) = a

Theorem 1

Let (L, £) be a lattice with the binary operations * and Å denote the operations of meet and

join respectively For any a, b Î L,

a £ b ó a * b = a ó a Å b = b

Proof

Suppose that a £ b. we know that a £ a, a £ GLB {a, b}, i.e., a £ a * b. But from the definition

of a * b, we get a * b £ a.

Hence a £ b => a * b = a ....(1)

Now we assume that a * b = a; but is possible only if a £ b,

that is a * b = a => a £ b.. (2)

From (1) and (2), we get a £ b ó a * b = a.

Suppose a * b = a.   then b Å (a * b) = b Å a = a Å b..................................(3)

but b Å (a * b) = b (by iv) ....(4)

Hence a Å b = b, from (3) => (4)

Suppose a Å b = b, i.e., LUB {a, b} = b, this is possible only if a £ b, thus(3) => (1) (1) => (2)

=> (3) => (1). Hence these are equivalent.

Let us assume a * b = a.

Now (a * b) Å b = a Å b

We know that by absorption law , (a * b) Å b = b

so that a Å b = b, therefore a * b = a Þ a Å b = b (5)

similarly, we can prove a Å b = b Þ a * b = a (6)

From (5) and (6), we get

a * b = a Û a Å b = b Hence the theorem.

Theorem2 For any a, b, c Î L, where (L, £) is a lattice. b

£ c => { a * b £ a * c and

{ a Å b £ a Å c

Proof Suppose b £ c. we have proved that b £ a ó b * c = b   **(1)**

Now consider

(a * b) * (a * c) = (a * a) * (b * c)= a * (b * c)

(by Idempotent)

= a * b (by (1))

Thus (a * b) * (a * c) = a * b which => (a * b) £ (a * c) Similarly (a Å b) Å (a Å c) = (a Å a) Å (b Å c)

= a Å (b Å c)

= a Å c which => (a Å b) £ (a Å c)

note:These properties are known as **isotonicity.**

Lattices as partially ordered sets

If (L, ≤) is a partially ordered set (poset), and S ⊆ L is an arbitrary subset, then an element u ∈ L is said to be an upper bound of S if s ≤ u for each s ∈ S. A set may have many upper bounds, or none at all. An upper bound u of S is said to be its least upper bound, or join, or supremum, if u ≤ x for each upper bound x of S. A set need not have a least upper bound, but it cannot have more than one. Dually, l ∈ L is said to be a lower bound of S if l ≤ s for each s ∈ S. A lower bound l of S is said to be its greatest lower bound, or meet, or infimum, if x ≤ l for each lower bound x of S. A set may have many lower bounds, or none at all, but can have at most one greatest lower bound.

Sublattices

A sublattice of a lattice L is a nonempty subset of L that is a lattice with the same meet and join operations as L. That is, if L is a lattice and M ≠ {\displaystyle \varnothing } \varnothing is a subset of L such that for every pair of elements a, b in M both a ∧ b and a ∨ b are in M, then M is a sublattice of L.[2]

A sublattice M of a lattice L is a convex sublattice of L, if x ≤ z ≤ y and x, y in M implies that z belongs to M, for all elements x, y, z in L.

Algebraic structures

Algebraic systems:

An algebraic system, loosely speaking, is a set, together with some operations on the set. Before formally defining what an algebraic system is, let us recall that a n -ary operation (or operator) on a set A is a function whose domain is An and whose range is a subset of A . Here, n is a non- negative integer. When n=0 , the operation is usually called a nullary operation, or a constant, since one element of A is singled out to be the (sole) value of this operation. A finitary operation on A is just an n -ary operation for some non-negative integer n.

Definition. An algebraic system is an ordered pair (A O) , where A is a set, called the underlying set of the algebraic system, and O is a set, called the operator set, of finitary operations on A .

We usually write **A** , instead of (A O) , for brevity.

A prototypical example of an algebraic system is a group, which consists of the underlying set G

, and a set O consisting of three operators: a constant e called the multiplicative identity, a unary operator called the multiplicative inverse, and a binary operator called the multiplication.

•   An algebraic system is also called algebra for short. Some authors require that A be non-empty. Note that A is automatically non-empty if Ocontains constants. A finite algebra is an algebra whose underlying set is finite.

•   By definition, all operators in an algebraic system are finitary. If we allow O to contain infinitary operations, we have an infinitary algebraic system. Other generalizations are possible. For example, if the operations are allowed to be multivalued, the algebra is said to be a multialgebra. If the operations are not everywhere defined, we get a partial algebra. Finally, if more than one underlying set is involved, then the algebra is said to be many-sorted.

The study of algebraic systems is called the theory of universal algebra. The first important thing in studying algebraic system is to compare systems that are of the same ``type''. Two algebras are said to have the same type if there is a one-to-one correspondence between their operator sets such that an n -ary operator in one algebra is mapped to an n -ary operator in the other algebra.

Examples:

Some recurring universes: **N**=natural numbers; **Z**=integers; **Q**=rational numbers;

R=real numbers; **C**=complex numbers.

N is a pointed unary system, and under addition and multiplication, is both the standard interpretation of Peano arithmetic and a commutative semiring.

Boolean algebras are at once semigroups, lattices, and rings. They would even be abelian groups if the identity and inverse elements were identical instead of complements.

Group-like structures

•   Nonzero **N** under addition (+) is a magma.

•   **N** under addition is a magma with an identity.

•   **Z under subtraction (−) is a quasigroup.**

•   Nonzero **Q** under division (÷) is a quasigroup.

•   Every group is a loop, because $a * x = b$ if and only if $x = a$ and $y * a = b$ if

and only if $y = b * a$.

•   2x2 matrices(of non-zero determinant) with matrix multiplication form a group.

•   **Z** under addition (+) is an abelian group.

•   Nonzero **Q** under multiplication (×) is an abelian group.

•   Every cyclic group G is abelian, because if x, y are in G, then $xy = a$

•   A monoid is a category with a single object, in which case the composition of morphisms and the identity morphism interpret monoid multiplication and identity element, respectively.

•   The Boolean algebra **2** is a boundary algebra.

General Properties:

Property of Closure

If we take two real numbers and multiply them together, we get another real number. (The real numbers are all the rational numbers and all the irrational numbers.) Because this is always true, we say that the real numbers are "closed under the operation of multiplication": there is no way to escape the set. When you combine any two elements of the set, the result is also included in the set. Real numbers are also closed under addition and subtraction. They are not closed under the square root operation, because the square root of -1 is not a real number.

Inverse

The inverse of something is that thing turned inside out or upside down. The inverse of an operation undoes the operation: division undoes multiplication.

A number's additive inverse is another number that you can add to the original number to get the additive identity. For example, the additive inverse of 67 is -67, because 67 + -67 = 0, the additive identity.

Similarly, if the product of two numbers is the multiplicative identity, the numbers are multiplicative inverses. Since 6 * 1/6 = 1 (the multiplicative identity), the multiplicative inverse of 6 is 1/6.

Zero does not have a multiplicative inverse, since no matter what you multiply it by, the answer is always 0, not 1.

Equality

The equals sign in an equation is like a scale: both sides, left and right, must be the same in order for the scale to stay in balance and the equation to be true.

The addition property of equality says that if a = b, then a + c = b + c: if you add the same number to (or subtract the same number from) both sides of an equation, the equation continues to be true.

The multiplication property of equality says that if a = b, then a * c = b * c: if you multiply (or divide) by the same number on both sides of an equation, the equation continues to be true.

The reflexive property of equality just says that a = a: anything is congruent to itself: the equals sign is like a mirror, and the image it "reflects" is the same as the original.

The symmetric property of equality says that if a = b, then b = a.

The transitive property of equality says that if a = b and b = c, then a = c.

Semi groups and monoids:

In the previous section, we have seen several algebraic system with binary operations. Here we consider an algebraic system consisting of a set and an associative binary operation on the set and then the algebraic system which possess an associative property with an identity element. These algebraic systems are called semigroups and monoids.

Semi group

Let S be a nonempty set and let * be a binary operation on S. The algebraic system (S, *) is called a semi-group if * is associative

i.e. if a * (b*c) = (a * b) * c for all a, b, c $\hat{\mathbf{I}}$ S.

Example The N of natural numbers is a semi-group under the operation of usual addition of numbers.

