Bachelor of Computer Applications

(BCA)

Discrete Mathematics

(DBCACO202T24)

Self-Learning Material

(SEM II)

Jaipur National University
Centre for Distance and Online Education

Established by Government of Rajasthan

Approved by UGC under Sec 2(f) of UGC ACT 1956

&

NAAC A+ Accredited

TABLE OF CONTENT

Discrete Mathematics

EXPERT COMMITTEE

Prof. Sunil gupta
(School of Computer and Systems Sciences)

Dr. Satish Pandey
(School of Computer and Systems Sciences)

Dr. Yogesh Chandra Sharma
(Dept. of Basic Sciences JNU, Jaipur)

COURSE COORDINATOR

Mr. Praveen Kumar
(Dept. of Basic Sciences
JNU, Jaipur)

UNIT PREPARATION

Unit Writers	Assisting & Prof Reading	Unit Editor
Mr. Praveen Kumar (Department of Basic Sciences JNU, Jaipur) (Unit 1 - 4)	Prof. (Dr.) Hoshiyar Singh Dept. of Basic Sciences JNU, Jaipur	Dr. Yogesh Khandelwal Dept. of Basic Sciences JN Jaipur
Dr. Yogesh Khandelwal (Department of Basic Sciences JNU, Jaipur) (Unit 5 - 11)		

Secretarial Assistance:

Mr. Mukesh Sharma,PA

COURSE INTRODUCTION

Discrete Mathematics is a foundational course that introduces students to the concepts and techniques essential for understanding and solving problems in mathematics and computer science. Unlike continuous mathematics, which deals with real numbers and continuous functions, discrete mathematics focuses on countable, distinct elements. This course covers a range of topics, including logic, set theory, combinatorics, graph theory, and algorithms, providing the mathematical underpinning for many areas in computer science and related disciplines.

The course is divided into 11 units. Each Unit is divided into sub topics. This course is ideal for students pursuing degrees in computer science, mathematics, engineering, or related fields. It is also suitable for anyone interested in developing strong logical and problem-solving skills. The course provides essential knowledge for understanding the theoretical foundations of computer science and for developing algorithms and data structures.

There are sections and subsections inside each unit. Each unit starts with a statement of objectives that outlines the goals we hope you will accomplish. Every segment of the unit has many tasks that you need to complete. We wish you pleasure in the Course.

Course Outcomes:

1. Simplify and evaluate basic logic statements including compound statements, implications, inverses, converses, and contrapositives using truth tables and the properties of logic.
2. Express a logic sentence in terms of predicates, quantifiers, and logical connectives
3. Apply the operations of sets and use Venn diagrams to solve applied problems; solve problems using the principle of inclusion-exclusion.
4. Determine the domain and range of a discrete or non-discrete function, graph functions, identify one-to-one functions, perform the composition of functions, find and/or graph the inverse of a function, and apply the properties of functions to application problems.
5. Verify that a simple program segment with given initial and final assertions is correct using the rule of inference for verification of partial correctness and loop invariants.

Acknowledgements:

The content we have utilized is solely educational in nature. The copyright proprietors of the materials reproduced in this book have been tracked down as much as possible. The editors apologize for any violation that may have happened, and they will be happy to rectify any such material in later versions of this book.

UNIT - 1

Introductions to Sets

Learning objectives

- Understanding Basic Algebraic Structures, gain familiarity with fundamental algebraic structures such as groups, rings, and fields. Learn their definitions, properties, and examples.

Structure

1.1 Sets and Equivalence Relations

Set theory is a stream of mathematical logic that study sets, which are collection of items. It's a primary theory in mathematics, providing a basis for various other fields. Here's a concise overview of key concepts in set theory:

Basic Concepts

1. **Set**
2. **Element**
3. **Subset**
4. **Power Set**
5. **Union**

Examples of Sets

Some standard sets in mathematics are:

- Set of natural numbers, $\mathbb{N} = \{1, 2, 3, ...\}$
- Set of whole numbers, $\mathbb{W} = \{0, 1, 2, 3, ...\}$
- Set of integers, $\mathbb{Z} = \{..., -3, -2, -1, 0, 1, 2, 3, ...\}$
- Set of rational numbers, $\mathbb{Q} = \{p/q \mid q \text{ is an integer and } q \neq 0\}$
- Set of irrational numbers, $\mathbb{Q}' = \{x \mid x \text{ is not rational}\}$
- Set of real numbers, $\mathbb{R} = \mathbb{Q} \cup \mathbb{Q}'$

All these are infinite sets. But there can be finite sets as well.

A = {2, 4, 6, 8} is an EX of a finite set that may be used to represent the collection of even natural numbers smaller than 10.

Parts of a Set

Either elements or members of a set are the objects that make up the set. Curly brackets encompass the components of a set, which are separated by commas. '∈' is the symbol indicates that an element is contained in the set.

Ex:- A = {2, 4, 6, 8}

Here, 2 ∈ A, 4 ∈ A, 6 ∈ A, 8 ∈ A .

The sign '∉' is indicate an element that is not a part of the set.

Ex: $3 \notin A$. i.e. 3 is not a part of the set A.

Cardinality (cardinal number or Number of element) of a Set:

A set's cardinal number, cardinality, or order indicates how many items there are in total. For natural even integers n (A) = 4 that are smaller than 10. A collection of distinct elements is referred to as a set. All of a set's elements must be connected to one another and possess a common property in order for a set to be defined. For instance, if we establish a set whose members are the names of the months in a year, we may state that the months themselves make up every element of the set.

1.2 Representation of the Sets:

There are different set notations are used in set theory to represent sets. Each of them has a separate set of components. There are three set notations used to represent sets:

- Roster form
- Set builder form

Let us understand each of these forms with an EX.

Set of first five even natural numbers		
Semantic Form	Roster Form	Set Builder Form
A set of first five even natural numbers	{2, 4, 6, 8, 10}	{x ∈ N | x ≤ 10 and x is even}

Table 1.1 : Representation of Sets

Venn diagram for the Visual Representation of Sets

A Venn diagram is a depiction of sets in which a circle represents each set. The elements of a set are included within each circle. There are instances where a rectangle encircled by circles is used to symbolize the universal set. The relationships between the given sets are depicted in the Venn diagram.

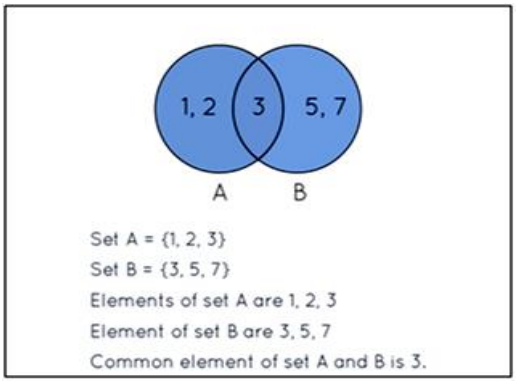


Figure 1.1 : Venn diagram

Sets Symbols

The components of a particular set are denoted by set symbols. The set theory symbols and their meanings are displayed in the following table.

Symbols	Meaning
{ }	Symbol of set
U	Universal set
n(X)	Cardinal number of set X
b ∈ A	'b' is an element of set A
a ∉ B	'a' is not an element of set B
∅	Null or empty set
A U B	Set A union set B
A ∩ B	Set A intersection set B
A ⊆ B	Set A is a subset of set B
B ⊇ A	Set B is the superset of set A

Table 1.2 : Sets Symbols

Types of Sets

A set in mathematics is a group of unique items that are each regarded as a separate entity. In mathematics, sets are basic objects. Here are various types of sets with brief explanations:

1. **Countable (Finite) Set**: A set with a limited number of elements.

   Ex: {1,2,3,4,5}

2. **Uncountable (Infinite) Set**: A set having infinite number of elements.

   Ex: {0, 1,2,3,…}

3. **Void Set :**A having no elements.

   Denoted as " $\emptyset$ or {}".

4. **Singleton Set**: "A If a set having only one element is called a singleton set".

   Ex: {a}

5. **Subset**: If A and B are two sets then the "set B is called the subset of set A if every element of set B contained in set A".

   Ex: If $A = \{1,2,3,4\}$ and $B = \{3,4\}$, then B is a subset of A.

6. **Power Set**: It is the set of all subsets of a particular set.

