INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad -500 043

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LECTURE NOTES

ON

DISCRETE MATHEMATICAL STRUCTURES

Course Code : AHS013

BRANCH/YEAR : CSE/II

SEMESTER : II

PREPARED BY

Dr. N. Rajasekhar, Professor

Dr. J. Sirisha Devi, Professor

Mr. P.V.Narasimha Rao, Assistant Professor

Ms. A. Jayanthi, Assistant Professor

Ms. B. Pravallika Assistant Professor

Ms. B. Dhanalaxmi, Assistant Professor

UNIT- I

1. Mathematical logic

1.1. Statements and notations

1.2. Connectives

1.3. Well-formed formulas

1.4. Truth tables, tautology

1.5. Equivalence implication

1.6. Normal forms

1.7. Disjunctive normal forms

1.8. Conjunctive normal forms

1.9. Principle disjunctive normal forms

1.10. Principle conjunctive normal forms.

2. Predicate calculus

2.1. Predicative logic

2.2. Statement functions

2.3. Variables and quantifiers

2.4. Free and bound variables,

2.5. Rules of inference

2.6. Consistency

2.7. Proof of contradiction

2.8. Automatic theorem proving.

UNIT-I

MATHEMATICAL LOGIC

Introduction

Discrete mathematics is mathematics that deals with discrete objects. Discrete objects are those which are separated from (not connected to/distinct from) each other. Integers (aka whole numbers), rational numbers (ones that can be expressed as the quotient of two integers), automobiles, houses, people etc. are all discrete objects. On the other hand real numbers which include irrational as well as rational numbers are not discrete. As you know between any two different real numbers there is another real number different from either of them. So they are packed without any gaps and can not be separated from their immediate neighbors. In that sense they are not discrete. In this course we will be concerned with objects such as integers, propositions, sets, relations and functions, which are all discrete. We are going to learn concepts associated with them, their properties, and relationships among them among others.

Types of Mathematics:

Mathematics can be broadly classified into two categories:

- Continuous Mathematics
- Discrete Mathematics

Continuous Mathematics is based upon continuous number line or the real numbers. It is characterized by the fact that between any two numbers, there are almost always an infinite set of numbers. For example, a function in continuous mathematics can be plotted in a smooth curve without breaks.

Discrete Mathematics, on the other hand, involves distinct values; i.e. between any two points, there are a countable number of points. For example, if we have a finite set of objects, the function can be defined as a list of ordered pairs having these objects, and can be presented as a complete list of those pairs.

Logic

Logic is a language for reasoning. It is a collection of rules we use when doing logical reasoning. Human reasoning has been observed over centuries from at least the times of Greeks, and patterns appearing in reasoning have been extracted, abstracted, and streamlined. The foundation of the logic we are going to learn here was laid down by a British mathematician George Boole in the middle of the 19th century, and it was further developed and used in an attempt to derive all of mathematics by Gottlob Frege, a German mathematician, towards the end of the 19th century. A British philosopher/mathematician, Bertrand Russell, found a flaw in basic assumptions in Frege's attempt but he, together with Alfred Whitehead, developed Frege's work further and repaired the damage. The logic we study today is more or less along this line.

In logic we are interested in true or false of statements, and how the truth/falsehood of a statement can be determined from other statements. However, instead of dealing with individual specific statements, we are going to use symbols to represent arbitrary statements so that the results can be used in many similar but different cases. The formalization also promotes the clarity of thought and eliminates mistakes.

There are various types of logic such as logic of sentences (propositional logic), logic of objects (predicate logic), logic involving uncertainties, logic dealing with fuzziness, temporal logic etc. Here we are going to be concerned with propositional logic and predicate logic, which are fundamental to all types of logic.

Propositional logic

Propositional logic is a logic at the sentential level. The smallest unit we deal with in propositional logic is a sentence. We do not go inside individual sentences and analyze or discuss their meanings. We are going to be interested only in true or false of sentences, and major concern is whether or not the truth or falsehood of a certain sentence follows from those of a set of sentences, and if so, how. Thus sentences considered in this logic are not arbitrary sentences but are the ones that are true or false. This kind of sentences are called **propositions**.

Statements and notations

Statements or Propositions:

A Statement or Proposition is a declarative sentence which is either true or false but not both. The truth or falsity of a statement is called its truth value. If a proposition is true, then we say it has a **truth value** of "**true**"; if a proposition is false, its truth value is "**false**". The truth value of a true statement is denoted by 'T' and the false statement is denoted by 'F'. They are also denoted by 1 or 0.

Examples:

1. There are 5 days in a week. 2. 5+4=9

3. y+3=8

- It will rain tomorrow
- There are 12 months in a year
- Examples (2) and (5) are true statements.
- Example (1) is a false statement.

- In example (3), its truth value depends upon the value of y. If y is 5 then sentence is true and if y not equal to 5 then sentence is false. Therefore (4) is not a statement.
- In example (4), its truth value cannot be predicted at this moment but it can be definitely determined tomorrow. Hence it is a statement.

The following are not propositions:

- "A is less than 2". It is because unless we give a specific value of A, we cannot say whether the statement is true or false.
- Also "x is greater than 2", where x is a variable representing a number, is not a proposition, because unless a specific value is given to x we can not say whether it is true or false, nor do we know what x represents.
- Similarly "x = x" is not a proposition because we don't know what "x" represents hence what "=" means. For example, while we understand what "3 = 3" means, what does "Air is equal to air" or

"Water is equal to water" mean? Does it mean a mass of air is equal to another mass or the concept of air is equal to the concept of air? We don't quite know what "x = x" mean. Thus we cannot say whether it is true or not. Hence it is not a proposition.

Connectives

Simple sentences which are true or false are basic propositions. Larger and more complex sentences are constructed from basic propositions by combining them with **connectives**. Thus **propositions** and **connectives** are the basic elements of propositional logic. Though there are many connectives, we are going to use the following **five basic connectives** here:

OR (∨), AND (∧), Negation/ NOT (¬), Implication / if-then (→), If and only if (⇔).

OR (∨) − The OR operation of two propositions A and B (written as A ∨ B) is true if at least any of the propositional variable A or B is true.

The truth table is as follows −

A	**B**	**A ∨ B**
True	True	True
True	False	True
False	True	True
False	False	False

AND (∧) − The AND operation of two propositions A and B (written as A ∧ B) is true if both the propositional variable A and B is true.

The truth table is as follows −

A	**B**	**A ∧ B**
True	True	True
True	False	False
False	True	False
False	False	False

Negation (¬) − The negation of a proposition A (written as ¬A) is false when A is true and is true when A is false.

The truth table is as follows −

A	**¬A**
True	False
False	True

Implication / if-then (→) − An implication A→B is False if A is true and B is false. The rest cases are true.

The truth table is as follows −

A	**B**	**A → B**
True	True	True
True	False	False
False	True	True
False	False	True

Biconditional / If and only if (⇔) − A⇔B is bi-conditional logical connective which is true when p and q are both false or both are true.

The truth table is as follows −

A	**B**	**A ⇔ B**
True	True	True
True	False	False
False	True	False
False	False	True

Well formed formulas

Subjects to be Learned

1. wff (well formed formula)
2. atomic formula
3. syntax of wff

Contents

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 is a predicate name, and x and y are variables, is called an **atomic formula**.

Wffs are constructed using the following rules:

1. True and False are wffs.
2. Each propositional constant (i.e. specific proposition), and each propositional variable (i.e. a variable representing propositions) are wffs.

3. Each atomic formula (i.e. a specific predicate with variables) is a wff.
4. If A, B, and C are wffs, then so are A, (A B), (A B), (A B), and (A B).
5. 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 .

For example, "The capital of Virginia is Richmond." is a specific proposition. Hence it is a well formed formula by Rule 2.
Let **B** be a predicate name representing "being blue" and let **x** be a variable. Then **B(x)** is an atomic formula meaning "**x** is blue". Thus it is a wff by Rule 3. above. By applying Rule 5. to **B(x)**, **xB(x)** is a wff and so is **xB(x)**. Then by applying Rule 4. to them **x B(x)** **x B(x)** is seen to be a wff. Similarly if **R** is a predicate name representing "being round". Then **R(x)** is an atomic formula. Hence it is a wff. By applying Rule 4 to **B(x)** and **R(x)**, a wff **B(x)** **R(x)** is obtained.
In this manner, larger and more complex wffs can be constructed following the rules given above. Note, however, that strings that can not 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)) .**

One way to check whether or not an expression is a wff is to try to state it in English. If you can translate it into a correct English sentence, then it is a wff.

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.

From well formed formulas to propositions

Subjects to be Learned

4. interpretation
5. satisfiable wff
6. invalid wff (unsatisfiable wff)
7. valid wff
8. equivalence of wffs

Contents Interpretation

A wff is, in general, not a proposition. For example, consider the wff **x P(x)**. Assume that **P(x)** means that **x** is non-negative (greater than or equal to 0). This wff is true if the universe is the set **{1, 3, 5}**, the set **{2, 4, 6}** or the set of natural numbers, for example, but it is not true if the universe is the set **{-1, 3, 5}**, or the set of integers, for example.
Further more the wff **x Q(x, y)**, where **Q(x, y)** means **x** is greater than **y**, for the universe **{1, 3, 5}** may be true or false depending on the value of **y**.

As one can see from these examples, the truth value of a wff is determined by the universe, specific predicates assigned to the predicate variables such as **P** and **Q**, and the values assigned to the **free** variables. **The specification of the universe and predicates, and an assignment of a value to each free variable in a wff** is called an **interpretation** for the wff.

For example, specifying the set **{1, 3, 5}** as the universe and assigning **0** to the variable **y**, for example, is an interpretation for the wff **x Q(x, y)**, where **Q(x, y)** means **x** is greater than **y**. **x Q(x, y)** with that interpretation reads, for example, "Every number in the set **{1, 3, 5}** is greater than **0**".

As can be seen from the above example, **a wff becomes a proposition when it is given an interpretation.**

There are, however, wffs which are always true or always false under any interpretation. Those and related concepts are discussed below.

Satisfiable, Unsatisfiable and Valid Wffs

A wff is said to be **satisfiable** if there exists an interpretation that makes it true, that is if there are a universe, specific predicates assigned to the predicate variables, and an assignment of values to the free variables that make the wff true.

For example, **x N(x)**, where **N(x)** means that **x** is non-negative, is satisfiable. For if the universe is the set of natural numbers, the assertion x N(x) is true, because all natural numbers are non-negative. Similarly x N(x) is also satisfiable.

However, **x [N(x) N(x)]** is not satisfiable because it can never be true. A wff is called **invalid** or **unsatisfiable**, if there is no interpretation that makes it true.

A wff is **valid** if it is true for every interpretation[*].
For example, the wff **x P(x) x P(x)** is valid for any predicate name **P ,** because **x P(x)** is the negation of **x P(x).**
However, the wff **x N(x)** is satisfiable but not valid.

Note that **a wff is not valid iff it is unsatisfiable** for a valid wff is equivalent to true. Hence its negation is false.

Equivalence

Two wffs **W_1** and **W_2** are **equivalent** if and only if **W_1 W_2** is valid, that is if and only if **W_1 W_2** is true for all interpretations.
For example **x P(x)** and **x P(x)** are equivalent for any predicate name **P .** So are **x [P(x) Q(x)]** and **[x P(x) x Q(x)]** for any predicate names **P** and **Q .**

Transcribing English to Predicate Logic wffs

Subjects to be Learned

Translating English sentences to wff

Contents

English sentences appearing in logical reasoning can be expressed as a wff. This makes the expressions compact and precise. It thus eliminates possibilities of misinterpretation of sentences. The use of symbolic logic also makes reasoning formal and mechanical, contributing to the simplification of the reasoning and making it less prone to errors.
Transcribing English sentences into wffs is sometimes a non-trivial task. In this course we are concerned with the transcription using given predicate symbols and the universe.
To transcribe a proposition stated in English using a given set of predicate symbols, first restate in English the proposition using the predicates, connectives, and quantifiers. Then replace the English phrases with the corresponding symbols.

Example: Given the sentence "Not every integer is even", the predicate "E(x)" meaning x is even, and that the universe is the set of integers,
first restate it as "It is not the case that every integer is even" or "It is not the case that for every object x in the universe, x is even."
Then "it is not the case" can be represented by the connective " ", "every object x in the universe" by " x", and "x is even" by E(x).
Thus altogether wff becomes   x E(x).

This given sentence can also be interpreted as "Some integers are not even". Then it can be restated as "For some object x in the universe, x is not integer". Then it becomes x E(x).

More examples: A few more sentences with corresponding wffs are given below. The universe is assumed to be the set of integers, E(x) represents x is even, and O(x), x is odd.

"Some integers are even and some are odd" can be translated as
x E(x) x O(x)

"No integer is even" can go to
x E(x)

"If an integer is not even, then it is odd" becomes
x [E(x) O(x)]

"2 is even" is
E(2)

More difficult translation: In these translations, properties and relationships are mentioned for certain type of elements in the universe such as relationships between integers in the universe of numbers rather than the universe of integers. In such a case the element type is specified as a precondition using if_then construct.

Examples: In the examples that follow the universe is the set of numbers including real numbers, and complex numbers. I(x), E(x) and O(x) representing "x is an integer", "x is even", and "x is odd", respectively.

"All integers are even" is transcribed as
x [I(x) E(x)]

It is first restated as "For every object in the universe (meaning for every numnber in this case) if it is integer, then it is even". Here we are interested in not any arbitrary object(number) but a specific type of objects, that is integers. But if we write x it means "for any object in the universe". So we must say "For any object, if it is integer .." to narrow it down to integers.

"Some integers are odd" can be restated as "There are objects that are integers and odd", which is expressed as
x [I(x) E(x)]

"A number is even only if it is integer" becomes
x [E(x) I(x)]

"Only integers are even" is equivalent to "If it is even, then it is integer". Thus it is translated to
x [E(x) I(x)]

Truth tables

Truth table is a powerful concept that constructs truth tables for its component statements. It is the most preferred tool in Boolean algebra. Truth table is a mathematical table specifically in connection with Boolean algebra, Boolean functions and propositional calculus. A mathematical table that displays all the possible truth values of a logical operation, is known as a truth table. It computes the functional values of logical expressions on each of their functional arguments.

A logical expression, such as ((p AND q) or r), has a truth value that depends on the truth values of the propositions (the p's, q's, and r's) that make up the expression. A truth table is an organized way of writing down the truth value of an expression, by exhaustively considering every possible set of truth values for the propositions that make up the expression.

In a complete truth table, every new column is made as the "and", "or", "implies", "if and only if", or "not" of earlier columns. Two expressions are called logically equivalent if their truth value is the same in EVERY condition, that is, if their truth value is the same in every row.

Truth Tables For Unary Operations:

The unary logical operations are those operations which contains only one logical variable.

Truth Table for Logical True

Logical true returns a true value for whatever every input. Its truth table is:

P	**T(P)**
T	T
F	T

Truth Table for Logical False

Logical false gives a false value for whatever the input is. Its truth table is as follows.

P	**F(P)**

T	F
F	F

Truth table for negation

Logical negation is a unary operation which typically returns opposite value of a proposition. If input is true, then output is false and vice versa. It is represented by NOT, or ¬p or Np or ~p. The truth table for NOT is given below.

P	**~P**
T	F
F	T

Truth Tables For Binary Operations

In logical mathematics, the binary operations are the logical operations that have two logical input variables. The truth tables of most important binary operations are given below.

Truth Table For Conjunction

Conjunction is a binary logical operation which results in a true value if both the input variables are true. This operator is represented by P AND Q or P ∧ Q or P . Q or P & Q, where P and Q are input variables. Its truth table is given below :

P	**Q**	**(P∧Q) And**
T	T	T
T	F	F
F	T	F
F	F	F

Truth Table For Disjunction:

Logical disjunction returns a true when at least input operands is true, i.e. either one of them or both are true.It is denoted by the symbols P OR Q, P ∨ Q or P + Q. Its truth table is shown below.

P	**Q**	**(P∨Q) or**
T	T	T
T	F	T

F	T	T
F	F	F

Truth Table for Implication:

Logical implication typically produces a value of false in singular case that the first input is true and the second is either false or true. It is associated with the condition, if P then Q and is denoted by P →→ Q or P ⇒⇒ Q. The truth table for implication is as follows:

P	**Q**	**(P → Q)**
T	T	T
T	F	F
F	T	T
F	F	T

We can learn another logic operations with truth tables. The propositional logic truth tables are standard one. So we can't change the propositional value.

Logical NAND

The NAND is a binary logical operation which similar to applying NOT on AND operation. In other words, NAND produces a true value if at least one of input variables is false. It is denoted by
P NAND Q or P | Q or P ↑ Q. Have a look at its truth table.

P	**Q**	**(P∧Q) And**
T	T	F
T	F	T
F	T	T
F	F	T

Logical NOR

The logical NOR is a logical operation which is obtained by applying a NOT operation to an OR operation. We can say that NOR results in a true value if both the input variables are false. It is represented by P NOR Q or P ↓ Q. Take a look at its truth table.

P	**Q**	**(P∨Q) or**
T	T	F
T	F	F
F	T	F
F	F	T

Tautology

A Tautology is a formula which is always true for every value of its propositional variables.

Example − Prove $[(A\rightarrow B)\wedge A]\rightarrow B[(A\rightarrow B)\wedge A]\rightarrow B$ is a tautology. The truth table is as follows −

A	B	$A \rightarrow B$	$(A \rightarrow B) \wedge A$	$[(A \rightarrow B) \wedge A] \rightarrow B$
True	True	True	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	True

As we can see every value of $[(A\rightarrow B)\wedge A]\rightarrow B[(A\rightarrow B)\wedge A]\rightarrow B$ is "True", it is a tautology.

Contradictions

A Contradiction is a formula which is always false for every value of its propositional variables.

Example − Prove $(A\vee B)\wedge[(\neg A)\wedge(\neg B)](A\vee B)\wedge[(\neg A)\wedge(\neg B)]$ is a contradiction The truth table is as follows –

A	B	$A \vee B$	$\neg A$	$\neg B$	$(\neg A) \wedge (\neg B)$	$(A \vee B) \wedge [(\neg A) \wedge (\neg B)]$
True	True	True	False	False	False	False
True	False	True	False	True	False	False
False	True	True	True	False	False	False
False	False	False	True	True	True	False

As we can see every value of $(A\lor B)\land[(\neg A)\land(\neg B)](A\lor B)\land[(\neg A)\land(\neg B)]$ is “False”, it is a contradiction.

Contingency

A Contingency is a formula which has both some true and some false values for every value of its propositional variables.

Example − Prove $(A\lor B)\land(\neg A)(A\lor B)\land(\neg A)$ a contingency The truth table is as follows −

A	**B**	**A ∨ B**	**¬ A**	**(A ∨ B) ∧ (¬ A)**
True	True	True	False	False
True	False	True	False	False
False	True	True	True	True
False	False	False	True	False

As we can see every value of $(A\lor B)\land(\neg A)(A\lor B)\land(\neg A)$ has both “True” and “False”, it is a contingency.

Implication

There is another fundamental type of connectives between statements, that of implication or more properly conditional statements. In English these are statements of the form ‘If p then q’ or ‘p implies q’. The following implications are some of the relationships between propositions that can be derived from the definitions(meaning) of connectives.
below corresponds to and it means that the implication always holds. That is it is a tautology.

These implications are used in logical reasoning. When the right hand side of these implications is substituted for the left hand side appearing in a proposition, the resulting proposition is implied by the original proposition, that is, one can deduce the new proposition from the original one.

Definition 1 The compound statement $p \Rightarrow q$ (‘If p then q’) is defined by the following truth table:

p	q	$p \Rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

In an implicative statement, p ⇒ q, we call p the premise and q the conclusion.

The first two rows make perfect sense from our linguistic understanding of ‘If p then q’, but the second two rows are more problematical. What are we to do if p is false?

Note that we must do something, otherwis⇒e p □ q would not be a well defined statement, since it would not be defined as either true or false on all the possible inputs.

We make the convention that p ⇒ q is always true if p is false.

The major reason for defining things this way is the following observation made by Bertrand Russell (1872 - 1970).

From a false premise it is possible to prove any conclusion.

Theorem 1: ‘If 0 = 1 then I am the Pope’.

Proof:

We assume 0 = 1 and show that ‘I am the Pope’ follows. 0 = 1, by adding 1 to both sides we conclude that 1 = 2. The Pope and I are two.
But 2 = 1, hence the Pope and I are one and the same! Q

The word any is very important here. It means literally anything, including things which are true. It is a common mistake in proofs to assume something along the waywhich is not true, thenproving the result is always possible. This is referred to as ‘arguing from false premises’.

Oneproblem is that in language we donotgenerallyuse implicativestatements in whichthepremise is false, or in which the premise and and conclusion are unrelated.

We usually assume that an implicative statement implies a connection, this is not so in logic. In logic we can make no such presumption, who would enforce ‘relatedness’? How would we define it?

When we wish to prove an implicative statement of the⇒form p q we assume that p is true and show that q follows under this assumption. Since, with our definition, if p is ⇒false p q is true irrespective of the truth value of q, we only have to consider the case when p is true.

Converse, Inverse and Contrapositive

Given an implicative statement, p ⇒ q, we can define the following statements:

1. The contrapositive is q ⇒ p.
2. The converse is q ⇒ p.
3. The inverse is p ⇒ q.

Theorem 2: p ⇒ q is logically equivalent to its contrapositive.

Proof:

p	q	p ⇒ q	¬q	¬p	¬q ⇒¬p
T	T	T	F	F	T

T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	T

Note that the converse is the contrapositive of the inverse.

A common method of proof is to in fact prove the contrapositive of an implicative statement. Thus, for example, if we wish to prove that

For all p prime, if p divides n^2 then p divides n. it is easier to prove the contrapositive:

For all p prime, if p does not divide n then p does not divide n^2.

Only if and Biconditionals

Definition If p and q are statements:

4. p only if q means ‘If not q then not p’ or equivalently ‘If p then q’.
i.e. p only if q means $\Rightarrow q$.

5. p if q means ‘If q then p’ i.e. $q \Rightarrow p$.

6. The biconditional ‘p if and only if q’ is true when p and q have the same truth value and false otherwise. It is denoted $p \Leftrightarrow q$.

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

Notes

i) ‘if and only if’ is often abbreviated to iff.

ii) In language it is common to say ‘If p then q’ when what we really mean is ‘p if and only if q’ - careful. See the remarks on page 26.

Theorem 3: $p \Leftrightarrow q \equiv (p \Rightarrow q) \wedge (q \Rightarrow p)$.

Proof:

p	q	$p \Rightarrow q$	$q \Rightarrow p$	$(p \Rightarrow q) \wedge (q \Rightarrow p)$	$p \Leftrightarrow q$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	T

Thus $p \Leftrightarrow q$ means that both $p \Rightarrow q$ and its converse are true.
When we wish to prove biconditional statements we must prove each direction separately. Thus we first prove p q (if p is tru⇒e then so is q) and then independently we prove q   p (if q is true then so is p⇒).

Necessary and Sufficient

Definition: Given two statements p and q
'p is a necessary condition for q' means $\sim p \Rightarrow \sim q$ or equivalently $q \Rightarrow p$.
'p is a sufficient condition for q' means $p \Rightarrow q$.

Notes

1. If p is a necessary condition for q this means that if q is true then so is p. However if p is true q may not be – there may be other things that must be true in order for q to happen.

2. If p is a sufficient condition for q then if p is true then q **must** happen as a consequence (all the conditions for q are fulfilled). However q may happen in some other way
– so q True but p False is a possibility

Duality Principle

Duality principle states that for any true statement, the dual statement obtained by interchanging unions into intersections (and vice versa) and interchanging Universal set into Null set (and vice versa) is also true. If dual of any statement is the statement itself, it is said **self-dual** statement.

Example − The dual of $(A\cap B)\cup C(A\cap B)\cup C$ is $(A\cup B)\cap C$

Normal Forms

Let $A(P_1, P_2, P_3, \ldots, P_n)$ be a statement formula where $P_1, P_2, P_3, \ldots, P_n$ are the atomic variables. If A has truth value T for all possible assignments of the truth values to the variables $P_1, P_2, P_3, \ldots, P_n$, 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.

Disjunctive normal form (DNF) is the normalization of a logical formula in Boolean mathematics. In other words, a logical formula is said to be in disjunctive normal form if it is a disjunction of conjunctions with every variable and its negation is present once in each conjunction. All disjunctive normal forms are non-unique, as all disjunctive normal forms for the same proposition are mutually equivalent. Disjunctive normal form is widely used in areas such as automated theorem proving.

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.
A clause that contains only ∨ is called a <u>disjunctive clause</u> and only ∧ is called a <u>conjunctive</u>

clause.

1. Negation is allowed, but only directly on variables.
2. $p\lor\neg q\lor r$: a disjunctive clause
3. $\neg p\land q\land\neg r$: a conjunctive clause
4. $\neg p\land\neg q\lor r$: neither

Putting conjunctive clauses together with $\lor$, it is called disjunctive normal form.

5. For example: $(p\land\neg q\land r)\lor(\neg q\land\neg r)$ is in disjunctive normal form.

If we put a bunch of disjunctive clauses together with $\land$, it is called conjunctive normal form.

6. For example: $(p\lor r)\land(\neg q\lor\neg r)\land q$ is in conjunctive normal form.

7. More examples:
1. $(p\land q\land\neg r\land s)\lor(\neg q\land s)\lor(p\land s)$ is in disjunctive normal form.
2. $(p\lor q\lor\neg r\lor s)\land(\neg q\lor s)\land\neg s$ is in conjunctive normal form.
3. $(p\lor r)\land(q\land(p\lor\neg q))$ is not in a normal form.
4. $\neg p\lor q\lor r$ and $\neg p\land q\land r$ are in both normal forms.

8. It turns out we can turn any proposition into either normal form.
1. We can use the definitions to get rid of →, ↔, and ⊕.
2. Use DeMorgan's laws to move any ¬ in past parens, so they sit on the variables.
3. Use double negation to get rid of any ¬¬ that showed up.
4. Use the distributive rules to move things in/out of parens as we need to.

9. For example, converting to conjunctive normal form:

$\neg ((\neg p \to \neg q) \land \neg r) \equiv \neg ((\neg\neg p \lor \neg q) \land \neg r)$ (Definition)
$\equiv \neg ((p \lor \neg q) \land \neg r)$ (Double Negation)
$\equiv \neg (p \lor \neg q) \lor \neg\neg r$ (De Morgan’s)
$\equiv \neg (p \lor \neg q) \lor r$ (Double Negation)
$\equiv (\neg p \land \neg\neg q) \lor r$ (De Morgan’s)
$\equiv (\neg p \land q) \lor r \equiv (\neg p \lor r) \land (q \lor r)$ (Distributive)

10. It was actually in disjunctive normal form in the second-last step.

Why would we want to convert to a normal form?

11. May be easier to prove equivalence: to show $A \equiv B$, convert both to normal form, and then re- write one proof backwards.
12. Maybe we simplify a lot: if we end up with $(p \lor \neg p \lor \cdots)$ terms, we know they are true.
13. Proving theorems about all propositions: only have to handle boolean expressions in a normal form and that covers every proposition.
14. Shows that we can use circuitry to calculate any boolean expression with two layers of logic gates.

Example 1:

Things that ARE in disjunctive normal form

15. &notC
16. W v T
17. W &and T
18. &notW v (R &and F &and G) v (A &and &notF)
19. &notA$_2$ v R v (B &and C) v (Y &and &notC)

Things that are NOT in disjunctive normal form

20. &not(A v C v G)
21. A &and (B v F)
22. &not(A → B*)*

Conjunctive Normal Forms (CNF)

Conjunctive normal form (CNF) is an approach to Boolean logic that expresses formulas as conjunctions of clauses with an AND or OR. Each clause connected by a conjunction, or AND, must be either a literal or contain a disjunction, or OR operator. CNF is useful for automated theorem proving.

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.

Let $A(P_1, P_2, P_3, \ldots, P_n)$ be a statement formula where $P_1, P_2, P_3, \ldots, P_n$ are the atomic variables. If A has truth value T for all possible assignments of the truth values to the variables $P_1, P_2, P_3, \ldots, P_n$, 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.