Monoids

Let M be a nonempty set with a binary operation * defined on it. Then (M, *) is called a monoid if * is associative (i.e) a * (b * c) = (a * b) * c for all a, b, c Î M and there exists an element e in M such that a * e = e * a = a for all a Î M, e is called the identity element in (M,*).

It is easy to prove that the identity element is unique. From the definition it follows that (M,*) is a semigroup with identity.

Example1 Let S be a nonempty set and r(S) be its power set. The algebras (r(S),U) and (r(S), Ç) are monoids with the identities f and S respectively.

Example2 Let N be the set of natural numbers, then (N,+), (N, X) are monoids with the identities 0 and 1 respectively.

Groups Sub Groups:

An algebraic system (S, *) is a semigroup if the binary operation * is associative. If there exists an identity element e Î S, then (S,*) is monoid. A further condition is imposed on the elements of the monoid, i.e., the existence of an inverse for each element of S then the algebraic system is called a group.

Definition

Let G be a nonempty set, with a binary operation * defined on it. Then the algebraic system (G,*) is called a **group** if

• 　　　* is associative i.e. a * (b * c) = (a * b) * c for all a, b, c,Î G.

• 　　　there exists an element e in G such that a * e = e * a = a for all a Î G

• 　　　for each a Î G there is an element denoted by a-1 in G such that a * a-1 = a-1 * a = e, a⁻¹ is called the inverse of a.

From the definition it follows that (G,*) is a monoid in which each element has an inverse w.r.t.

* in G.

A group (G,*) in which * is commutative is called an abelian group or a commutative group.

If * is not commutative then (G,*) is called a non-abelian group or non-commutative group.

The order of a group (G,*) is the number of elements of G, when G is finite and is denoted by o(G) or |G|

Examples 1. (Z5, +5) is an abelian group of order 5.

2. G = {1, -1, i, -i} is an abelian group with the binary operation x is defined as 1 x 1 = 1, -1 x -1 = 1, i x i = -1 , -i x -i = **1, …**

Homomorphism of semigroups and monoids

Semigroup homomorphism.

Let (S, *) and (T, D) be any two semigroups. A mapping g: S ® T such that any two elements a, b Î S , g(a * b) = g(a) D g(b) is called a semigroup homomorphism.

Monoid homomorphism

Let (M, *,eM) and (T, D,eT) be any two monoids. A mapping g: M® T such that any two elements a, b Î M ,g(a * b) = g(a) D g(b) and g(eM) = eT is called a monoid homomorphism.

Theorem 1 Let (s, *) , (T, D) and (V, Å) be semigroups. A mapping g: S ® T and h: T ® V be semigroup homomorphisms. Then (hog): S ® V is a semigroup homomorphism from (S,*) to(V,Å).

Proof. Let a, b Î S. Then

(h o g)(a * b) = h(g(a* b))

= h(g(a) D g(b))

= h(g(a)) Å h(g(b))

= (h o g)(a) Å (h o g)(b)

Theorem 2 Let (s,*) be a given semigroup. There exists a homomorphism g: S ® SS, where (SS, o) is a semigroup of function from S to S under the operation of composition.

Proof For any element a Î S, let g(a) = fa where f aÎ SS and f a is defined by

f a(b) = a * b for any a, bÎ S

g(a * b) = f a*b

Now f a*b(c) = (a * b) * c = a*(b * c)

where = f a(f b(c)) = (f a o f b) (c).

Therefore, g(a * b) = f a*b = f a o f b = g(a) o g(b), this shows that g: S ® SS is a homomorphism.

Theorem 3 For any commutative monoid (M, *),the set of idempotent elements of M forms a submonoid.

Proof. Let S be the set of idempotent elements of M.

Since the identity element e Î M is idempotent, e Î S. Let a, b Î S, so that a*a = a and b * b = b

Now (a * b) * (a * b) = (a * b) * (b * a)   [(M, *) is a commutative monoid]

= a * (b * b) * a

= a * b * a

= a * a * b

= a * b Hence a * b Î S and (S, *) is a submonoid.

Isomorphism:

In abstract algebra, an isomorphism is a bijective map f such that both f and its inverse f are homomorphisms, i.e., structure-preserving mappings. In the more general setting of category theory, an isomorphism is a morphism f: X → Y in a category for which there exists an Informally, an isomorphism is a kind of mapping between objects, which shows a relationship between two properties or operations. If there exists an isomorphism between two structures, we call the two structures isomorphic. In a certain sense, isomorphic structures are structurally identical, if you choose to ignore finer-grained differences that may arise from how they are defined.

Purpose:

Isomorphisms are studied in mathematics in order to extend insights from one phenomenon to others: if two objects are isomorphic, then any property which is preserved by an isomorphism and which is true of one of the objects, is also true of the other. If an isomorphism can be found from a relatively unknown part of mathematics into some well studied division of mathematics, where many theorems are already proved, and many methods are already available to find answers, then the function can be used to map whole problems out of unfamiliar territory over to "solid ground" where the problem is easier to understand and work with.

Ring:

The algebraic structure (R, +, .) which consisting of a non-empty set R along with two binary operations like addition(+) and multiplication(.) then it is called a ring.

An algebraic (or mathematically) system (R, *, o) consisting of a non-empty set R any two binary operations * and o defined on R such that:

(R, *) is an abelian group and (R, 0) is a semigroup.

The operation o is the distributive over the operation * is said to be the ring.

A ring is a set R equipped with two binary operations + and · satisfying the following three sets of axioms, called the ring axioms.

R is an abelian group under addition, meaning that:

(a + b) + c = a + (b + c) for all a, b, c in R (that is, + is associative).

a + b = b + a for all a, b in R (that is, + is commutative).

There is an element 0 in R such that a + 0 = a for all a in R   (that is, 0 is the additive identity).

For each a in R there exists −a in R such that a + (−a) = 0 (i.e, −a is the additive inverse of a).

R is a monoid under multiplication, meaning that: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all a, b, c in R (that is, $\cdot$ is associative).

There is an element 1 in R such that $a \cdot 1 = a$ and $1 \cdot a = a$ for all a in R   (that is, 1 is the multiplicative identity).[5]

Multiplication is distributive with respect to addition, meaning that:

$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ for all a, b, c in R (left distributivity).

$(b + c) \cdot a = (b \cdot a) + (c \cdot a)$ for all a, b, c in R (right distributivity).

Elementary Combinatorics

Basis of counting:

If X is a set, let us use |X| to denote the number of elements in X.

Two Basic Counting Principles

Sum Rule: **The principle of disjunctive counting** :

If a set X is the union of disjoint nonempty subsets S1, ….., Sn, then $| X | = | S1 | + | S2 | +$ ….. $+| Sn |$.

We emphasize that the subsets S1, S2, …., Sn must have no elements in common.Moreover, since X = S1 U S2 U ……U Sn, each element of X is in exactly one of the subsets Si. In other words, S1, S2, …., Sn is a partition of X.If the subsets S1, S2, …., Sn were allowed to overlap, then a more profound principle will be needed--the principle of inclusion and exclusion.

Frequently, instead of asking for the number of elements in a set perse, some problems ask for how many ways a certain event can happen.

The difference is largely in semantics, for if A is an event, we can let X be the set of ways that A can happen and count the number of elements in X. Nevertheless, let us state the sum rule for counting events.

If E1, ……, En are mutually exclusive events, and E1 can happen e1 ways, E2 happen e2 ways,…. ,En can happen en ways, E1 or E2 or …. or En can happen e1 + e2 + +en ways.

Again we emphasize that mutually exclusive events E1 and E2 mean that E1 or E2 can happen but both cannot happen simultaneously.

The sum rule can also be formulated in terms of choices: If an object can be selected from a reservoir in e1 ways and an object can be selected from a separate reservoir in e2 ways and an object can be selected from a separate reservoir in e2 ways, then the selection of one object from either one reservoir or the other can be made in e1 + e2 ways.

Product Rule: The principle of sequencing counting

If S1, , Sn are nonempty sets, then the number of elements in the Cartesian product

S1 x S2 x ….. x Sn is the product $\prod_{i=1}^{n} |S_i|$. That is,

$| S_1 \times S_2 \times .... \times S_n | = \prod_{i=1}^{n} | S_i |$.