   Ex: A={1,2,@}, then power set of A is $\{\emptyset, \{1\}, \{2\}\{@\}, \{1,2\}, \{2, @\}, \{1,2, @\}\}$

7. **Universal Set**: The set that contains all the objects under consideration, usually denoted by U.

   Ex: If we are considering all natural numbers, then U could be the set of all natural numbers.

8. **Complement of a Set**: The set of elements that belong to U but are not part of A is known as the complement of A, assuming that U is the universal set and A is a subset.

   Ex: If U = $\{1,2,3,4,5,6\}$ and $A = \{1,2\}$, then complement of A is $\{3,4,5,6\}$.

9. **Union of Sets**: A set containing all elements of the given sets.

   Notation: A∪B

   Ex: If A={444 , 222} and B={12,34}, then A∪B={444,222,12,34}.

10. **Intersection of two Sets**: A set containing only the common elements of the given sets.

    Notation: A∩B

    Ex: If $A = \{1, \#\}$ and $B = \{\#,3\}$, then $A \cap B = \{\#\}$.

11. **Disjoint Sets**: Having no elements in common.

Ex: A = {1,2} and B = {3,4}

12. **Equivalent Sets**: If there is a one-to-one correspondence between the constituents of A and the fundamentals of B, two sets, A and B, are said to be equivalent.

    Ex: A={1,2,3} and B={a,b,c}

13. **Cartesian product**: "The set of all ordered pairs from two sets".

Understanding these types of sets and their properties is fundamental in set theory and various areas of mathematics.

Venn Diagram for Different types of sets

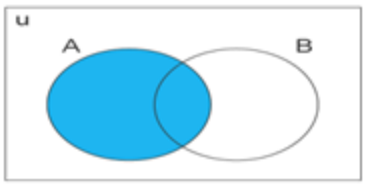


Figure 1.2 : Set A

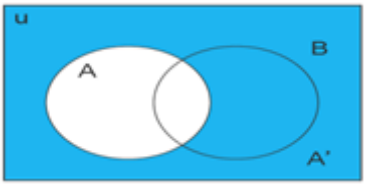


Figure 1.3 : A' is the complement of A

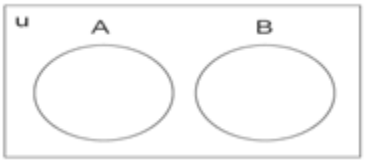


Figure 1.4 : Disjoint sets of A and B

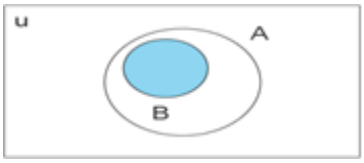


Figure 1.5 : B is a Proper subset of A

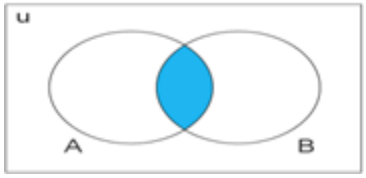


Figure 1.6 : A n B

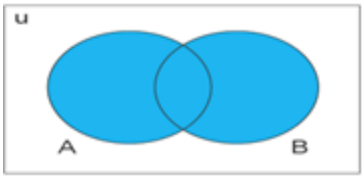


Figure 1.7 : A U B

In the above figure, the shaded portions in "blue" show the set that they are labelled with.

Sets Formulas in Set Theory

For any two overlapping sets A and B,

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

For any two sets A and B that are disjoint,

- $n(A \cup B) = n(A) + n(B)$
- $A \cap B = \emptyset$

Laws of Sets

1. **Commutative Law**:
    - **Union**: $A \cup B = B \cup A$
    - **Intersection**: $A \cap B = B \cap A$
2. **Associative Law**:
    - **Union**: $(A \cup B) \cup C = A \cup (B \cup C)$
    - **Intersection**: $(A \cap B) \cap C = A \cap (B \cap C)$
3. **Distributive Law**:
    - **Union over Intersection**: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
    - **Intersection over Union**: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
4. **Identity Law**:
    - **Union with Empty Set**: $A \cup \emptyset = A$

- **Intersection with Universal Set**: $A \cap U = A$
- **Union with Universal Set**: $A \cup U = U$
- **Intersection with Empty Set**: $A \cap \emptyset = \emptyset$

5. **Idempotent Law**:
   - **Union**:

     $$A \cup A = A$$

6. **Intersection**:

   $$A \cap A = A$$

7. **Complementary Law**:
   - **Complement of Complement**:

     $$(Ac)c = A(Ac)c$$

   - **De Morgan's Laws**:
     - $(A \cup B)c = Ac \cap Bc$
     - $(A \cap B)c = Ac \cup Bc$

8. **Null and Universal Sets**:
   - **Union with Universal Set**: $A \cup U = U$
   - **Intersection with Universal Set**: $A \cap U = A$
   - **Union with Empty Set**: $A \cup \emptyset = A$
   - **Intersection with Empty Set**: $A \cap \emptyset = \emptyset$
9. **Subset Property**:

   If $A \subseteq B$, then:
   - $A \cup B = B$
   - $A \cap B = A$

Understanding these properties is essential for working with sets, as they provide the rules for how sets interact and how to manipulate them in various mathematical contexts.

1.3 Summary

Sets in mathematics are collections, they provide a foundational concept for organizing and studying data in various fields. Key aspects of sets include their elements, which are unordered

and unique, and operations such as union, intersection, and complementation. Sets are versatile tools used to define relationships, establish criteria, and classify entities across disciplines.

1.4 Keywords

Keywords for the preliminaries of abstract algebra typically include:

- Elements
- Subset
- Union
- Intersection
- Complement
- Set theory
- Power set
- Cardinality

1.5 Self -Assessment Questions

1. Define a group and list its four defining properties.
2. Explain what a subgroup is and provide an example..
3. What is Lagrange's theorem, and what does it state about the order of subgroups?
4. Define a normal subgroup and explain its significance in group theory.
5. Describe what a quotient group is and how it is constructed.

1.6 Case Study

Cryptography is the practice of secure communication in the presence of adversaries. It relies heavily on abstract algebra, particularly group theory, for designing secure cryptographic systems.

Question: Alice and Bob want to communicate securely over an insecure channel, such as the internet, without Eve, the eavesdropper, intercepting their messages.

1.7 References

- "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
- "Discrete Mathematics" by Richard Johnsonbaugh

Unit - 2

Relations

Learning objectives

- Understand what a relation is and how it relates pairs of elements from two sets.
- Learn different ways to represent relations.
- Understand the properties that characterize different types of relations.

Structure

2.1 Introduction

In mathematics, the term "relation" typically refers to a set of ordered pairs. An ordered pair is a pair of elements where the order in which the elements appear matters. For example, in the ordered pair (x, y), x is related to y.

The term "AND" doesn't directly apply to relations in mathematics in the same way it does in logic or computer programming. However, you can think of certain operations involving relations that could be analogous to the logical AND operation.

For instance, if R={(1,2),(3,4),(5,6)} and S={(1,3),(3,5),(5,7)} the relation that satisfies both R and S simultaneously would be R∩S={(1,3)}.

So, in this context, you can think of the intersection of relations as a way to combine them in a manner similar to the logical AND operation. However, it's essential to note that this analogy has limitations, as relations in mathematics are more complex structures than simple logical conditions.

Defination: The relation is a set of ordered pairs. An ordered pair consists of two objects (which may be numbers, elements of sets, etc.) in a specific order.

Ordered Pairs: An ordered pair is a pair of elements where the order matters. For example, (2, 3) is different from (3, 2).

Cartesian Product: Cartesian product are the set of all ordered pairs (a,b) where a∈A and b∈B is the Cartesian product A×B given two sets A and B. It is defined formally as:

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$$

Subset of Cartesian Product: "If R is a relation from set A to set B is a subset of the Cartesian product A × B". This means that R contains some (or all) of the ordered pairs that are possible between elements of A and B.

Relations can be represented graphically using diagrams like directed graphs, directed arrows, or tables of ordered pairs. They're fundamental in various areas of mathematics, including algebra, calculus, discrete mathematics, and more. Relations help describe connections, dependencies, and interactions between elements of different sets.

2.2 Types of Relation

Relations can be grouped into different kinds in light of their properties and attributes. A portion of the normal sorts of relations include:

Reflexive Relation: If every element in A has a relation to itself, then a relation R is reflexive. If (a,a) belongs to R. For instance, the "is equal to" relation on the real number set is reflexive.