In conjunctive normal form, statements in Boolean logic are conjunctions of clauses with clauses of disjunctions. In other words, a statement is a series of ORs connected by ANDs.

For example **$(p\lor r)\land(\neg q\lor\neg r)\land q$** is in conjunctive normal form.

1. More examples:

1. $(p\land q\land\neg r\land s)\lor(\neg q\land s)\lor(p\land s)$ is in disjunctive normal form.
2. $(p\lor q\lor\neg r\lor s)\land(\neg q\lor s)\land\neg s$ is in conjunctive normal form.
3. $(p\lor r)\land(q\land(p\lor\neg q))$ is not in a normal form.
4. $\neg p\lor q\lor r$ and $\neg p\land q\land r$ are in both normal forms.

2. It turns out we can turn any proposition into either normal form.

1. We can use the definitions to get rid of →, ↔, and ⊕.
2. Use DeMorgan's laws to move any ¬ in past parens, so they sit on the variables.
3. Use double negation to get rid of any ¬¬ that showed up.
4. Use the distributive rules to move things in/out of parens as we need to.

3. For example, converting to conjunctive normal form:

$\neg((\neg p\to\neg q)\land\neg r)\equiv\neg((\neg\neg p\lor\neg q)\land\neg r)$ (Definition)

$\equiv\neg((p\lor\neg q)\land\neg r)$ (Double Negation)

$\equiv\neg(p\lor\neg q)\lor\neg\neg r$ (De Morgan’s)

$\equiv\neg(p\lor\neg q)\lor r$ (Double Negation)

$\equiv$ $(\neg p\land\neg\neg q)\lor r$ (De Morgan’s)

$\equiv$ $(\neg p\land q)\lor r\equiv(\neg p\lor r)\land(q\lor r)$ (Distributive)

4. It was actually in disjunctive normal form in the second-last step.

Why would we want to convert to a normal form?

5. May be easier to prove equivalence: to show A≡B, convert both to normal form, and then re- write one proof backwards.
6. Maybe we simplify a lot: if we end up with (p∨¬p∨⋯) terms, we know they are true.
7. Proving theorems about all propositions: only have to handle boolean expressions in a normal form and that covers every proposition.
8. Shows that we can use circuitry to calculate any boolean expression with two layers of logic gates.

Principal disjunctive normal forms

Let us assume A and B be two statement variables. All possible formulas by using conjunction are given as follows. The total number of formulas for two variables A and B are 2^2 formulas. They are A □ B, A □ □B, □A □ B and □A □ □ B.

These are called minterms or Boolean conjunctions of A and B. The minterms (2^n terms) are denoted by M_0, M_1, … ,M_{2^n-1}.

A formula equivalent to a given formula consisting of the disjunction of minterms only is called Principal disjunctive normal form (PDNF) of the given formula.

Principal Normal form

Let p and q be two statement variables, then p □ q, p □ □q, □p □ □q are called minterms of p and q, they are also called Boolean conjunctives of p and q. The number of minterms with n variables is 2^n . None of the minterms should contain both a variable and it's negation. □p □ □ is not minterm. The dual of minterm is called a maxterm.

Principal conjunctive Normal form (PCNF)

A statement formula which consists of a conjunction of maxterms only is called the principal conjunctive normal form.

The duals of minterms are called maxterms. For a given number of variables the maxterm consists of disjunctions in which each variable or its negation, but not both, appears only once.

Therefore for a given formula, an equivalent formula consisting of conjunctions of maxterms only is known as **its principal conjunctive normal form**. This is also called the **product of sums canonical form**.

Predicate Logic

Predicate Logic – Definition

A predicate is an expression of one or more variables defined on some specific domain. A predicate with variables can be made a proposition by either assigning a value to the variable or by quantifying the variable. The following are some examples of predicates −

1. Let E(x, y) denote "x = y"
2. Let X(a, b, c) denote "a + b + c = 0"
3. Let M(x, y) denote "x is married to y"

Predicate calculus

we'll symbolize the word "every" (or equivalently "all", "any", etc.) with an upside down 'A': ∀. And we'll use a backwards-E for "some" and it's synonyms: ∃ .

Symbolizing "All" and "Some"

Example1:(∀x)W(x): read this as "every x is such that it is valid" or as "all x make 'W(x)' true" or “every x, every member of the universe of discourse, is a valid argument:”

Example2:(∃ x)W(x): read this "there is an x such that x is valid" or as "there is an x making 'Wx' true.", or “some x, some member of the universe of discourse, is a valid argument.”

Symbolizing "No" meaning "None"

We have two logically equivalent ways to think about our "no" statement: "**No** cats are reptiles". This statement can be understood to be it's not true that some cat is a reptile ('~(∃x)R(x)') or it can be equivalently rendered as all cats are non- reptiles ('(∀x)~R(x)'). (To say that all cats are non-reptiles is to say that every cat fails to be a reptile.)

We will see these two ways of expressing "no" again and again, so let's put it in a box:

QN: "no cats are reptiles" can be symbolized '~(∃x)R(x)' or, equivalently, '(∀x)~R(x)'.

We will return to the PL symbolizations later. For now keep in mind that "**no**", "**some**", and "**all**" have this complicated relation just called '**QN**'.

Categorical Statements

These three examples are of categorical statements. They relate categories. For example, "All dogs are mammals" takes the subject term "dogs" and relates it to the predicate term "mammals".

1. A categorical statement of this form...

All S are P

we call a "universal categorical statement". We'll call the form "universal".

2. A statement of this "existential form":

Some S are P: is an existential categorical statement.

3. A statement of the last or "negative form":

No S are P is a negative categorical statement.

4. **Obversion** has two forms to remember:

"All S are P" is logically equivalent to "No S are non-P". "No S are P" is logically equivalent to "All S are non-P". **e)Conversion has two forms for equivalence**:

"Some S are P" is logically equivalent to "Some P are S". "No S are P" is logically equivalent to "No P are S".

f)Contraposition:

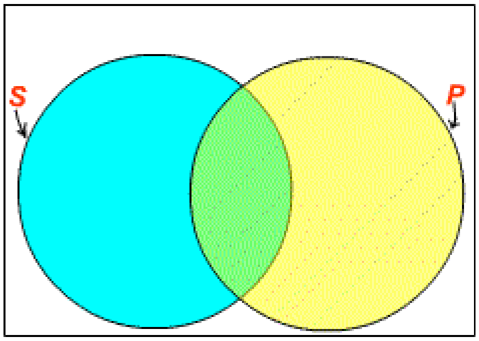
"All S are P" is logically equivalent to "All non-P are non-S".

Venn diagrams

Venn diagrams are the easy way to go to understand the meaning of categorical sentences. Think of everything in the universe (of discourse) as being contained within the box just below.

Everything that falls into the category S is located in a circle (the left one). Everything falling into category P is located in the other circle (the right one).

If something is **both** S and P, then it's located in the area of overlap of the two circles: the area in green. Anything which is **neither** S nor P, is outside both circles and in the white area.

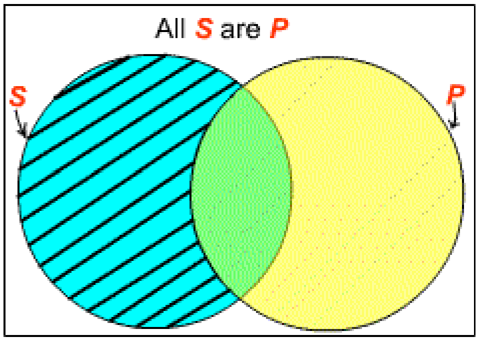


Now, let's use these diagrams for understanding categorical statements.

We have three "official" categorical forms.

1. **universal categorical form**:

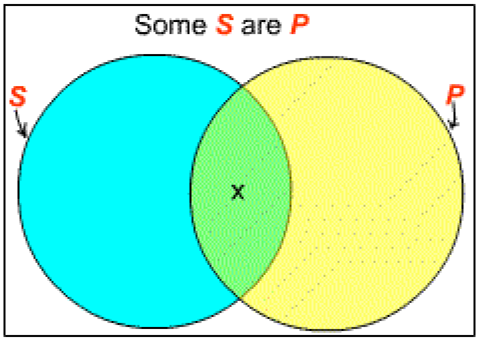
All S are P



The lines mean that the area inside S but outside P is empty. So, any S is P. Thus we have a diagram of our universal form.

2. **existential categorical form**:

Some S are P



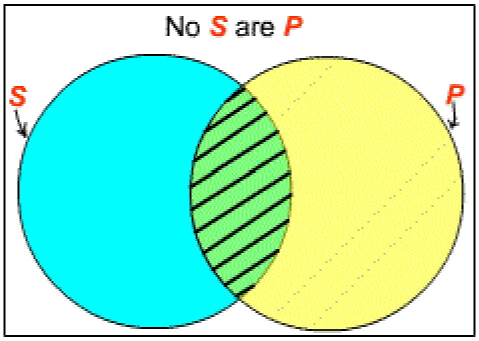
The 'x' is an arbitrary stand-in for an object. The diagram simply says that there is something in the green area of overlap between S and P.

3. **negative categorical form**:

No S are P

This diagram represents there being nothing in the overlap. So, there is nothing that is both S and P.

Symbolization into strict categorical form



4. When we first wrote up a Venn Diagram for the universal form, we put it this way:

This diagram specifies the meaning of "All S are P". The idea is that everything that is in P is also in S. I.e., there is nothing in S that is not also in P. This last just means:

No S are non-P.

There is nothing in the overlap of S and non-P.

Second, a little thought shows that we've also justified the logical equivalence for this similar transposition:

"No S are P" is logically equivalent to "All S are non-P".

Variables and quantifiers

A predicate with variables is not a proposition. For example, the statement x > 1 with variable x over the universe of real numbers is neither true nor false since we don't know what x is. It can be true or false depending on the value of x.

For x > 1 to be a proposition either we substitute a specific number for x or change it to something like "There is a number x for which x > 1 holds", or "For every number x, x > 1 holds".

More generally, a predicate with variables (called an **atomic formula**) can be made a **proposition** by applying one of the following two operations to each of its variables:

1. assign a value to the variable
2. quantify the variable using a **quantifier** (see below).

For example, x > 1 becomes 3 > 1 if 3 is assigned to x, and it becomes a true statement, hence a proposition.

In general, a quantification is performed on formulas of predicate logic (called wff), such as **x > 1** or **P(x)**, by using quantifiers on variables. There are two types of quantifiers: universal quantifier and existential quantifier.

The **universal quantifier** turns, for example, the statement x > 1 to "for every object x in the universe, x > 1", which is expressed as " x x > 1". This new statement is true or false in the universe of discourse. Hence it is a proposition once the universe is specified.

Similarly the **existential quantifier** turns, for example, the statement x > 1 to "for some object x in the universe, x > 1", which is expressed as " x x > 1." Again, it is true or false in the universe of discourse, and hence it is a proposition once the universe is specified.

Universe of Discourse

The universe of discourse, also called **universe**, is the set of objects of interest. The propositions in the predicate logic are statements on objects of a universe. The universe is thus the domain of the (individual) variables. It can be the set of real numbers, the set of integers, the set of all cars on a parking lot, the set of all students in a classroom etc. The universe is often left implicit in practice. But it should be obvious from the context.

The Universal Quantifier

The expression: x P(x), denotes the **universal quantification** of the atomic formula P(x). Translated into the English language, the expression is understood as: "For all x, P(x) holds", "for each x, P(x)holds" or "for every x, P(x) holds". is called the **universal quantifier**, and x means all the objects x in the universe. If this is followed by P(x) then the meaning is that P(x) is true for every object x in the universe. For example, "All cars have wheels" could be transformed into the propositional form, x P(x), where:

1. P(x) is the predicate denoting: **x has wheels**, and
2. the universe of discourse is only populated by cars.

Universal Quantifier and Connective AND

If all the elements in the universe of discourse can be listed then the universal quantification x P(x) is equivalent to the conjunction: **$P(x_1)$)** **$P(x_2)$** **$P(x_3)$** **...** **$P(x_n)$** .

For example, in the above example of x P(x), if we knew that there were **only** 4 cars in our universe of discourse (c1, c2, c3 and c4) then we could also translate the statement as: **P(c1)** **P(c2)** **P(c3)**

P(c4)

The Existential Quantifier

The expression: xP(x), denotes the **existential quantification** of P(x). Translated into the English language, the expression could also be understood as: "There exists an x such that P(x)" or "There is at least one x such that P(x)" is called the **existential quantifier**, and x means at least one object x in the universe. If this is followed by P(x) then the meaning is that P(x) is true for at least one object x of the universe. For example, "Someone loves you" could be transformed into the propositional form, x P(x), where:

3. P(x) is the predicate meaning: **x loves you**,
4. The universe of discourse contains (but is not limited to) all living creatures.

Existential Quantifier and Connective OR

If all the elements in the universe of discourse can be listed, then the existential quantification xP(x) is equivalent to the disjunction: **$P(x_1)$** **$P(x_2)$** **$P(x_3)$** **...** **$P(x_n)$**.

For example, in the above example of x P(x), if we knew that there were **only** 5 living creatures in our universe of discourse (say: me, he, she, rex and fluff), then we could also write the statement

as: **P(me)** **P(he)** **P(she)** **P(rex)** **P(fluff)**

An appearance of a variable in a wff is said to be **bound** if either a specific value is assigned to it or it is quantified. If an appearance of a variable is not bound, it is called **free**. The extent of the application(effect) of a quantifier, called the **scope** of the quantifier, is indicated by square brackets **[ ]**. If there are no square brackets, then the scope is understood to be the smallest **wff** following the quantification.
For example, in **x P(x, y)**, the variable **x** is bound while **y** is free. In **x [** **y P(x, y)** **Q(x, y)]**

, **x** and the **y** in **P(x, y)** are bound, while **y** in **Q(x, y)** is free, because the scope of **y** is **P(x, y)**. The scope of **x** is **[y P(x, y) Q(x, y)] .**

How to read quantified formulas

When reading quantified formulas in English, **read them from left to right.** **x** can be read as "for every object **x** in the universe the following holds" and **x** can be read as "there exists an object **x** in the universe which satisfies the following" or "for some object **x** in the universe the following holds". Those do not necessarily give us good English expressions. But they are where we can start. Get the correct reading first then polish your English without changing the truth values.

For example, let the universe be the set of airplanes and let **F(x, y)** denote "**x** flies faster than **y**". Then
x y F(x, y) can be translated initially as "For every airplane **x** the following holds: **x** is faster than every (any) airplane **y**". In simpler English it means "Every airplane is faster than every airplane (including itself !)".
x y F(x, y) can be read initially as "For every airplane **x** the following holds: for some airplane **y**, **x** is faster than **y**". In simpler English it means "Every airplane is faster than some airplane".
x y F(x, y) represents "There exist an airplane **x** which satisfies the following: (or such that) for every airplane **y**, **x** is faster than **y**". In simpler English it says "There is an airplane which is faster than every airplane" or "Some airplane is faster than every airplane".
x y F(x, y) reads "For some airplane **x** there exists an airplane **y** such that **x** is faster than **y**", which means "Some airplane is faster than some airplane".

Order of Application of Quantifiers

When more than one variables are quantified in a wff such as **y x P(x, y)**, they are applied from the inside, that is, the one closest to the atomic formula is applied first. Thus **y x P(x,**
y) reads **y [x P(x, y)] ,** and we say "there exists an **y** such that for every **x**, **P(x, y)** holds" or "for some **y**, **P(x, y)** holds for every **x**".

The positions of the same type of quantifiers can be switched without affecting the truth value as long as there are no quantifiers of the other type between the ones to be interchanged.
For example **x y z P(x, y , z)** is equivalent to **y x z P(x, y , z)**, **z y x P(x, y , z)**, etc. It is the same for the universal quantifier.

However, the positions of different types of quantifiers can **not** be switched.
For example **x y P(x, y)** is **not** equivalent to **y x P(x, y).** For let **P(x, y)** represent **x < y** for the set of numbers as the universe, for example. Then **x y P(x, y)** reads "for every number **x**, there is a number **y** that is greater than **x**", which is true, while **yx P(x, y)** reads "there is a number that is greater than every (any) number", which is not true.

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

Implications

I1 P∧Q =>P } Simplification I2 PQ∧ =>Q I3 P=>PVQ } Addition I4 Q =>PVQ
I5 □P => P→ Q I6 Q => P→ Q I□ □(P→Q) =>P I8 □(P → Q) => 7Q I9 P, Q => P ∧ Q
I10 □P, PVQ => Q (disjunctive syllogism) I11 P, P→ Q => Q (modus ponens)
I12 □Q, P → Q => □P (modus tollens)
I13 P → Q, Q → R => P → R (hypothetical syllogism) I14 P V Q, P → Q, Q → R => R (dilemma)

Equivalences

E1 □ □P <=>P

E2 P ∧ Q <=> Q ∧ P } Commutative laws

E3 P V Q <=> Q V P

E4 (P ∧ Q) ∧ R <=> P ∧ (Q ∧ R) } Associative laws

E5 (P V Q) V R <=> PV (Q V R)

E6 P ∧ (Q V R) <=> (P ∧ Q) V (P ∧ R) } Distributive laws

E7 P V (Q ∧ R) <=> (P V Q) ∧ (PVR)

E8 □(P ∧ Q) <=> □P V□Q

E9 □(P V Q) <=> □P ∧ □Q } De Morgan’s laws

E10 P V P <=> P E11 P ∧ P <=> P
E12 R V (P ∧ □P) <=>R
E13 R ∧ (P V □P) <=>R
E14 R V (P V □P) <=>T
E15 R ∧ (P ∧ □P) <=>F
E16 P → Q <=> □P V Q
E17 □ (P→ Q) <=> P ∧ □Q
E18 P → Q <=> □Q → □P
E19 P → (Q → R) <=> (P ∧ Q) → R
E20 □(PD Q) <=> P D □Q
E21 PDQ <=> (P → Q) ∧ (Q → P)
E22 (PDQ) <=> (P ∧ Q) V (□P ∧ □Q)

Example 1.Show that R is logically derived from P → Q, Q → R, and P

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

Example 3. Show that □Q, P→ Q => □P

Solution .	{1}	(1)	P → Q	Rule P
	{1}	(2)	□P → □Q	T, and E 18
	{3}	(3)	□Q	P
	{1, 3}	(4)	□P	T, (2), (3), and I11 .

Example 4 .Prove that R ^ (P V Q) is a valid conclusion from the premises PVQ , Q → R, P → M and □M.

Solution . {1}	(1)	P → M	P
{2}	(2)	□M	P
{1, 2}	(3)	□P	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.

Note. 1. Rule CP follows from the equivalence E10 which states that (P ∧ R) → S óP → (R → S).

- ➢ 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 5 .Show that R → S can be derived from the premises P → (Q → S), □R V P , and Q.

Solution.	{1}	(1)	□R 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 6.Show that P → S can be derived from the premises, □P V Q,
□Q V R, and R → S .

Solution.

{1}	(1)	□P V Q	P
{2}	(2)	P	P, assumed premise
{1, 2}	(3)	Q	T, (1), (2) and I11
{4}	(4)	□Q 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

Example 7. ” If there was a ball game , then traveling was difficult. If they arrived on time, then traveling was not difficult. They arrived on time. Therefore, there was no ball game”. Show that these statements

constitute a valid argument.

Solution. Let P: There was a ball game

Q: Traveling was difficult. R: They arrived on time.

Given premises are: P → Q, R → □Q and R conclusion is: □P

{1}	(1)	P → Q	P
{2}	(2)	R → □Q	P
{3}	(3)	R	P
{2, 3}	(4)	□Q	T, (2), (3), and I11
{1, 2, 3}	(5)	□P	T, (2), (4) and I12

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 ∧ □R

Where R is any formula. Note that R ∧ □R 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 □C as an additional premise. If the new set of premises is inconsistent, so that they imply a contradiction, then the assumption that □C 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. E**xample 8:** Show that □(P

We introduce — — (P∧ Q) as an additional premise and show that this additional

∧ Q) follows from □P∧ □Q.

Here (6) P∧ □P is a contradiction. Thus {1, 4} viz. □ □(P∧ Q) and □P∧ □Q leads to a contradiction P ∧ □P.

Example 9: 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 ofbooks. S: Jack is uneducated.

The premises are P→ Q, Q → S, R→ □S 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→ □S	P
{4}	(5)	S → □R	T, (4), and E18
{1, 2, 4}	(6)	P→□R	T, (3), (5) and I13
{1, 2, 4}	(7)	□PV □R	T, (6) and E16
{1, 2, 4}	(8)	□(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**
$p\to(p\lor q)$ p $\therefore \overline{p\lor q}$		Addition Simplification
$p\land q$	$(p\land q)\to p$	
Conjunction p q $\therefore \overline{p\land q}$	$((p)\land(q))\to(p\land q)$	
Modus ponens p $p\to q$ $\therefore \overline{q}$	$((p\land(p\to q))\to q$	
Modus tollens $\neg q$ $p\to q$ $\therefore \overline{\neg p}$	$((\neg q\land(p\to q))\to\neg p$	
Hypothetical syllogism $p\to q$ $q\to r$ $\therefore \overline{p\to r}$	$((p\to q)\land(q\to r))\to(p\to r)$	
Disjunctive syllogism $p\lor q$ $\neg p$ $\therefore \overline{q}$	$((p\lor q)\land\neg p)\to q$	
Resolution $p\lor q$ $\neg p\lor r$ $\therefore \overline{q\lor r}$	$((p\lor q)\land(\neg p\lor r))\to(q\lor r)$	

$\neg p$

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:

$$\begin{array}{r l} & p \rightarrow q \\ & q \rightarrow r \\ \therefore & \overline{p \rightarrow r} \end{array}$$

Example 2

Let us consider a more complex set of assumptions: "It is not sunny today and it is colder than yesterday". "We will go swimming only if it is sunny", "If we do not go swimming, then we will have a barbecue", and "If we will have a barbecue, then we will be home by sunset" lead to the conclusion

"We will be home before sunset." Proof by rules of inference: Let p be the proposition "It is sunny this today", q the proposition "It is colder than yesterday", r the proposition "We will go swimming", s the proposition "We will have a barbecue", and t the proposition "We will be home by sunset". Then the hypotheses become $\neg p \wedge q, r \rightarrow p, \neg r \rightarrow s$ and $s \rightarrow t$. Using our intuition we conjecture that the conclusion might be t. Using the Rules of Inference table we can proof the conjecture easily:

Step	**Reason**
1. $\neg p \wedge q$	Hypothesis
2.	Simplification using Step 1

3.	Hypothesis
4.	Modus tollens using Step 2 and 3
5.	Hypothesis
6. s	Modus ponens using Step 4 and 5
7.	Hypothesis
8. t	Modus ponens using Step 6 and 7

Proof by 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:

i. Assume that a given proposition is untrue.

ii. 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.)

For Example:

Let’s 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.

More Examples:

1. Irrationality of the square root of 2

A classic proof by contradiction from mathematics is the proof that the square root of 2 is irrational. If it were rational, it could be expressed as a fraction a/b in lowest terms, where a and b are integers, at least one of which is odd. But if $a/b = \sqrt{2}$, then $a^2 = 2b^2$. Therefore, a^2 must be even. Because the square of an odd number is odd, that in turn implies that a is even. This means that b must be odd because a/b is in lowest terms.

On the other hand, if a is even, then a^2 is a multiple of 4. If a^2 is a multiple of 4 and $a^2 = 2b^2$, then $2b^2$ is a multiple of 4, and therefore b^2 is even, and so is b.

So b is odd and even, a contradiction. Therefore the initial assumption—that $\sqrt{2}$ can be expressed as a fraction—must be false.

2. The length of the hypotenuse

The method of proof by contradiction has also been used to show that for any non-degenerate right triangle, the length of the hypotenuse is less than the sum of the lengths of the two remaining sides.[4] The proof relies on the Pythagorean theorem. Letting c be the length of the hypotenuse and a and b the lengths of the legs, the claim is that $a + b > c$.

The claim is negated to assume that $a + b \leq c$. Squaring both sides results in $(a + b)^2 \leq c^2$ or, equivalently, $a^2 + 2ab + b^2 \leq c^2$. A triangle is non-degenerate if each edge has positive length, so it may be assumed that a and b are greater than 0. Therefore, $a^2 + b^2 < a^2 + 2ab + b^2 \leq c^2$. The transitive relation may be reduced to $a^2 + b^2 < c^2$. It is known from the Pythagorean theorem that $a^2 + b^2 = c^2$. This results in a contradiction since strict inequality and equality are mutually exclusive. The latter was a result of the Pythagorean theorem and the former the assumption that $a + b \leq c$. The contradiction means that it is impossible for both to be true and it is known that the Pythagorean theorem holds. It follows that the assumption that $a + b \leq c$ must be false and hence $a + b > c$, proving the claim.

3. No least positive rational number

Consider the proposition, P: "there is no smallest rational number greater than 0". In a proof by contradiction, we start by assuming the opposite, ¬P: that there is a smallest rational number, say, r.

Now r/2 is a rational number greater than 0 and smaller than r. But that contradicts our initial assumption, ¬P, that r was the smallest rational number. (In the above symbolic argument, "r is the smallest rational number" would be Q and "r/2 is a rational number smaller than r" would be ¬Q.) So we can conclude that the original proposition, P, must be true — "there is no smallest rational number greater than 0".

4. Prove the following statement by Contradiction:

The difference of any rational number and any irrational number is irrational.

Proof:

Suppose not. [We take the negation of the theorem and suppose it to be true.] Suppose ∃ a rational number x and an irrational number y such that (x − y) is rational. [We must derive a contradiction.] By definition of rational, we have x = a/b for some integers a and b with b ≠ 0. and x − y = c/d for some integers c and d with d ≠ 0.
By substitution, we have

x − y = c/d a/b − y = c/d

y = a/b − c/d

= (ad − bc)/bd

But (ad − bc) are integers [because a, b, c, d are all integers and products and differences of integers are integers], and bd ≠ 0 [by zero product property]. Therefore, by definition of rational, y is rational. This contradicts the supposition that y is rational. [Hence, the supposition is false and the theorem is true.]

And this completes the proof.

5. Prove the following statement by contradiction: The negative of any irrational number is irrational.

First, translate given statement from informal to formal language:

∀ real numbers x, if x is irrational, then −x is irrational.

Proof:

Suppose not. [we take the negation of the given statement and suppose it to be true.] Assume, to the contrary, that

$\exists$ irrational number x such that −x is rational.

[We must deduce the contradiction.] By definition of rational, we have

$-x = a/b$ for some integers a and b with $b \neq 0$.

Multiply both sides by −1, gives $x = -(a/b)$

$= -a/b$

But −a and b are integers [since a and b are integers] and $b \neq 0$ [by zero product property.] Thus, x is a ratio of the two integers −a and b with $b \neq 0$. Hence, by definition of ration x is rational, which is a contradiction. [This contradiction shows that the supposition is false and so the given statement is true.]

This completes the proof.

6. Prove the following statement by contradiction: For all integers n, if n^2 is odd, then n is odd.

Proof:

Suppose not. [We take the negation of the given statement and suppose it to be true.] Assume, to the contrary, that $\exists$ an integer n such that n^2 is odd and n is even. [We must deduce the contradiction.]
By definition of even, we have

$n = 2k$ for some integer k.

So, by substitution we have

$n \,.\, n = (2k) \,.\, (2k)$

$= 2\,(2.k.k)$

Now (2.k.k) is an integer because products of integers are integer; and 2 and k are integers. Hence,

$n \,.\, n = 2 \,.$ (some integer)

or $n^2 = 2.$ (some integer)

and so by definition of n^2 even, is even.

So the conclusion is since n is even, n^2, which is the product of n with itself, is also even. This contradicts the supposition that n^2 is odd. [Hence, the supposition is false and the proposition is true.]

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.

UNIT -II

1. **Relations**

    1.1. Properties of binary relations

    1.2. Equivalence, transitive closure

    1.3. Compatibility and partial ordering relations

    1.4. Lattices

    1.5. Hasse diagram

2. **Functions**

    2.1. Inverse function

    2.2. Composition of functions

    2.3. Recursive functions.

3. **Lattices**

    3.1. Lattices as partially ordered sets

    3.2. Definition and examples

    3.3. Properties of lattices

    3.4. Lattices as algebraic system

    3.5. Sub lattices

    3.6. Direct product and homomorphism

    3.7. Some special lattices.

Relations

A **Function** assigns to each element of a set, exactly one element of a related set. Functions find their application in various fields like representation of the computational complexity of algorithms, counting objects, study of sequences and strings, to name a few. The third and final chapter of this part highlights the important aspects of functions.