Observe that there are 5 branches in the first stage corresponding to the 5 elements of S1 and to each of these branches there are 3 branches in the second stage corresponding to the 3 elements of S2 giving a total of 15 branches altogether. Moreover, the Cartesian product S1 x S2 can be partitioned as (a1 x S2) U (a2 x S2) U (a3 x S2) U (a4 x S2) U (a5 x S2), where (ai x S2)= {(ai, b1), (ai i, b2), (ai, b3)}. Thus, for example, (a3 x S2) corresponds to the third branch in the first stage followed by each of the 3 branches in the second stage.

More generally, if a1,….., an are the n distinct elements of S1 and b1,….,bm are the m distinct elements of S2, then S1 x S2 = Uin =1 (ai x S2).

For if x is an arbitrary element of S1 x S2 , then x = (a, b) where a Î S1 and b Î S2. Thus, a = ai for some i and b = bj for some j. Thus, x = (ai, bj) Î(ai x S2) and therefore x Î Uni =1(ai x S2).

Conversely, if x Î Uin =1(ai x S2), then x Î (ai x S2) for some i and thus x = (ai, bj) where bj is some element of S2. Therefore, x Î S1 x S2.

Next observe that (ai x S2) and (aj x S2) are disjoint if i ≠ j since if

x Î (ai x S2) ∩ (aj x S2) then x = (ai, bk) for some k and x = (aj, b1) for some l.

But then (ai, bk) = (aj, bl) implies that ai = aj and bk = bl. But since i ≠ j , ai ≠ a j.

Thus, we conclude that S1 x S2 is the disjoint union of the sets (ai x S2). Furthermore |ai x S2| = |S2| since there is obviously a one-to-one correspondence between the sets ai x S2 and S2, namely, (ai, bj) → bj.

Then by the sum rule |S1 x S2| = $\sum_{i=1}^{n}$ | ai x S2|

We can reformulate the product rule in terms of events. If events E1, E2 , …., En can happen e1, e2,…., and en ways, respectively, then the sequence of events E1 first, followed by E2,…., followed by En can happen e1e2 …en ways.

In terms of choices, the product rule is stated thus: If a first object can be chosen e1 ways, a second e2 ways , …, and an nth object can be made in e1e2….en ways.

Combinations & Permutations:

Definition.

A combination of n objects taken r at a time (called an r-combination of n objects) is an **unordered selection** of r of the objects.

A permutation of n objects taken r at a time (also called an r-permutation of n objects) is an **ordered selection** or **arrangement** of r of the objects.

Note that we are simply defining the terms r-combinations and r-permutations here and have not mentioned anything about the properties of the n objects.

For example, these definitions say nothing about whether or not a given element may appear more than once in the list of n objects.

In other words, it may be that the n objects do not constitute a set in the normal usage of the word.

SOLVED PROBLEMS

Example1. Suppose that the 5 objects from which selections are to be made are: a, a, a, b, c. then the 3-combinations of these 5 objects are : aaa, aab, aac, abc. The permutations are: aaa, aab, aba, baa, aac, aca, caa, abc, acb, bac, bca, cab, cba. Neither do these definitions say anything about any rules governing the selection of the r- objects: on one extreme, objects could be chosen where all repetition is forbidden, or on the other extreme, each object may be chosen up to t times, or then again may be some rule of selection between these extremes; for instance, the rule that would allow a given object to be repeated up to a certain specified number of times.We will use expressions like {3 . a , 2. b ,5.c} to indicate either

(1) that we have **3 + 2 + 5 =10 objects including 3a's , 2b's and 5c's, or (2) that we have 3** objects a, b, c, where selections are constrained by the conditions that a can be selected at most three times, b can be selected at most twice, and c can be chosen up to five times.

The numbers 3, 2 and 5 in this example will be called repetition numbers.

Example 2   The 3-combinations of {3. a, 2. b, 5. c} are:

aaa, aab, aac, abb, abc, ccc, ccb, cca, cbb.

Example 3. The 3-combinations of {3 . a, 2. b, 2. c , 1. d} are: aaa, aab, aac, aad, bba, bbc, bbd,cca, ccb, ccd, abc, abd, acd, bcd.

In order to include the case where there is no limit on the number of times an object can be repeated in a selection (except that imposed by the size of the selection) we use the symbol ∞ as a repetition number to mean that an object can occur an infinite number of times.

Example 4. The 3-combinations of {∞. a, 2.b, ∞.c} are the same as in Example 2 even though a and c can be repeated an infinite number of times. This is because, in 3-combinations, 3 is the limit on the number of objects to be chosen.

In particular, a selection of r objects in this case will be called r-combinations with unlimited repetitions and any ordered

arrangement of these r objects will be an r-permutation with unlimited repetitions.

Example5 The combinations of a ,b, c, d with unlimited repetitions are the 3-combinations

of $\{\infty \cdot a, \infty \cdot b, \infty \cdot c, \infty \cdot d\}$. These are 20 such 3-combinations, namely:

aaa, aab, aac, aad, bbb, bba, bbc, bbd, ccc, cca, ccb, ccd, ddd, dda, ddb, ddc, abc, abd, acd, bcd.

Moreover, there are $4^3 = 64$ of 3-permutations with unlimited repetitions since the first position can be filled 4 ways (with a, b, c, or d), the second position can be filled 4 ways, and likewise for the third position. The 2-permutations of $\{\infty \cdot a, \infty \cdot b, \infty \cdot c, \infty \cdot d\}$ do not present such a formidable list and so we tabulate them in the following table.

2-combinations with Unlimited Repetitions	2-permutations with Unlimited Repetitions
aa	Aa
ab	ab, ba
ac	ac, ca
ad	ad, da
bb	Bb
bc	bc, cb
bd	bd, db
cc	Cc
cd	cd, dc
dd	Dd
10	16

We list some more examples just for concreteness. We might, for example, consider selections of $\{\infty \cdot a, \infty \cdot b, \infty \cdot c\}$ where b can be chosen only even number of times. Thus, 5-combinations with these repetition numbers and this constraint would be those 5-combinations with unlimited repetitions and where b is chosen 0, 2, or 4 times.

Example6 The 3-combinations of $\{\infty \cdot a, \infty \cdot b, 1 \cdot c, 1 \cdot d\}$ where b can be chosen only an even number of times are the 3-combinations of a, b, c, d where a can be chosen up 3 times, b can be chosen 0 or 2 times, and c and d can be chosen at most once. The 3-cimbinations subject to these constraints are:aaa, aac, aad, bbc, bbd, acd.

As another example, we might be interested in, selections of $\{\infty \cdot a, 3 \cdot b, 1 \cdot c\}$ where a can be chosen a prime number of times. Thus, the 8-combinations subject to these constraints would be all those 8-combinations where a can be chosen 2, 3, 5, or 7 times, b can chosen up to 3 times, and c can be chosen at most once.

While there may be an infinite variety of constraints, we are primarily interested in two major types: one we have already described—combinations and permutations with unlimited repetitions, the other we now describe.

Combinations And Permutations With Repetitions:

General formulas for enumerating combinations and permutations will now be presented. At this time, we will only list formulas for combinations and permutations without repetitions or with unlimited repetitions. We will wait until later to use generating functions to give general techniques for enumerating combinations where other rules govern the selections.

Let P (n, r) denote the number of r-permutations of n elements without repetitions.

Theorem 5.3.1.(Enumerating r-permutations without repetitions).

P(n, r) = n(n-**1)…….** **(n –** r + 1) = n! / (n-r)!

Proof. Since there are n distinct objects, the first position of an r-permutation may be filled in n ways. This done, the second position can be filled in n-1 ways since no repetitions are allowed and there are n – 1 objects left to choose from. The third can be filled in n-2 ways. By applying the product rule, we conduct that

P (n, r) = n(n-1)(n-**2)…….** **(n –** r + 1). From the definition of factorials, it follows that

P (n, r) = n! / (n-r)!

When r = n, this formula becomes

P (n, n) = n! / 0! = n!

When we explicit reference to r is not made, we assume that all the objects are to be arranged; thus we talk about the permutations of n objects we mean the case r=n. Corollary 1. There are n! permutations of n distinct objects.

Example 1.

There are 3! = 6 permutations of {a, b, c}.