Symmetric Relation: If every pair (a,b) in relation R on a set A also has a pair (b,a) in relation R, then the relation is symmetric. Example: There is symmetry in the relation "is a sibling of".

Transitive Relation: When (a,b) and (b,c) are the element of R, then (a,c) is also an element of R, indicating that a relation R on a set A is transitive. For instance, the relation "is less than" is transitive on the real number set.

Antisymmetric Relation: If (a,b) is in R for each pair (a,b) where an is not equal to b, then (b,a) is not in R, then a relation R on a set A is antisymmetric. For instance, the relationship "is less than or equal to" on the integer set is asymmetric..

Equivalence Relation: An equivalency relation is one that is transitive, symmetric, and reflexive on a set A. As an illustration, consider the equivalency relation "is congruent modulo n" on the set of integers.

Partial Order Relation:

A binary relation $\leq$ on a set P is called a **partial order** if it satisfies three properties:

1. **Reflexivity**: For every element $a \in P$, $a \leq a$.
2. **Antisymmetry**: For every pair of elements $a, b \in P$, if $a \leq b$ and $b \leq a$, then $a = b$.
3. **Transitivity**: For every three elements $a, b, c \in P$, if $a \leq b$ and $b \leq c$, then $a \leq c$.

Total Order Relation: If every pair of elements in A is comparable under a relation R and the relation is partial order, then the relation R on the set A is a complete order relation. Example: On the set of real numbers, the relation "is less than or equal to" is a total order relation.

These are just a few examples of the types of relations that can exist. Relations are fundamental in mathematics and are used to describe various kinds of connections or associations between elements in sets.

Example 1 : Find the cartesian products of set A = {@ ,#, $} and B={3, 4, 5}.

Solution: Cartesian product be R,

R = A × B

= {(@,3), (@,4), (@,5), (#,3), (#,4), (#,5), ($,3), ($,4), ($,5)}

Example 2 : Let the set $A = \{1,2,3,4\}$ and $B = \{a, b, c, d\}$. Find the relation R from set A to set B as follows:

R={(1,a),(2,b),(3,c),(4,d)}

1.Determine if the relation R is reflexive, symmetric, and transitive.

2.Find the range of the relation R.

Solution:

1. To determine if the relation R is refle., symm., and trans.:
    - Reflexive: R is reflexive if for every a in A, (a,a) is in R. Since R contains no pairs of the form (a,a) it is not reflexive.
    - Symmetric: Given relation R is symmetric if (a,b) belong to R, then (b,a) is also belong to R. Since R does not contain pairs like (b,a) where aaa and bbb are different elements, it is symmetric.
    - Transitive: R is transitive if (a,b)) and (b,c) belong to R, then (a,c) is also belong R, it is vacuously transitive.
2. Set of all second coordinates of the ordered pairs in R is called range of R, which is {a,b,c,d}.

2.3 Summary

A relation is a set of ordered pairs, linking elements from two sets. Formally, if A and B are two sets, then the relation R is a subset of the Cartesian product A×B. This means each element in R is a pair (a,b), where a∈A and b∈B. Relations are used to model associations between elements of different sets. }. Relations are foundational in mathematics, underpinning functions, equivalences, and orderings.

2.4 Keywords

- Ordered pairs
- Cartesian product
- Sets
- Subset
- Association
- Mapping

2.5 Self- Assessment Questions

1. Q.1 If $A = \{1,2,3\}$ and $B = \{4,5\}$, how many ordered pairs are possible in the Cartesian product $A \times B$?
2. Q2. Given the relation $R = \{(1,4), (2,5), (3,4)\}$, how many elements does the relation R contain?
3. Q3. If a relation R on set A is defined as $R = \{(x, y) \mid x \leq y\}$, how many ordered pairs are there in R if $A = \{1,2,3\}$?
4. Q4. For sets $A = \{a, b\}$, how many different relations can be formed from A to B?
5. Q5. If R is a relation on set $A = \{1,2,3,4\}$ defined as $R = \{(x, y) \mid x + y = 5\}$, what are the ordered pairs in R?

2.6 Case Study

A university wants to create a digital system to manage course enrollments. The system needs to track which students are enrolled in which courses, facilitating easy access for administrators, professors, and students.

Objective

Develop a model using relations to represent and manage the enrollments effectively.

Sets Involved

1. **Students (S)**: The set of all students at the university.
    - Example: $S = \{s1, s2, s3, s4\}$
2. **Courses (C)**: The set of all courses offered by the university.
    - Example: $C = \{c1, c2, c3\}$

Question:

Define a relation R where each ordered pair (s,c) indicates that student s is enrolled in course c.

2.7 References

- "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
- "Discrete Mathematics" by Richard Johnsonbaugh

Unit - 3

Functions

Learning objectives

- Perform operations on functions, such as addition, subtraction, multiplication, division, composition, and inversion.
- Understand how these operations affect the domain, range, and behavior of functions.
- Understand how changes in equations affect the graphical representation of functions

Structure

3.1 Introduction

A function in mathematics is a relationship where each input has a single relationship with each conceivable output (sometimes referred to as the codomain or range) and a set of inputs (often referred to as the domain).

A correspondence or rule that assigns the identity of exactly one element y in set B to each element x in set A is the formal definition of a function f from set A to B. Mathematically, this is expressed as:

f:A→B

1. If any element x in A,then there exists exactly one y belong to B implies that f(x)=yf(x)
2. It is possible that some elements in B are not the image of any element in A.

Functions are often represented in various ways, including algebraic expressions, tables, graphs, mappings, or verbal descriptions. They are used extensively in almost all branches of mathematics, as well as in numerous real-world applications, to model relationships, analyze data, and solve problems.

3.2 Function and Relationship Representation

Relations: Definition

A subset of A× B is a relation R that connects set A to set B. If (a,b)∈R, we write aRb to indicate that a and b are connected by R.

Example:

Consider the sets:

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

$B = \{a, b\}$

A relation R from A to B can be:

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

Functions: Definition

“A function f from set A to set B written f:A→B is a relation such that for every a∈A, then there exists a unique element b∈B such that the ordered pair(a,b)∈f. Then this unique element b as f(a)”.

Consider the sets:

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

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

A function f from A to B can be:

$f = \{(1, a), (2, b), (3, c)\}$

This means:

$f(1) = a$

$f(2) = b$

$f(3) = c$

3.3 Domain , Codomain, Range definition

Particularly in the context of functions, the terms "domain," "codomain," and "range" are used to describe different aspects of the function's behavior and its relationship between inputs and outputs.

Domain: A function's domain refers to the set of all input values, also known as the independent variable, for which the function is defined. Put otherwise, it's the collection of all x values for which the function f(x) is defined. It specifies where the function begins. dom(f) or just A when f:A→B is used to indicate the domain

Codomain: "The collection of all potential output values that a function can generate", commonly referred to as the dependent variable, is its codomain. It is the collection of all potential values that f(x) is capable of. In contrast to the domain, which is a collection of particular input values, the codomain denotes the full range of possible outputs. Often, the codomain is indicated by codom(f) or just B for f:A→B.

Range: The set of all actual output values obtained by evaluating a function throughout its whole domain is called the function's range. Put differently, it represents the entire set of values that the function can accept as x fluctuates across the domain. A portion of the codomain makes up the range. It is a representation of the range of values that the function "hits" or "reaches" when using the domain. Often, the range is indicated by range(f).

Here's a simplified breakdown:

- **Domain**: All possible input values.
- **Codomain**: All possible output values.
- **Range**: Actual output values obtained by evaluating the function over its domain.

It's essential to note that the range can be equal to or smaller than the codomain. If the range is equal to the codomain, it means that the function hits every possible output value. If the range is smaller, it means that there are some output values in the codomain that the function never reaches.

3.4 Types of Function

Functions can be classified into various types based on different criteria. Here are some common types of functions:

1. Linear Function
2. Quadratic Function
3. Polynomial Function
4. Exponential Function
5. Logarithmic Function
6. Trigonometric Function
7. Piecewise Function
8. Rational Function
9. Trigonometric Polynomial
10. Periodic Function
11. Inverse Function

These are just a few examples of the types of functions encountered in mathematics. Each type has its own characteristics, properties, and applications in various fields

One-One Function(Injection): An injection, or one-to-one function, is a function type in which every element in the domain maps to a different element in the codomain. Stated differently, no two distinct elements within the domain are mapped to an identical element inside the codomain.