Function - Definition

A function or mapping (Defined as $f:X\to Yf:X\to Y$) is a relationship from elements of one set X to elements of another set Y (X and Y are non-empty sets). X is called Domain and Y is called Codomain of function ‘f’.

Function ‘f’ is a relation on X and Y such that for each $x\in Xx\in X$, there exists a unique $y\in Yy\in Y$ such that $(x,y)\in R(x,y)\in R$. ‘x’ is called pre-image and ‘y’ is called image of function f.

A function can be one to one or many to one but not one to many.

Injective / One-to-one function

A function $f:A\to Bf:A\to B$ is injective or one-to-one function if for every $b\in Bb\in B$, there exists at most one $a\in Aa\in A$ such that $f(s)=tf(s)=t$.

This means a function **f** is injective if $a1\neq a2a1\neq a2$ implies $f(a1)\neq f(a2)f(a1)\neq f(a2)$.

Example

- $f:N\to N,f(x)=5xf:N\to N,f(x)=5x$ is injective.
- $f:N\to N,f(x)=x2f:N\to N,f(x)=x2$ is injective.
- $f:R\to R,f(x)=x2f:R\to R,f(x)=x2$ is not injective as $(-x)2=x2(-x)2=x2$

Surjective / Onto function

A function $f:A\to Bf:A\to B$ is surjective (onto) if the image of f equals its range. Equivalently, for every $b\in Bb\in B$, there exists some $a\in Aa\in A$such that $f(a)=bf(a)=b$. This means that for any y in B, there exists some x in A such that $y=f(x)y=f(x)$.

Example

- $f:N\to N,f(x)=x+2f:N\to N,f(x)=x+2$ is surjective.
- $f:R\to R,f(x)=x2f:R\to R,f(x)=x2$ is not surjective since we cannot find a real number whose square is negative.

Bijective / One-to-one Correspondent

A function $f:A\to Bf:A\to B$ is bijective or one-to-one correspondent if and only if **f** is both injective and surjective.

Problem

Prove that a function $f:R\to Rf:R\to R$ defined by $f(x)=2x–3f(x)=2x–3$ is a bijective function.

Explanation − We have to prove this function is both injective and surjective.

If $f(x1)=f(x2)f(x1)=f(x2)$, then $2x1–3=2x2–32x1–3=2x2–3$ and it implies that $x1=x2x1=x2$.
Hence, f is **injective**.

Here, $2x–3=y2x–3=y$
So, $x=(y+5)/3x=(y+5)/3$ which belongs to R and $f(x)=yf(x)=y$.
Hence, f is **surjective**.

Since **f** is both **surjective** and **injective**, we can say **f** is**bijective**.

Inverse of a Function

The **inverse** of a one-to-one corresponding function $f:A\to Bf:A\to B$, is the function $g:B\to Ag:B\to A$, holding the following property −
$f(x)=y\Leftrightarrow g(y)=xf(x)=y\Leftrightarrow g(y)=x$
The function f is called **invertible**, if its inverse function g exists.

Example

- A Function $f:Z\to Z,f(x)=x+5f:Z\to Z,f(x)=x+5$, is invertible since it has the inverse function $g:Z\to Z,g(x)=x−5g:Z\to Z,g(x)=x−5$.
- A Function $f:Z\to Z,f(x)=x2f:Z\to Z,f(x)=x2$ is not invertiable since this is not one-to-one as $(−x)2=x2(−x)2=x2$.

Composition of Functions

Two functions $f:A\to Bf:A\to B$ and $g:B\to Cg:B\to C$ can be composed to give a composition gofgof. This is a function from A to C defined by $(gof)(x)=g(f(x))(gof)(x)=g(f(x))$

Example

Let $f(x)=x+2f(x)=x+2$ and $g(x)=2x+1g(x)=2x+1$, find $(fog)(x)(fog)(x)$ and $(gof)(x)(gof)(x)$.

Solution

$(fog)(x)=f(g(x))=f(2x+1)=2x+1+2=2x+3(fog)(x)=f(g(x))=f(2x+1)=2x+1+2=2x+3$
$(gof)(x)=g(f(x))=g(x+2)=2(x+2)+1=2x+5(gof)(x)=g(f(x))=g(x+2)=2(x+2)+1=2x+5$
Hence, $(fog)(x)\neq(gof)(x)(fog)(x)\neq(gof)(x)$

Some Facts about Composition

- If f and g are one-to-one then the function (gof)(gof) is also one-to-one.
- If f and g are onto then the function (gof)(gof) is also onto.
- Composition always holds associative property but does not hold commutative property.

Inverse function:

A function f:A->B is said to be inverse if their exists a function g:B->A such that gof=I_A and fog=I_B where I_A is identity function on A and I_B is the identity function on B then g is called inverse of f and we write

$g=f^{-1}$

Notation

Note: f^{-1} (x) $\neq 1/$ f(x) .

It is very important not to confuse function notation with negative exponents.

Does the Function have an Inverse?

Only One-to-One Functions have an inverse function.

Examples – Now let’s look at a few examples to help demonstrate what a one-to-one function is.

Example 1:

Determine if the function f = {(7, 3), (8, –5), (–2, 11), (–6, 4)} is a one-to-one function.

The function f is a one-to-one functionbecause each of the y-values in the ordered pairs is unique ; none of the y-values appear more than once. Since the function f is a one-to-one function, the function f must have an inverse.

Example 2:

Determine if the function h = {(–3, 8), (–11, –9), (5, 4), (6, –9)} is a one-to-one function.

The function h is not a one-to-one function because the y-value of –9 is not unique; the y-value of –9 appears more than once. Since the function h is not aone-to-one function, the function h does not have an inverse.

Remember that only one-to-one function have an inverse

How to solve the inverse of the function

Here are the steps required to find the inverse function:

Step 1: Determine if the function has an inverse. Is the function a one-to-one function? If the function is a one-to-one function, go to step 2. If the function is not a one-to-one function, then say that the function does not have an inverse and stop.

Step 2: Change f(x) to y.

Step 3: Switch x and y.

Step 4: Solve for y.

Step 5: Change y back to f (x).

Example 1 Given $f\left(x\right)=3x-2$ find $f^{-1}\left(x\right)$.

Solution

Now, we already know what the inverse to this function is as we’ve already done some work with it. However, it would be nice to actually start with this since we know what we should get. This will work as a nice verification of the process.

So, let’s get started. We’ll first replace $f\left(x\right)$ with y.

$$y=3x-2$$

Next, replace all x’s with y and all y’s with x.

$$x = 3y - 2$$

Now, solve for y.

$$\begin{aligned} x+2 &= 3y \\ \frac{1}{3}(x+2) &= y \\ \frac{x}{3}+\frac{2}{3} &= y \end{aligned}$$

Finally replace y with $f^{-1}(x)$.

$$f^{-1}(x) = \frac{x}{3}+\frac{2}{3}$$

Now, we need to verify the results. We already took care of this in the previous section, however, we really should follow the process so we'll do that here. It doesn't matter which of the two that we check we just need to check one of them. This time we'll check that $\left(f \circ f^{-1}\right)(x) = x$ is true.

Example 2 Given $g(x) = \sqrt{x-3}$ find $g^{-1}(x)$.

Solution

The fact that we're using $g(x)$ instead of $f(x)$ doesn't change how the process works. Here are the first few steps.

$$\begin{aligned} y &= \sqrt{x-3} \\ x &= \sqrt{y-3} \end{aligned}$$

Now, to solve for y we will need to first square both sides and then proceed as normal.

$$\begin{aligned} x &= \sqrt{y-3} \\ x^2 &= y-3 \\ x^2+3 &= y \end{aligned}$$

This inverse is then,

$$g^{-1}(x) = x^2+3$$

Finally let’s verify and this time we’ll use the other one just so we can say that we’ve gotten both down somewhere in an example.

$$\begin{aligned}\left(g^{-1} \circ g\right)(x) &= g^{-1}\left[g(x)\right]\\ &= g^{-1}\left(\sqrt{x-3}\right)\\ &= \left(\sqrt{x-3}\right)^2 + 3\\ &= x - 3 + 3\\ &= x\end{aligned}$$

So, we did the work correctly and we do indeed have the inverse.

Composition of functions :

Composition of functions is when one function is inside of another function.
The notation used for the composition of functions looks like this,
(f og)(x)= f(g(x)), notice that in the case the function g is inside of the function f.

How Do You Find the Composition of Two Functions

Here are the steps we can use to find the composition of two functions:
Step 1:
Rewrite the composition in a different form. For example, the composition
(f o g)(x) needs to rewritten as f(g(x)).
Step 2:
Replace each occurrence of x found in the outside function with the inside function. For example, in the composition of(f og)(x) = f(g(x)), we need to replace each x found in f(x), the outside function, with g(x), the inside function.
Step 3:
Simplify the answer.

Example 1:
If f(x) = –4x + 9 and g(x) = 2x– 7, find(f og)(x).

Solution
(f og)(x) =f(g(x))
=-4(2x- 7) +9
=-8x+28+7
=-8x+37
Thus,
(f og)(x) = –8x+ 37.

Example 2:

If $f(x) = –4x + 9$ and $g(x) = 2x– 7$, find$(go\ f)(x)$.

Solution

$(go\ f)(x)= g(f(x))$

$=2(-4x+9)-7$

$=-8x+18-7$

$=-8x+11$

Thus,

$(g\ of)(x) = –8x+11$

Example 3:

If $h(x) = 3x– 5$ and $g(x) = 2x^2 – 7x$, find $(goh)(x)$.

Solution

$(g\ oh)(x) = g(h(x))$

$=2(3x- 5)^2- 7(3x- 5)$

$= 2(9x^2-30x+ 25) -7(3x-5)$

$=18x^2- 60x +50-21x+ 35$

$=18x^2 -81x +85$

thus,

$(g\ oh)(x) = 18x^2 – 81x + 85$.

Recursive function:

A function called itself is called a recursive function any function $f: N^n\text{->}N$ is called total because it is defined for every n-tuple in N^n

If $f:D\text{->}N$ where $D\subseteq N^n$ then f is called partial

Example

$F(x,y)=x+y$ which is defined $\forall$ x,y€N and hence it is a total function $g(x,y)=x-y$ is defined for only those x,y€N which satisfy $x\geq y$ hence $g(x,y)$is partial. A set of three functions called the initial function, which are used in defining other function the initial functions are

Zero function, $Z:Z(x)=0$

Successor function, $S:S(x)=x+1$

Projection function, U_i^n: $U_i^n(x_1,x_2,x^{\cdots\cdots}x_n)=x_i$

Note:

The projection function is also called the generalized identity function

$U_1^2(x,y)=x$

$U_2^3(2,4,6)=4$…etc

Example:

Show that {<x, x>/x€N} which defines the relation of equality is primitive recursive

Solution:

given function $f(x,y)=x-y$ now

$x-y+1$

$(x-y)+1$

$f(x-y)+1$

$f(x,y)+1$

$s(x,y)$

define (x,y) as $f(x,0)=x=p_1^1(x)$

$f(x,y+1)=s(p_3^3(x,y,f(x,y))$

projection function

Therefore sp_1^1,p_2^2,p_3^3 are initial function

Therefore f(x,y) is primitive recursive function

Example:

Show that f(x,y)=x+y,(x,y)€N is primitive recursive

Solution:

$f(x,y)=x+y$

$f(x,0)=x+0$

$x=U_1^1(x)$

$f(x,y+1)=x+y+1$

$=s(x+y)$

$=s(f(x,y))$

$=s(u_3^3(x.y\ f(x.y))$

Therefore it follows that f comes from primitive recursive of u_1^1 and u_3^3

f is primitive recursive

Example:

Find the value of f(2,5) by using f(x,y)=x+y and the initial value (2,0)=2

Solution:

$f(x,y)=x+y$

$f(2,1)=f(2,0)+1=2+1=3$

$f(2,2)=f(2,1)+1=3+1=4$

$f(2,3)=f(2,2)+1=4+1=5$

$f(2,4)=f(2,3)+1=5+1=6$

$f(2,5)=f(2,4)+1=6+1=7$

Some more examples of functions:

Example:

Let f:R->R and g:R->R and h:R->R is defined as f(x)=2x+1 ∀ X€ R h(x)=2x-2 ∀ X€ R and g(x)=3x+2 then find

1. fog
2. gof
3. fo(goh)
4. fo(hog)
5. go(foh)
6. go(fof)
7. ho(gof)

Solution

f(x)=2x+1

$g(x)=3x+2$

$h(x)=2x-2$

1. $fog(x)=f(g(x))$

$=f(3x+2)$

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

$= 6x+4+1$

$= 6x+5$

2.$gof(x)=g(f(x))$

$=g(2x+1)$

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

$=6x+3+2$

$=6x+5$

3.$fo(goh)=f(g(h(x)))$

$=f(g(2x-2))$

$=f(3(2x-2)+2)$

$=f(6x-4)$

$=2(6x-4)+1$

$=12x-8+1$

$=12x-7$

4. $fo(hog)=f(h(g(x)))$

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

$=f(2(3x+2)-2)$

$=f(6x+4-2)$

$=f(6x+2)$

$=2(6x+2)+1$

$=12x+4+1$

$=12x+5$

5. $go(foh)=g(f(h(x)))$

$=g(f(2x-2))$

$=g(2(2x-2)+1)$

$=g(4x-4+1)$

$=g(4x-3)$

$=3(4x-3)+2$

$=12x-9+2$

$=12x-7$

6. $go(fof)=g(f(f(x))$

$=g(f(2x+1))$

$=g(2(3x+2)-2)$

$=g(6x+4-2)$

$=g(6x+2)$

$=2(6x+2)+1$

=12x+4+1

=12x+5

7. ho(gof)=h(g(f(x)))

=h(g(2x+1))

=h(3(2x+1)+2)

=h(6x+3+2)

=h(6x+5)

=2(6x+5)-2

=12x+10-2

=12x+8

Example:

The function f:R->R be defined by

1. $f(x)=x^2+1$ Find $f^{-1}(-8)$ $f^{-1}(17)$

2.$f(x)=x^2+3$ Find $f^{-1}(7)$ $f^{-1}(19)$

Solution:

$f(x)=x^2+1$

f(x)=y

$x^2+1=y$

$x^2=y-1$

$x=\sqrt{y-1}$

$y=f(x)\Rightarrow f^{-1}(y)=\sqrt{y-1}\ \forall$ y€R

$f^{-1}(x)=\sqrt{x-1}$

$f^{-1}(8)$ means that x=-8

$\sqrt{-8-1}=\sqrt{-9}=\pm 3i$

=(3i,-3i)= ∅

$f^{-1}(17)$ means that x=17

$\sqrt{17-1}$ =16=4

=(4,-4)

ii) $f(x)=x^2+3$

f(x)=y

$x^2+3=y$

$x^2=y-3$

$x=\sqrt{y-3}$

y=f(x)=f-1(y)= $\sqrt{y-1}\ \forall$ y € R

$f^{-1}(7)$ means that x=7

$\sqrt{7-3}=\sqrt{4}=\pm 2$

$f^{-1}(19)$ means that x=19

$\sqrt{19-3} = \sqrt{16} = \pm 4$

LATTICE:

Lattice introduced as poset $(p,\leq)$ in which every pair has a greatest lower bound(GLB) and least upper bound(LUP) is called lattice.

GLB:-(greatest lower bound) : greatest lower bound of(a,b)=a*b(or) a.b (or) gcd of a and b (or) $a \cap b$

Example:

GLB of (2,3)=6

Gcd of (2,3) =6

LUB:-(least upper bound): least upper bound of(a,b)=a+b=$a \oplus b$=lcm of a and b=aUb

Example problems:

1. Let p={2,3,6,12} then prove that $(p,\leq)$ this notation is lattice(or) not

Solution:

Given that p=(2,3,6,12}

Consider one pair(2,3) from set p

GLB of(2,3) $=1 \notin p$ means it is not GLB from set p

LUB of (2,3)$=6 \in p$ then $(p,\leq)$ is not a lattice

2. if A is finite set and p(a) is power set then prove that $(p(a), \leq)$ is lattice for

i) A={a}

ii) A={a,b}

Solution:

iii) A={a}

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

GLB of ($\emptyset$,{a})=$\emptyset \cap${a}

$=\emptyset \in p(a)$

Therefore ($\emptyset$, {a}) has a GLB

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

LUB of ($\emptyset$,{a})=$\emptyset \cap${a}

={a} $\in$ p(a)

Therefore ($\emptyset$, {a}) has a LUB

$(p(a), \leq)$ is a lattice

iv) A={a,b}

P(a)={{$\emptyset$},{a},{b},{a,b}}

GLB of ($\emptyset$,{a})=$\emptyset \cap${a}

$=\emptyset \in p(a)$

Therefore $(\emptyset, \{a\})$ has a GLB

LUB of $(\emptyset,\{a\})=\emptyset\cup\{a\}$

$=\{a\}\in p(a)$

Therefore $(\emptyset, \{a\})$ has a LUB

Therefore $(\emptyset, \{a\})$ has GLB and LUB-----□1

GLB of $(\emptyset,\{b\})=\emptyset\cap\{b\}$

$=\emptyset\in \quad p(a)$

Therefore $(\emptyset, \{b\})$ has a GLB

LUB of $(\emptyset,\{b\})=\emptyset\cup\{b\}$

$=\{b\}\in p(a)$

Therefore $(\emptyset, \{b\})$ has a LUB

Therefore $(\emptyset, \{b\})$ has GLB and LUB-----□2

GLB of $(\emptyset,\{a,b\})=\emptyset\cap\{a,b\}$

$=\emptyset\in \quad p(a)$

Therefore $(\emptyset, \{a,b\})$ has a GLB

LUB of $(\emptyset,\{a,b\})=\emptyset\cup\{a,b\}$

$=\{a,b\}\in p(a)$

Therefore $(\emptyset, \{a,b\})$ has a LUB

Therefore $(\emptyset, \{a,b\})$ has GLB and LUB-----□3

GLB of $(\{a\},\{b\})=\{a\}\cap\{b\}$

$=(a,b\}\in p(a)$

Therefore $(\{a\}\{b\})$ has a GLB

LUB of $(\{a\},\{b\})=\{a\}\cup\{b\}$

$=\{a,b\}\in p(a)$

Therefore $(\{a\},\{b\})$ has a LUB

Therefore $(\{a\},\{b\})$ has GLB and LUB-----□4

GLB of $(\{b\},\{a,b\})=\{b\}\cap\{a,b\}$

$=(b\}\in p(a)$

Therefore has $(\{b\},\{a,b\})$ a GLB

LUB of $(\{b\},\{a,b\})=\{b\}\cup\{a,b\}$

$=\{a,b\}\in p(a)$

Therefore $(\{b\},\{a,b\})$has a LUB

Therefore $(\{b\},\{a,b\})$ has GLB and LUB-----□5

GLB of $(\{b\},\{a,b\})=\{a\}\cap\{a,b\}$

$=\{a\}\in p(a)$

Therefore has $(\{a\},\{a,b\})$ a GLB

LUB of $(\{a\},\{a,b\})=\{a\}\cup\{a,b\}$

$=\{a,b\}\in p(a)$

Therefore $(\{a\},\{a,b\})$has a LUB

Therefore $(\{a\},\{a,b\})$ has GLB and LUB-----□6

From equation 1,2,3,4,5,6 $(p(a),\leq)$ is lattice

PROPERTIES OF LATTICE

We shall discuss some properties of two binary operations of meet and join denoted by$\otimes$ *and* $\oplus$ on lattice (L,$\leq$).for all a,b,c$\in$L WE HAVE

1. a*a=a $\quad$ a$\oplus$a=a $\quad$ (idempotent)
2. a*b=b*a $\quad$ a$\oplus$b=b$\oplus$a $\quad$ (commutative)
3. (a*b)*c=a*(b*c) $\quad$ (a$\oplus$b) $\oplus$c=a$\oplus$(b$\oplus$c) $\quad$ (associative)
4.a*(a$\oplus$b)=a $\quad$ a$\oplus$(a*b)=a $\quad$ (absorption law)

LATTICE AS ALGEBRAIC SYSTEM:

A lattice is an algrbraic system (L,*,$\oplus$) with two binary operations *,$\oplus$ on L which are both commutative and associative and satisfy the absorption law.

Here consider the below

a*a=a*(a$\oplus$(a*a))=a

we have replaced the second a in a*a by a$\oplus$(a*a) and then from fourth law we obtained a in the second stepthe identity a$\oplus$a=a can be proved in a similar manner or by the principle of duality

let us define a relation R on L such that for a,b$\in$L

aRb□a*b=a

for any a$\in$L,a*a=a, so that aRa, or the relation R is reflexive now for some a,b$\in$L. let us assume that aRb and bRa,so that a*b=a and b*a=b.but a*b=b*a,and so a=b.the assumptions aRb and bRa imply a=b,or that the relation R is antisymmetric. finally let us assume that for some a,b,c$\in$L, aRb and bRc. this requires that a*b=a, or aRc. the last step shows that the relation R is transitive.from this we can conclude that R is a partial ordering relation.

It is easy to show that a*b=a□a $\oplus\oplus$b=b. hence we could have defined the same partial ordering relation R on L as

aRb□a$\oplus$b=b for any a,b$\in L$

our next step is to show that for any two elements a,b$\in L$,the greatest lower bound and the least upper bound of {a,b}$\subseteq$L with respect to the partial ordering R are a*b and a$\oplus b$respectively.

From the absorption laws a*(a $\oplus b$)=a and b*(a$\oplus b$)=b,we have a R(a$\oplus b$) and bR(a$\oplus b$).

Let us now assume that there exist an element c$\in L$ such that aRc and bRc

This means that

a$\oplus c$=c and b$\oplus c$=c

BOUNDED LATTICE:

A Lattice (L,R) is said to be bounded lattice if it has greatest element and least element

In the bounded lattice a gretest element is denoted by I and least element is denoted by O

NOTE:

$a \vee o=a$
$a \wedge o=o$
$a \vee I=I$
$a \wedge I=a$

DISTRIBUTIVE LATTICE:

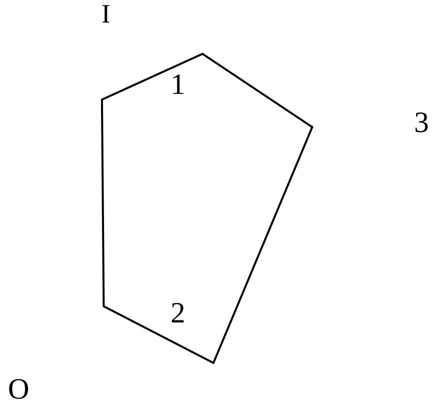
A LATTICE (L,R) is said to be distributive if for any $a,b,c \in L$,the following distributive laws hold,

$a \wedge (b \vee c)=(a \wedge b) \vee (a \wedge c)$

$a \vee (b \wedge c)=(a \vee b) \wedge (a \vee c)$

EXAMPLE

Prove that the hasse diagram is distributive lattice



Assume that

A=1

B=2

C=3

By distributive law we get

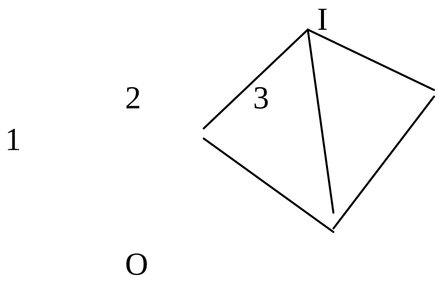
$1 \wedge (2 \vee 3)=(1 \wedge 2) \vee (1 \wedge 3)$

$1 \wedge I=2 \vee O$

$1=2$

Therefore it is not distributive lattice

EXAMPLE
Prove that the hasse diagram is distributive lattice or not



Assume that
A=1
B=2
C=3

By distributive law we get

$1 \wedge (2\vee 3)=(1\wedge 2) \vee(1\wedge 3)$
$1\wedge I=2\vee O$
$1=O$

Therefore it is not distributive lattice

BOUNDED LATTICE:

A Lattice (L,R) is said to be bounded lattice if it has greatest element and least element
In the bounded lattice a gretest element is denoted by I and least element is denoted by O

NOTE:
$a\vee o=a$
$a\wedge o=o$
$a\vee I=I$
$a\wedge I=a$

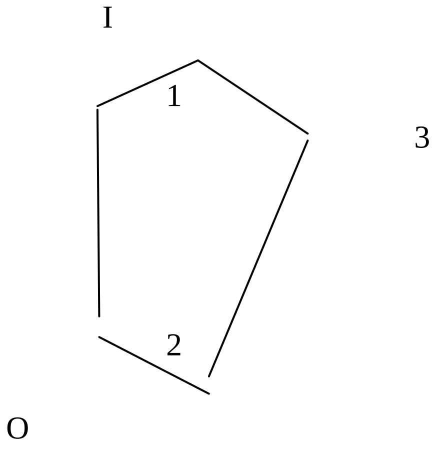
DISTRIBUTIVE LATTICE:

A LATTICE (L,R) is said to be distributive if for any $a,b,c\in L$,the following distributive laws hold,

$a\wedge(b\vee c)=(a\wedge b) \vee(a\wedge c)$
$a \vee (b\wedge c)=(a\vee b) \wedge (a\vee c)$

EXAMPLE

Prove that the hasse diagram is distributive lattice



Assume that

A=1

B=2

C=3

By distributive law we get

$1 \wedge (2 \vee 3)=(1 \wedge 2) \vee (1 \wedge 3)$

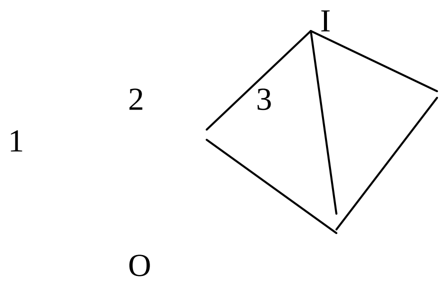
$1 \wedge I=2 \vee O$

1=2

Therefore it is not distributive lattice

EXAMPLE

Prove that the hasse diagram is distributive lattice or not



Assume that

A=1

$B=2$
$C=3$

By distributive law we get

$1 \wedge (2 \vee 3)=(1 \wedge 2) \vee (1 \wedge 3)$
$1 \wedge I=2 \vee O$
$1=O$

Therefore it is not distributive lattice

UNIT-III

1. **Algebraic structures**
    - 1.1 Algebraic systems
    - 1.2 Examples and general properties
    - 1.3 Semi groups and monoids
    - 1.4 Groups
    - 1.5 Sub groups
    - 1.6 Homomorphism
    - 1.7 Isomorphism,
    - 1.8 Rings.
2. **Combinatory:**
    - 2.1 The fundamental counting principles
    - 2.2 Permutations
    - 2.3 Disarrangements
    - 2.4 Combinations
    - 2.5 Permutations and combinations with repetitions
    - 2.6 The binomial theorem
    - 2.7 Multinomial theorem
    - 2.8 Generalized inclusion exclusion principle

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.

For a more comprehensive listing of examples, please see this entry.

Remarks:

- ➢ 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 O contains 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
- 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.
- 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.

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:

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.

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

If a * (b*c) = (a * b) * c for all a, b, c

Example:

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

A Semi group is an algebra which consists of a set and a binary associative operation. There need not be an identity element or inverses for all elements.

A finite or infinite set ‘S′ with a binary operation ‘o′ (Composition) is called semigroup, if it holds following two conditions simultaneously.

Closure Property:

For every pair (a,b)∈ S,(aob) has to be present in the set S.

Associative Property: For every element a,b,c∈ S,(aob)oc=ao(boc) must hold.

The set of positive integers (excluding zero) with addition operation is a semigroup.

For example:

S={1,2,3,…}

Here closure property holds as for every pair (a,b)∈ S,(a+b) is present in the set S.