There are 4! = 24 permutations of (a, b, c, d). The number of 2-permutations {a, b, c, d, e} is P(5, 2) = 5! /(5 - 2)! = 5 x 4 = 20.

The number of 5-letter words using the letters a, b, c, d, and e at most once is P (5, 5) = 120.

Example 2 There are P (10, 4) = 5,040 4-digit numbers that contain no repeated digits since each such number is just an arrangement of four of the digits 0, 1, 2, 3 , …., 9 (leading zeroes are allowed). There are P (26, 3) P(10, 4) license plates formed by 3 distinct letters followed by 4 distinct digits.

Example3. In how many ways can 7 women and 3 men be arranged in a row if the 3 men must always stand next to each other?

There are 3! ways of arranging the 3 men. Since the 3 men always stand next to each other, we treat them as a single entity, which we denote by X. Then if W1, W2, ….., W7 represents

the women, we next are interested in the number of ways of arranging {X, W1, W2, W3,……., W7}. There are 8! permutations these 8 objects. Hence there are (3!) (8!) permutations altogether. (of course, if there has to be a prescribed order of an arrangement on the 3 men then there are only 8! total permutations).

Example4.   In how many ways can the letters of the English alphabet be arranged so that there are exactly 5 letters between the letters a and b?

There are P (24, 5) ways to arrange the 5 letters between a and b, 2 ways to place a and b, and then 20! ways to arrange any 7-letter word treated as one unit along with the remaining 19 letters. The total is P (24, 5) (20!) (2).

permutations for the objects are being arranged in a line. If instead of arranging objects in a line, we arrange them in a circle, then the number of permutations decreases.

Example 5. In how many ways can 5 children arrange themselves in a ring?

Solution. Here, the 5 children are not assigned to particular places but are only arranged relative to one another. Thus, the arrangements (see Figure 2-3) are considered the same if the children are in the same order clockwise. Hence, the position of child C1 is immaterial and it is only the position of the 4 other children relative to C1 that counts. Therefore, keeping C1 fixed in position, there are 4! Arrangements of the remaining children.

Binomial coefficients:

In mathematics, the binomial coefficient $\binom{n}{.}$ is the coefficient of the nth term in the polynomial expansion of the binomial power $(1 + x)^n$. In combinatorics, $\binom{n}{k}$ is interpreted as the number of k-element subsets (the k-combinations) of an n-element set, that is the number of ways that k things can be "chosen" from a set of n things.

Hence, is often read as "n choose k" and is called the choose function of n and k. This notation was introduced by Andreas von Ettingshausen. Alternative notations include C(n, k), C_k^n, C_n^k,nCk, in all of which the C stands for combinations or choices.

For natural numbers (taken to include 0) n and k, the binomial coefficient $\binom{n}{.}$can be defined as the coefficient of the monomial X in the expansion of (1 + X). The same coefficient also occurs (if k ≤ n) in the binomial formula

$$(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

(valid for any elements x,y of a commutative ring), which explains the name "binomial coefficient".

Binomial Multinomial theorems:

Binomial theorem:

In elementary algebra, the **binomial theorem** describes the algebraic expansion of powers of a binomial.According to the theorem, it is possible to expand the power (x + y) into a sum b c involving terms of the form ax y , where the coefficient of each term is a positive integer, and

the sum of the exponents of x and y in each term is n. For example,

$$(x + y)^4 = x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4.$$

The coefficients appearing in the binomial expansion are known as binomial coefficients. They

are the same as the entries of Pascal's triangle, and can be determined by a simple formula

n−k k

involving factorials. These numbers also arise in combinatorics, where the coefficient of xy is equal to the number of different combinations of k elements that can be chosen from an n-element set.

According to the theorem, it is possible to expand any power of x + y into a sum of the form

$$(x + y)^n = \binom{n}{0}x^n y^0 + \binom{n}{1}x^{n-1}y^1 + \binom{n}{2}x^{n-2}y^2 + \binom{n}{3}x^{n-3}y^3 + \cdots$$
$$\cdots + \binom{n}{n-1}x^1 y^{n-1} + \binom{n}{n}x^0 y^n,$$

where $\binom{n}{k}$ denotes the corresponding binomial coefficient. Using summation notation, the formula above can be written

$$(x + y)^n = \sum_{k=0}^{n} \binom{n}{k}x^{n-k}y^k.$$

This formula is sometimes referred to as the **Binomial Formula** or the **Binomial Identity**.

A variant of the binomial formula is obtained by substituting 1 for x and x for y, so that it involves only a single variable. In this form, the formula reads

$$(1 + x)^n = \binom{n}{0}x^0 + \binom{n}{1}x^1 + \binom{n}{2}x^2 + \cdots + \binom{n}{n-1}x^{n-1} + \binom{n}{n}x^n,$$

or equivalently

$$(1 + x)^n = \sum_{k=0}^{n} \binom{n}{k}x^k.$$

Example

Simplify (x+v(x2-1)) + (x- v(x2-1))6 Solution: let vx2-1 = a, so we have:

(x=a)6 + (x-a)6

= [x6+6C1x5.a+6C2.x4.a2 + 6C3x3a3 + 6C4x2a4 + 6C5xa5 +6C6a6] [x6-6C1x5a+6C2.x4.a2 – 6C3x3a3 + 6C4x2a4 – 6C5xa5 +6C6a6]

x6+6C2x4a2+6C4x2a4+6C6a6] x6+15x4(x2-1)+15x2(x2-1)2+(x2-1)3]

x6+15x6-15x4+15x6+15x2-30x4+x6-1-3x4+3x3] 32x6-48x4+18x2-1]

Multinomial theorem:

In mathematics, the **multinomial theorem** says how to write a power of a sum in terms of powers of the terms in that sum. It is the generalization of the binomial theorem to polynomials.

For any positive integer m and any nonnegative integer n, the multinomial formula tells us how a polynomial expands when raised to an arbitrary power:

$$(x_1 + x_2 + \cdots + x_m)^n = \sum_{k_1+k_2+\cdots+k_m=n} \binom{n}{k_1, k_2, \ldots, k_m} x_1^{k_1} x_2^{k_2} \cdots x_m^{k_m}.$$

The summation is taken over all sequences of nonnegative integer indices k_1 through k_m such the sum of all k_i is n. That is, for each term in the expansion, the exponents must add up to n.

Also, as with the binomial theorem, quantities of the form x that appear are taken to equal 1 (even when x equals zero). Alternatively, this can be written as

$$(x_1 + \cdots + x_m)^n = \sum_{|\alpha|=n} \binom{n}{\alpha} x^{\alpha}$$ **where α = ($α_1, α_2, \ldots, α_m$) and x**

Example

$(a + \mathbf{b} + \mathbf{c})^3 = a^3 + b^3 + c^3 + 3a^2b + 3a^2c + 3b^2c + 3b^2a + 3c^2a + 3c^2a + 6abc$

However calculation in above process is slow, and can be avoided by using the multinomial theorem. The multinomial theorem "solves" this process by giving us the closed form for any coefficient we might want. It is possible to "read off" the multinomial coefficients from the terms by using the multinomial coefficient formula. For example:

$$\binom{3}{2, 0, 1} = \frac{3!}{2! \cdot 0! \cdot 1!} = \frac{6}{2 \cdot 1 \cdot 1} = 3$$

$$\binom{3}{1, 1, 1} = \frac{3!}{1! \cdot 1! \cdot 1!} = \frac{6}{1 \cdot 1 \cdot 1} = 6$$

We could have also had a 'd' variable, or even more variables—hence the multinomial theorem.

The principles of Inclusion – Exclusion:

In combinatorics (combinatorial mathematics), the **inclusion–exclusion principle** is a counting technique which generalizes the familiar method of obtaining the number of elements in the union of two finite sets; symbolically expressed as

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

where A and B are two finite sets and |S| indicates the cardinality of a set S (which may be considered as the number of elements of the set, if the set is finite). The formula expresses the fact that the sum of the sizes of the two sets may be too large since some elements may be counted twice. The double-counted elements are those in the intersection of the two sets and the count is corrected by subtracting the size of the intersection.

The principle is more clearly seen in the case of three sets, which for the sets A, B and C is given by

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

Ex: How many integers in {1,...,100} are not divisible by 2, 3 or 5?