Many-One Function:

A many-to-one function, also known simply as a "non-injective" function, is a type of function where multiple elements in the domain can map to the same element in the codomain. In other words, there can be more than one input that produces the same output.

In simpler terms, a function is one-to-one if different inputs always produce different outputs. Graphically, a one-to-one function never intersects itself when plotted on a graph.

Onto Function(Surjection)

Every element in the codomain has at least one pre-image in the domain when it comes to an onto function, also referred to as a surjection. That is to say, at least one element in the domain maps to each element in the codomain.

Into Function:

Not every element in the codomain has a pre-image in the domain; this is referred to as an into function, also called a non-surjective function. Alternatively said, some elements in the codomain exist that are not mapped to any element in the domain.

One-One Onto Function

An onto function that is one-to-one (injective) and one-to-one (surjective) is referred to as a one-to-one onto function, or bijective function. Put otherwise, this function translates each element in the domain to a unique element in the codomain and assigns each element in the codomain to a single pre-image in the domain.

3.5 Summary

Functions and relations are fundamental concepts in mathematics. A relation between sets A and B is a subset of their Cartesian product A×B, consisting of ordered pairs. A function is a special type of relation where each element in A maps to a unique element in BBB. Functions can be injective (one-to-one), surjective (onto), or bijective (both). Relations and functions can be represented as sets of ordered pairs, mapping diagrams, or graphs. These concepts are essential for understanding mappings, transformations, and dependencies between elements in different sets.

Formally, a function f is defined by a rule or formula that assigns to each element x in the domain D a unique element f(x)f(x) in the codomain C. Key components of functions include their domain (set of permissible inputs) and range (set of possible outputs), which together define the function's behavior and scope. Functions can be represented in various forms, such as algebraic expressions, tables, graphs, or verbal descriptions.

3.6 Keywords

- Domain
- Range
- Input
- Output
- Mapping
- Notation

3.7 Self- Assessment Questions

1. Q1. Given the function $f(x) = 2x^2 - 3x + 1$, find f(3).
2. Q2. Determine the domain and range of the function $(x) = \sqrt{4 - x^2}$.
3. Q3. If $f(x) = 2x + 1$ and $g(x) = x^2 - 3$, find $(f \circ g)(2)$.
4. Q4. If $h(x) = x + 1/3$, find h^{-1}.
5. Q5. Consider the function $y = -2x^2 + 4x - 1$. What are the coordinates of the vertex of its graph?

3.8 Case Study

A city planner needs to analyze population growth trends to plan infrastructure development. They want to use functions to model population growth and predict future demographics.

Questions:

1. Develop a function-based model to describe the population growth of a city over the next 10 years.
2. Use the model to predict future population sizes and analyze demographic trends.

3.9 References

- "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
- "Discrete Mathematics" by Richard Johnsonbaugh

Unit- 4

Permutations and Combinations

Learning Objectives:

- Understand the definitions of permutations and combinations.
- Understand and use factorial notation n! in permutations and combinations calculations.
- Differentiate between permutations (arrangements) and combinations (selections).

Structure:

4.1 Permuations

A permutation is a mathematical calculation that determines how many different configurations there are for a particular set. Simply said, a permutation is the number of alternative configurations or ordering for an item.. The arrangement's order is important when using permutations. There are two kinds of permutations: one without repetition and the other with repetition. Permutations deviate from combinations in that the order in which the data is selected is irrelevant.

- An array's permutation is the total number of possible configurations for a collection or object.
- The numerical sequence is important when using a permutation.
- Circular and multi-set permutations are two less common forms of permutations, although the two basic varieties are those with and without repetition.
- A single combination can result in more than one permutation.
- Combinations, which are selections of data from a group where order is irrelevant, are not the same as permutations.

Mathematicians employ the idea of permutations. They show the breadth of configurations that can exist inside a group. In the case of permutations, order is crucial. This distinguishes it from the idea of a combination, where the order is irrelevant. Permutations resemble ordered combinations in several ways. Below, we go over combinations in a bit more detail.

Permutations formula:

$$ {}_nP_r = \frac{n!}{(n-r)!} $$

When all the potential permutations are listed, a permutation can also be computed manually. Any order of the pieces can occur in a combination, which is occasionally mistaken for a permutation.

The amount of ways a three-digit keypad sequence can be organized is a basic method to display

a permutation. The number of combinations utilizing the numbers 0 (0) through 9 (9), and only using a particular digit on the keypad once.

Examples of Permuations

These two instances demonstrate the operation of permutations. The first two are from the business and financial sectors. Let us imagine that a portfolio manager selected 25 equities for a new fund after eliminating 100 firms. Since there won't be an equal weight distribution across these 25 holdings, ranking will occur. There will be several options to order the money.

$$P(100,25) = 100! \div (100\text{-}25)! = 100! \div 75! = 3.76E + 48$$

A simpler example would be for a business to expand its nationwide warehouse network. Three of the five potential locations will be committed to by the corporation. They will be constructed in a sequential order, therefore order matters. The quantity of combinations is:

$P(5,3) = 5! \div (5\text{-}3)! = 5! \div 2! = 60$

4.2 Combination

It is composed of discrete parts that are carried one at a time, partially at a time, or all at once from a common set of parts. For instance, if there are two components, A and B, there is only one option to choose: choose both of them.

Combination Formula :

$$ {}_nC_r = \left(\begin{matrix} n \\ r \end{matrix} \right) = \frac{n!}{r!(n-r)!} $$

On the other hand, a combination is a kind of pack. Again, if sets are made with two of those three numbers—1, 2, and 3—the combinations are (1, 2), (1, 3), and (2, 3).

Unlike permutations where they are distinct, (1, 2) and (2, 1) are identical here. The writing for this is 3C2. Generally speaking, the quantity of n unique item combinations taken r at a time is,

$$ {}^{\mathbf{n}}\mathbf{C}_{\mathbf{r}} = \frac{\mathbf{n!}}{\mathbf{r!} \times (\mathbf{n}-\mathbf{r})!} = \frac{{}^{\mathbf{n}}\mathbf{P}_{\mathbf{r}}}{\mathbf{r!}} $$

4.3 Difference Between Permutation and Combination

Permutation	Combination
The order of arrangement matters in permutation. As an illustration, AB and BA are distinct pairings.	The arrangement's sequence in combination is irrelevant. AB and BA, for instance, are identical combinations.
When sorting or arranging several types of objects, a permutation is employed.	When sorting items of the same kind, combinations are employed.
$\mathbf{n\,Pr = n!/(n - r)!}$	$\mathbf{n\,Cr = n!\,/\{r!\times(n - r)!\}}$

Example 1: No of permutations and combinations taking n = 9 and r = 3**.**

Solution:

Given, $n = 9, r = 3$

Using the formula given above:

For Permutation:

$^nP_r = (n!) / (n - r)!$

$\Rightarrow {}^nP_r = (9!) / (9 - 3)!$

$\Rightarrow {}^nP_r = 9! / 6! = (9 \times 8 \times 7 \times 6!)/ 6!$

$\Rightarrow {}^nP_r = 504$

For Combination:

$^nC_r = n!/r!(n - r)!$

$\Rightarrow {}^nC_r = 9!/3!(9 - 3)!$

$\Rightarrow {}^nC_r = 9!/3!(6)!$

$\Rightarrow {}^nC_r = 9 \times 8 \times 7 \times 6!/3!(6)!$

$\Rightarrow {}^nC_r = 84$

Example 2:

How many options are there to choose from among six men and five women a committee of two women and four men?

Choose 4 men out of 6 men = ${}^6C_4$ ways = 15 ways

Choose 2 women out of 5 women = ${}^5C_2$ ways = 10 ways

The committee can be chosen in ${}^6C_4 \times {}^5C_2$ = 150 ways.

Example 3:

Using the letters from the word "FABLE," how many 3-letter words may be created?

Solution

Using the permutation formula, we get:

5P3 = 5! / (5 – 3)! = 5! / 2! = 5 x 4 x 3 = 60

4.4 Summary

Permutations and combinations are fundamental concepts in combinatorics. Permutations focus on the arrangement of items where the order matters. $P(n, r) = n!/(n - r)!$ indicates the number of permutations of n elements taken r at a time... Contrarily, combinations are choices in which the sequence is irrelevant. $C(n, r) = n!/r!\,(n - r)!$ gives the total number of combinations of n items taken r at a time.. Both concepts are used in probability, statistics, and various fields requiring counting of distinct arrangements or selections.