For example:
1+2=3∈ S
Associative property also holds for every element a,b,c∈ S,(a+b)+c=a+(b+c)

For example:
(1+2)+3=1+(2+3)=5

Let S- be a nonempty set and is a binary operation on S, then the algebraic system(S,*) is called a semigroup, if the operation * is associative. The algebraic system is called semigroup.

$$\forall\, a,b,c \in S,\ a*(b*c)=(a*b)*c$$

It is to note that since the characteristic property of a binary operation on a set S is the closure property, it is not necessary to mention it explicitly when algebraic system is defined.

Monoid

A monoid is a semigroup with an identity element. The identity element (denoted by e or E) of a set S is an element such that $(a \omicron e) = a$, for every element $a \in S$. An identity element is also called a unit element. So, a monoid holds three properties simultaneously − Closure, Associative, Identity element.

Example:

The set of positive integers (excluding zero) with multiplication operation is a monoid.

Here closure property holds as for every pair is present in the set S.

Associative property also holds for every element. Identity property also holds for every element. Here identity element is 1.

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.

Theorem: 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.

Group:

A group is a monoid with an inverse element. The inverse element denotedbyIdenotedbyI of a set S is an element such that (aoI)=(Ioa)=a(aoI)=(Ioa)=a, for each element a∈Sa∈S. So, a group holds four properties simultaneously - i) Closure, ii) Associative, iii) Identity element, iv) Inverse element. The order of a group G is the number of elements in G and the order of an element in a group is the least positive integer n such that an is the identity element of that group G.

Examples:

The set of N×NN×N non-singular matrices form a group under matrix multiplication operation.
The product of two N×NN×N non-singular matrices is also an N×NN×N non-singular matrix which holds closure property. Matrix multiplication itself is associative. Hence, associative property holds. The set of N×NN×N non-singular matrices contains the identity matrix holding the identity element property.

As all the matrices are non-singular they all have inverse elements which are also nonsingular matrices. Hence, inverse property also holds.

Abelian Group:

An abelian group G is a group for which the element pair (a,b)∈G(a,b)∈G always holds commutative law. So, a group holds five properties simultaneously - i) Closure, ii) Associative, iii) Identity element, iv) Inverse element, v) Commutative.

Example:

The set of positive integers includingzeroincludingzero with addition operation is an abelian group. G={0,1,2,3,…}G={0,1,2,3,…}.Here closure property holds as for every pair (a,b)∈S,(a+b)(a,b)∈S,(a+b) is present in the set S. [For example, 1+2=2∈S1+2=2∈S and so on].
Associative property also holds for every element a,b,c∈S,(a+b)+c=a+(b+c)a,b,c∈S,(a+b)+c=a+(b+c)
[For example, (1+2)+3=1+(2+3)=6(1+2)+3=1+(2+3)=6 and so on].

Identity property also holds for every element a∈S,(a×e)=aa∈S,(a×e)=a [For example, (2×1)=2,(3×1)=3(2×1)=2,(3×1)=3 and so on]. Here, identity element is 1.

Commutative property also holds for every element a∈S,(a×b)=(b×a)a∈S,(a×b)=(b×a) [For example, (2×3)=(3×2)=3(2×3)=(3×2)=3 and so on]

Cyclic Group and Subgroup:

A **cyclic group** is a group that can be generated by a single element. Every element of a cyclic group is a power of some specific element which is called a generator. A cyclic group can be generated by a generator 'g', such that every other element of the group can be written as a power of the generator 'g'.

Example:

The set of complex numbers {1,−1,i,−i}{1,−1,i,−i} under multiplication operation is a cyclic group.

There are two generators − ii and –i–i as i1=i,i2=−1,i3=−i,i4=1i1=i,i2=−1,i3=−i,i4=1 and also (–i)1=–i,(–i)2=−1,(–i)3=i,(–i)4=1(–i)1=–i,(–i)2=−1,(–i)3=i,(–i)4=1 which covers all the elements of the group. Hence, it is a cyclic group.

Note : A cyclic group is always an abelian group but not every abelian group is a cyclic group. The rational numbers under addition is not cyclic but is abelian.

A subgroup H is a subset of a group G denotedby$H≤G$denotedby$H≤G$ if it satisfies the four properties simultaneously − Closure, Associative, Identity element, and Inverse.

A subgroup H of a group G that does not include the whole group G is called a proper subgroup Denoted by $H<G$ Denoted by $H<G$. A subgroup of a cyclic group is cyclic and a abelian subgroup is also abelian.

Example:

Let a group G={1,i,−1,−i}G={1,i,−1,−i}

Then some subgroups are H1={1},H2={1,−1}H1={1},H2={1,−1},

This is not a subgroup − H3={1,i}H3={1,i} because that (i)−1=−i(i)−1=−i is not in H3H3

Partially Ordered Set POSETPOSET:

A partially ordered set consists of a set with a binary relation which is reflexive, antisymmetric and transitive. "Partially ordered set" is abbreviated as POSET.

Examples:

- The set of real numbers under binary operation less than or equal to (≤)(≤) is a poset.
- Let the set S={1,2,3}S={1,2,3} and the operation is ≤≤
- The relations will be {(1,1),(2,2),(3,3),(1,2),(1,3),(2,3)}{(1,1),(2,2),(3,3),(1,2),(1,3),(2,3)}
- This relation R is reflexive as {(1,1),(2,2),(3,3)}∈R{(1,1),(2,2),(3,3)}∈R
- This relation R is anti-symmetric, as
- {(1,2),(1,3),(2,3)}∈R and {(1,2),(1,3),(2,3)}∉R{(1,2),(1,3),(2,3)}∈R and {(1,2),(1,3),(2,3)}∉R
- This relation R is also transitive as {(1,2),(2,3),(1,3)}∈R{(1,2),(2,3),(1,3)}∈R.
- Hence, it is a poset.
- The vertex set of a directed acyclic graph under the operation ‘reachability’ is a poset.

By homomorphism we mean a mapping from one algebraic system with a like algebraic system which preserves structures.

Definition:
Let GG and G′G′ be any two groups with binary operation ∘ ∘ and ∘ ′∘ ′ respectively. Then a mapping f:G→G′f:G→G′ is said to be a homomorphism if for all a,b∈ Ga,b∈ G,
f(a∘ b)=f(a)∘ ′f(b)
A homomorphism ff which at the same time is also onto is said to be an epimorphism.
A homomorphism ff which at the same time is also one-one is said to be an monomorphism.
A group G′G′ is called a homomorphism image of a group GG, if there exists a homomorphism ff of GG onto G′G′. A homomorphism of a group GG into itself is called an edomorphism.

Examples:
(i) Let GG be any group under binary operation ∘ ∘ . If f(x)=xf(x)=x for every x∈ Gx∈ G then f:G→Gf:G→G is a homomorphism because
f(xy)=f(x)f(y)f(xy)=f(x)f(y)
(ii) Let GG be the group of integers under addition, let G′G′ be the group of integers under addition modulo nn. If f:G→G′f:G→G′ be defined by f(x)=f(x)=remainder of xx on division by nn, then this is a homomorphism.

(iii) Let GG be any group under addition. If f(x)=ef(x)=e, ∀ x∈ G∀ x∈ G then the mapping f:G→Gf:G→G is a homomorphism because for all x,y∈ Gx,y∈ G, f(x,y)=ef(x,y)=e and f(x)+f(y)=e+e=ef(x)+f(y)=e+e=e, so that
f(x+y)=f(x)+f(y)

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^{-1} 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 "inverse" f^{-1}: Y → X, with the property that both $f^{-1}f = \mathrm{id}_X$ and $f f^{-1} = \mathrm{id}_Y$.

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.

Definition:

A structure (R, +, ·) is a ring if R is a non-empty set and + and are binary operations: such that

$+ \ : R \times R \to R, (a, b) \rightarrowtail a + b$

$. \ : R \times R \to R, (a,b) \rightarrowtail a \cdot b$

Addition: (R, +) is an abelian group, that is,

Associativity: For all a, b, c ∈ R we have

a + (b + c) = (a + b) + c

Zero element: There exists 0 ∈ R such that for all a ∈ R we have a + 0 = 0 + a = a

Inverse: For any a ∈ R there exists −a ∈ R such that

a + (−a) = (−a) + a = 0

Commutativity: For all a, b ∈ R we have

a + b = b + a

Multiplication:

- Associativity: For all a, b, c ∈ R we have a·(b·c) = (a·b)·c

Addition and multiplication together:

- For all a, b, c ∈ R,

  a · (b + c) = a · b + a · c and (a + b) · c = a · b + b · c

- We sometimes say ‘R is a ring’, taken it as given that the ring operations are denoted + and ·. As in ordinary arithmetic we shall frequently suppress · and write ab instead of a · b
- We do NOT demand that multiplication in a ring be commutative. As a consequence we must postulate distributivity as 2 laws, since neither follows from the other in general.

- All of Z, Q, R and C are commutative rings with identity (with the number 1 as the identity).
- N is NOT a ring for the usual addition and multiplication. These are binary operations and we do have a zero element, namely 0, so axiom holds. However (existence of additive inverses) fails: there is no n ∈ N for which 1+n=0.

- **Note:** Polynomials, with real coefficients, form a commutative ring with identity under the usual addition and multiplication; we denote this by R[x].

Assume that (R; +,) is a commutative ring. Let a, b, c R.

iii. If a + b = a + c then b = c.

iv. If a + a = a then a = 0.

v. −(−a) = a.

vi. 0a = 0.

vii. −(ab) = (−a)b = a(−b).

viii. (−1)a = −a. (Assume in addition that R has an identity 1)

ix. If a ∈ R has a multiplicative identity a^{-1} then ab = 0 implies b = 0.

Definition:

A field F is a set together with two binary operations + and ×, satisfying the following properties:

- (F,+) is a commutative group
- (F-{0},×) is a commutative group
- The distributive law holds in F:

$$(a + b) \times c = (a \times c) + (b \times c)$$

Note: A field is a commutative ring with identity where each non-zero element has a multiplicative inverse.

Combinatorics is the study of collections of objects. Specifically, counting objects, arrangement, derangement, etc. along with their mathematical properties. Counting objects is important in order to analyze algorithms and compute discrete probabilities. Originally, combinatorics was motivated by gambling: counting configurations is essential to elementary probability.

A simple example: How many arrangements are there of a deck of 52 cards?
In addition, combinatorics can be used as a proof technique.
A combinatorial proof is a proof method that uses counting arguments to prove a statement.

A simple example: How many arrangements are there of a deck of 52 cards?
In addition, combinatorics can be used as a proof technique.
A combinatorial proof is a proof method that uses counting arguments to prove a statement.

Fundamental Counting Principle can be used to determine the number of possible outcomes when there are two or more characteristics.
Fundamental Counting Principle states that if an event has m possible outcomes and another independent event has n possible outcomes, then there are m* n possible outcomes for the two events together.

Let's start with a simple example:

A student is to roll a die and flip a coin. How many possible outcomes will there be?

1H 2H 3H 4H 5H 6H
1T 2T 3T 4T 5T 6T

6*2 = 12 outcomes
12 outcomes

For a college interview, Robert has to choose what to wear from the following: 4 slacks, 3 shirts, 2 shoes and 5 ties. How many possible outfits does he have to choose from?

4*3*2*5 = 120 outfits

Example: If a password is 6,7,or 8 characters long; a character is an uppercase letters or a digit, and the password is required to include at least one digit - how many passwords can there be?
First, two most basic rules:

- Sum rule
- Product rule

Let us consider two tasks:

- m is the number of ways to do task 1
- n is the number of ways to do task 2

Tasks are independent of each other, i.e.
Performing task 1 does not accomplish task 2 and vice versa.
Sum rule: The number of ways that “either task 1 or task 2 can be done, but not both”, is m + n.

Let us consider two tasks:

- m is the number of ways to do task 1
- n is the number of ways to do task 2

Tasks are independent of each other, i.e.,
Performing task 1does not accomplish task 2 and vice versa.
Product rule: the number of ways that “both tasks 1 and 2 can be done” in mn.
If two events are not mutually exclusive (that is we do them separately), then we apply the product rule

Theorem: Product Rule

Suppose a procedure can be accomplished with two disjoint subtasks. If there are n_1 ways of doing the first task and n_2 ways of doing the second task, then there are $n_1.n_2$ ways of doing the overall procedure.
There is a natural generalization to any sequence of m tasks; namely the number of ways m mutually events can occur $n_1 + n_2 + \ldots + n_{m-1} + n_m$
We can give another formulation in terms of sets. Let $A_1, A_2, \ldots, A_m$ be pairwise disjoint sets. Then
$|A_1 \square A_2 \square \ldots \square A_m | = |A_1| \square |A_2| \square \ldots \square |A_m|$
(In fact, this is a special case of the general Principal of Inclusion-Exclusion (PIE))

A permutation is an arrangement of all or part of a set of objects, with regard to the order of the arrangement. For example, suppose we have a set of three letters: A, B, and C. we might ask how many ways we can arrange 2 letters from that set.

Permutation is defined and given by the following function:

Formula:

nPr=n!(n−r)!
Where ,
n = of the set from which elements are permuted.
r= size of each permutation.
n,r are non negative integers.

Example:

Problem Statement:

A computer scientist is trying to discover the keyword for a financial account. If the keyword consists only of 10 lower case characters (e.g., 10 characters from among the set: a, b, c... w, x, y, z) and no character can be repeated, how many different unique arrangements of characters exist?

Solution:

Step 1: Determine whether the question pertains to permutations or combinations. Since changing the order of the potential keywords (e.g., ajk vs. kja) would create a new possibility, this is a permutations problem.

Step 2: Determine n and r
n = 26 since the computer scientist is choosing from 26 possibilities (e.g., a, b, c... x, y, z).
r = 10 since the computer scientist is choosing 10 characters.

Step 3: Apply the formula

$26P10 = 26!(26-10)!$
$= 26!16!$
$= 26(25)(24)...(11)(10)(9)...(1)/(16)(15)...(1)$
$= 26(25)(24)...(17)$
$= 19275223968000$

Each of several possible ways in which a set or number of things can be ordered or arranged is called permutation Combination with replacement in probability is selecting an object from an unordered list multiple times.

Permutation with replacement is defined and given by the following probability function:
Formula:
$nPr=nr$

Where,
n= number of items which can be selected.
r = number of items which are selected.
nPr= Ordered list of items or permutations.

Example
Problem Statement:
Electronic device usually require a personal code to operate. This particular device uses 4-digits code. Calculate how many codes are possible.

Solution:
Each code is represented by r=4 permutation with replacement of set of 10 digits{0,1,2,3,4,5,6,7,8,9}

10P4=(10)4 =10000

A derangement of {1,2,…,n} is a permutation $i_1i_2…i_n$ of {1,2,…,n} in which no integer is in its natural position:

i_1□1,i_2□2,…,i_n□n.

We denote by D_n the number of derangements of {1,2,…,n}.

Theorem: For n□1,

$$D_n = n!(1-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+\cdots+(-1)^n\frac{1}{n!})$$

- **Proof:** Let S={1,2,…,n} and X be the set of all permutations of S. Then |X|=n!.
- For j=1,2,…,n, let p_j be the property that in a permutation, j is in its natural position. Thus the permutation $i_1,i_2,…,i_n$ of S has property p_j provided i_j=j. A permutation of S is a derangement if and only if it has none of the properties $p_1,p_2,…,p_n$.
- Let A_j denote the set of permutations of S with property p_j (j=1,2,…,n).

Example:

- (1)Determine the number of permutations of {1,2,3,4,5,6,7,8,9} in which no odd integer is in its natural position and all even integers are in their natural position.
- (2) Determine the number of permutations of {1,2,3,4,5,6,7,8,9} in which four integers are in their natural position.

- Permutations with relative forbidden position
- A Permutations of {1,2,…,n} with relative forbidden position is a permutation in which none of the patterns i,i+1(i=1,2,…,n) occurs.
- We denote by Q_n the number of the permutations of {1,2,…,n} with relative forbidden position.

Theorem : For n□1,

Q_n=n!-C(n-1,1)(n-1)!+C(n-1,2)(n-2)!-…+$(-1)^{n-1}$ C(n-1,n-1)1!

Proof: Let S={1,2,…,n} and X be the set of all permutations of S.

Then |X|=n!

j(j+1), p_j

A_j: p_j

$Q_n=D_n+D_{n-1}$

A combination is a selection of all or part of a set of objects, without regard to the order in which objects are selected. For example, suppose we have a set of three letters: A, B, and C. we might ask how many ways we can select 2 letters from that set.

Combination is defined and given by the following function:

Formula:

$C(n,r)=n!r!(n-r)!$

Where , n = the number of objects to choose from.

r= the number of objects selected.

Example:

Problem Statement:

How many different groups of 10 students can a teacher select from her classroom of 15 students?

Solution:

Step 1: Determine whether the question pertains to permutations or combinations. Since changing the order of the selected students would not create a new group, this is a combinations problem.

Step 2: Determine n and r

n = 15 since the teacher is choosing from 15 students.

r = 10 since the teacher is selecting 10 students.

Step 3: Apply the formula

$15C10=15!(15-10)!10!$

$=15!5!10!$

$=15(14)(13)(12)(11)(10!)5!10!$

$=15(14)(13)(12)(11)5!$

$=15(14)(13)(12)(11)5(4)(3)(2)(1)$

$=(14)(13)(3)(11)(2)(1)$

$=(7)(13)(3)(11)$

$=3003$

Each of several possible ways in which a set or number of things can be ordered or arranged is called permutation Combination with replacement in probability is selecting an object from an unordered list multiple times.

Combination with replacement is defined and given by the following probability function:

Formula:

$nCr=(n+r-1)!r!(n-1)!$

Where, n = number of items which can be selected.

r = number of items which are selected.

nCr= Unordered list of items or combinations

Example:

Problem Statement:

There are five kinds of frozen yogurt: banana, chocolate, lemon, strawberry and vanilla. You can have three scoops. What number of varieties will there be?

Solution:

Here n = 5 and r = 4. Substitute the values in formula,

$nCr=(n+r-1)!r!(n-1)! =(5+3+1)!3!(5-1)! =7!3!4! =50406\times24 =35$

Permutations and combinations with repetitions:

The number of permutations of “n” objects, “r” of which are alike, “s” of which are alike, ‘t” of which are alike, and so on, is given by the expression

$$\frac{n!}{r! \times s! \times t! \ldots}$$

Example 1: In how many ways can all of the letters in the word **SASKATOON** be arranged?

Solution: If all 9 letters were different, we could arrange then in 9! Ways, but because there are 2 identical S's, 2 identical A's, and 2 identical O's, we can arrange the letters in:

$$\frac{n!}{r! \times s! \times t! \ldots} = \frac{9!}{2! \times 2! \times 2!} = 45360$$

Therefore, there are 45360 different ways the letters can be arranged.

Example 2: Along how many different routes can one walk a total of 9 blocks by going 4 blocks north and 5 blocks east?

Solution: If you record the letter of the direction in which you walk, then one possible path would be represented by the arrangement NNEEEENENE. The question then becomes one to determine the number of arrangements of 9 letters, 4 are N's and 5 are E's.

$\frac{9!}{5! \times 4!} = 126$ □ Therefore, there are 126 different

Circular Permutations Principle

"n" different objects can be arranged in circle in (n – 1)! ways.

Ring Permutations Principle

"n" different objects can arranged on a circular ring in ways.

$$\frac{(n-1)!}{2}$$

Example 1: In how many different ways can 12 football players be arranged in a circular huddle?

Solution: Using the circular permutations principle there are:

(12 – 1)! = 11! = 39 916 800 arrangements

If the quarterback is used as a point of reference, then the other 11 players can be arranged in 11! ways.

<u>Example 2:</u> In how many ways can 8 different charms be arranged on a circular bracelet?

<u>Solution:</u> Using the ring permutation principle there are:

$$\frac{(n-1)!}{2}=\frac{(8-1)!}{2}=\frac{7!}{2}=2520\ ways$$

- A combination with repetition of objects from is a way of selecting objects from a list of . The selection rules are:
- The order of selection does not matter (the same objects selected in different orders are regarded as the same combination);
- Each object can be selected more than once.
- Thus, the difference between simple combinations and combinations with repetition is that objects can be selected only once in the former, while they can be selected more than once in the latter.
- A more rigorous definition of combination with repetition involves the concept of multiset, which is a generalization of the notion of set.
- The difference between a multiset and a set is the following: the same object is allowed to appear more than once in the list of members of a multiset, while the same object is allowed to appear only once in the list of members of an ordinary set.
- Like sets, multisets are unordered collections of objects, i.e. the order in which the elements of a multiset are listed does not matter.
- A **combination with repetition** of objects from the objects , is one of the possible ways to form a multiset containing objects taken from the set .

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 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 involving factorials. These numbers also arise in combinatorics, where the coefficient of x,y 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

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

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

Bionominal appropriation is a discrete likelihood conveyance. This distribution was discovered by a Swiss Mathematician James Bernoulli. It is used in such situation where an experiment results in two possibilities - success and failure. Binomial distribution is a discrete probability distribution which expresses the probability of one set of two alternatives-successes (p) and failure (q). Binomial distribution is defined and given by the following probability function:

Formula:

P(X−x)=nCxQn−x.px

Where, p= Probability of success.

q= Probability of failure = 1−p

n = Number of trials.

P(X−x)= Probability of x successes in n trials.

Example:

Problem Statement:

Eight coins are tossed at the same time. Discover the likelihood of getting no less than 6 heads.

Solution:

Let p=probability of getting a head.

q=probability of getting a tail.

Here,p=12,q=12,n=8, P(X−x)=nCxQn−x.px,P(at least 6 heads)=P(6H)+P(7H)+P(8H),8C6(12)2(12)6+8C7(12)1(12)7+8C8(12)8,=28×1256+8×1256+1×1256,=37256

The Binomial Theorem:

Here is the expansion of $(x + y)^n$ for $n = 0, 1,\ldots, 5$:

$(x + y)^0 = 1$
$(x + y)^1 = x + y$
$(x + y)^2 = x^2 +2xy + y^2$
$(x + y)^3 = x^3 +3x^2y + 3xy^2 + y^3$
$(x + y)^4 = x^4 +4x^3y + 6x^2y^2 +4xy^3 + y^4$
$(x + y)^5 = x^5 +5x^4y + 10x^3y^2 +10x^2y^3 +5xy^4 + y^5$

Example: Write out the expansion of $(x + y)^7$.

$(x + y)^7 = x^7 +7x^6y + 21x^5y^2 +35x^4y^3 +35x^3y^4 +21x^2y^5 +7xy^6 + y^7$.

When the terms of the binomial have coefficient(s), be sure to apply the exponents to these coefficients.

Example: Write out the expansion of $(2x + 3y)^4$.

$$\begin{aligned}(2x + 3y)^4 &= (2x)^4 +4(2x)^3(3y) + 6(2x)^2(3y)^2 +4(2x)(3y)^3 + (3y)^4\\ &= 16x^4 +4(8x^3)(3y) + 6(4x^2)(9y^2) + 4(2x)(27y^3) + 81y^4\\ &= 16x^4 +96x^3y + 216x^2y^2 +216xy^3 +81y^4.\end{aligned}$$

Example: Write out the expansion of $(5x - y)^3$.

$$\begin{aligned}(5x - y)^3 &= (5x)^3 +3(5x)^2(- y) + 3(5x)(- y)^2 + (- y)^3\\ &= 125x^3 +3(25x^2)(- y) + 3(5x)(y^2) + (- y^3)\\ &= 125x^3 -75x^2y + 15xy^2 - y^3.\end{aligned}$$

Problem : Write out the expansion of $(x + y)^6$.

$(x + y)^6 = x^6 +6x^5y + 15x^4y^2 +20x^3y^3 +15x^2y^4 +6xy^5 + y^6$.

Problem : Write out the expansion of $(x - 5)^4$.

$(x - 5)^4 = x^4 - 20x^3 + 150x^2 - 500x + 625$.

Problem : Write out the expansion of $(2x + 7y)^3$.

$(2x + 7y)^3 = 8x^3 + 84x^2y + 294xy^2 + 343y^3$.

Problem : Write out the expansion of $(1 - x)^7$.

$(1 - x)^7 = 1 - 7x + 21x^2 - 35x^3 + 35x^4 - 21x^5 + 7x^6 - x^7$.

Problem : Write out the expansion of $(x^2 + 3y)^4$.

$(x^2 + 3y)^4 = x^8 + 12x^6y + 54x^4y^2 + 108x^2y^3 + 81y^4$.

1. Counting Subsets (Binomial Coefficients)

1. Counting all subsets

1. Notation: We denote the set of all subsets of a set A by 2^A. For example, $2^{\{a,b,c\}} = \{\phi,\{a\},\{b\},\{c\},\{a,b\},\{a,c\},\{b,c\},\{a,b,c\}\}$. Many books use the notation $P(A)$ (usually with script P) or #(A).

2. Theorem: A set with n elements has 2^n subsets. A brief way of saying this is that if the set A is finite, then $\left|2^A\right| = 2^{|A|}$. Proof: Let $A = [n]$. Then to form a subset of A perform n different tasks. First decide whether to include 1 in the subset (there are 2 choices). Then decide whether to include 2 in the subset (2 choices). Continue in this way for each of the n elements of A. By the multiplication rule, there are 2^n ways to form a subset.

3. Example: $\left|2^{\{a,b,c\}}\right| = \left|\{\phi,\{a\},\{b\},\{c\},\{a,b\},\{a,c\},\{b,c\},\{a,b,c\}\}\right| = 8 = 2^3$.

4. Example: If a pizza parlor offers eight toppings, then you have $2^8 = 256$ ways to top a pizza, not allowing repeated toppings.

2. Counting subsets of a fixed size

1. Definition: We define $\binom{n}{k}$ to be the number of k-subsets of an n-set for nonnegative integers n and k. We call this number the k-th binomial coefficient of order n (for reasons that will soon appear). Note that this does not yet give us a formula for computing binomial coefficients.

2. Examples: Certain values are easy to find without a formula

1. Clearly $\binom{n}{0} = \binom{n}{n} = 1$ for all n, since the first term counts the empty set and the second counts the whole n-set itself.

2. Similarly $\binom{n}{1} = \binom{n}{n-1} = n$ since the first term counts the ways to choose a single element (a 1-subset) and the second couns the ways to leave a single element out.

3. By brute force we can count the 2-subsets of [4] , getting $\{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$. This demonstrates that $\binom{4}{2} = 6$.

4. If k exceeds n then $\binom{n}{k} = 0$.

3. Theorems

1. Theorem 1.9: For all $n,k \in N$ it holds that $\binom{n}{k} = \frac{n^{\underline{k}}}{k!}$. Proof: Suppose we want to count the permutations of [n] taken k at a time. We might do this in two different ways. First we already know the answer is $n^{\underline{k}}$. Second we might choose k elements from [n] to be in the permutation and then we might arrange those k elements in some order. There are $\binom{n}{k}$ ways to do the first task and k! ways to do the second. By the multiplication rule there are $\binom{n}{k}k!$ ways to do the whole job. Since these two approaches both count the permutations of [n] taken k at a time they must be equal. Thus $\binom{n}{k}k! = n^{\underline{k}}$. Dividing both sides of the equation by k! yields the desired result.

2. Notes to theorem 1.9

1. Recall that 0!=1.

2. This is an example of a combinatorial proof. That is, it proves equality of two formulas by showing they count the same quantity. Combinatorial proofs are often far simpler and more intuitive than algebraic proofs of the same results.

3. Multiplying the numerator and denominator of the formula in Theorem 1.9 by (n-k)! yields the familiar formula $\binom{n}{k} = \frac{n^{\underline{k}}}{k!} = \frac{n!}{k!(n-k)!}$. The last formula has the disadvantage, however, that it fails when k exceeds n.

4. Many books call $\binom{n}{k}$ the number of combinations of n things taken k at a time. AVOID THIS TERMINOLOGY! A combination simply means a subset, but the term combination is much more confusing than the simple term subset. Explain that $\binom{n}{k}$ is simply the number of k-subsets of an n-set.

5. Example: Suppose you want to buy a three-topping pizza at a pizza parlor that offers 15 toppings. If repeated toppings are not allowed, how many different pizzas can you order? You simply want a 3-subset of the 15-set of toppings. Thus there are $\binom{15}{3}$ possible three-topping pizzas.