Let S = {1,...,100} and P_1 the property that an integer is divisible by 2, P_2 the property that an integer is divisible by 3 and P_3 the property that an integer is divisible by 5. Letting A_i be the subset of S whose elements have property P_i we have by elementary counting: $|A_1| = 50$, $|A_2| = 33$, and $|A_3| = 20$. There are 16 of these integers divisible by 6, 10 divisible by 10 and 6 divisible by 15. Finally, there are just 3 integers divisible by 30, so the number of integers not divisible by any of 2, 3 or 5 is given by:

$$100 - (50 + 33 + 20) + (16 + 10 + 6) - 3 = 26.$$

Recurrence Relation

Generating Functions:

In mathematics, a **generating function** is a formal power series in one indeterminate, whose Coefficients encode information about a sequence of numbers a_n that is indexed by the natural numbers. Generating functions were first introduced by Abraham de Moivre in 1730, in order to solve the general linear recurrence problem. One can generalize to formal power series in more than one indeterminate, to encode information about arrays of numbers indexed by several natural numbers.

Generating functions are not functions in the formal sense of a mapping from a domain to a codomain; the name is merely traditional, and they are sometimes more correctly called **generating series**.

Ordinary generating function

The ordinary generating function of a sequence a_n is

$$G(a_n; x) = \sum_{n=0}^{\infty} a_n x^n.$$

When the term generating function is used without qualification, it is usually taken to mean an ordinary generating function.

If a_n is the probability mass function of a discrete random variable, then its ordinary generating function is called a probability-generating function.

The ordinary generating function can be generalized to arrays with multiple indices. For example, the ordinary generating function of a two-dimensional array $a_{m, n}$ (where n and m are natural numbers) is

$$G(a_{m,n}; x, y) = \sum_{m,n=0}^{\infty} a_{m,n} x^m y^n.$$

Example:

$$G(n^2; x) = \sum_{n=0}^{\infty} n^2 x^n = \frac{x(x+1)}{(1-x)^3}$$

Exponential generating function

$$\mathrm{EG}(a_n; x) = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}.$$ The exponential generating function of a sequence a_n is

$$\mathrm{EG}(n^2; x) = \sum_{n=0}^{\infty} \frac{n^2 x^n}{n!} = x(x+1)e^x$$

Function of Sequences:

Generating functions giving the first few powers of the nonnegative integers are given in the following table.

n^p	$f(x)$	
	$\frac{x}{1-x}$	$x + x^2 + x^3 + \ldots$
n	$\frac{x}{(1-x)^2}$	$x + 2x^2 + 3x^3 + 4x^4 + \ldots$
n^2	$\frac{x(x+1)}{(1-x)^3}$	$x + 4x^2 + 9x^3 + 16x^4 + \ldots$
n^3	$\frac{x(x^2+4x+1)}{(1-x)^4}$	$x + 8x^2 + 27x^3 + \ldots$
n^4	$\frac{x(x+1)(x^2+10x+1)}{(1-x)^5}$	$x + 16x^2 + 81x^3 + \ldots$

There are many beautiful generating functions for special functions in number theory. A few particularly nice examples are

$$f(x) \quad = \quad \frac{1}{(x)_\infty}$$

$$= \quad \sum_{n=0}^{\infty} P(n) x^n$$

$$= \quad 1 + x + 2x^2 + 3x^3 + \ldots$$

for the partition function P, where $(q)_\infty$ is a q-Pochhammer symbol, and

$$f(x) \quad = \quad \frac{x}{1-x-x^2}$$

$$= \quad \sum_{n=0}^{\infty} F_n x^n$$

$$= \quad x + x^2 + 2x^3 + 3x^4 + \ldots$$

for the Fibonacci numbers F_n.

mMGenerating functions are very useful in combinatorial enumeration problems. For example, the subset sum problem, which asks the number of ways $C_{m,s}$ to select out of given integers such that their sum equals s, can be solved using generating functions.

Calculating Coefficient of generating function:

By using the polynomial expansions, we can calculate the coefficient of a generating function.

Example

Find the coefficient of x^{16} in $(x^2 + x^3 + x^4 + \ldots)^5$

x^{16} in $x^{10}(1-x)^{-5}$ [i.e, the x^6 term in $(1-x)^{-5}$ is

multiplied by x^{10} to become the x^{16} term in $x^{10}(1-x)^{-5}$]

To simplify the expression, we extract x^2 from each polynomial factor and the apply identity (2).

$$(x^2 + x^3 + x^4 + \ldots)^5 = [x^2(1 + x + x^2 + \ldots)]^5$$

$$= x^{10}(1 + x + x^2 + \ldots)^5$$

$$= x^{10}\frac{1}{(1-x)^5}$$

Thus the coefficient of x^{16} in $(x^2 + x^3 + x^4 + \ldots)^5$ is the coefficient of

x^{16} in $x^{10}(1-x)^{-5}$ [i.e., the x^6 term in $(1-x)^{-5}$ is multiplied by to become the x^{16} term in $x^{10}(1-x)^{-5}$]

$$\frac{1}{(1-x)^n} = 1 + C(1+n-1,1)x + C(2+n-1,2)x^2 + \ldots + C(r+n-1,r)x^r + \ldots$$

From expansion (5) we see that the coefficient of x^6 in $(1-x)^{-5}$ is $C(6+5-1,6)$

More generally, the coefficient of x^r in

x^r in $x^{10}(1-x)^{-5}$ equals the coefficient of x^{r-10} in

$(1-x)^{-5}$, namely, $C((r-10)+5-1,(r-10))$.

Recurrence relations:

Introduction :A recurrence relation is a formula that relates for any integer $n \geq 1$, the n-th term of a sequence $A = \{a_r\}\infty r=0$ to one or more of the terms $a_0, a_1, \ldots, a_{n-1}$.

Definition. Suppose n and k are nonnegative integers. A recurrence relation of the form $c_0(n)a_n$

$+ c_1(n)a_{n-1} + \ldots + c_k(n)a_{n-k} = f(n)$ for $n \geq k$, where $c_0(n)$, $c_1(n), \ldots, c_k(n)$, and $f(n)$ are functions of n is said to be a linear recurrence relation. If $c_0(n)$ and $c_k(n)$ are not identically zero, then it is said to be a linear recurrence relation degree k. If $c_0(n)$, $c_1(n), \ldots, c_k(n)$ are constants, then the recurrence relation is known as a linear relation with constant coefficients. If $f(n)$ is identically zero, then the recurrence relation is said to be homogeneous; otherwise, it is inhomogeneous.

There are no general techniques that will enable one to solve all recurrence relations. There are techniques that will enable us to solve linear recurrence relations with constant coefficients.

SOLVING RECURRENCE RELATIONS BY SUSTITUTION AND GENERATING FUNCTIONS

We shall consider four methods of solving recurrence relations in this and the next two sections:

•   Substitution (also called iteration),

•   Generating functions,

•   Characteristics roots, and

In the substitution method the recurrence relation for an is used repeatedly to solve for a general expression for an in terms of n. We desire that this expression involve no other terms of the sequence except those given by boundary conditions.

Example

Solve the recurrence relation an = a n-1 + f(n) for n ³1 by substitution a1= a0 + f(1)

a2 = a1 + f(2) = a0 + f(1) + f(2))

a3 = a2 + f(3)= a0 + f(1) + f(2) + f(3)

.

.

.

an = a0 + f(1) + f(2) +….+ f(n) n

= a0 + ∑ f(k)

K = 1

Thus, an is just the sum of the f(k) „s plus a0.

More generally, if c is a constant then we can solve an = c a n-1 + f(n) for n ³1 in the same way: a1 = c a0 + f(1)

a2 = c a1 + f(2) = c (c a0 + f(1)) + f(2) = c2 a0 + c f(1) + f(2)

a3= c a2 + f(3) = c(c 2 a0 + c f(1) + f(2)) + f(3) =c3 a0 + c2 f(1) + c f(2) + f(3)

.

.

.

an = c a n-1 + f(n) = c(c n-1 a0 + c n-2 f(1) +. . . + c n-2 + f(n-1)) +

f(n) =c n a0 + c n-1 f(1) + c n-2 f(2) +. . .+ c f(n-1) + f(n)

Or

an = c n a0 + ∑c n-k f(k)

Generating function examples:

Find the generating functions for the following sequences.