4.5 Keywords

- Permutations
- Combinations
- Arrangement
- Selection
- Order
- Factorial

4.6 Self- Assessment Questions

1. How many different ways can 5 books be arranged on a shelf?
2. In how many ways can a committee of 3 members be chosen from a group of 10 people?
3. If it is forbidden to repeat letters, how many different 4-letter words may be made from the letters A, B, C, and D?
4. If a password consists of 3 letters followed by 2 digits, how many unique passwords can be created if letters and digits can be repeated?
5. How many ways can you distribute 5 identical apples among 3 different children?

4.7 Case Study

A school is organizing a science fair and has received 8 project submissions from students. The science fair committee needs to select 5 projects to present at the event, and they also need to determine the order in which these projects will be presented.

Questions:

1. In how many ways can the committee select the 5 projects out of the 8 submissions?

2. Once the 5 projects are selected, in how many ways can the committee arrange the order of presentation?

4.8 References

- "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
- "Discrete Mathematics" by Richard Johnsonbaugh

Unit - 5

Mathematical Induction

Learning Objectives:

- Understand the importance of the base case in establishing the validity of the induction process.
- Understand how the inductive hypothesis serves as an assumption to prove the inductive step.
- Understand how to modify the induction process for different contexts, such as strong induction or induction on different domains

Structure:

5.1 Mathematical Induction

A basic method of creating proofs is mathematical induction, which may be used to demonstrate a point with any collection that is well-structured. It is customary to establish claims or demonstrate results whenever something is expressed in terms of the natural number n. The Principle of Mathematical Induction can be used to prove P(n), assuming it is a statement for n natural numbers. First, we will demonstrate P(1). Next, we will assume P(k) to be true and demonstrate P(k+1). By the concept of mathematical induction, we can conclude that P(n) is true if P(k+1) holds.

Mathematical induction can be likened to falling dominoes. A domino falling causes the subsequent domino to fall as well. One domino falls on top of the other, and so on. The first one knocks down the second. Ultimately, every domino will go over. However, a few requirements must be met:

In order for the knocking process to begin, the first domino needs to fall. For every two adjacent dominoes, the distance between them must be the same. If not, one domino could topple over another without colliding. Following that, the series of reactions will end. For any integer k ≥ a, P(k) → P(k + 1) if the inter-domino distance is maintained equal. This is where induction begins.

5.2 Properties of Mathematical Induction

1. $1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}, n \geq 1$
2. $1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}, n \geq 1$
3. $1^3 + 2^3 + 3^3 + \ldots + n^3 = \frac{n^2(n+1)^2}{4}, n \geq 1$
4. $1 + 3 + 5 + \ldots + (2n-1) = n^2, n \geq 1$
5. $2^n > n, n \geq 1$

5.3 Mathematical Induction Example

Example 1: Sum of the First n Natural Numbers

Statement: Let P(n): n(n+1) / 2.

Base Case (n=1):

P(1) = 1(1+1) / 2 = 1

The sum of the first 1 natural number is 1, so the base case holds.

suppose P(k) is true for some k≥1

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

Now we have to prove that P(k+1) is true, i.e., that:

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

Consider the sum up to k+1:

$$\sum_{i=1}^{k+1} i = \left(\sum_{i=1}^{k} i\right) + (k+1)$$

By the inductive hypothesis:

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

Combine the terms:

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

Hence, P(k+1) holds.

Hence, the statement is true for all n≥1

Example 2: Prove $2^n > n$ for all $n \geq 1$

Solution

Base Case (n=1):

$2^1 = 2 > 1$

The base case holds.

Assume $2^k > k$ for some $k \geq 1$. Now we have to prove $2^{k+1} > k+1$.

$2^k > k$

Multiply both sides by 2:

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

Since $k \geq 1$, we have $2k \geq k+1$:

Thus, $2^{k+1} > k+1$.

Hence , the statement is true for all $n \geq 1$.

5.4 Application of Mathematical Induction in Real Life

Mathematical induction, while primarily a theoretical tool, has applications that extend to various real-life scenarios. Here are some examples and applications where mathematical induction plays a role:

Computer Science and Algorithm Analysis

1. **Correctness of Algorithms**:
    - Mathematical induction is used to prove the correctness of recursive algorithms. For instance, proving that a sorting algorithm like quick sort or merge sort correctly sorts an array can be done using induction on the size of the array.
2. **Runtime Analysis**:
    - Induction can help establish the time complexity of algorithms, particularly those that are recursive. For example, proving that the runtime of a recursive function grows at a certain rate can be achieved using mathematical induction.

Network Design and Analysis

3. **Routing and Communication Protocols**:
    - Induction can be used to verify that protocols or algorithms for data routing in networks work correctly for any number of nodes or hops. For instance, ensuring that a packet reaches its destination regardless of the network size.
4. **Distributed Systems**:
    - In distributed systems, mathematical induction helps prove properties like consensus or reliability, ensuring that algorithms work for any number of processes or nodes.

Software Engineering

5. **Proof of Program Correctness**:
    - Induction is employed to prove that software programs meet their specifications, especially in loop invariants and recursive function correctness. This is crucial in formal verification and in proving that a program will not run into infinite loops or errors.

Mathematics and Theoretical Applications

6. **Mathematical Theorems**:
    - Many mathematical results and theorems are proven using induction. This is essential in number theory, combinatorics, and other areas where proving results for all natural numbers is necessary.
7. **Combinatorial Proofs**:
    - Induction is used in combinatorial proofs to establish properties of structures like graphs, trees, and sets. For example, proving that every tree with nnn nodes has n−1n-1n−1 edges can be done using induction.

Economics and Finance

8. **Economic Models**:
    - Inductive proofs are used to verify properties of economic models that operate over time or across multiple agents. For instance, proving that a certain strategy will lead to an equilibrium in a game-theoretic model.

Biology and Medicine

9. **Population Dynamics**:

- Inductive reasoning can help in modeling population growth or the spread of diseases. Proving that certain patterns hold as populations grow or as time progresses can be done using induction.

5.5 Summary

The process of proving a proposition holds true for all natural numbers is known as mathematical induction. The inductive step and the base case are the two primary steps involved. The statement is checked for the beginning value in the base case, which is typically n=1. In the inductive stage, the assertion is assumed to be true for any natural number n = k (the inductive hypothesis), and it is then proven to be true for n = k+1. By using these procedures, it may be demonstrated that the assertion is true for all n if it is true for n=k and n=k+1.

5.6 Keywords

- Mathematical Induction
- Base Case
- Inductive Step
- Sequence
- Strong Induction
- Induction on Integers
- Assumption

5.7 Self- Assessment Questions

1. Prove that the sum of the first n natural numbers is $n(n+1)/2$ using mathematical induction.
2. Using mathematical induction, show that $2^n > n$ for all natural numbers $n \geq 1$.
3. Show that the Fibonacci sequence defined by F1=1, F2=1, and $F_n = F_{n-1} + F_{n-2}$ satisfies $F_n < 2^n$ for all $n \geq 1$ using mathematical induction.
4. Use mathematical induction to demonstrate that for any natural number $n, 7^n - 1$ is divisible by 6.

5.8 Case Study

A software engineer is designing an algorithm that involves summing the squares of the first n positive integers. The engineer proposes the formula $S(n) = n(n + 1)(2n + 1)/6$ for the sum of the squares of the first n integers.

Question

1. Prove that the formula is correct using mathematical induction.
2. Explain how this proof ensures the reliability of the algorithm for any positive integer n.

5.9 References

- "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
- "Discrete Mathematics" by Richard Johnsonbaugh

Unit - 6

Asymptotic Notations

Learning Objectives:

- Compare and contrast different asymptotic notations to understand their implications for algorithm performance.
- Explore lower bound analysis using Big-Omega notation.
- Understand scenarios where Big-Theta notation is appropriate.

Structure:

6.1 Asymptotic Notations

- Mathematical techniques called asymptotic notations are used to examine algorithm performance by determining how an algorithm's efficiency increases with increasing input size.
- The behavior of the time or space complexity of an algorithm as the input size gets closer to infinity can be succinctly expressed with these notations.