6. Tabulating the values of the binomial coefficients produces Pascal's Triangle (see p. 12 in the book).

3. Theorem 1.10: $\binom{n}{k} = \binom{n}{n-k}$ for integers $0 \le k \le n$. Proof: Combinatorially this is obvious because each way to choose a k-subset also excludes the remaining (n-k)-subset. Algebraically it is also trivial.

4. Theorem 1.11: For $n \in N$ and $k \in P$ it always holds that $\binom{n}{0} = 1$ and $\binom{0}{k} = 0$. Otherwise for $n, k \in P$ the recurrence $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$. Proof: (We prove only the recurrence, the rest being obvious.) Count the k-subsets of [n] according to whether they contain the number n. If so, then the remaining elements form a (k-1)-subset of [n-1]. If not, then the remaining elements form a k-subset of [n-1]. By the addition rule the result follows.

5. Note to Theorem 1.11: This recurrence relation gives the standard rule for constructing Pascal's triangle — namely, each entry is the sum of the two entries "above" it. In the tabular arrangement in the book, this means the sum of the entry directly above with the entry above and to the left.

6. Theorem 1.12: For all nonnegative integers n, $\sum_{k=0}^{n} \binom{n}{k} = 2^n$. Proof: Both sides of the equation count the subsets of [n], the left-hand side (LHS) according to the size of the subset.

7. Theorem 1.13: For positive integers n and k we have $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$. Proof: Suppose we need to appoint a committee of size k with chair from a group of n people. We might first choose the k people and then make one of them the chair. There are $\binom{n}{k}k$ such possibilities. Alternatively we might first appoint a chair and then fill in the remaining k-1 positions on the committee from the remaining n-1 people. There are $n\binom{n-1}{k-1}$ possibilities. Equate these two formulas and divide by k to get the desired result.

8. Theorem 1.14 (Vandermonde's Identity): For nonnegative integers m, n, and r we have $\binom{m+n}{r} = \sum_{j=0}^{r} \binom{m}{j}\binom{n}{r-j}$. Proof: This astonishing equation has a simple combinatorial

proof. Suppose you have a group of m men and n women. Both sides of the equation count the number of ways to appoint a committee of r people from this group. The LHS does it by definition of binomial coefficient. The RHS does it according to the number, j, of men on the committee.

2. The Binomial Theorem

1. Purpose: The binomial theorem tells us what the results will be in multiplying out the expression $(x+y)^n$ using the distributive property. Specifically it tells how what the various "like terms" will be and how many of each will appear (e.g., what the coefficient will be).

2. Explanation: Consider what happens in multiplying out $(x+y)^3$. We get

$$(x+y)^3 = \underbrace{(x+y)}_{A}\underbrace{(x+y)}_{B}\underbrace{(x+y)}_{C} = \underbrace{x}_{A}\underbrace{(x+y)}_{B}\underbrace{(x+y)}_{C} + \underbrace{y}_{A}\underbrace{(x+y)}_{B}\underbrace{(x+y)}_{C}$$

$$= \underbrace{x}_{A}\underbrace{x}_{B}\underbrace{(x+y)}_{C} + \underbrace{x}_{A}\underbrace{y}_{B}\underbrace{(x+y)}_{C} + \underbrace{y}_{A}\underbrace{x}_{B}\underbrace{(x+y)}_{C} + \underbrace{y}_{A}\underbrace{y}_{B}\underbrace{(x+y)}_{C}$$

$$= xxx + xxy + xyx + xyy + yxx + yxy + yyx + yyy$$

$$= x^3 + 3x^2y + 3xy^2 + y^3.$$

3. Further Explanation: Each factor in $(x+y)(x+y)(x+y)$ contributes an x or a y to every word (term) in the final expansion. Since there are two choices (x or y) in each of three positions, by the multiplication rule we get 8 different words (terms) in the expansion. These are all the three-letter words on x and y. We then collect like terms by putting together terms with the same number of x's and y's. For instance, the term $3x^2y$ at the end indicates that it is possible to form three different three-letter words with two x's and one y.

4. More Explanation: Now suppose we want to expand $(x+y)^9$. The terms will be all the nine-letter words on x and y, so when we collect like terms, the sum of the exponents in each term must be nine. How can we find the coefficient of, say, x^7y^2? We must determine how many different nine-letter words have 7 x's and 2 y's. This turns out to be easy: Draw nine spaces and choose seven of them to receive x's (or two to receive y's). The number of ways to do this is $\binom{9}{7} = \binom{9}{2}$. Thus the simplified expansion has a term $\binom{9}{7}x^7y^2 = 36x^7y^2$.

5. Yet More Explanation: Thus $(x+y)^9 = \binom{9}{0}x^9 + \binom{9}{1}x^8y + \binom{9}{2}x^7y^2 + \cdots + \binom{9}{8}xy^8 + \binom{9}{9}y^9$

(Note that you can reverse the order of the coefficients because of theorem 1.10. Similarly you can reverse the order of the terms while keeping the coefficients unchanged.)

6. The Binomial Theorem: $(x+y)^n = \sum_{k=0}^{n}\binom{n}{k}x^k y^{n-k} = \sum_{k=0}^{n}\binom{n}{k}x^{n-k}y^k$. Proof: We have already given the essential idea.

7. Consequences

1. The Binomial Theorem yields an alternative proof of Theorem 1.16. By setting $x = y = 1$ we get $2^n = (1+1)^n = \sum_{k=0}^{n} \binom{n}{k} 1^k 1^{n-k} = \sum_{k=0}^{n} \binom{n}{k}$.

2. Expanding $(1-1)^n$ we see that the sum of the binomial coefficients $\binom{n}{k}$ with k odd equals the sum of them with k even. Thus every set has the same number of odd and even subsets.

3. Frequently in enumeration we want to use the Binomial Theorem "backwards" to express a complicated polynomial as some power of a binomial. The third, fourth, and fifth examples on p. 17 illustrate this technique nicely. The trick is to fiddle with the sum until it looks like a binomial expansion, perhaps with a term missing at the beginning or end.

3. Trinomial and Multinomial Coefficients

1. Example: Suppose we want to form nine-letter words comprising 4 x's, 3 y's, and 2 z's. How many such words are there (words are different if they are visually distinct). Denote the number by $\binom{9}{4,3,2}$. Suppose for a moment that we distinguish the letters by attaching subscripts to them: x_1, x_2, x_3, x_4, y_1, etc. Now there are 9! different nine-letter words using these nine symbols. We can count them in another way, however: First choose the positions for the x's, y's, and z's (without subscripts. There are $\binom{9}{4,3,2}$ ways to do this. Next there are 4! ways to assign subscripts to the x's, 3! ways to do it for the y's, and 2! ways to do it for the z's. Thus $\binom{9}{4,3,2} 4!3!2! = 9!$. Dividing yields $\binom{9}{4,3,2} = \frac{9!}{4!3!2!}$.

2. Definition: Given nonnegative integers n, n_1, n_2, n_3 with $n_1 + n_2 + n_3 = n$, we define the trinomial coefficient by $\binom{n}{n_1, n_2, n_3} = \frac{n!}{n_1!n_2!n_3!}$.

3. Theorem 1.17: Under these circumstances, the trinomial coefficient $\binom{n}{n_1, n_2, n_3} = \frac{n!}{n_1!n_2!n_3!}$ counts the number of words (sequences) of length n with n_1 x's, n_2 y's, and n_3 z's. Equivalently it counts the ways to distribute n labeled balls among three labeled urns such that the first gets n_1 balls, the second gets n_2, and the third gets n_3. (Think of the position numbers in the word corresponding to the balls and the letters x, y, and z being the labels on the urns). Proof: The above example illustrates the central idea.

4. Theorem 1.18: For nonnegative n, the sum of all trinomial coefficients of order n is 3^n. That is, $\sum_{\substack{n_1+n_2+n_3=n \\ n_i \in N}} \binom{n}{n_1, n_2, n_3} = 3^n$. Proof: Summing the trinomial coefficients counts every word of length n on x, y, and z. There are 3^n such words.

5. Theorem 1.19 (The Trinomial Theorem): For nonnegative n we have $(x+y+z)^n = \sum_{\substack{n_1+n_2+n_3=n \\ n_i \in N}} \binom{n}{n_1, n_2, n_3} x^{n_1} y^{n_2} z^{n_3}$. Proof sketch: In expanding the nth power of the trinomial on the left, we get every word of length n on x, y, and z. Much as in the binomial case, the coefficient on a particular term is the number of words with the specified number of x's, y's, and z's.

6. Example: In the expansion of $(x+y+z)^6$ the coefficient of xy^2z^3 is $\binom{6}{1,2,3} = \frac{6!}{1!2!3!} = 60$.

7. Note: There is a trinomial recurrence that produces a "Pascal's Pyramid."

8. Definition: We can generalize the notion of trinomial coefficients to multinomial coefficients $\binom{n}{n_1, n_2, n_3, \ldots, n_k} = \frac{n!}{n_1! n_2! n_3! \ldots n_k!}$, where the n_i sum to n. This coefficient counts the number of words on n letters in which one letter appears n_1 times, another appears n_2 times, etc. It also counts the ways to put n labeled balls in k labeled urns with specified occupancies (n_1 balls in the first urn, etc.)

9. Example: The letters in the word TENNESSEE can be arranged in $\binom{9}{4,2,2,1}$ visually distinct ways.

10. Theorem 1.18 generalizes to multinomial coefficients. That is, the sum of all k-nomial coefficients of order n is k^n. Proof: Analogous to the trinomial case.

11. The Trinomial Theorem generalizes in the obvious fashion to produce a Multinomial theorem (Theorem 1.22).

Generalized inclusion exclusion principle:

6. Say there are two events, e_1 and e_2, for which there are n_1 and n_2 possible outcomes respectively.

7. Now, say that only <u>one</u> event can occur, not both

8. In this situation, we cannot apply the sum rule. Why?

… because we would be over counting the number of possible outcomes.

1. Instead we have to count the number of possible outcomes of e_1 and e_2 minus the number of possible outcomes in common to both; i.e., the number of ways to do both tasks

2. If again we think of them as sets, we have

$|A_1 \square A_2| =|A_1| + |A_2| - |A_1\square A_2|$

8. More generally, we have the following

9. Lemma: Let A, B, be subsets of a finite set U. Then

a. $|A\square B| = |A| + |B| - |A\square B|$

b. $|A \square B| \square \min \{|A|, |B|\}$

c. $|A\backslash B| = |A| - |A\square B| \square |A|-|B|$

d. $|A| = |U| - |A|$

e. $|A\square B| =|A\square B|-|A\square B|= |A|+|B|-2|A\square B|= |A\backslash B|+ |B\backslash A|$

f. $|A \square B| = |A|\square|B|$

3. Theorem: Let $A_1,A_2, \ldots,A_n$ be finite sets, then

$$\begin{aligned}|A_1\square A_2 \square\ldots\square A_n|= {} & \square_i|A_i| \\ & - \square_{i<j}|A_i \square A_j| \\ & + \square_{i<j<k}|A_i \square A_j \square A_k| \\ & - \ldots \\ & +(-1)^{n+1} |A_1\square A_2\square\ldots\square A_n|\end{aligned}$$

Each summation is over

6.1.1. all i,

6.1.2. pairs i,j with i<j,

6.1.3. triples with i<j<k, etc.

v) To illustrate, when n=3, we have

$$\begin{aligned}|A_1\square A_2 \square A_3|= {} & |A_1|+ |A_2| +|A_3| \\ & - [|A_1\square A_2|+|A_1\square A_3|+|A_2\square A_3|] \\ & +|A_1 \square A_2 \square A_3|\end{aligned}$$

3. To illustrate, when n=4, we have

$$\begin{aligned}|A_1\square A_2\square A_3\square A_4|= {} & |A_1|+|A_2|+|A_3|+|A_4| \\ & - [|A_1\square A_2|+|A_1\square A_3|+|A_1\square A_4| \\ & \quad +|A_2\square A_3|+|A_2\square A_4|+|A_3\square A_4|] \\ & + [|A_1\square A_2\square A_3|+|A_1\square A_2\square A_4| \\ & \quad +|A_1\square A_3\square A_4|+|A_2\square A_3\square A_4|] \\ & - |A_1 \square A_2 \square A_3 \square A_4|\end{aligned}$$

1. How many integers between 1 and 300 (inclusive) are

1. Divisible by at least one of 3,5,7?
2. Divisible by 3 and by 5 but not by 7?
3. Divisible by 5 but by neither 3 or 7?

2. Let

$A = \{n\square Z \mid (1 \square n \square 300)\square (3|n)\}$

B = {n□Z | (1 □ n □ 300)□ (5|n)}

C = {n□Z | (1 □ n □ 300)□ (7|n)}

23. How big are these sets? We use the floor function

|A| = □300/3□ = 100

|B| = □300/5□ = 60

|C| = □300/7□ = 42

1. How many integers between 1 and 300 (inclusive) are divisible by at least one of 3,5,7?

Answer: |A□B □C|

9. By the principle of inclusion-exclusion

|A□B □C|= |A|+|B|+|C|-[|A□B|+|A□C|+|B□C|]+|A□B□C|

➢ How big are these sets? We use the floor function

|A| = □300/3□ = 100 |A□B| = □300/15□ = 20

|B| = □300/5□ = 60 |A□C| = □300/21□ = 100

|C| = □300/7□ = 42 |B□C| = □300/35□ = 8

|A□B□C| = □300/105□ = 2

➢ Therefore:

|A□B □C| = 100 + 60 + 42 - (20+14+8) + 2 = 162

* How many integers between 1 and 300 (inclusive) are divisible by 3 and by 5 but not by 7?

Answer: |(A □ B)\C|

1. By the definition of set-minus

|(A □ B)\C| = |A □ B| - |A □ B □ C| = 20 – 2 = 18

4. Knowing that

|A| = □300/3□ = 100 |A□B| = □300/15□ = 20

|B| = □300/5□ = 60 |A□C| = □300/21□ = 100

|C| = □300/7□ = 42 |B□C| = □300/35□ = 8

|A□B□C| = □300/105□ = 2

UNIT-IV

1. **Recurrence relation**
   - 1.1. Generating functions
   - 1.2. Function of sequences calculating coefficient of generating function
   - 1.3. Recurrence relations
   - 1.4. Solving recurrence relation by substitution and generating functions
   - 1.5. Characteristics roots solution of homogeneous recurrence relation.

GENERATING FUNCTIONS

Consider a sequence of real numbers $a_0,a_1,a_2,\ldots$. Suppose there exists a function

$f(x)=a_0+a_1x+a_2x^2+\ldots..a_{n-1}x^{n-1}+\ldots=\sum_{r=0}^{\infty} x^r a_r$.

Then f(x) is the generating function for the sequence $a_0,a_1,a_2,\ldots.a_n$

Examples:

Since $(1-x)^{-1}=1+x+x^2+x^3\ldots..=\sum_{r=0}^{\infty} x^r$,

$f(x)=(1-x)^{-1}$ is a generating function for the sequence 1,1,1,1…..

Similarly

Since $(1+x)^{-1}=1-x+x^2-x^3\ldots.. =\sum_{r=0}^{\infty}(-1)x^r$,

$f(x)=(1+x)^{-1}$ is a generating function for the sequence 1,-1,1,-1…..

Examples for finding Generating Functions:

The generating functions for the following:

5. 1, 2, 3, 4…

Here $a_0=1$, $a_1=2$, $a_2=3$, $a_3=4$

$f(x) = 1x^0+2x^1+3x^2+4x^3+\ldots..$

$= 1+2x+3x^2+4x^3+\ldots\ldots$

$=$ $\mathbf{(1-x)^{-2}}$ is the generating function for the given sequence.

6. 1, -2, 3, -4…

Here $a_0=1$, $a_1= -2$, $a_2=3$, $a_3= -4$

$f(x) = 1x^0-2x^1+3x^2-4x^3+\ldots..$

$=1-2x+3x^2-4x^3+\ldots\ldots$

$=$ $\mathbf{(1+x)^{-2}}$ is the generating function for the given sequence.

7. 0, 1, 2, 3, 4…

Here $a_0=0$, $a_1=1$, $a_2=2$, $a_3=3$

$f(x) = 0x^0+1x^1+2x^2+3x^3+\ldots..$

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

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

$=$ $\mathbf{x\ (1-x)^{-2}}$ is the generating function for the given sequence.

8. 0, -1, 2, -3, 4…

Here $a_0=0$, $a_1= -1$, $a_2=2$, $a_3= -3$

f(x) $= 0x^0-1x^1+2x^2-3x^3+\ldots..$

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

$= -x(1-2x+3x^2-4x^3+\ldots\ldots)$

$=$ **$-x\,(1+x)^{-2}$** is the generating function for the given sequence.

CALCULATING CO-EFFICIENT

We have formulas for Calculating Co-efficient

3. $(1+x)^n = \sum_{r=0}^{\infty} nc_r x^r$

4. $(1+x)^{-n} = \sum_{r=0}^{\infty}(-1)(n+r-1c_r)x^r$

5. $(1-x)^n = \sum_{r=0}^{\infty} nc_r(-1)x^r$

6. $(1-x)^{-n} = \sum_{r=0}^{\infty}(n+r-1c_r)x^r$

Examples:

Determine the coefficient of

5. X^{12} in $x^3(1-2x)^{10}$

$$x^3(1-2x)^{10} = x^3\sum_{r=0}^{10}\binom{10}{r}(-2x)^r$$

$$= \sum_{r=0}^{10}\binom{10}{r}(-2)^r x^{r+3}$$

Therefore the coefficient of X^{12} is

$C_{12} = (-2)^9\binom{10}{9}$

= **-5210.**

6. X^5 in $(1-2x)^{-7}$

$$(1-2x)^{-7} = \sum_{r=0}^{\infty}\binom{7+r-1}{r}(2x)^r$$

$$= \sum_{r=0}^{\infty}\binom{6+r}{r}(2x)^r$$

Therefore the coefficient of X^5 is

$C_5 = (2)^5\binom{11}{5}$

= **14,784.**

7. X^0 in $(3x^2-(2/x))^{15}$

We have $(3x^2-(2/x))^{15} = (3x^2)^{15}(1-\frac{2}{3x^3})^{15}$

$= (3^{15}x^{30})\sum_{r=0}^{15}\binom{15}{r}(-\frac{2}{3x^3})^r$

$= 3^{15}\sum_{r=0}^{15}\binom{15}{r}(-\frac{2}{3})^r x^{30-3r}$

Therefore the coefficient of X^0 is

$C_0 = (3)^{15}\binom{15}{10}(-\frac{2}{3})$

$=$ $\mathbf{3^5*2^{10}*}\binom{15}{10}$

8. X^{10} in $(x^3-5x)/(1-x)^3$

We have $(x^3-5x)/(1-x)^3 = (x^3-5x)(1-x)^{-3}$

$= (x^3-5x)\sum_{r=0}^{\infty}\binom{3+r-1}{r}(x)^r$

$= (x^3-5x)\sum_{r=0}^{\infty}\binom{2+r}{r}(x)^r$

Therefore the coefficient of X^0 is

$C_{10} = \binom{9}{7}-5\binom{11}{9}$

= **-239.**

COUNTING TECHNIQUE

Suppose we wish to determine number of integer solutions of the equation

$\mathbf{x_1+x_2+x_3+\ldots\ldots+x_n = r}$, where $n \geq r \geq 0$,

under constraints that

x_1 can take integer values $p_{11},p_{12},p_{13}\ldots..$

x_2 can take integer values $p_{21},p_{22},p_{23}\ldots..$

……

x_n can take integer values $p_{n1},p_{n2},p_{n3}\ldots..$

To solve this problem, we first define the functions f1(x), $f_2(x)\ldots f_n(x)$ as follows

$f_1(x) = x^{p11}+ x^{p12+} x^{p13}+\ldots..$

$f_2(x) = x^{p21}+ x^{p22+}\ x^{p23}+\ldots..$

$\ldots\ldots\ldots\ldots$

$f_n(x) = x^{pn1}+ x^{pn2+}\ x^{pn3}+\ldots..$

We then consider

$f(x) = f_1(x) . f_2(x) . f_3(x)\ldots. f_n(x)$

Here **f(x) is generating function** for the problem.

Examples:

Using generating function find the number of

➢ Non-negative

➢ Positive

integer solutions of the equation

$x_1+x_2+x_3+x_4=25$

➢ In case of Non-negative integer solutions, x_i can take values 0, 1, 2, 3… Accordingly whose

$f_i(x) = x^{0}+ x^{1+}\ x^2+ x^3+\ldots..$ for i=1, 2,3,4.

Therefore generating function is

$f(x) = f_1(x) . f_2(x). f_3(x).f_4(x)$

$= (x^{0}+ x^{1+}\ x^2+ x^3+\ldots..)^4$

$= ((1-x)^{-1})^4$

$= (1-x)^{-4}$

$= \sum_{r=0}^{\infty}\binom{3+r}{r}(x)^r$

The **coefficient of X^{25}** in this is

$\binom{3+25}{25} =$ **3276.**

Thus the given equation has 3276 Non-negative integer solutions.

➢ In case of Positive integer solutions, x_i can take values 1, 2, 3… Accordingly whose

$f_i(x) = x^{1+}\ x^2+ x^3+\ldots..$ for i=1, 2,3,4.

Therefore generating function is

$f(x) = f_1(x) . f_2(x). f_3(x).f_4(x)$

$= (x^{1+}\ x^2+ x^3+\ldots..)^4$

$=x^4(1+x^{+}\ x^2+ x^3+\ldots..)4$

$= x^4\ ((1-x)^{-1})^4$

$= x^4\ (1-x)^{-4}$

$= x^4 \sum_{r=0}^{\infty}\binom{3+r}{r}(x)^r$

The **coefficient of X^{25}** in this is

$\binom{3+21}{21} =$ **2024.**

Thus the given equation has 2024 Positive integer solutions.

2) Find the number of integer solutions of the equation

$x_1+x_2+x_3+x_4+x_5=30$

under constraints $x_i>=0$ for i=1,2,3,4,5 and further x_2 is even and x_3 is odd. So, We take

$f_1(x)= (x^{0}+ x^1+ x^2+ x^3+\ldots..) = (1-x)^{-1}$

$f_2(x)= (x^{0}+ x^2+ x^4+\ldots\ldots..) = (1-x^2)^{-1}$

$f_3(x)= (x + x^3+ x^5+\ldots\ldots..) = x(1-x^2)^{-1}$

$f_4(x)= (x^{0}+ x^1+ x^2+ x^3+\ldots..) = (1-x)^{-1}$

$f_5(x)= (x^{0}+ x^1+ x^2+ x^3+\ldots..) = (1-x)^{-1}$

Therefore generating function is

$f(x) = f_1(x) . f_2(x). f_3(x).f_4(x).f_5(x)$

$= x (1-x^2)^{-2} (1-x)^{-3}$

$= x \sum_{r=0}^{\infty}\binom{2+r-1}{r}(x^2)^r \ \sum_{s=0}^{\infty}\binom{3+s-1}{s}(x)^s$

The **coefficient of X^{25}** in this is

$\mathbf{C_{30}} =$ $\binom{1}{0}\binom{31}{29}+\binom{2}{2}\binom{29}{27}+\ldots..+\binom{15}{14}\binom{3}{1}$ is required number.

SUBSTITUTION METHOD

Recurrence Relation

A recurrence relation is an equation that defines a sequence based on a rule that gives the next term as a function of the previous term(s).

A recurrence relation is an equation that recursively defines a sequence where the next term is a function of the previous terms (Expressing F_n as some combination of F_i with i<n)

Example – i) Fibonacci series $F_n=F_{n-1}+F_{n-2}$

ii) Tower of Hanoi $F_n= 2F_{n-1}+1$

Solving Recurrence Relation by Substitution Method

In this method we solve relations by substituting values for n and from those results we can get solution for that recurrence relation.

Solving Recurrence Relation by Substitution Method Examples

x. Solve Recurrence Relation $\mathbf{a_n=a_{n-1}+n}$, n>=1 where $a_0=2$ by Substitution Method

Given Recurrence Relation $a_n=a_{n-1}+n$

If n=1 then $a_1 = a_{1-1}+1$

$= a_0+1$

$= 2+1$

$= 3$

If n=2 then $a_2 = a_{2-1}+2$

$= a_1+2$

$= (a_0+1) +2$

$= 3+2$

$= 5$

If n=3 then $a_3 = a_{3-1}+3$

$= a_2+3$

$= (a_1+2) +3$

= (a_0+1) +2+3

= 5+3

= 8

…..

So **$a_n = a_0+1 +2+3+…. +n.$**

$$a_n = a_0 + \frac{n(n+1)}{2}$$

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

xi. Solve Recurrence Relation **$a_n=a_{n-1}+n^3$**, n>=1 where a_0=5 by Substitution Method

Given Recurrence Relation $a_n=a_{n-1}+n^3$

If n=1 then $a_1 = a_{1-1}+1$

= a_0+1

= 5+1

= 6

= a_0+1^3

If n=2 then $a_2 = a_{2-1}+8$

= a_1+8

= (a_0+1) +8

= $a_0+1^3+2^3$

So **$a_n = a_0+1^3+2^3+3^3+…+n^3$**

$a_n = a_0 + \sum n^3$

xii. Solve Recurrence Relation **$a_n=a_{n-1}+n^2$**, n>=1 where a_0=4 by Substitution Method

Given Recurrence Relation $a_n=a_{n-1}+n^2$

If n=1 then $a_1 = a_{1-1}+1$

= a_0+1

=4+1

= a_0+1^2

If n=2 then $a_2 = a_{2-1}+4$

= a_1+4

= (a_0+1) +4

= $a_0+1^2+2^2$

So **$a_n = a_0+1^2+2^2+3^2+…+n^2$**

$a_n = a_0 + \sum n^2$

FIRST ORDER RECURRENCE RELATIONS

We consider for solution recurrence relations of the form

$$\mathbf{a_n = ca_{n-1}+f(n)}, \quad \text{for } n\geq 1$$

Where c is a constant and f(n) is a known function. Such a relation is called recurrence relation of first order with constant coefficient.

The solution for this relation is

$$\mathbf{a_n = c^n a_0} + \sum_{k=1}^{n} c^{n-k} f(k)$$

If f(n)=0, the relation is called homogeneous, otherwise non- homogeneous relation.

So, the solution for homogeneous relation where f(n)=0, is

$\mathbf{a_n = c^n a_0}$

i.e. if the recurrence relation is of the form

$\mathbf{a_n=ca_{n-1}}$

then solution for this is $\mathbf{a_n = c^n a_0}$

Examples

7. Solve the recurrence relation $\mathbf{a_{n+1} = 4a_n}$ for n>0 and $a_0=3$

Given recurrence relation is $a_{n+1} = 4a_n$ which is homogeneous.

Its solution is $\mathbf{a_n = 4^n a_0}$ for $n\geq 1$.

It is given that $a_0=3$

So we get

$\mathbf{a_n = 3.4^n}$ for $n\geq 1$ is the required solution.

8. Solve the recurrence relation $\mathbf{a_n = 7a_{n-1}}$ for $n\geq 1$and $a_0=98$

Given recurrence relation can be written as $a_{n+1} = 7a_n$ for $n\geq 0$ which is homogeneous.

Its solution is $\mathbf{a_n = 7^n a_0}$ for $n\geq 1$.

It is given that $a_0=98$

So we get

$\mathbf{a_n = 98.7^n}$ for $n\geq 1$ is the required solution.

9. Solve the recurrence relation $\mathbf{a_n = na_{n-1}}$ for $n\geq 1$and $a_0=1$

From the given recurrence relation we find that

$a_1 = 1^* a_0$

$a_2 = 2^* a_1 = (2^*1) a_0$

$a_3 = 3^* a_2 = (3^*2^*1) a_0$ and so on.

Its solution is $a_n = n!a_0$ for $n\geq1$.

It is given that $a_0=1$

So we get

$a_n = n!$ for $n\geq1$ is the required solution.

10. If a_n is a solution of recurrence relation **$a_{n+1} = k\ a_n$** for $n\geq0$ and $a_3=153/49$ and $a_5=1377/2401$, what is k?

The solution of relation is **$a_n = k^n a_0$** for $n\geq1$

From this we get $a_3 = k^3a_0$ and $a_5 = k^5a_0$,

so that $a_5/ a_3 = k^2$

Using the given values we get $k^2 =9/49$.