(a) 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, …

(b) 1, 1, 1, 1, 1, …

(c) 1, 3, 3, 1, 0, 0, 0, 0, …

Solution:

(a) The generating function is

$$G(x) = 1 + 1x + 1x^2 + 1x^3 + 1x^4 + 1x^5 + 0x^6 + 0x^7 + \cdots$$
$$= 1 + x + x^2 + x^3 + x^4 + x^5$$

We can apply the formula for the sum of a geometric series to rewrite $G(x)$ as $G(x) = \dfrac{1 - x^6}{1 - x}$.

(b) The generating function is

$$G(x) = 1 + x + x^2 + x^3 + x^4 + \cdots.$$

When $|x| < 1$, we can apply the formula for the sum to infinity of a geometric series to rewrite $G(x)$ as $G(x) = \dfrac{1}{1 - x}$. In working with generating functions, we shall ignore the question of convergence and simply say $G(x) = \dfrac{1}{1 - x}$.

(c) The generating function is $G(x) = 1 + 3x + 3x^2 + 1$, and of course, the binomial theorem enables us to simplify the answer as $G(x) = (1 + x)^3$.

Characteristic roots of homogeneous recurrence relation:

Ex: Let ak = 5ak-1 - 6ak-2 . Find the general solution.

 The relation has characteristic equation:

x^2 = 5x - 6,

so x^2 - 5x + 6 = 0

hence (x - 2)(x - 3) = 0

implying either (x - 2) = 0 or (x - 3) = 0

thus x = 2,3

General Solution is an = C(2^n) + D(3^n).

Graph Theory

Representation of Graphs:

There are two different sequential representations of a graph. They are Adjacency Matrix representation and Path Matrix representation

Adjacency Matrix Representation

Suppose G is a simple directed graph with m nodes, and suppose the nodes of G have been ordered and are called v1, v2, . . . , vm. Then the adjacency matrix $A = (a_{ij})$ of the graph G is the m x m matrix defined as follows:

1 if vi is adjacent to Vj, that is, if there is an edge (Vi, Vj) a_{ij} =0 otherwise

Suppose G is an undirected graph. Then the adjacency matrix A of G will be a symmetric matrix, i.e., one in which $a_{ij} = a_{ji}$; for every i and j.

Drawbacks

• It may be difficult to insert and delete nodes in G.

• If the number of edges is 0(m) or 0(m log2 m), then the matrix A will be sparse, hence a great deal of space will be wasted.

Path Matrix Representation

Let G be a simple directed graph with m nodes, v1,v2, . . . ,vm. The path matrix of G is the m-square matrix $P = (p_{ij})$ defined as follows:

1 if there is a path from Vi to Vj P_{ij} =0 otherwise

Isomorphism:

Let G1 and G1 be two graphs and let f be a function from the vertex set of G1 to the vertex set of G2. Suppose that f is one-to-one and onto & f(v) is adjacent to f(w) in G2 if and only if v is adjacent to w in G1.
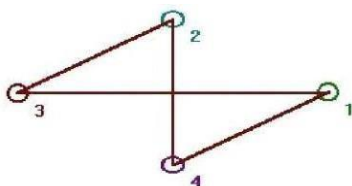
Then we say that the function f is an isomorphism and that the two graphs G1 and G2 are isomorphic. So two graphs G1 and G2 are isomorphic if there is a one-to-one correspondence between vertices of G1 and those of G2 with the property that if two vertices of G1 are adjacent then so are their images in G2. If two graphs are isomorphic then as far as we are concerned they are the same graph though the location of the vertices may be different.

Example:
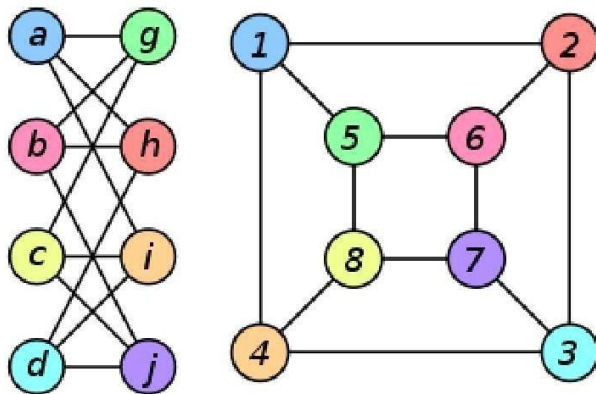
The two graphs shown below are isomorphic, despite their different looking drawings.

Graph G	Graph H	An isomorphism between G and H
		$f(a) = 1$
		$f(b) = 6$
		$f(c) = 8$
		$f(d) = 3$
		$f(g) = 5$
		$f(h) = 2$
		$f(i) = 4$
		$f(j) = 7$

Euler circuits:

In graph theory, an **Eulerian trail** is a trail in a graph which visits every edge exactly once. Similarly, an **Eulerian circuit** is an Eulerian trail which starts and ends on the same vertex. They were first discussed by Leonhard Euler while solving the famous Seven Bridges of Königsberg problem in 1736. Mathematically the problem can be stated like this:

Given the graph on the right, is it possible to construct a path (or a cycle, i.e. a path starting and ending on the same vertex) which visits each edge exactly once?

Euler proved that a necessary condition for the existence of Eulerian circuits is that all vertices in the graph have an even degree, and stated without proof that connected graphs with all vertices of even degree have an Eulerian circuit. The first complete proof of this latter claim was published in 1873 by Carl Hierholzer.

The term **Eulerian graph** has two common meanings in graph theory. One meaning is a graph with an Eulerian circuit, and the other is a graph with every vertex of even degree. These definitions coincide for connected graphs.

For the existence of Eulerian trails it is necessary that no more than two vertices have an odd degree; this means the Königsberg graph is not Eulerian. If there are no vertices of odd degree, all Eulerian trails are circuits. If there are exactly two vertices of odd degree, all Eulerian trails start at one of them and end at the other. Sometimes a graph that has an Eulerian trail but not an Eulerian circuit is called **semi-Eulerian**.

An **Eulerian trail**, **Eulerian trail** or **Euler walk** in an undirected graph is a path that uses each edge exactly once. If such a path exists, the graph is called **traversable** or **semi-eulerian**.
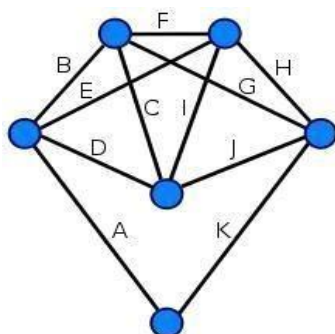
An **Eulerian cycle**, **Eulerian circuit** or **Euler tour** in an undirected graph is a cycle that uses each edge exactly once. If such a cycle exists, the graph is called **unicursal**. While such graphs are Eulerian graphs, not every Eulerian graph possesses an Eulerian cycle.

For directed graphs path has to be replaced with directed path and cycle with directed cycle.

The definition and properties of Eulerian trails, cycles and graphs are valid for multigraphs as well.
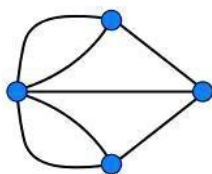
This graph is not Eulerian, therefore, a solution does not exist.

Every vertex of this graph has an even degree, therefore this is an Eulerian graph. Following the edges in alphabetical order gives an Eulerian circuit/cycle.

Hamiltonian graphs:

In the mathematical field of graph theory, a **Hamiltonian path** (or **traceable path**) is a path in an undirected graph which visits each vertex exactly once. A **Hamiltonian cycle** (or **Hamiltonian circuit**) is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex. Determining whether such paths and cycles exist in graphs is the Hamiltonian path problem which is NP-complete.

Hamiltonian paths and cycles are named after William Rowan Hamilton who invented the Icosian game, now also known as Hamilton's puzzle, which involves finding a Hamiltonian cycle in the edge graph of the dodecahedron. Hamilton solved this problem using the Icosian Calculus, an algebraic structure based on roots of unity with many similarities to the quaternions (also invented by Hamilton). This solution does not generalize to arbitrary graphs.

A Hamiltonian path or traceable path is a path that visits each vertex exactly once. A graph that contains a Hamiltonian path is called a **traceable graph**. A graph is **Hamilton-connected** if for every pair of vertices there is a Hamiltonian path between the two vertices.