- Instead of explicitly comparing algorithms, asymptotic analysis aims to comprehend the relative rates at which the complexity of algorithms rise.
- By removing implementation specifics and machine-specific constants, it makes it possible to compare the efficiency of algorithms by concentrating on underlying patterns.
- Asymptotic analysis looks at an algorithm's performance characteristics as the input size changes, making it possible to compare the space and time complexity of different methods.
- By classifying algorithms according to their worst-case, best-case, or average-case time or space complexity, we may get important insights into their efficiency by employing asymptotic notations, such as Big O, Big Omega, and Big Theta.

There are mainly three asymptotic notations:

1. **Big − O Notation**
2. **Omega Notation**
3. **Theta Notation**

6.2 Theta Notation (Θ-Notation):

The function is enclosed from above and below by theta notation. Since it gives the upper and lower bounds on the method's execution time, the average-case complexity of an algorithm may be examined.

Any function that accepts a set of natural numbers as input and returns itself may be implemented using g and f. When constants c1, c2 > 0 are present and $c1 * g(n) < f(n) \leq c2 * g(n)$ for every $n \geq n0$, a function f is said to be Θ(g).

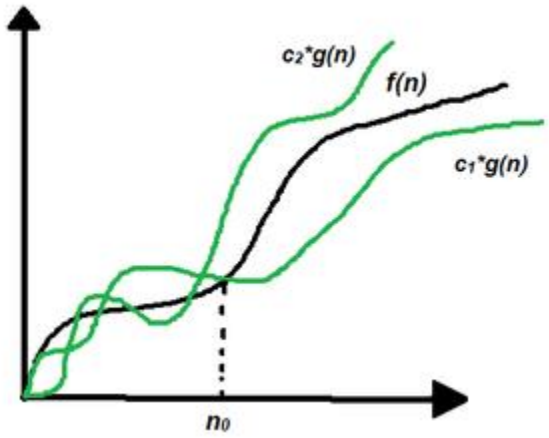


Figure 6.1 : Theta Notation

Mathematical Representation of Theta notation:

If f(n) is a function of Θ(g(n)) if there exist +ve constants c1,c2, and n0 such that for all n≥n0,

$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

Properties:

- **Tight Bound**: f(n) is bounded both above and below by g(n) for sufficiently large n.
- **Formal Definition**: If there exist constants c1,c2>and such that the condition $c1 \cdot g(n) \leq f(n) \leq c2 \cdot g(n)$ for all n≥n0 .
- **Useful for Exact Analysis**: Provides a precise characterization of the time complexity when both upper and lower bounds are the same.

6.3 Big-O Notation (O-notation):

Big-O notation is used to indicate an algorithm's upper bound on running time. It is the notation most commonly used in asymptotic analysis.It describes the maximum value of a function.An algorithm's worst-case time complexity is the maximum amount of time it can take.The largest yield value (big-O) is input.Big-Oh: (The Worst Case) It is clear as the situation in which an algorithm can carry out an operation for the maximum amount of time that is practical.

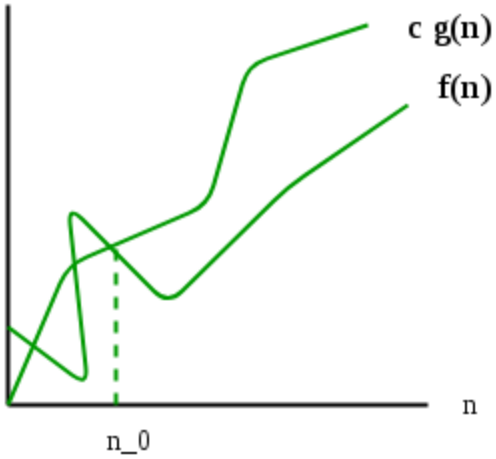


Figure 6.2 : Big-O Notation

Mathematical Formation of Big-O Notation:

A function f(n) is O(g(n)) if there exist positive constants c and n_0 such that for all n≥n0, f(n)≤c·g(n)

Example: If an algorithm's running time is $f(n)=3n^2+2n+1$, it is Θ(n2) because n^2 provides both an upper and lower bound, and we can find constants c_1,c_2, and n_0 such that c1·n2≤f(n)≤c2·n2

Properties:

- **Upper Bound**: f(n) is bounded above by g(n) for sufficiently large n.
- **Formal Definition**: "If constants c>0 and n0 such that f(n)≤c·g(n) for all n≥n0".
- **Best for Worst-case Analysis**: Useful for determining the worst-case time complexity of algorithms.

6.4 Omega Notation (Ω-Notation):

The letters G and f stand for the function that converts the set of natural numbers back to itself. Function f is defined as follows: for any n that is more than or equal to n0, if c*g(n) < f(n), then function f exists.

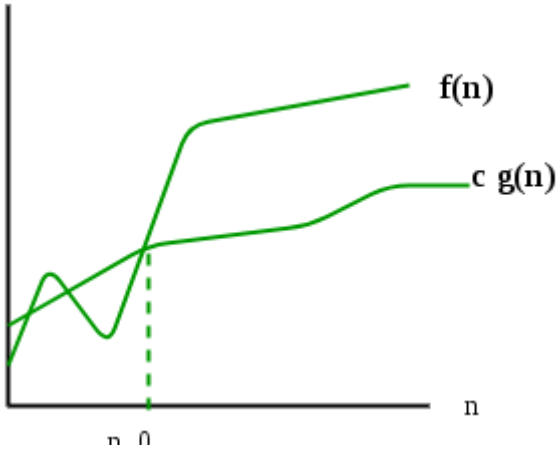


Figure 6.3: Omega Notation

Mathematical Representation of Omega notation :

Mathematically, this is expressed as:

$$f(n) \geq c \cdot g(n) \text{ for all } n \geq n0$$

Here, g(n) acts as a lower bound or a best-case scenario for f(n).

Properties:

- **Lower Bound**: f(n) is bounded below by g(n) for sufficiently large n.
- **Formal Definition**: "If constants c>0 and n0 such that $f(n) \geq c \cdot g(n)$ for all $n \geq n0$".
- **Best for Best-case Analysis**: Useful for determining the best-case time complexity of algorithms.

6.5 Summary

Asymptotic notations, like Big-O, Big-Theta, and Big-Omega, quantify the behavior of functions as their inputs grow large. Big-O describes the upper bound of a function's growth rate, indicating the worst-case scenario. Big-Theta provides a tight bound, capturing both upper and lower limits, defining the exact growth rate. Big-Omega signifies the lower bound, showing the best-case growth rate. These notations are crucial in algorithm analysis, allowing comparisons of efficiency and scalability. They simplify complexity analysis by focusing on dominant terms and constants, aiding in algorithm design, optimization, and understanding how algorithms perform as input sizes increase.

6.6 Keywords

- Asymptotic Notations
- Big-O
- Big-Theta
- Big-Omega
- Upper Bound
- Lower Bound
- Tight Bound
- Worst-case
- Best-case
- Average-case

6.7 Self- Assessment Questions

1. What does Big-O notation represent in algorithm analysis?
2. Compare and contrast Big-O, Big-Theta, and Big-Omega notations.
3. Determine the Big-O notation for the following functions:

$$g(n) = 2n^3 + 4n^2 + 3n$$

4. Explain how asymptotic notations help in algorithm selection and optimization.
5. Provide an example where Big-Theta notation is more appropriate than Big-O or Big-Omega.

6.8 Case Study

A software development team needs to choose between two sorting algorithms for a new feature implementation. They are considering the Merge Sort algorithm and the Bubble Sort algorithm. The team needs to decide based on their respective time complexities.

Questions

1. Describe the time complexity of Merge Sort using Big-O notation and explain its significance.
2. Describe the time complexity of Bubble Sort using Big-O notation and explain its significance.

3. Based on the time complexities, recommend which sorting algorithm would be more suitable for sorting a large dataset, and justify your recommendation.

6.9 References

1. "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
2. "Discrete Mathematics" by Richard Johnsonbaugh

Unit -7

Recurrence Relation

Learning Objectives:

- Understanding Recurrence Relations.
- Analyzing and Solving Recurrence Relations.
- Understand the application of recurrence relations in the design and analysis of algorithms, particularly in analyzing time and space complexity.