Therefore **$k = \pm 3/7$.**

SECOND ORDER LINEAR HOMOGENEOUS RECURRENCE RELATIONS

We consider solution for recurrence relations of the form

$$c_na_n+ c_{n-1}a_{n-1}+ c_{n-2}a_{n-2} = 0, \text{ for } n\geq2,$$

Where c_n, c_{n-1}, c_{n-2} are real constants with $c_n \neq 0$.

A relation of this type is called recurrence relation of second order linear homogeneous relation with constant coefficients.

We write this as

$c_nk^2+ c_{n-1}k+ c_{n-2} = 0$

This quadratic equation is called auxiliary equation or characteristic equation. To solve this relation following three cases will arise:

Case 1: If the roots k_1 and k_2 of the above equation are **real and distinct** then the general solution is

$$a_n = A\ k_1^n + B\ k_2^n$$

Where A and B are arbitrary real constants.

Case 2: If the roots k_1 and k_2 of the above equation are **real and equal** then the general solution is

$$a_n = (A+Bn)\ k^n$$

Where A and B are arbitrary real constants.

Case 3: If the roots k_1 and k_2 of the above equation are **complex**. So that if $k_1 = p + iq$, $k_2 = p - iq$, then the general solution is

$$a_n = r^n(A \cos n\theta + B \sin n\theta)$$

Where A and B are arbitrary complex constants, $r = | k_1 | = | k_2 | = \sqrt{}\ (p^2 + q^2)$ and $\theta = \tan^{-1}(q/p)$.

Examples how to solve these type of recurrence relations:

Example for Case 1:

- Solve the recurrence relation

$a_n + a_{n-1} + 6a_{n-2} = 0$, for $n \geq 2$, with $a_0 = -1$ and $a_1 = 8$

Sol: Given

$$a_n + a_{n-1} - 6a_{n-2} = 0$$

We can write this as characteristic equation

$k^2+k-6 = 0$

$$(k-2)(k+3) = 0$$

The roots are $k_1 = 2$, $k_2 = -3$ which are **real and distinct.**

So, the general solution is

$$a_n = A k_1^n + B k_2^n$$

$a_n = A(-3)^n + B 2^n$

where A and B arbitrary constants. From this solution we get

$$a_0 = A + B$$

$$a_1 = A(-3) + B(2)$$

Using the given values $a_0 = -1$ and $a_1 = 8$ we get

$$-1 = A + B$$

$$8 = -3A + 2B$$

Solving these, we get A = -2, B = 1. Putting these in our general solution, we get

$a_n = -2(-3)^n + 2^n$ is the required solution.

Example for Case 2:

- Solve the recurrence relation

$a_n - 6a_{n-1} + 9a_{n-2} = 0$, for $n \geq 2$, with $a_0 = 5$ and $a_1 = 12$

Sol: Given

$$a_n + a_{n-1} + 6a_{n-2} = 0$$

We can write this as characteristic equation

$k^2-6k+9 = 0$

$$(k-3)(k-3) = 0$$

The roots are $k_1 = k_2 = 3$ which are **real and equal.**

So, the general solution is

$$a_n = (A+Bn) k^n$$

$a_n = (A+Bn) 3^n$

where A and B arbitrary constants. From this solution we get

$$a_0 = A$$

$$a_1 = (A + B) 3$$

Using the given values $a_0 = 5$ and $a_1 = 12$ we get

$$5 = A$$

$$12 = 3(A + B)$$

Solving these, we get A = 5, B = -1. Putting these in our general solution, we get

$a_n = (5 - n)3^n$ is the required solution.

Example for Case 3:

- Solve the recurrence relation

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

Sol: Given

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

We can write this as characteristic equation
$k^2-2k+2 = 0$

Whose roots are $k = (2 \pm \sqrt{(4 - 8)}) / 2$

$k = 1 \pm i$ which are **complex.**

So, the general solution is

$$a_n = r^n(A \cos n\,\theta + B \sin n\,\theta)$$

where A and B arbitrary constants, $r = \sqrt{2}$ and $\tan \theta = 1$ which yields $\theta = \pi/4$. Thus

$$a_n = (\sqrt{2}\,)^n(A \cos n\,\pi/4 + B \sin n\,\pi/4)$$

Using the given values $a_0 = 1$ and $a_1 = 2$ we get

$$1 = A$$

$$2 = (\sqrt{2}\,)(A \cos\ \pi/4 + B \sin\ \pi/4) = A+B$$

Solving these, we get A = 1, B = 1. Putting these in our general solution, we get

$$\mathbf{a_n = (\sqrt{2}\,)^n(\cos n\,\pi/4 + \sin n\,\pi/4)}$$

is the required solution.

THIRD AND HIGHER ORDER LINEAR HOMOGENEOUS RECURRENCE RELATIONS

We consider solution for recurrence relations of the form

$$\mathbf{c_na_n+ c_{n-1}a_{n-1}+ c_{n-2}a_{n-2}+\ldots..+ c_{n-k}a_{n-k}= 0,\ \ for\ n\geq k\geq 3,}$$

Where c_n, $c_{n-1},\ldots..c_{n-k}$ are real constants with $c_n \neq 0$.

A relation of this type is called recurrence relation of third and higher order linear homogeneous relation with constant coefficients.

The method of solving these relations is same as second order linear homogeneous relation.

Examples:

1. Solve the recurrence relation

$\mathbf{2a_{n+3} = a_{n+2} + 2a_{n+1} -a_n,}$ for $n\geq 0$, with $a_0 = 0$, $a_1 = 1$, $a_2 = 2$

Given relation is same as

$$2a_n- a_{n-1}- 2a_{n-2}+a_{n-3}= 0 \qquad \text{for } n\geq 3$$

We can write this as characteristic equation
$2k^3-k-2k+1 = 0$

$$(2k-1)\,(k^2-1) = 0$$

The roots are $k_1 = 1/2$, $k_2 = 1$, $k_3 = -1$, which are real and distinct.

So, the general solution is

$$a_n = A\,(1/2)^n + B\,1^n + C\,(-1)^n$$

Where A, B, C are arbitrary constants. To determine A,B,C we use given values $a_0 = 0$, $a_1 = 1$, $a_2 = 2$. We get

$$0 = a_0 = A\,(1/2)^0 +B\,(1)^0 + C\,(-1)^0$$

$$1 = a_1 = A\,(1/2)^1 +B\,(1)^1 + C\,(-1)^1$$

$$2 = a_2 = A\,(1/2)^2 +B\,(1)^2 + C\,(-1)^2$$

These can be written as

$$A + B + C = 0$$

$A + 2B - 2C = 2$

$A + 4B + 4C = 8$

Solving these, we get $A = -8/3$, $B = 5/2$, $C = 1/6$ Putting these in our general solution, we get

$a_n = -8/3 (1/2)^n + 5/2 (1)^n + 1/6 (-1)$^{n} is the required solution.

2. Solve the recurrence relation

$a_n+a_{n-1}- 8a_{n-2}-12a_{n-3}= 0$ for $n\geq3$, with $a_0 = 0$, $a_1 = 5$, $a_2 = 1$.

Given relation

$a_n+a_{n-1}- 8a_{n-2}-12a_{n+3}= 0$ for $n\geq3$

We can write this as characteristic equation

$k^3+k^2-8k-12 = 0$

$(k+2)^2 (k-3) = 0$

Whose roots are $k_1 = k_2 = -2$, $k_3 = 3$, which are real and distinct.

So, the general solution is

$a_n = (A + Bn) (-2)^n + C (3)^n$

Where A, B, C are arbitrary constants. To determine A,B,C we use given values $a_0 = 1$, $a_1 = 5$, $a_2 = 1$. We get

$1 = a_0 = A + C$

$5 = a_1 = -2(A + B) + 3C$

$1 = a_2 = 4(A+2B) + 9C$

Solving these, we get $A = 0$, $B = -1$, $C = 1$. Putting these in our general solution, we get

$a_n = (-n) (-2)^n + (3)^n$ is the required solution.

3. Solve the recurrence relation

$a_n = 6a_{n-1}- 12a_{n-2}+8a_{n-3}$ for $n\geq3$, with $a_0 = 1$, $a_1 = 4$, $a_2 =28$.

Given relation

$a_n - 6a_{n-1}+12a_{n-2}-8a_{n-3} = 0$ for $n\geq3$

We can write this as characteristic equation

$k-6k^2+12k-8 = 0$

$(k-2)^3 = 0$

Whose roots are $k_1 = k_2 = k_3 = 2$, which are real and distinct.

So, the general solution is

$a_n = (A + Bn+Cn^2) (2)^n$

Where A, B, C are arbitrary constants. To determine A,B,C we use given values $a_0 = 1$, $a_1 = 4$, $a_2 = 28$. We get

$1 = a_0 = A$

$4 = a_1 = 2(A + B + C)$

$28 = a_2 = 4(A + 2B + 4C)$

Solving these, we get $A = 1$, $B = -1$, $C = 2$. Putting these in our general solution, we get

$a_n = (1 – n + 2n^2) (2)^n$ is the required solution.

SECOND AND HIGHER ORDER LINEAR NONHOMOGENEOUS RECURRENCE RELATIONS

We consider solution for recurrence relations of the form

$$\mathbf{c_na_n+ c_{n-1}a_{n-1}+ c_{n-2}a_{n-2}+\ldots..+ c_{n-k}a_{n-k}= f(n),} \quad \text{for } n\geq k\geq 2,$$

Where c_n, c_{n-1}, ….c_{n-k} are real constants with $c_n \neq 0$ and f(n) is a real valued function of n.

The general solution of this relation is given by

$\mathbf{a_n = a_n^h + a_n^p}$

Where $\mathbf{a_n^h}$ is the general solution of homogeneous part namely f(n)=0 and $\mathbf{a_n^p}$ is the general solution of particular part.

The solution for homogeneous part ($\mathbf{a_n^h}$) is same as of characteristic equation.

The solution for particular part ($\mathbf{a_n^p}$) has four cases.

Case 1: Suppose f(n) is a polynomial of degree q and 1 is not a root of characteristic equation of homogeneous part of the relation. Then

$$\mathbf{a_n^p = A_0+A_1n+A_2n^2+\ldots+A_qn^q}$$

Where A_0, A_1, A_2....... A_q are constants to be evaluated by using the fact that $\mathbf{a_n = a_n^p}$ satisfies the relation.

Case 2: Suppose f(n) is a polynomial of degree q and 1 is a root of multiplicity m of characteristic equation of homogeneous part of the relation. Then

$$\mathbf{a_n^p = n^m (A0+A_1n+A_2n^2+\ldots+A_qn^q)}$$

Where A_0, A_1, A_2....... A_q are constants to be evaluated by using the fact that $\mathbf{a_n = a_n^p}$ satisfies the relation.

Case 3: Suppose $f(n) = \alpha\ b^n$ where α is a constant and b is not a root of characteristic equation of homogeneous part of the relation. Then

$$\mathbf{a_n^p = A_0\ b^n}$$

Where A_0, A_1, A_2....... A_q are constants to be evaluated by using the fact that $\mathbf{a_n = a_n^p}$ satisfies the relation.

Case 4: Suppose $f(n) = \alpha b^n$ where α is a constant and b is a root of multiplicity m of the characteristic equation of homogeneous part of the relation. Then

$$\mathbf{a_n^p = n^m\, A_0\ b^n}$$

Where A_0, A_1, A_2....... A_q are constants to be evaluated by using the fact that $\mathbf{a_n = a_n^p}$ satisfies the relation.

SECOND AND HIGHER ORDER LINEAR NONHOMOGENEOUS RECURRENCE RELATIONS

Example for Case 1:

Solve the Recurrence Relation

$$\mathbf{a_n+4a_{n-1}+4a_{n-2} = 8}, \quad \text{where } a_0=1,\ a_1=2.$$

Given relation is

$a_n+4a_{n-1}+4a_{n-2} = 8$ in this the homogeneous part is

$a_n+4a_{n-1}+4a_{n-2} = 0$

The characteristic equation for this is

$$k^2+4k+4=0$$
$$(k+2)^2 = 0$$
$$k_1 = k_2 = -2 = k$$

So the general solution for this homogeneous part is

$\mathbf{a_n^h = (A + B\ n)\ k^n}$

$= (A + B\ n)\ (-2)^n$

Next we move to solution for particular part.

Here f(n)=8, is a polynomial degree of 0 and 1 is not a root of characteristic equation.

Then the solution for particular part is

$\mathbf{a_n^p = A_0}$

Substitute this part in given recurrence relation a_n, we get

$A_0 + 4\ A_0 + 4\ A_0 = 8$

$9\ A_0 = 8$

$A_0 = 8/9$

The solution for given recurrence relation is

$\mathbf{a_n = a_n^h + a_n^p}$

$a_n = (A + B\ n)\ (-2)^n + 8/9$

To find A, B values substitute initial conditions a_0=1, a_1=2 in above equation.

$1 = a_0 = A + 8/9$

$2 = a_1 = (A+B)\ (-2) + 8/9$

By solving these we get A = 1/9, B = -2/3. Putting these values in our solution we get

$\mathbf{a_n = (1/9 + -2/3\ n)\ (-2)^n + 8/9}$ is the required solution.

Example for Case 2:

Solve the Recurrence Relation

$\mathbf{a_{n+2} - 4a_{n+1}+3a_n = -\ 200,}$ where a_0=3000, a_1=3300.

Given relation is

$a_{n+2} - 4a_{n+1}+3a_n = -\ 200$ in this the homogeneous part is

$a_{n+2} - 4a_{n+1}+3a_n = 0$

The characteristic equation for this is

$k^2-4k+3=0$

$(k-3)\ (k-1) = 0$

$k_1 = 1,\ k_2 = 3$

So the general solution for this homogeneous part is

$\mathbf{a_n^h = A\ k_1^n + B\ k_2^n}$

$= A\ (1)^n + B\ (3)^n$

Next we move to solution for particular part.

Here f(n)= -200, is a polynomial degree of 0 and 1 is a root of characteristic equation.

Then the solution for particular part is

$a_n^p = n^m A_0$

Here multiplicity m = 1 so

$a_n^p = nA_0$

Substitute this part in given recurrence relation a_n, we get

$A_0(n+2) - 4\ [A_0(n+1)] + 3\ (A_0 n) = -200$

$-2\ A_0 = -200$

$A_0 = 100$

So, $\mathbf{a_n^p = 100n}$

The solution for given recurrence relation is

$\mathbf{a_n = a_n^h + a_n^p}$

$a_n = A\ (1)^n + B\ (3)^n + 100n$

To find A, B values substitute initial conditions a_0=3000, a_1=3300 in above equation.

$3000 = a_0 = A + B + 100$

$3300= a_1 = A + 3B + 100$

By solving these we get A = 2900, B = 100. Putting these values in our solution we get

$\mathbf{a_n = 2900\ (1)^n + 100\ (3)^n + 100n}$ is the required solution.

Example for Case 3:

Solve the Recurrence Relation

$\mathbf{a_{n+2} +3a_{n+1}+2a_n = 3^n,}$ where $a_0 = 0$, $a_1 = 1$.

Given relation is

$a_{n+2} +3a_{n+1}+2a_n = 3^n$ in this the homogeneous part is

$a_{n+2} +3a_{n+1}+2a_n = 0$

The characteristic equation for this is

$k^2+3k+2=0$

$(k+2)\ (k+1) = 0$

$k_1 = -2$, $k_2 = -1$

So the general solution for this homogeneous part is

$\mathbf{a_n^h = A\ k_1^n + B\ k_2^n}$

$= A\ (-2)^n + B\ (-1)^n$

Next we move to solution for particular part.

Here $f(n) = 3^n$ here $f(n)= \alpha b^n$ where α is a constant and 1 is not a root of characteristic equation.

Then the solution for particular part is

$\mathbf{a_n^p = A_0 b^n}$

$= A_0 3^n$

Substitute this part in given recurrence relation a_n, we get

$A_0(3^{n+2}) +3\ [A_0(3^{n+1})] + 2\ (A_0 3^n) = 3^n$

$3^n\ A_0\ (20) = 3^n$

$A_0 = 1/20$

$a_n^p = (1/20)\ 3^n$

The solution for given recurrence relation is

$\mathbf{a_n = a_n^h + a_n^p}$

$a_n = A\ (-2)^n + B\ (-1)^n + (1/20)\ 3^n$

To find A, B values substitute initial conditions $a_0=0$, $a_1=1$ in above equation.

$0 = a_0 = A + B + (1/20)\ 3^n$

$1= a_1 = -2A - B + (1/20)\ 3^n$

By solving these we get A = -4/5, B = 3/4. Putting these values in our solution we get

$\mathbf{a_n = -4/5\ (-2)^n + 3/4\ (-1)^n + (1/20)\ 3^n}$ is the required solution.

Example for Case 4:

Solve the Recurrence Relation

$\mathbf{a_n+4a_{n-1}+4a_{n-2} = 5 * (-2)^n,}$ $n\geq 2$

Given relation is

$a_n+4a_{n-1}+4a_{n-2} = 5 * (-2)^n$ in this the homogeneous part is

$a_n+4a_{n-1}+4a_{n-2} = 0$

The characteristic equation for this is

$k^2+4k+4=0$

$(k+2)^2 = 0$

$k_1 = k_2 = -2$

So the general solution for this homogeneous part is

$$\mathbf{a_n^h = (A + B\ n)\ k^n}$$
$$= (A + B\ n)\ (-2)^n$$

Next we move to solution for particular part.

Here $f(n) = (-2)^n$ i.e., $f(n)= \alpha b^n$ where α is a constant and $b(-2)$ is a repeated root of characteristic equation.

Then the solution for particular part is

$$\mathbf{a_n^p = n^m A_0 b^n} \text{ here multiplicity } m=2$$
$$= n^2 A_0(-2)^n$$

Substitute this part in given recurrence relation a_n, we get

$$A_0\, n^2\ (-2)^n + 4A_0\,(n-1)^2\ (-2)^{n-1} + 4A_0(n-2)^2\ (-2)^{n-2} = 5*(-2)^n$$
$$A_0 = 5/2$$
$$a_n^p = (5/2)\ n^2\ (-2)^n$$

The solution for given recurrence relation is

$$\mathbf{a_n = a_n^h + a_n^p}$$
$$a_n = (A + B\ n)\ (-2)^n + (5/2)\ n^2\ (-2)^n$$
$$= [A + B\ n + (5/2)\ n^2\ ]\ (-2)^n$$
$$\mathbf{a_n = [A + B\ n + (5/2)\ n^2\ ]\ (-2)^n} \text{ is the required solution.}$$

METHOD OF GENERATING FUNCTIONS

In this section, we describe the method of solving linear recurrence relations with constant coefficients through use of generating functions.

We consider the cases of first order and second order relations separately.

Method of Generating Functions for First order Recurrence Relations

Suppose the recurrence relation to be solved is of the form

$$\mathbf{a_n = ca_{n-1} + F(n),} \quad \text{for } n\geq 1$$

or equivalently

$$\mathbf{a_{n+1} - ca_n = \phi(n)}\ , \quad \text{for } n\geq 0$$

Where c is a constant and $\phi(n) = F(n+1)$.

Therefore the Generating Function for this relation is

$$\mathbf{f(x) = (a_0 + x\ g(x)\ )\ /\ (1-cx)}$$

Where $\mathbf{g(x) =} \sum_{n=0}^{\infty} \phi(n)(x)^n$ and $\mathbf{f(x) =} \sum_{n=0}^{\infty} a_n x^n$

Example: Solve the recurrence relation using generating function

$$\mathbf{a_{n+1} - a_n = 3^n}, \ n\geq 0 \text{ and } a_0 = 1.$$

The given relation is of the form

$a_{n+1} - ca_n = \phi(n)$ with $c = 1$ and $\phi(n) = 3^n$

Therefore the generating function for this relation is

$$f(x) = (a_0 + x\ g(x)\ )\ /\ (1-cx) \quad \text{where}$$

$g(x) = \sum_{n=0}^{\infty} \phi(n)(x)^n$

$= \sum_{n=0}^{\infty} (3)^n\ (x)^n$

$= \sum_{n=0}^{\infty} (3x)^n$

$= (1-3x)^{-1}$

Also, it is given that $a_0 = 1$. Using these, we get

$$f(x) = (1 + x\,(1-3x)^{-1}) / (1-x)$$
$$= (1-3x) + x / ((1-3x)(1-x))$$
$$= (1-2x) / ((1-3x)(1-x)).$$

This is the required generating function.

Let $(1-2x) / ((1-3x)(1-x)) = \frac{A}{(1-x)} + \frac{B}{(1-3x)}$

$$(1-2x) = A\,(1-3x) + B\,(1-x)$$

Equating the corresponding coefficients in this, we get

$$1 = A + B$$
$$-2 = -3A-B$$

Solving these we get A = B = 1/2. Thus,

$$(1-2x) / ((1-3x)(1-x)) = 1/2\ \left(\frac{1}{(1-x)} + \frac{1}{(1-3x)}\right)$$

Therefore $f(x) = 1/2\ \left(\frac{1}{(1-x)} + \frac{1}{(1-3x)}\right)$

$$= 1/2\ \left(\sum_{n=0}^{\infty} (x)^n + \sum_{n=0}^{\infty} (3x)^n\right.$$
$$=1/2\ \left(\sum_{n=0}^{\infty} (1 + 3^n)(x)^n\right)$$

Accordingly, since $f(x) = \sum_{n=0}^{\infty} a_n x^n$, we find that

$\mathbf{a_n = ½ (1+3^n)}$ is the required solution.

Method of Generating Functions for Second order Recurrence Relations

Suppose the recurrence relation to be solved is of the form

$\mathbf{a_n + Aa_{n-1} + Ba_{n-2} = F(n)}$ for $n \geq 2$

or equivalently

$\mathbf{a_{n+2} + Aa_{n+1} + Ba_n = \phi(n)}$ **,** for $n \geq 0$

Where A and B are constants and $\phi(n) = F(n+2)$.

Therefore the Generating Function for this relation is

$$\mathbf{f(x) = [a_0 +(a_1 + a_0 A)x+x^2 g(x)] / (1+Ax+Bx^2)}$$

Where **g(x) =** $\sum_{n=0}^{\infty} \phi(n)(x)^n$ and **f(x) =** $\sum_{n=0}^{\infty} a_n x^n$

Example: Solve the recurrence relation using generating function

$\mathbf{a_{n+2} - 3a_{n+1}+2\,a_n = 0,}$ $n \geq 0$ and $a_0 = 1$, $a_1 = 6$.

The given relation is of the form

$\mathbf{a_{n+2} + Aa_{n+1} + Ba_n = \phi(n)}$ with A = -3, B = 2 and $\phi(n) = 0$ means it is a homogeneous relation.

Therefore g(x) = 0

Also it is given that $a_0 = 1$, $a_1 = 6$.

Therefore the generating function for this relation is

$$f(x) = [a_0 +(a_1 + a_0 A)x+x^2 g(x)] / (1+Ax+Bx^2)$$
$$= (1+3x) / (1-3x+2x^2)$$
$$= (1+3x) / [(1-x)(1-2x)]$$

This is the required generating function.

Now taking $(1+3x) / [(1-x)(1-2x)] = \frac{A}{(1-x)} + \frac{B}{(1-2x)}$

$$(1+3x) = A\,(1-2x) + B\,(1-x)$$

Equating the corresponding coefficients in this, we get

$$1 = A + B$$

3 = -2A-B

Solving these we get A = -4, B = 5. Thus,

(1+3x) / [(1-x)(1-2x)] = $(\frac{-4}{(1-x)} + \frac{5}{(1-2x)})$

Therefore f(x) = $(\frac{-4}{(1-x)} + \frac{5}{(1-2x)})$

= -4(1-x)$^{-1}$ +5(1-2x)$^{-1}$

= -4 ($\sum_{n=0}^{\infty} (x)^n + 5\sum_{n=0}^{\infty} (2x)^n$

= $\sum_{n=0}^{\infty} (5 * 2^n - 4)(x)^n$

Accordingly, since f(x) = $\sum_{n=0}^{\infty} a_n x^n$, we find that

a_n = 5*2^n - 4 is the required solution.

UNIT-V

1. **Graphs**

    1.1. Basic concepts of graphs

    1.2. Isomorphic graphs

    1.3. Euler graphs

    1.4. Hamiltonian graphs

    1.5. Planar graphs

    1.6. Graph coloring,

    1.7. Digraphs

    1.8. Directed acyclic graphs

    1.9. Weighted digraphs

    1.10. Region graph

    1.11. Chromatic numbers

2. **Trees**

    2.1. Tree

    2.2. Spanning trees

    2.3. Minimal spanning trees.

Basic graph concepts

A graph is a mathematical object that is used to model different situations – objects and processes:

1. Linked list
2. Tree
3. Flowchart of a program
4. Structure chart of a program
5. Finite state automata
6. City map
7. Electric circuits
8. Course curriculum

Definition

A graph is a collection (nonempty set) of vertices and edges

Vertices: can have names and properties

Edges: connect two vertices, can be labeled, can be directed

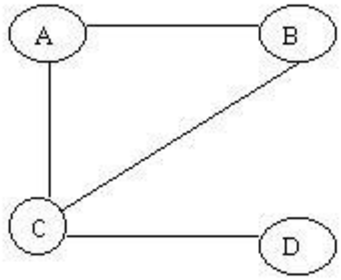
Adjacent vertices: if there is an edge between them.

Example:

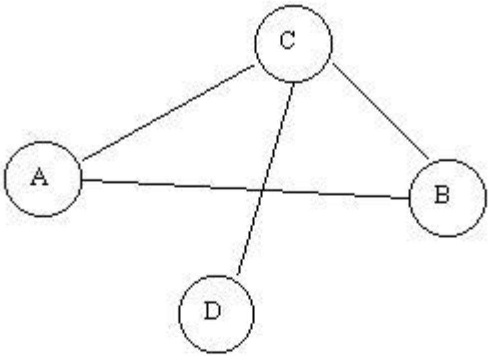
Vertices: A,B,C,D

Edges: AB, AC, BC, CD

Graph1:



Same graph given in another way:



Directed graphs and undirected graphs

There are two basic types of graphs - directed and undirected.

In undirected graphs the edges are symmetrical, e.g. if A and B are vertices,
A B and B A are one and the same edge.

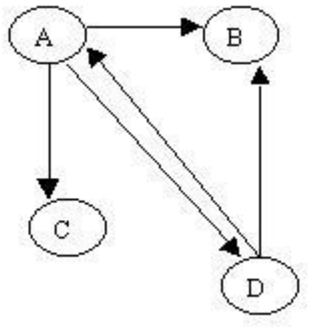
Graph1 above is undirected.

In directed graphs the edges are oriented, they have a beginning and an end.
Thus A B and B A are different edges.

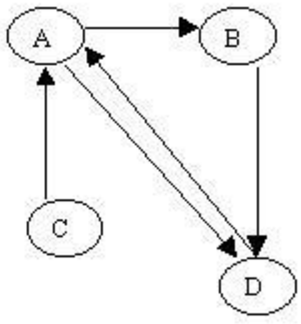
Sometimes the edges of a directed graph are called **arcs**.

Examples of directed graphs

Graph2:



Graph3:



Graph2 and **Graph3** are different graphs

Some definitions of basic graph concepts differ slightly depending on whether we talk about directed or undirected graphs.

Paths

A path is a list of vertices in which successive vertices are connected by edges

Examples

Some paths in **Graph1** :

A B C D

A C B A C D

A B

D C B

C B A

Some paths in **Graph2**:

D A B

A D A C

Note that D A B is a path in **Graph3**, however A D A C is not a path in **Graph3**because A C is not an edge (the edge is C A).

Note: In different textbooks you may find different definitions for a path in a graph. Apart from the usual confusion, there is nothing wrong in this as long as the definitions are followed throughout the presented theory.

Simple path No vertex is repeated.

Examples:

In **Graph1,** D C B A is a simple path, while D C B A C is not a simple path

In **Graph2,** D A B is a simple path, while D A D B is not a simple path

Cycles

A cycle is a simple path with distinct edges, where the first vertex is equal to the last.