A Hamiltonian cycle, Hamiltonian circuit, vertex tour or graph cycle is a cycle that visits each vertex exactly once (except the vertex which is both the start and end, and so is visited twice). A graph that contains a Hamiltonian cycle is called a **Hamiltonian graph**.

Similar notions may be defined for directed graphs, where each edge (arc) of a path or cycle can only be traced in a single direction (i.e., the vertices are connected with arrows and the edges traced "tail-to-head").

A **Hamiltonian decomposition** is an edge decomposition of a graph into Hamiltonian circuits.

Examples

- a complete graph with more than two vertices is Hamiltonian
- every cycle graph is Hamiltonian
- every tournament has an odd number of Hamiltonian paths
- every platonic solid, considered as a graph, is Hamiltonian

Planar Graphs:

In graph theory, a **planar graph** is a graph that can be embedded in the plane, i.e., it can be drawn on the plane in such a way that its edges intersect only at their endpoints.

A planar graph already drawn in the plane without edge intersections is called a **plane graph** or **planar embedding of the graph**. A plane graph can be defined as a planar graph with a mapping from every node to a point in 2D space, and from every edge to a plane curve, such

that the extreme points of each curve are the points mapped from its end nodes, and all curves are disjoint except on their extreme points. Plane graphs can be encoded by combinatorial maps.

It is easily seen that a graph that can be drawn on the plane can be drawn on the sphere as well, and vice versa.

The equivalence class of topologically equivalent drawings on the sphere is called a **planar map**. Although a plane graph has an **external** or **unbounded** face, none of the faces of a planar map have a particular status.

Applications

- Telecommunications – e.g. spanning trees

- Vehicle routing – e.g. planning routes on roads without underpasses • VLSI – e.g. laying out circuits on computer chip.

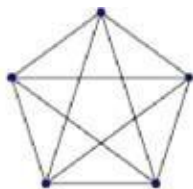•The puzzle game Planarity requires the player to "untangle" a planar graph so that none of its edges intersect.

Example graphs

Planar   **non planar**
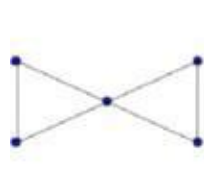

Butterfly graph   K5

Chromatic Numbers:

In graph theory, **graph coloring** is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color; this is called a **vertex coloring**. Similarly, an **edge coloring** assigns a color to each edge so that no two adjacent edges share the same color, and a **face coloring** of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color.

Vertex coloring is the starting point of the subject, and other coloring problems can be transformed into a vertex version. For example, an edge coloring of a graph is just a vertex coloring of its line graph, and a face coloring of a planar graph is just a vertex coloring of its planar dual. However, non-vertex coloring problems are often stated and studied as is. That is

partly for perspective, and partly because some problems are best studied in non-vertex form, as for instance is edge coloring.

The convention of using colors originates from coloring the countries of a map, where each face is literally colored. This was generalized to coloring the faces of a graph embedded in the plane. By planar duality it became coloring the vertices, and in this form it generalizes to all graphs. In mathematical and computer representations it is typical to use the first few positive or nonnegative integers as the "colors". In general one can use any finite set as the "color set". The nature of the coloring problem depends on the number of colors but not on what they are.

Graph coloring enjoys many practical applications as well as theoretical challenges. Beside the classical types of problems, different limitations can also be set on the graph, or on the way a color is assigned, or even on the color itself. It has even reached popularity with the general public in the form of the popular number puzzle Sudoku. Graph coloring is still a very active field of research.
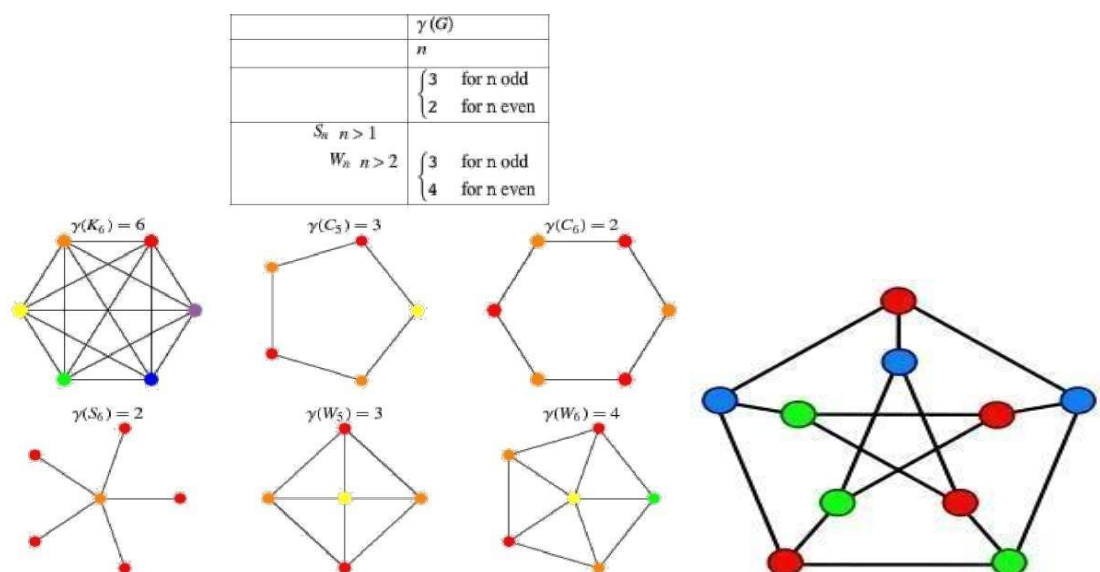

A proper vertex coloring of the Petersen graph with 3 colors, the minimum number possible.

Vertex coloring

When used without any qualification, a **coloring** of a graph is almost always a proper vertex coloring, namely a labelling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color. Since a vertex with a loop could never be properly colored, it is understood that graphs in this context are loopless.

The terminology of using colors for vertex labels goes back to map coloring. Labels like red and blue are only used when the number of colors is small, and normally it is understood that the labels are drawn from the integers {1,2,3,...}.
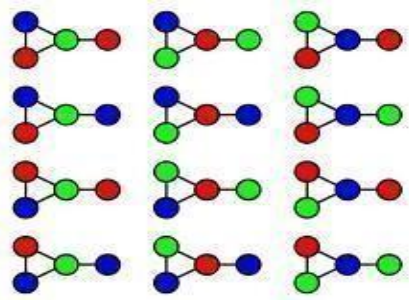
A coloring using at most k colors is called a (proper) **k-coloring**. The smallest number of colors needed to color a graph G is called its **chromatic number**, $\chi(G)$. A graph that can be assigned a (proper) k-coloring is **k-colorable**, and it is **k-chromatic** if its chromatic number is exactly k. A subset of vertices assigned to the same color is called a color class, every such class forms an independent set. Thus, a k-coloring is the same as a partition of the vertex set into k independent sets, and the terms k-partite and k-colorable have the same meaning.

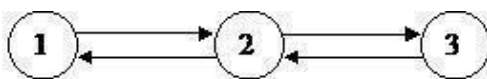**Directed Graphs**

A directed graph G, also called a digraph or graph is the same as a multigraph except that each edge e in G is assigned a direction, or in other words, each edge e is identified with an ordered pair (u, v) of nodes in G.

Indegree : The indegree of a vertex is the number of edges for which v is head

Outdegree :The outdegree of a node or vertex is the number of edges for which v is tail.

Example

Outdegree of 1 =1

Outdegree of 2 =2

Indegree of 1=1

Indegree of 2 = 2

Simple Directed Graph

A directed graph G is said to be simple if G has no parallel edges. A simple graph G may have loops, but it cannot have more than one loop at a given node.

Directed Acyclic Graph (DAG)

A directed acyclic graph (DAG) is a finite directed graph with no directed cycles. That is, it consists of finitely many vertices and edges (also called *arcs*), with each edge directed from one vertex to another, such that there is no way to start at any vertex *v* and follow a consistently-directed sequence of edges that eventually loops back to *v* again. Equivalently, a DAG is a directed graph that has a topological ordering, a sequence of the vertices such that every edge is directed from earlier to later in the sequence. Every directed acyclic graph has

a topological ordering, an ordering of the vertices such that the starting endpoint of every edge occurs earlier in the ordering than the ending endpoint of the edge. The existence of such an ordering can be used to characterize DAGs: a directed graph is a DAG if and only if it has a topological ordering.