Structure:

7.1 Introduction

Recurrence relations in discrete mathematics are equations that recursively define a sequence. These relations are instrumental in combinatorics, algorithm analysis, and other areas where discrete structures are analyzed. Here's an in-depth look at recurrence relations in the context of discrete mathematics:

Definition

A recurrence relation for a sequence {an} is an equation that expresses a_n in terms of one or more previous terms $a_{n-1}, a_{n-2}, \ldots, a_{n-k}$ along with some initial conditions.

Linear Recurrence Relations:

- **Homogeneous Linear Recurrence Relation:**

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

Example: Fibonacci sequence, $F_n = F_{n-1} + F_{n-2}$, with $F_0 = 0$ and $F_1 = 1$.

- **Non-Homogeneous Linear Recurrence Relation:**

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

Example: $an = 2a_{n-1} + 3$, with $a_0 = 1$.

Recurrence relations	Initial values	Solutions
$F_n = F_{n-1} + F_{n-2}$	$a_1 = a_2 = 1$	Fibonacci number
$F_n = F_{n-1} + F_{n-2}$	$a_1 = 1, a_2 = 3$	Lucas Number
$F_n = F_{n-2} + F_{n-3}$	$a_1 = a_2 = a_3 = 1$	Padovan sequence
$F_n = 2F_{n-1} + F_{n-2}$	$a_1 = 0, a_2 = 1$	Pell number

Table 7.1: Recurrence Relations

7.2 Methods for Solving Recurrence Relations

Substitution Method (Iteration):

- Repeatedly substitute the recurrence relation into itself.
  - Example: For $a_n = 2a_{n-1} + 1$ with $a_0 = 1$,

$$a_1 = 2a_0 + 1 = 3$$

$$a_2 = 2a_1 + 1 = 2 \cdot 3 + 1 = 7$$

Generalizing, we find a pattern and derive $a_n = 2^n + 2^n - 1$.

Characteristic Equation Method (for Linear Homogeneous Recurrences):

- Solve the characteristic polynomial.
  - Example: For $a_n = 3a_{n-1} - 2a_{n-2}$,

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

Solving, $r = 1$ and $r = 2$. The general solution is $a_n = A \cdot 1^n + B \cdot 2^n$.

Generating Functions Method:

- Use generating functions to transform the recurrence relation into an algebraic equation.
  - Example: For $a_n = a_{n-1} + n$ with $a_0 = 0$,

$$A(x) = \sum_{n=0}^{\infty} a_n x^n$$

Transform the recurrence into $A(x) - a_0 = x(A(x) + A'(x))$.

Matrix Methods:

- Represent the recurrence relation using matrices and solve using matrix exponentiation.

- Example: For the Fibonacci sequence Fn=Fn−1+Fn−2

$$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix}$$

7.3 Applications

1. **Algorithm Analysis:**

   Used to determine the time complexity of recursive algorithms. For example, the recurrence for merge sort is T(n)=2T(n/2)+O(n).

2. **Counting Problems:**

   In combinatorics, recurrence relations are used to count objects. For instance, the number of ways to climb n stairs taking 1 or 2 steps at a time can be modeled by an=an−1+an−2.

3. **Graph Theory:**

   Recurrence relations can describe properties of graph sequences, such as the number of certain types of paths in a graph.

4. **Dynamic Programming:**

   Many dynamic programming problems, like the knapsack problem, rely on solving recurrence relations.

Degree of the Recurrence relations:

The level of a difference equation is define to be the maximum power of f (x) or $a_r=y_k$

Example 1: The formation $y^3_{k+3}+2y^2_{k+2}+2y_{k+1}=0$ has the degree 3, as the maximum power of y_k is 3.

Example 2: The formation $a^4_r+3a^3_{r-1}+6a^2_{r-2}+4a_{r-3}=0$ has the degree 4, as the maximum power of a_r is 4.

7.4 Rules For Finding the CF

Consider the difference equation $f(E)y_n = F(n)$ ... ①

$\Rightarrow (a_0 E^r + a_1 E^{r-1} + \cdots + a_{r-1} E + a_r)y_n = F(n)$

Step 1: Find the Auxiliary Equation (A.E) given by $f(E) = 0$

$\Rightarrow (a_0 E^r + a_1 E^{r-1} + \cdots + a_{r-1} E + a_r) = 0$ ...②

Step 2: Solve the auxiliary equation given by ②

i. If the n roots of A.E. are real and distinct say $m_1, m_2, \ldots, m_n$
   C.F. = $c_1 m_1{}^n + c_2 m_2{}^n + \cdots + c_n m_n{}^n$

ii. If two or more roots are equal i.e. $m_1 = m_2 = \cdots = m_k$, $k \leq n$
   C.F. = $(c_1 + c_2 n + c_3 n^2 + \cdots + c_k n^{k-1}) m_1{}^n + \cdots + c_n m_n{}^n$

iii. If A.E. has a pair of imaginary roots i.e. $m_1 = \alpha + i\beta$, $m_2 = \alpha - i\beta$

   C.F. = $r^n(c_1 \cos n\theta + c_2 \sin n\theta) + c_3 m_3{}^n + \cdots + c_n m_n{}^n$

   where r = $\sqrt{\alpha^2 + \beta^2}$ and $\theta = \tan^{-1}\frac{\beta}{\alpha}$

Example 3: Solve

$$U_{n+3} - 2U_{n+2} - 5U_{n+1} + 6U_n = 0$$

Solution

$\Rightarrow (E^3 - 2E^2 - 5E + 6)U_n = 0$

$\Rightarrow E^3 - E^2 - E^2 + E - 6E + 6 = 0$

$\Rightarrow E^2(E-1) - E(E-1) - 6(E-1) = 0$

$\Rightarrow (E-1)(E^2 - E - 6) = 0$

$\Rightarrow (E-1)(E-3)(E+2) = 0$

$\Rightarrow E = 1, 3, -2$

$\therefore C.F. = c_1 + c_2 3^n + c_2 3^n$

The solution is y_n = C.F.

$\Rightarrow y_n = c_1 + c_2 3^n + c_2 3^n$

Example 4: Solve

$$y_{n+3} - 2y_{n+1} + 4y_n = 0$$

Solution

$$\Rightarrow (E^3 - 2E + 4)y_n = 0$$

Auxiliary equation is: $E^3 - 2E + 4 = 0$ ...①

By hit and trial $(E + 2)$ is a factor of ①

∴① May be rewritten as

$$E^3 + 2E^2 - 2E^2 - 4E + 2E + 4 = 0$$

$$\Rightarrow E^2(E + 2) - 2E(E + 2) + 2(E + 2) = 0$$

$$\Rightarrow (E + 2)(E^2 - 2E + 2) = 0$$

$$\Rightarrow E = -2, 1 \pm i$$

C.F. $= c_1(-2)^n + r^n(c_2 \cos n\theta + c_3 \sin n\theta)$

$r = \sqrt{\alpha^2 + \beta^2} = \sqrt{1 + 1} = \sqrt{2}$, where $1 \pm i \equiv \alpha \pm i\beta$

$\theta = \tan^{-1}\frac{\beta}{\alpha} = \tan^{-1} 1 = \frac{\pi}{4}$

∴ C.F. $= c_1(-2)^n + r^n(c_2 \cos n\theta + c_3 \sin n\theta)$

Since $F(n) = 0$, solution is given by $y_n = $ C.F

$$\Rightarrow y_n = c_1 e^{-2x} + \left(\sqrt{2}\right)^n \left(c_2 \cos\frac{n\pi}{4} + c_3 \sin\frac{n\pi}{4}\right)$$

7.5 Summary

A recurrence relation defines a sequence where each term is expressed in terms of previous terms. It typically consists of an initial condition and a recurrence formula. Solving recurrence relations involves finding a closed-form solution or understanding the asymptotic behavior of the sequence. Techniques like substitution, iteration, and the Master Theorem are used to analyze and solve them. Recurrence relations are fundamental in algorithm design and analysis, helping to understand time complexity and recursion depth. They bridge mathematical concepts with real-world applications, providing a powerful tool for modeling iterative processes and understanding patterns in sequences and algorithms.