Examples:

Cycles in **Graph1:** C A B C, C B A C, A B C A, A C B A, B A C B, B C A B

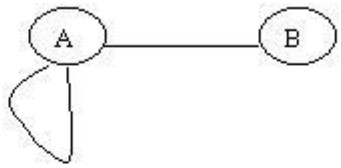
A B A is not a cycle, because the edge A B is the same as B A

Cycles in **Graph3:** A D A, D A B D

A graph without cycles is called **acyclic graph**

Loop

An edge that connects the vertex with itself



Connected graphs

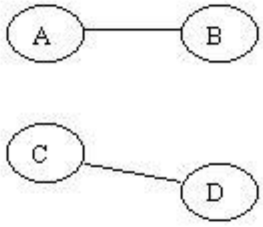
Connected graph: There is a path between each two vertices

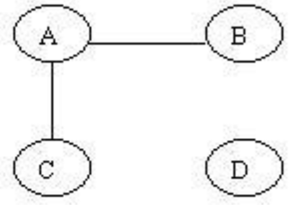
Graph1, Graph2 and **Graph3** are connected graphs.

Disconnected graph: There are at least two vertices not connected by a path.

Examples of disconnected graphs**:**

Graph4		**Graph5**	
Vertices:	A,B,C,D	Vertices:	A,B,C,D
Edges:	AB, CD	Edges:	AB, AC





Isomorphic graphs

Two graphs which contain the same number of graph vertices connected in the same way are said to be isomorphic. Formally, two graphs G and H with graph vertices $V_n = \{1, 2, \ldots, n\}$ are said to be isomorphic if there is a permutation p of V_n such that $\{u, v\}$ is in the set of graph edges $E(G)$ iff $\{p(u), p(v)\}$ is in the set of graph edges $E(H)$.

Two graphs G_1 and G_2 are said to be isomorphic if −

- Their number of components (vertices and edges) are same.
- Their edge connectivity is retained.

Note − In short, out of the two isomorphic graphs, one is a tweaked version of the other. An unlabelled graph also can be thought of as an isomorphic graph.

There exists a function 'f' from vertices of G_1 to vertices of G_2

[f: $V(G_1) \Rightarrow V(G_2)$], such that

Case (i): f is a bijection (both one-one and onto)

Case (ii): f preserves adjacency of vertices, i.e., if the edge $\{U, V\} \in G_1$, then the edge $\{f(U), f(V)\} \in G_2$, then $G_1 \equiv G_2$.

Note

If $G_1 \equiv G_2$ then −

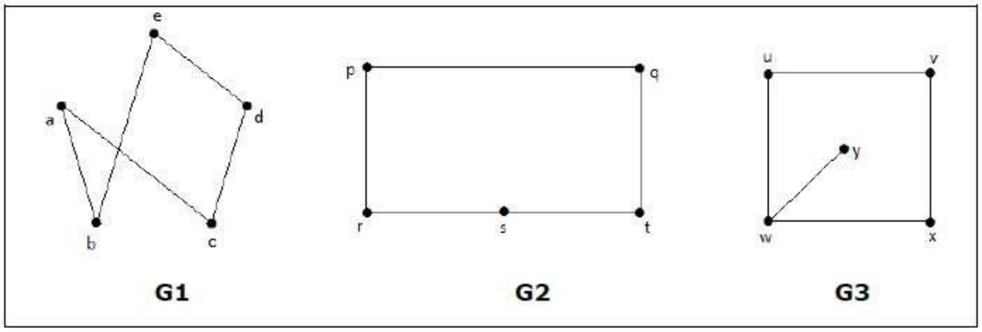
- $|V(G_1)| = |V(G_2)|$
- $|E(G_1)| = |E(G_2)|$
- Degree sequences of G_1 and G_2 are same.
- If the vertices $\{V_1, V_2, .. V_k\}$ form a cycle of length K in G_1, then the vertices $\{f(V_1), f(V_2),\ldots f(V_k)\}$ should form a cycle of length K in G_2.

All the above conditions are necessary for the graphs G_1 and G_2 to be isomorphic, but not sufficient to prove that the graphs are isomorphic.

- $(G_1 \equiv G_2)$ if and only if $(G_1- \equiv G_2-)$ where G_1 and G_2 are simple graphs.
- $(G_1 \equiv G_2)$ if the adjacency matrices of G_1 and G_2 are same.
- $(G_1 \equiv G_2)$ if and only if the corresponding subgraphs of G_1 and G_2(obtained by deleting some vertices in G_1 and their images in graph G_2) are isomorphic.

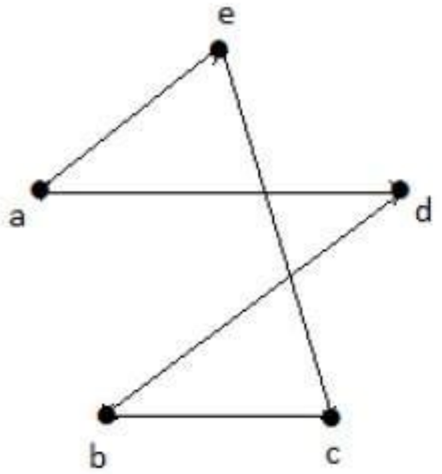
Example

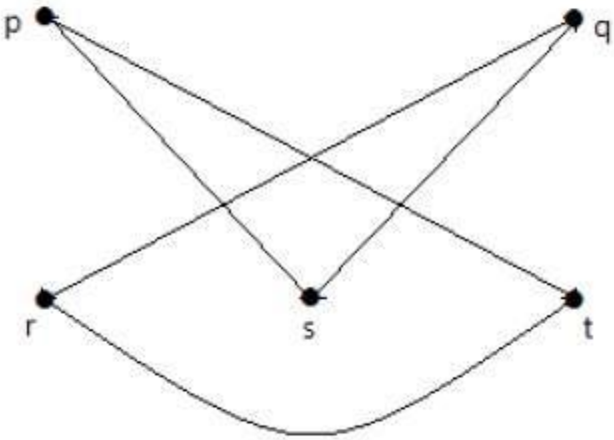
Which of the following graphs are isomorphic?



In the graph G_3, vertex 'w' has only degree 3, whereas all the other graph vertices has degree 2. Hence G_3 not isomorphic to G_1 or G_2.

Taking complements of G_1 and G_2, you have −



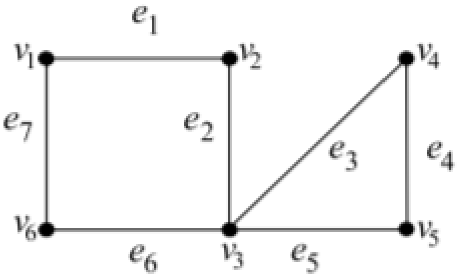


Here, $(G_1- \equiv G_2-)$, hence $(G_1 \equiv G_2)$.

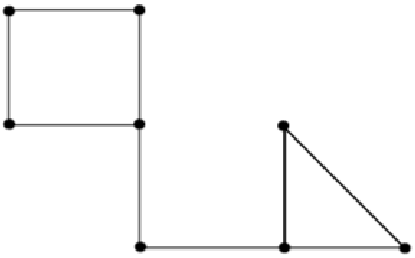
Euler graph

A closed walk in a graph G containing all the edges of G is called an Euler line in G. A graph containing an Euler line is called an Euler graph. We know that a walk is always connected. Since the Euler line (which is a walk) contains all the edges of the graph, an Euler graph is connected except for any isolated vertices the graph may contain. As isolated vertices do not contribute anything to the understanding of an Euler graph, it is assumed now onwards that Euler graphs do not have any isolated vertices and are thus connected.

Example Consider the graph shown in Figure. Clearly, v1 e1 v2 e2 v3 e3 v4 e4 v5 e5 v3 v6 e7 v1 in (a) is an Euler line, whereas the graph shown in (b) is non-Eulerian.

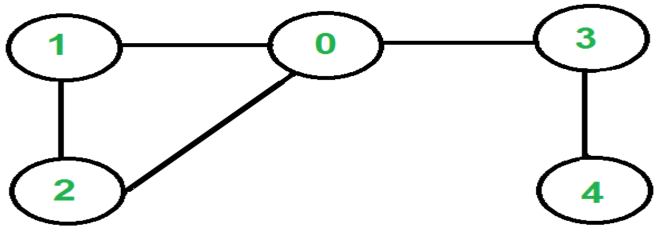


Eulerian graph

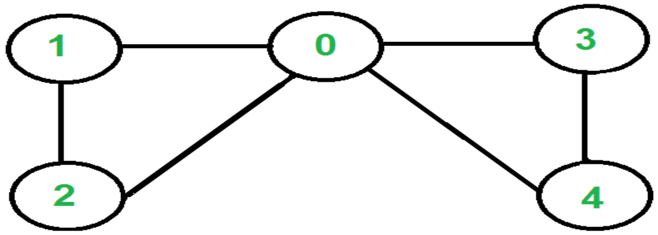


Non- Eulerian graph

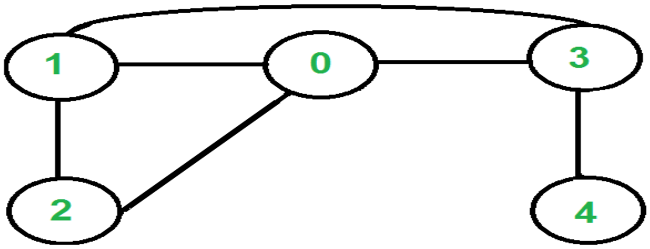
Eulerian Path is a path in graph that visits every edge exactly once. Eulerian Circuit is an Eulerian Path which starts and ends on the same vertex.



The graph has Eulerian Paths, for example "4 3 0 1 2 0", but no Eulerian Cycle. Note that there are two vertices with odd degree (4 and 0)



The graph has Eulerian Cycles, for example "2 1 0 3 4 0 2"
Note that all vertices have even degree



The graph is not Eulerian. Note that there are four vertices with odd degree (0, 1, 3 and 4)

How to find whether a given graph is Eulerian or not?

The problem is same as following question. “Is it possible to draw a given graph without lifting pencil from the paper and without tracing any of the edges more than once”.

A graph is called Eulerian if it has an Eulerian Cycle and called Semi-Eulerian if it has an Eulerian Path. The problem seems similar to Hamiltonian Path which is NP complete problem for a general graph. Fortunately, we can find whether a given graph has a Eulerian Path or not in polynomial time. In fact, we can find it in O(V+E) time.

Following are some interesting properties of undirected graphs with an Eulerian path and cycle. We can use these properties to find whether a graph is Eulerian or not.

Eulerian Cycle

An undirected graph has Eulerian cycle if following two conditions are true.

….a) All vertices with non-zero degree are connected. We don’t care about vertices with zero degree because they don’t belong to Eulerian Cycle or Path (we only consider all edges).

….b) All vertices have even degree.

Eulerian Path

An undirected graph has Eulerian Path if following two conditions are true.

….a) Same as condition (a) for Eulerian Cycle

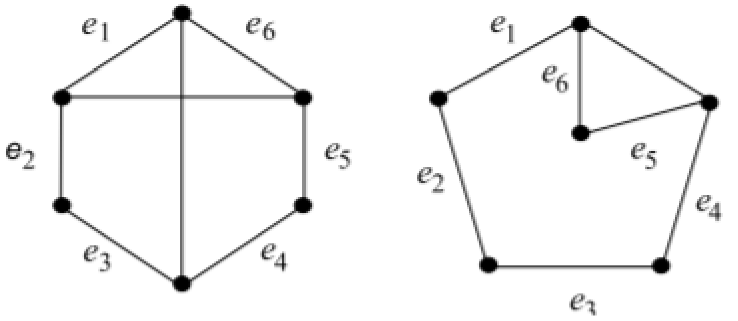
….b) If zero or two vertices have odd degree and all other vertices have even degree. Note that only one vertex with odd degree is not possible in an undirected graph (sum of all degrees is always even in an undirected graph)

Note that a graph with no edges is considered Eulerian because there are no edges to traverse.

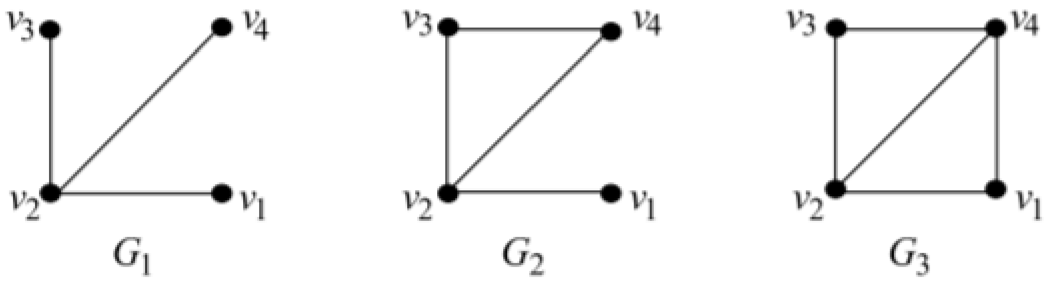
How does this work?
In Eulerian path, each time we visit a vertex v, we walk through two unvisited edges with one end point as v. Therefore, all middle vertices in Eulerian Path must have even degree. For Eulerian Cycle, any vertex can be middle vertex, therefore all vertices must have even degree.

Hamiltonian graphs

A cycle passing through all the vertices of a graph is called a Hamiltonian cycle. A graph containing a Hamiltonian cycle is called a Hamiltonian graph. A path passing through all the vertices of a graph is called a Hamiltonian path and a graph containing a Hamiltonian path is said to be traceable. Examples of Hamiltonian graphs are given in Figure.

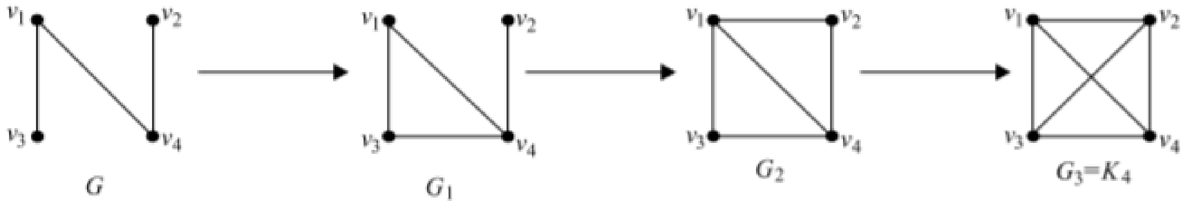


If the last edge of a Hamiltonian cycle is dropped, we get a Hamiltonian path. However, a non-Hamiltonian graph can have a Hamiltonian path, that is, Hamiltonian paths cannot always be used to form Hamiltonian cycles. For example, in Figure, G1 has no Hamiltonian path, and so no Hamiltonian cycle; G2 has the Hamiltonian path v1v2v3v4, but has no Hamiltonian cycle, while G3 has the Hamiltonian cycle v1v2v3v4v1.



A multigraph or general graph is Hamiltonian if and only if its underlying graph is Hamiltonian, because if G is Hamiltonian, then any Hamiltonian cycle in G remains a Hamiltonian cycle in the underlying graph of G. Conversely, if the underlying graph of a graph G is Hamiltonian, then G is also Hamiltonian.

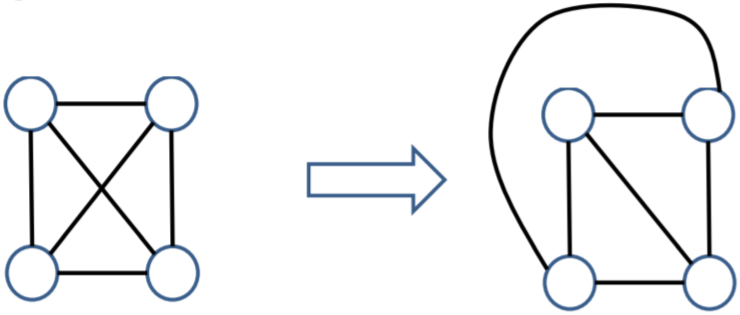
Let G be a graph with n vertices. Clearly, G is a subgraph of the complete graph Kn. From G, we construct step by step supergraphs of G to get Kn, by adding an edge at each step between two vertices that are not already adjacent.



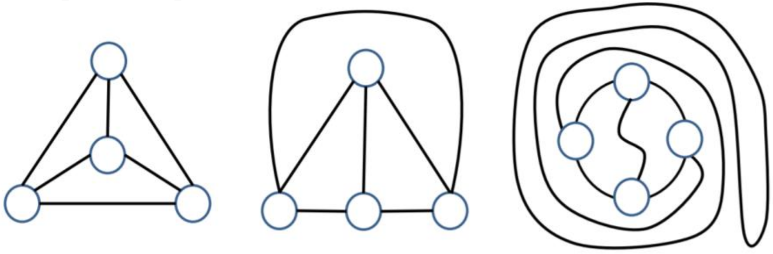
Now, let us start with a graph G which is not Hamiltonian. Since the final outcome of the procedure is the Hamiltonian graph Kn, we change from a non-Hamiltonian graph to a Hamiltonian graph at some stage of the procedure. For example, the non-Hamiltonian graph G1 above is followed by the Hamiltonian graph G2. Since supergraphs of Hamiltonian graphs are Hamiltonian, once a Hamiltonian graph is reached in the procedure, all the subsequent supergraphs are Hamiltonian.

Planar graph

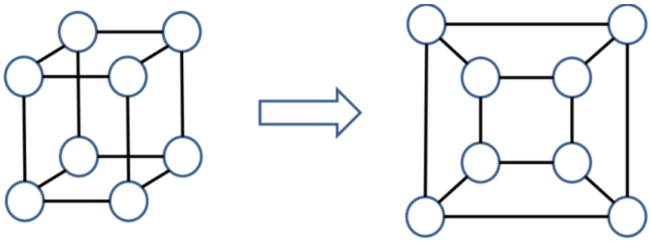
A planar graph is an undirected graph that can be drawn on a plane without any edges crossing. Such a drawing is called a planar representation of the graph in the plane. Ex : K4 is a planar graph



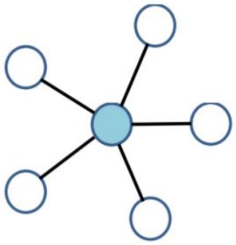
Other planar representations of K4



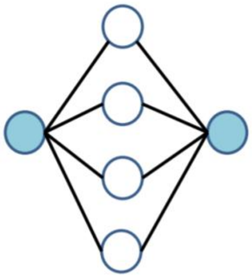
Q3 is a planar graph



K1,n and K2,n are planar graphs for all n



$K_{1,5}$

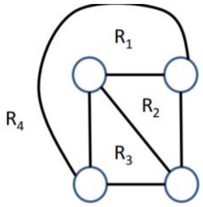


$K_{2,4}$

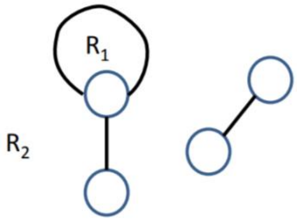
Euler's Planar Formula

Definition : A planar representation of a graph splits the plane into regions, where one of them has infinite area and is called the infinite region.

Ex :



4 regions



2 regions

Euler's Planar Formula

Let G be a connected planar graph, and consider a planar representation of G.

Let V = # vertices, E = # edges, F = # regions.

Theorem : V + F = E + 2.

Graph Coloring

Graph coloring is the procedure of assignment of colors to each vertex of a graph G such that no adjacent vertices get same color. The objective is to minimize the number of colors while coloring a graph. The smallest number of colors required to color a graph G is called its chromatic number of that graph. Graph coloring problem is a NP Complete problem.

Method to Color a Graph

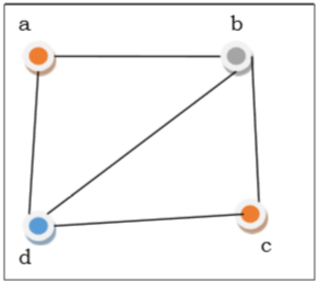
The steps required to color a graph G with n number of vertices are as follows −

Step 1 − Arrange the vertices of the graph in some order.

Step 2 − Choose the first vertex and color it with the first color.

Step 3 − Choose the next vertex and color it with the lowest numbered color that has not been colored on any vertices adjacent to it. If all the adjacent vertices are colored with this color, assign a new color to it. Repeat this step until all the vertices are colored.

Example



In the above figure, at first vertex aa is colored red. As the adjacent vertices of vertex a are again adjacent, vertex bb and vertex dd are colored with different color, green and blue respectively. Then vertex cc is colored as red as no adjacent vertex of cc is colored red. Hence, we could color the graph by 3 colors. Hence, the chromatic number of the graph is 3.

Applications of Graph Coloring

Some applications of graph coloring include −

- Register Allocation
- Map Coloring
- Bipartite Graph Checking
- Mobile Radio Frequency Assignment
- Making time table, etc.

Graph Traversal

Graph traversal is the problem of visiting all the vertices of a graph in some systematic order. There are mainly two ways to traverse a graph.

- Breadth First Search
- Depth First Search

Breadth First Search

Breadth First Search (BFS) starts at starting level-0 vertex XX of the graph GG. Then we visit all the vertices that are the neighbors of XX. After visiting, we mark the vertices as "visited," and place them into level-1. Then we start from the level-1 vertices and apply the same method on every level-1 vertex and so on. The BFS traversal terminates when every vertex of the graph has been visited.

BFS Algorithm

The concept is to visit all the neighbor vertices before visiting other neighbor vertices of neighbor vertices.

- Initialize status of all nodes as “Ready”.
- Put source vertex in a queue and change its status to “Waiting”.

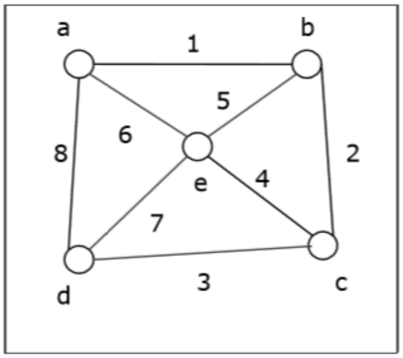
- Repeat the following two steps until queue is empty −

o Remove the first vertex from the queue and mark it as “Visited”.

o Add to the rear of queue all neighbors of the removed vertex whose status is “Ready”. Mark their status as “Waiting”.

Problem

Let us take a graph (Source vertex is ‘a’) and apply the BFS algorithm to find out the traversal order.



Solution −

- Initialize status of all vertices to “Ready”.

- Put *a* in queue and change its status to “Waiting”.

- Remove *a* from queue, mark it as “Visited”.

- Add *a*’s neighbors in “Ready” state *b, d* and *e* to end of queue and mark them as “Waiting”.

- Remove *b* from queue, mark it as “Visited”, put its “Ready” neighbor *c*at end of queue and mark *c* as “Waiting”.

- Remove *d* from queue and mark it as “Visited”. It has no neighbor in “Ready” state.

- Remove *e* from queue and mark it as “Visited”. It has no neighbor in “Ready” state.

- Remove *c* from queue and mark it as “Visited”. It has no neighbor in “Ready” state.

- Queue is empty so stop.

So the traversal order is −
a→b→d→e→ca→b→d→e→c
The alternate orders of traversal are −
a→b→e→d→ca→b→e→d→c
Or, a→d→b→e→ca→d→b→e→c
Or, a→e→b→d→ca→e→b→d→c
Or, a→b→e→d→ca→b→e→d→c

Or, a→d→e→b→ca→d→e→b→c

Application of BFS

- Finding the shortest path
- Minimum spanning tree for un-weighted graph
- GPS navigation system
- Detecting cycles in an undirected graph
- Finding all nodes within one connected component

Complexity Analysis

Let $G(V,E)G(V,E)$ be a graph with $|V||V|$ number of vertices and $|E||E|$ number of edges. If breadth first search algorithm visits every vertex in the graph and checks every edge, then its time complexity would be −

$O(|V|+|E|).O(|E|)O(|V|+|E|).O(|E|)$

It may vary between $O(1)O(1)$ and $O(|V2|)O(|V2|)$

Depth First Search

Depth First Search (DFS) algorithm starts from a vertex vv, then it traverses to its adjacent vertex (say x) that has not been visited before and marks as "visited" and goes on with the adjacent vertex of xx and so on.

If at any vertex, it encounters that all the adjacent vertices are visited, then it backtracks until it finds the first vertex having an adjacent vertex that has not been traversed before. Then, it traverses that vertex, continues with its adjacent vertices until it traverses all visited vertices and has to backtrack again. In this way, it will traverse all the vertices reachable from the initial vertex vv.

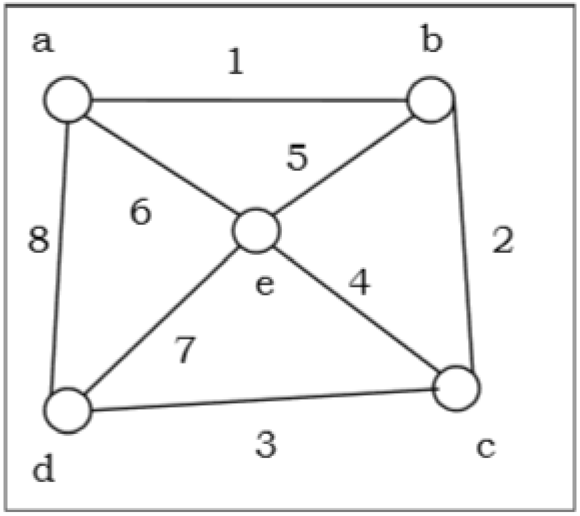
DFS Algorithm

The concept is to visit all the neighbor vertices of a neighbor vertex before visiting the other neighbor vertices.

- Initialize status of all nodes as “Ready”
- Put source vertex in a stack and change its status to “Waiting”
- Repeat the following two steps until stack is empty −
    - Pop the top vertex from the stack and mark it as “Visited”
    - Push onto the top of the stack all neighbors of the removed vertex whose status is “Ready”. Mark their status as “Waiting”.

Problem

Let us take a graph (Source vertex is ‘a’) and apply the DFS algorithm to find out the traversal order.



Solution

- Initialize status of all vertices to “Ready”.
- Push *a* in stack and change its status to “Waiting”.
- Pop *a* and mark it as “Visited”.
- Push *a*’s neighbors in “Ready” state *e, d* and *b* to top of stack and mark them as “Waiting”.
- Pop *b* from stack, mark it as “Visited”, push its “Ready” neighbor *c*onto stack.
- Pop *c* from stack and mark it as “Visited”. It has no “Ready” neighbor.
- Pop *d* from stack and mark it as “Visited”. It has no “Ready” neighbor.
- Pop *e* from stack and mark it as “Visited”. It has no “Ready” neighbor.
- Stack is empty. So stop.

So the traversal order is −
a→b→c→d→ea→b→c→d→e
The alternate orders of traversal are −
a→e→b→c→da→e→b→c→d
Or, a→b→e→c→da→b→e→c→d
Or, a→d→e→b→ca→d→e→b→c
Or, a→d→c→e→ba→d→c→e→b
Or, a→d→c→b→ea→d→c→b→e

Complexity Analysis

Let G(V,E)G(V,E) be a graph with |V||V| number of vertices and |E||E| number of edges. If DFS algorithm visits every vertex in the graph and checks every edge, then the time complexity is −
⊝(|V|+|E|)⊝(|V|+|E|)

Applications

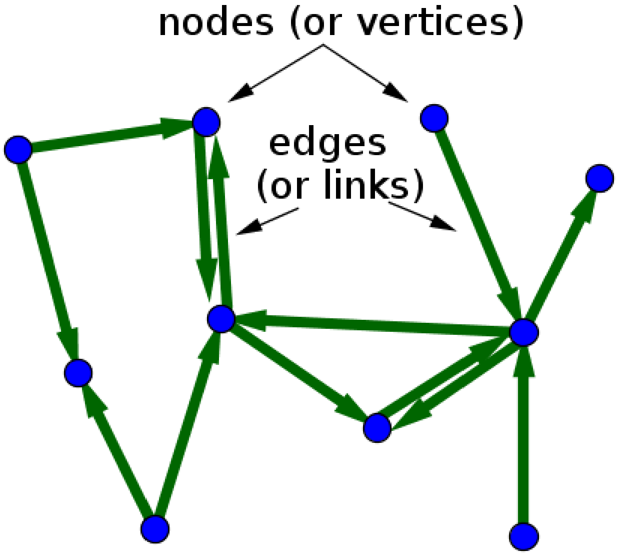
- Detecting cycle in a graph
- To find topological sorting
- To test if a graph is bipartite
- Finding connected components
- Finding the bridges of a graph
- Finding bi-connectivity in graphs
- Solving the Knight's Tour problem
- Solving puzzles with only one solution

Digraphs

A graph in which each graph edge is replaced by a directed graph edge, also called a digraph. A directed graph having no multiple edges or loops (corresponding to a binary adjacency matrix with 0s on the diagonal) is called a simple directed graph. A complete graph in which each edge is bidirected is called a complete directed graph. A directed graph having no symmetric pair of directed edges (i.e., no bidirected edges) is called an oriented graph. A complete oriented graph (i.e., a directed graph in which each pair of nodes is joined by a single edge having a unique direction) is called a tournament.