Labeled or Weighted Graph

If the weight is assigned to each edge of the graph then it is called as Weighted or Labeled graph.

The definition of a graph may be generalized by permitting the following:

- **Multiple edges**: Distinct edges e and e' are called multiple edges if they connect the same endpoints, that is, if e = [u, v] and e' = [u, v].

- **Loops:** An edge e is called a loop if it has identical endpoints, that is, if e = [u, u].

- **Finite Graph**:A multigraph M is said to be finite if it has a finite number of nodes and a finite number of edges.

(a) Graph.   (b) Multigraph.'
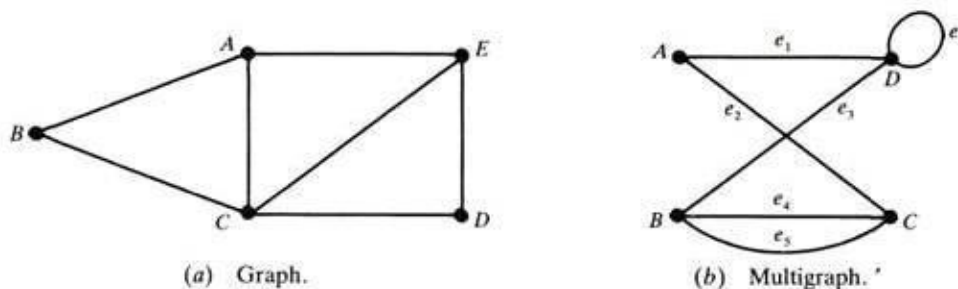
Trees:

A **tree** is an undirected graph in which any two vertices are connected by *exactly one* path. Every acyclic connected graph is a tree, and vice versa. A **forest** is a disjoint union of trees, or equivalently an acyclic graph that is not necessarily connected.

A *tree* is an undirected graph G that satisfies any of the following equivalent conditions:

- G is connected and acyclic (contains no cycles).

- G is acyclic, and a simple cycle is formed if any edge is added to G.

- G is connected, but would become disconnected if any single edge is removed from G.

- G is connected and the 3-vertex complete graph K_3 is not a minor of G.

- Any two vertices in G can be connected by a unique simple path.

If G has finitely many vertices, say n of them, then the above statements are also equivalent to any of the following conditions:

- G is connected and has $n − 1$ edges.

- G is connected, and every subgraph of G includes at least one vertex with zero or one incident edges. (That is, G is connected and 1-degenerate.)

- *G* has no simple cycles and has *n* − 1 edges.

Spanning Trees:

In the mathematical field of graph theory, a **spanning tree** T of a connected, undirected graph G is a tree composed of all the vertices and some (or perhaps all) of the edges of G. Informally, a spanning tree of G is a selection of edges of G that form a tree spanning every vertex. That is, every vertex lies in the tree, but no cycles (or loops) are formed. On the other hand, every bridge of G must belong to T.

A spanning tree of a connected graph G can also be defined as a maximal set of edges of G that contains no cycle, or as a minimal set of edges that connect all vertices.
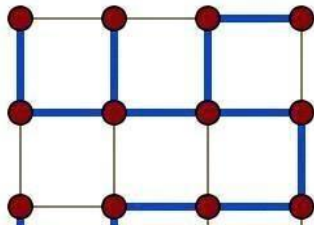
Given an undirected and connected graph G=(V,E), a spanning tree of the graph G is a tree that spans G(that is, it includes every vertex of G) and is a subgraph of G (every edge in the tree belongs to G)

Minimum Spanning Tree

The cost of the spanning tree is the sum of the weights of all the edges in the tree. There can be many spanning trees. Minimum spanning tree is the spanning tree where the cost is minimum among all the spanning trees. There also can be many minimum spanning trees.

There are two famous algorithms for finding the Minimum Spanning Tree:
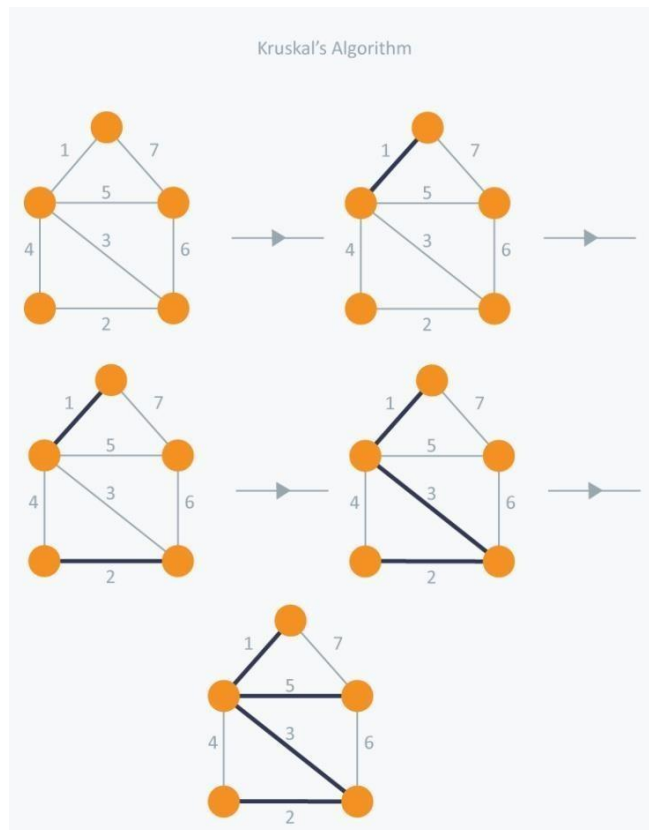
Kruskal's Algorithm

Kruskal's Algorithm builds the spanning tree by adding edges one by one into a growing spanning tree. Kruskal's algorithm follows greedy approach as in each iteration it finds an edge which has least weight and add it to the growing spanning tree.

Algorithm Steps:

- Sort the graph edges with respect to their weights.
- Start adding edges to the MST from the edge with the smallest weight until the edge of the largest weight.
- Only add edges which doesn't form a cycle , edges which connect only disconnected components.

In Kruskal's algorithm, at each iteration we will select the edge with the lowest weight. So, we will start with the lowest weighted edge first i.e., the edges with weight 1. After that we will select the second lowest weighted edge i.e., edge with weight 2. Notice these two edges

are totally disjoint. Now, the next edge will be the third lowest weighted edge i.e., edge with weight 3, which connects the two disjoint pieces of the graph. Now, we are not allowed to pick the edge with weight 4, that will create a cycle and we can't have any cycles. So we will select the fifth lowest weighted edge i.e., edge with weight 5. Now the other two edges will create cycles so we will ignore them. In the end, we end up with a minimum spanning tree with total cost 11 (= 1 + 2 + 3 + 5).

Kruskal's Algorithm

Prim's Algorithm

Prim's Algorithm also use Greedy approach to find the minimum spanning tree. In Prim's Algorithm we grow the spanning tree from a starting position. Unlike an **edge** in Kruskal's, we add **vertex** to the growing spanning tree in Prim's.

Algorithm Steps:

- Maintain two disjoint sets of vertices. One containing vertices that are in the growing spanning tree and other that are not in the growing spanning tree.
- Select the cheapest vertex that is connected to the growing spanning tree and is not in the growing spanning tree and add it into the growing spanning tree. This can be done using Priority Queues. Insert the vertices, that are connected to growing spanning tree, into the Priority Queue.
- Check for cycles. To do that, mark the nodes which have been already selected and insert only those nodes in the Priority Queue that are not marked.

In Prim's Algorithm, we will start with an arbitrary node (it doesn't matter which one) and mark it. In each iteration we will mark a new vertex that is adjacent to the one that we have already marked. As a greedy algorithm, Prim's algorithm will select the cheapest edge and mark the vertex. So we will simply choose the edge with weight 1. In the next iteration we have three options, edges with weight 2, 3 and 4. So, we will select the edge with weight 2 and mark the vertex. Now again we have three options, edges with weight 3, 4 and 5. But we can't choose edge with weight 3 as it is creating a cycle. So we will select the edge with weight 4 and we end up with the minimum spanning tree of total cost 7 (= 1 + 2 +4).