If G is an undirected connected graph, then one can always direct the circuit graph edges of G and leave the separating edges undirected so that there is a directed path from any node to another. Such a graph is said to be transitive if the adjacency relation is transitive.

When drawing a directed graph, the edges are typically drawn as arrows indicating the direction, as illustrated in the following figure.



A directed graph with 10 vertices (or nodes) and 13 edges

One can formally define a directed graph as G=(N,E)G=(N,E), consisting of the set NN of nodes and the set EE of edges, which are ordered pairs of elements of NN.

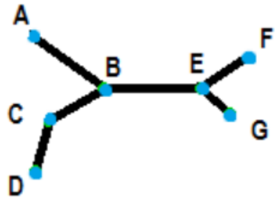
Directed acyclic graphs

Directed acyclic graphs (DAGs) are used to model probabilities, connectivity, and causality. A “graph” in this sense means a structure made from nodes and edges.

- **Nodes** are usually denoted by circles or ovals (although technically they can be any shape of your choosing).

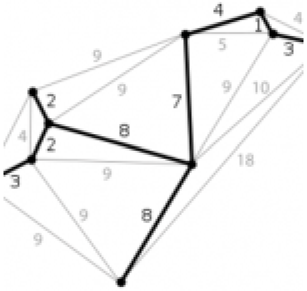
- **Edges** are the connections between the nodes. An edge connects two nodes. They are usually represented by lines, or lines with arrows.

DAGs are based on basic acyclic graphs.



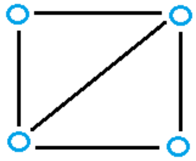
A tree with nodes A B C D E F and G.

An acyclic graph is a graph without cycles (a cycle is a complete circuit). When following the graph from node to node, you will never visit the same node twice.



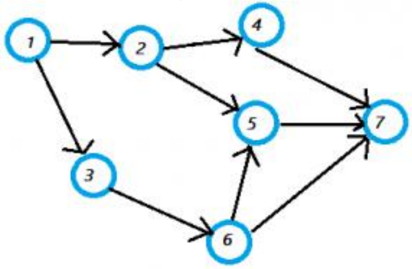
This graph (the thick black line) is acyclic, as it has no cycles (complete circuits).

A connected acyclic graph, like the one above, is called a **tree**. If one or more of the tree “branches” is disconnected, the acyclic graph is a called a **forest**.



This graph has a complete circuit and so is not acyclic.

A directed acyclic graph is an acyclic graph that has a direction as well as a lack of cycles.



The parts of the above graph are:

- **Integer** = the set for the the Vertices.
- **Vertices set** = {1,2,3,4,5,6,7}.
- **Edge set** = {(1,2), (1,3), (2,4), (2,5), (3,6), (4,7), (5,7), (6,7)}.

A directed acyclic graph has a **topological ordering**. This means that the nodes are ordered so that the starting node has a lower value than the ending node. A DAG has a unique topological ordering if it has a directed path containing all the nodes; in this case the ordering is the same as the order in which the nodes appear in the path.

In computer science, DAGs are also called *wait-for-graphs*. When a DAG is used to detect a deadlock, it illustrates that a resources has to *wait for* another process to continue.

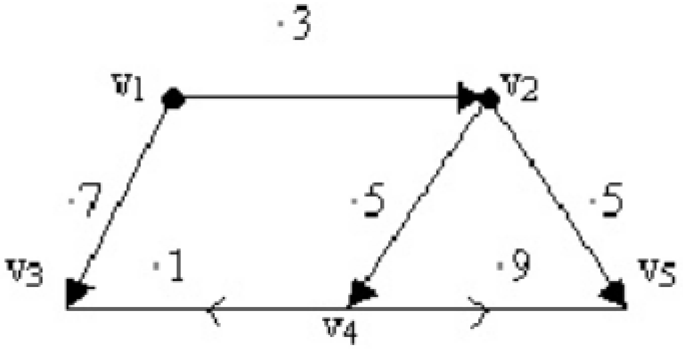
Weighted digraphs

We can assign numbers to the edges or vertices of a graph in order to enable them to be used in physical problems. Such an assignment is called the weight of the edges or vertices.

Weighted graphs are defined as the quadruples (V, E, f, g) or the triplets (V, E, f)or the triplets (V, E, g), where V is the set of vertices, E is the set of domains, f is the function with domain V, which assigns weights to vertices and g is the function with domain E, which assigns weights to edges

Example

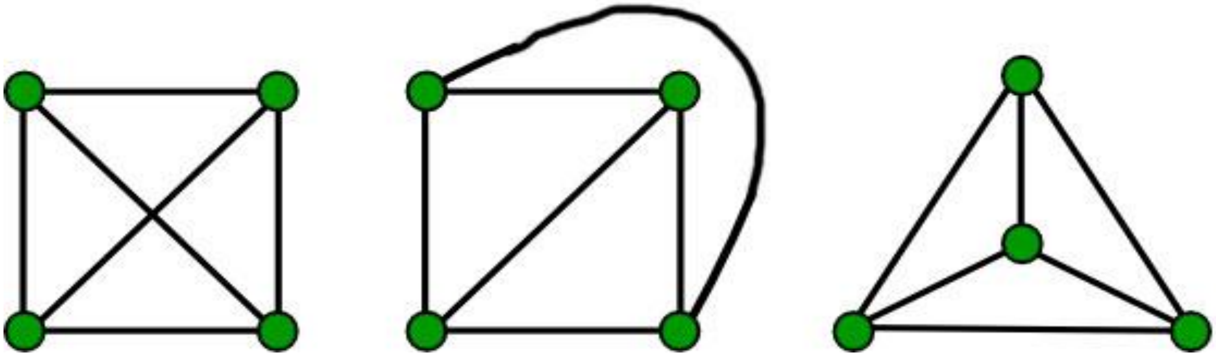
Following diagram is a weighted digraph which represents the communication network among five individuals v1,v2,v3,v4,v5. The number assigned for each directed edge gives the probability of their communication.



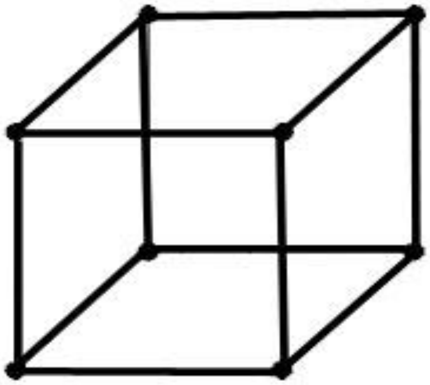
Planarity – “A graph is said to be planar if it can be drawn on a plane **without any edges crossing**. Such a drawing is called a planar representation of the graph.”

Important Note – A graph may be planar even if it is drawn with crossings, because it may be possible to draw it in a different way without crossings.

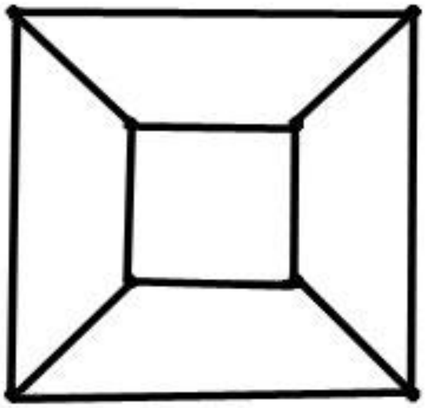
For example consider the complete graph k4 and its two possible planar representations –



- **Example** – Is the hypercube Q3 planar?

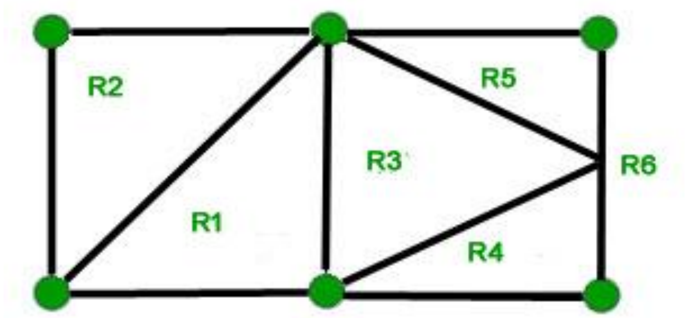


- **Solution** – Yes, Q3 is planar. Its planar representation-



Regions in Planar Graphs –

The planar representation of a graph splits the plane into **regions**. These regions are bounded by the edges except for one region that is unbounded. For example, consider the following graph ”



There are a total of 6 regions with 5 bounded regions and 1 bounded region R6. All the planar representations of a graph split the plane in the same number of regions. Euler found out the number of regions in a planar graph as a function of the number of vertices and number of edges in the graph.

Theorem – "Let G be a **connected simple planar graph** with e edges and v vertices. Then the number of regions r in the graph is equal to **e-v+2**."

- **Example –** What is the number of regions in a connected planar simple graph with 20 vertices each with a degree of 3?

- **Solution –** Sum of degrees of edges = 20 * 3 = 60. By handshaking theorem, 2e=60 which gives e=30 .
By Euler's theorem, the number of regions = e-v+2 which gives 12 regions.

An important result obtained by Euler's formula is the following inequality –

Note –"If G is a connected planar graph with e edges and v vertices, where v>=3, then **e<=ev-6**. Also G cannot have a vertex of degree exceeding 5."

- **Example –** Is the graph K5 planar?

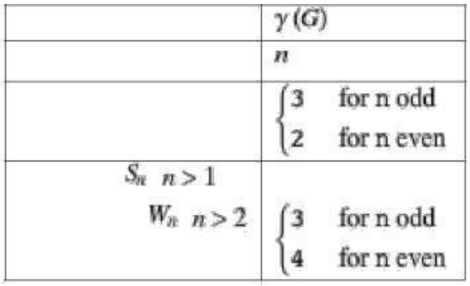
- **Solution –** Number of vertices and edges in K5 is 5 and 10 respectively. Since 10 > 3*5 – 6, 10 > 9 the inequality e<=3v-6 is not satisfied. Thus the graph is not planar.

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.

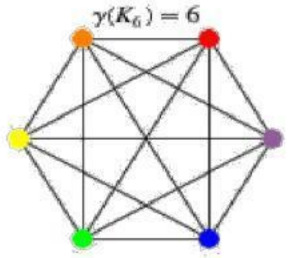
Vertex coloring is the starting point of the subject, and other coloring problems can be transformed into a vertex version. For example, an edge coloring of a graph is just a vertex coloring of its line graph, and a face coloring of a planar graph is just a vertex coloring of its planar dual. However,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

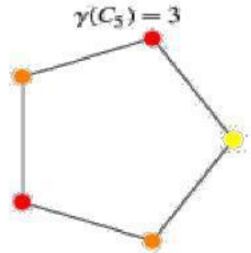
The convention of using colors originates from coloring the countries of a map, where each face is literally colored. This was generalized to coloring the faces of a graph embedded in the plane. By planar duality it became coloring the vertices, and in this form it generalizes to all graphs. In mathematical 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.

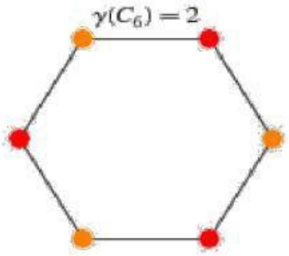
Graph coloring enjoys many practical applications as well as theoretical challenges. Beside the classical types of problems, different limitations can also be set on the graph, or on the way a color is assigned, or even on the color itself. It has even reached popularity with the general public in the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

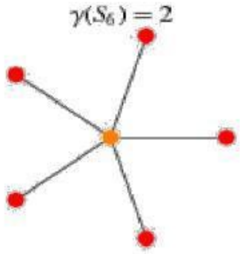


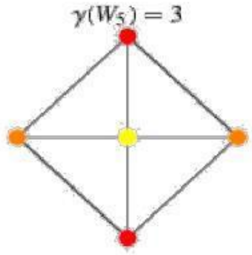
	$\gamma(G)$
	n
	$\begin{cases} 3 & \text{for n odd} \\ 2 & \text{for n even} \end{cases}$
S_n $n > 1$	
W_n $n > 2$	$\begin{cases} 3 & \text{for n odd} \\ 4 & \text{for n even} \end{cases}$

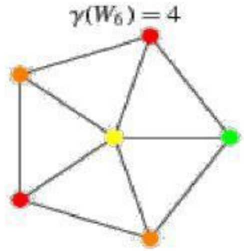


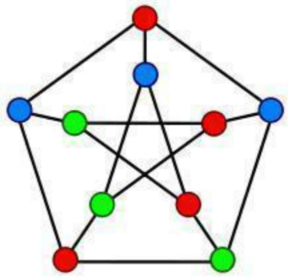












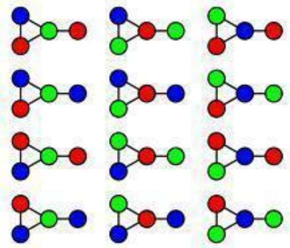
A proper vertex coloring of the Petersen graph with 3 colors, the minimum number possible

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.

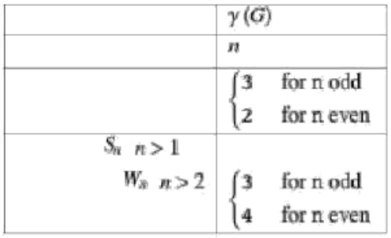
The terminology of using *colors* for vertex labels goes back to map coloring. Labels like *red* and *blue* are only used when the number of colors is small, and normally it is understood that the labels are drawn from the integers {1,2,3,...}.

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.

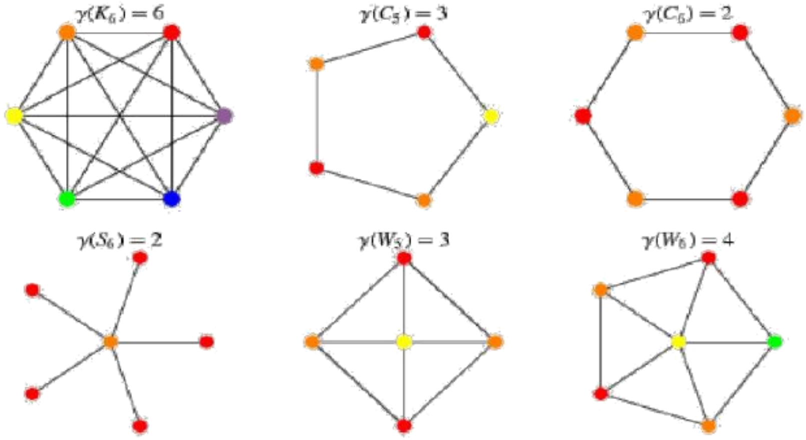


This graph can be 3-colored in 12 different ways.

The following table gives the chromatic number for familiar classes of graphs.



	$\gamma(G)$
	n
	$\begin{cases} 3 & \text{for n odd} \\ 2 & \text{for n even} \end{cases}$
S_n $n>1$	
W_n $n>2$	$\begin{cases} 3 & \text{for n odd} \\ 4 & \text{for n even} \end{cases}$



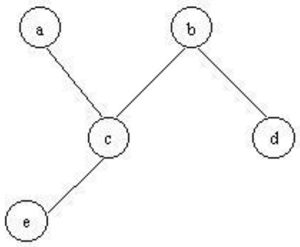
Trees

A tree is an undirected graph with no cycles and a vertex chosen to be the root of the tree.

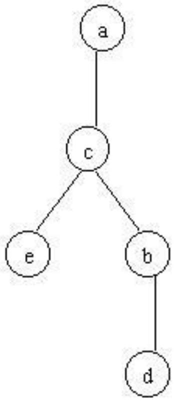
Note: in a tree, when we choose a root we impose an orientation. Given an acyclic graph, we may choose any node to be the root of a tree.

Example

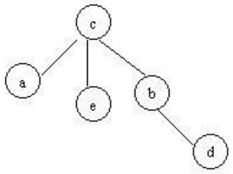
Acyclic graph:



If we choose vertex **a** to be the root, then we get the following tree:



If we choose **c** to be the root, then the tree will be:



Note also, that the graph does not specify the order of the children of a given node.

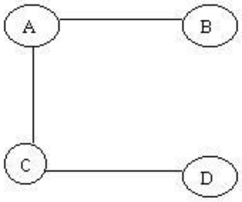
A spanning tree of a graph

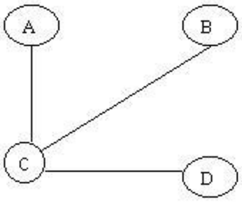
A spanning tree of an undirected graph is a subgraph that contains all the vertices, and no cycles. If we add any edge to the spanning tree, it forms a cycle, and the tree becomes a graph.

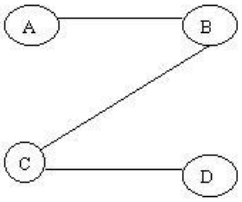
It is possible to define a spanning tree for directed graphs, however the definition is rather complicated and will not be discussed here.

Example:

Spanning trees for ***Graph1***:





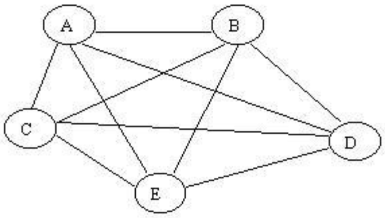


A tree with **N** vertices has **N-1** edges.
A graph with less than **N-1** edges is not connected.

Complete graphs

Graphs with all edges present – each vertex is connected to all other vertices, are called complete graphs.

Example:



Dense graphs: relatively few of the possible edges are missing
Sparse graphs: relatively few of the possible edges are present

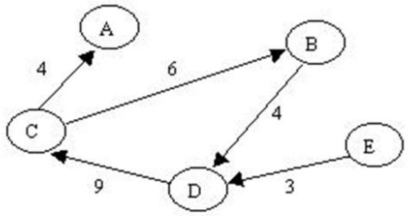
Weighted graphs

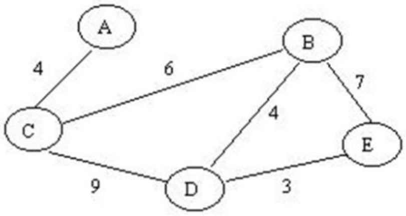
Weights are assigned to each edge (e.g. distances in a road map)

Networks - directed weighted graphs

Note: Some textbooks define networks to be undirected weighted graphs as well.

Examples:





Graph representation

a. Adjacency matrix

Vertices: A,B,C,D

Edges: AB, AC, BD, CD

	A	B	C	D
A	0	1	1	0
B	1	0	0	1
C	1	0	0	1
D	0	1	1	0

Note: Depending on the application sometimes the diagonal is set to 1

For not directed graphs the matrix is symmetrical. Depending on the application only half of the matrix may be used.

b. Adjacency list structure

Each vertex is associated with a list, that holds all adjacent vertices.

A: B, C

B: A, D

C: A, D

D: B, C

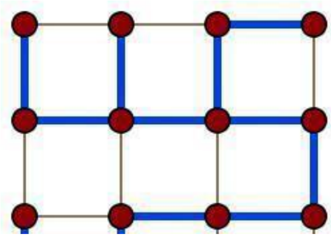
Other information, as weights, labels, names, can be represented using auxiliary arrays.

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.

Example:



A spanning tree (blue heavy edges) of a grid graph

Spanning forests

A **spanning forest** is a type of subgraph that generalises the concept of a spanning tree. However, there are two definitions in common use. One is that a spanning forest is a subgraph that consists of a spanning tree in each connected component of a graph. (Equivalently, it is a maximal cycle-free subgraph.) This definition is common in computer science and optimisation. It is also the definition used when discussing minimum spanning forests, the generalization to disconnected graphs of minimum spanning trees. Another definition, common in graph theory, is that a spanning forest is any subgraph that is both a forest (contains no cycles) and spanning (includes every vertex).

There are a few general properties of spanning trees.

1. A connected graph can have more than one spanning tree. They can have as many as where is the number of vertices in the graph.

2. All possible spanning trees for a graph G have the same number of edges and vertices.

3. Spanning trees do not have any cycles.

4. Spanning trees are all minimally connected. That is, if any one edge is removed, the spanning tree will no longer be connected.

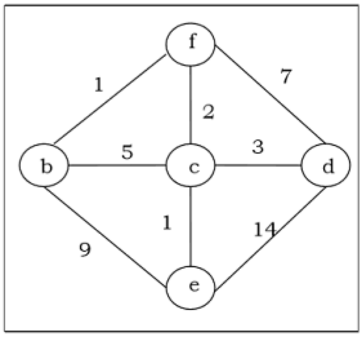
5. Adding any edge to the spanning tree will create a cycle. So, a spanning tree is maximally acyclic.

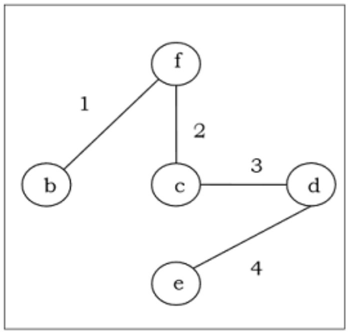
6. Spanning trees have |n|-1 edges, where |n| is the number of vertices

Minimum Spanning Tree

A spanning tree with assigned weight less than or equal to the weight of every possible spanning tree of a weighted, connected and undirected graph GG, it is called minimum spanning tree (MST). The weight of a spanning tree is the sum of all the weights assigned to each edge of the spanning tree.

Example





Kruskal's Algorithm

Kruskal's algorithm is a greedy algorithm that finds a minimum spanning tree for a connected weighted graph. It finds a tree of that graph which includes every vertex and the total weight of all the edges in the tree is less than or equal to every possible spanning tree.

Algorithm

Step 1 − Arrange all the edges of the given graph G(V,E)G(V,E) in non-decreasing order as per their edge weight.

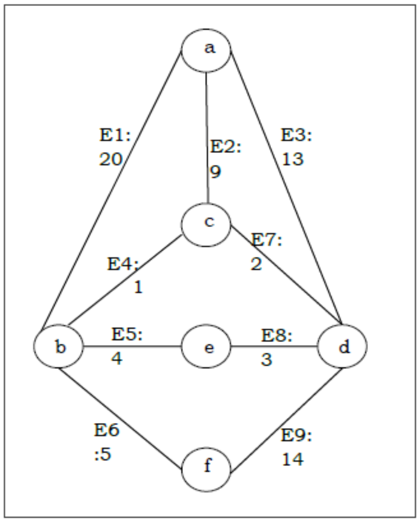
Step 2 − Choose the smallest weighted edge from the graph and check if it forms a cycle with the spanning tree formed so far.

Step 3 − If there is no cycle, include this edge to the spanning tree else discard it.

Step 4 − Repeat Step 2 and Step 3 until (V−1)(V−1) number of edges are left in the spanning tree.

Problem

Suppose we want to find minimum spanning tree for the following graph G using Kruskal’s algorithm.



Solution

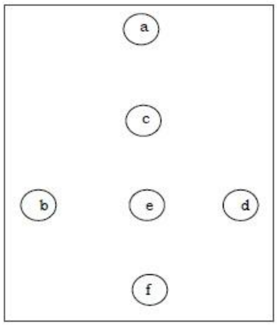
From the above graph we construct the following table −

Edge No.	Vertex Pair	Edge Weight
E1	(a, b)	20
E2	(a, c)	9
E3	(a, d)	13

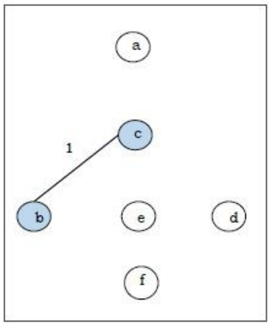
E4	(b, c)	1
E5	(b, e)	4
E6	(b, f)	5
E7	(c, d)	2
E8	(d, e)	3
E9	(d, f)	14

Now we will rearrange the table in ascending order with respect to Edge weight −

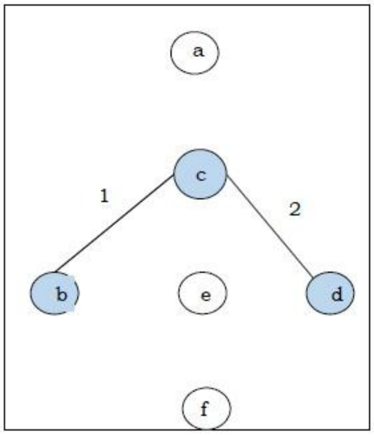
Edge No.	Vertex Pair	Edge Weight
E4	(b, c)	1
E7	(c, d)	2
E8	(d, e)	3
E5	(b, e)	4
E6	(b, f)	5
E2	(a, c)	9
E3	(a, d)	13
E9	(d, f)	14
E1	(a, b)	20



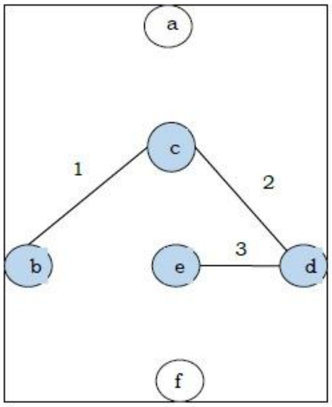
After adding vertices



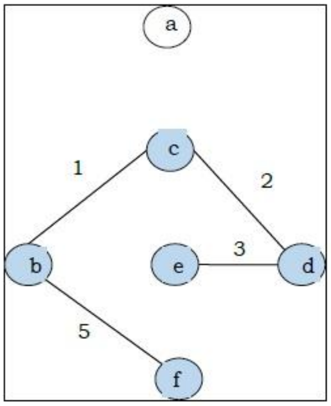
After adding edge E4



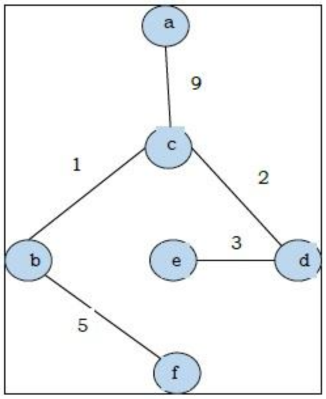
After adding edge E7



After adding edge E8



After adding edge E6
(don't add E5 since it forms cycle)



After adding edge E2

Since we got all the 5 edges in the last figure, we stop the algorithm and this is the minimal spanning tree and its total weight is (1+2+3+5+9)=20(1+2+3+5+9)=20.

Prim's Algorithm

Prim's algorithm, discovered in 1930 by mathematicians, Vojtech Jarnik and Robert C. Prim, is a greedy algorithm that finds a minimum spanning tree for a connected weighted graph. It finds a tree of that graph which includes every vertex and the total weight of all the edges in the tree is less than or equal to every possible spanning tree. Prim's algorithm is faster on dense graphs.

Algorithm

- Initialize the minimal spanning tree with a single vertex, randomly chosen from the graph.

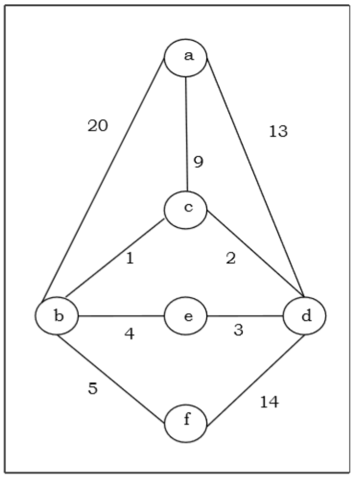
- Repeat steps 3 and 4 until all the vertices are included in the tree.

- Select an edge that connects the tree with a vertex not yet in the tree, so that the weight of the edge is minimal and inclusion of the edge does not form a cycle.

- Add the selected edge and the vertex that it connects to the tree.

Problem

Suppose we want to find minimum spanning tree for the following graph G using Prim's algorithm.



Solution

Here we start with the vertex 'a' and proceed.