7.6 Keywords

- Recurrence Relation
- Sequence
- Initial Condition
- Recurrence Formula
- Linear Recurrence
- Nonlinear Recurrence
- Homogeneous Recurrence
- Non-homogeneous Recurrence

7.7 Self- Assessment Questions

1. Solve the recurrence relation $T(n) = T(n-1) + 3$ with the initial condition $T(1) = 2$.
2. Find a closed-form solution for the recurrence relation $S(n) = 2S(n-1) + 3$ with the initial condition $S(0) = 1$.
3. Analyze the recurrence relation $T(n) = T(n/2) + 1$ with the initial condition $T(1) = 1$. Determine its asymptotic behavior.
4. Derive a recurrence relation for the Fibonacci sequence F(n) and solve it to find F(n).

7.8 Case Study

A biologist is studying the growth of a population of bacteria. The biologist observes that the population doubles every hour. Initially, there are 10 bacteria. Develop a recurrence relation to model the population growth over time.

Questions

1. Define the recurrence relation P(n) where n represents the number of hours elapsed.
2. Solve the recurrence relation to determine the population size after n hours.
3. Discuss the limitations of this model and suggest modifications for more realistic scenarios.

7.9 References

1. "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
2. "Discrete Mathematics" by Richard Johnsonbaugh

Unit -8

Pigeonhole principle

Learning Objectives:

- Develop strategies for identifying when and how to apply the Pigeonhole Principle in different types of problems.
- Be able to explain and illustrate the principle using simple, everyday examples.
- Use the Pigeonhole Principle to solve basic counting problems.

Structure:

8.1 Pigeonhole principle

According to the pigeonhole principle, if there are less boxes than there are available, fewer objects than there are boxes overall, and one must arrange objects in the boxes provided, then at least one box must contain more than one of these objects.

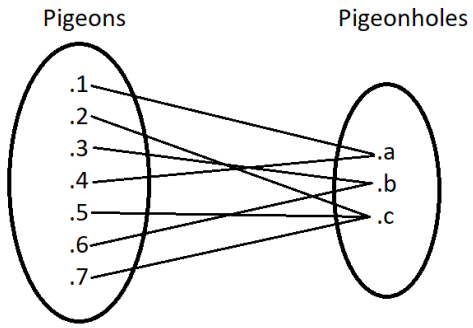


Figure 8.1 : Example of Pigeonhole of Principle

Therefore, if n pigeons are placed into n pigeonholes, then certain pigeonholes must contain more than one pigeon, according to the pigeonhole principle.

Demonstrating the identity of two potentially distinct values is a crucial stage in numerous proofs. There are situations when the Pigeonhole principle can be useful.

Theorem 8.1.1 ("Pigeonhole Principle"): "Suppose that n + 1 (or more) objects are put into n boxes". Then some box contains at least two objects.

Proof: Suppose each box contains at most one object.

Then the total number of objects is at most 1+1+...+1= n, Which shows a contradiction.

There are unexpected uses for this seemingly straightforward truth. Placing items in boxes in accordance with rules is usually the key, since this ensures that when two items wind up in the alike box, it's because they have a desirable association

Example 1: Two or more of the thirteen individuals have the same birth month.

Put the month names on the labels of 12 boxes. Place each individual in the box with the corresponding birth month label. At least two persons who have the same birth month will be in some box.

Example 2: "Suppose a1,...,an are integers. Then some "consecutive sum" $a_k + a_{k+1} + a_{k+2} +...+ a_{k+m}$ is divisible by n".

Let these n sums:

$$s_1 = a_1$$
$$s_2 = a_1 + a_2$$
$$s_3 = a_1 + a_2 + a_3$$
$$\vdots$$
$$s_n = a_1 + a_2 + \cdots + a_n$$

Since they are all successive sums, we are done if at least one of them can be divided by n. If not, there is a non-zero remainder when dividing each by n, therefore r1 = s1 mod n, r2 = s2 mod n, and so on. {1,2,3,...,n-1} is where these remainders are valued. Put these n - 1 values on the labels of the n - 1 boxes, and then place all of the n sums into the box labeled with its rest mod n.

8.2 "Chinese Remainder Theorem"

"Let m and n are relatively prime, and 0 < a <m and 0<b<n, then there is an integer x such that a mod m = a and x mod n = b".

Proof. The following integers have remainders of a when divided by m: a, a + m, a + 2m,..., a+ (n − 1)m. In order for one of these numbers to satisfy the required condition, we need to demonstrate that it has a remainder of b when divided by n.

If not, consider a contradiction. Tn-1 = a+ (n-1)m mod n; let the remainders be ro = a mod n, r1 = a+m mod n, and so on. Put the numbers 0,1,2,3,..., b-1, b+1,...n-1 on the labels of the n - 1 boxes. Place each r; in the corresponding box. When two leftovers, let's call ri and rj, arrive in the same box and j>i, ri = rj = r. This implies that

$$a + im = q_1 n + r \quad \text{and} \quad a + jm = q_2 n + r.$$

$$a + jm - (a + im) = q_2 n + r - (q_1 n + r)$$
$$(j - i)m = (q_2 - q_1)n.$$

This indicates that n | (j – i) since n is comparatively prime to m. However, n (ji) while i and j are different and in $\{0,1,2,\ldots,n-1\}, 0 < j - i < n$. The proof comes to an end with this

contradiction.

Essentially the same technique can be used to prove more generic variants of the Pigeonhole Principle. The following would be a natural generalization: There will be at least m objects in some box if X objects are placed inside n boxes.

8.3 Applications of Pigeonhole Principle

- **Birthday Problem:** In a group of people, the probability that at least two people share the same birthday is high because there are only 365 days in a year (pigeonholes) but potentially more than 365 people (pigeons).
- **Subset Sums:** In computer science and cryptography, the pigeonhole principle is used to analyze algorithms and situations where it's necessary to find combinations that sum to a certain value.

Example 1: Suppose you have 8 jars and 25 apples. If you distribute these apples into the jars, what can you conclude using the pigeonhole principle?

Solution

To solve this using the pigeonhole principle, we need to compare the number of items (apples) with the number of containers (jars).

- **Number of apples (n):** 25
- **Number of jars (m):** 8

Applying this to our scenario:

- n=25 (apples)
- m=8 (jars)

Since n>m, which is 25 > 8, there must be at least one jar that contains more than one apple. This conclusion follows directly from the pigeonhole principle because there are more apples than jars available to place them into.

Therefore, using the pigeonhole principle, we can conclude that at least one jar will contain more than one apple when distributing 25 apples into 8 jars.

8.4 Summary

The Pigeonhole Principle "states that if n items are placed into m containers, with n>m, then at least one container must contain more than one item". This fundamental concept in combinatorics is useful for solving problems involving counting and probability. The generalized version asserts that if n items are distributed among m containers, at least one will hold at least $\lceil n/m \rceil$ items. The principle aids in developing problem-solving strategies, constructing proofs, and understanding real-world applications in various fields such as computer science, scheduling, and error detection.

8.5 Keywords

- Combinatorics
- Counting
- Containers
- Distribution
- Generalized Pigeonhole Principle

8.6 Self- Assessment Questions

1. If you have 10 pairs of socks in a drawer, all mixed up, how many socks must you pull out to ensure you have at least one matching pair?
2. In a group of 50 people, how many people must have the same birth month at a minimum?
3. You have 25 students and 5 different classrooms. Prove that at least one classroom must have at least 5 students.
4. In a computer system with 1000 users and 100 possible passwords, prove that at least one password must be used by at least 10 users if each user must choose one password.

8.7 Case Study

You have a drawer containing 113 pairs of red shirts and 100 pairs of blue shirts, all mixed together. You are in the dark and cannot distinguish between the colors by touch.

Question : How many shirts must you pull out to be sure you have a pair of the same color?

8.8 References

1. "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
2. "Discrete Mathematics" by Richard Johnsonbaugh

Unit - 9

Graph Theory

Learning Objective

- The learning objectives of graph theory typically focus on understanding the fundamental concepts, properties, and algorithms associated with graphs
- Understand the basic terminology used in graph theory, such as vertices (nodes), edges, degree of a vertex, adjacency, and paths.
- Learn different ways to represent graphs, including adjacency matrix, adjacency list, and edge list.

Structure

9.1 Introduction

Graphs are mathematical structures use to depict couple wise relations among objects, are the subject of graph theory, a prominent field of discrete mathematics. Graph theory in discrete mathematics is examined in detail below:

An ordered pair, G=(V,E), connecting pairs of vertices, is the definition of a graph G. V is a collection of vertices, also known as nodes, and E is a set of edges, also known as arcs.

9.2 Types of Graphs:

1. Undirected graph

An undirected graph is one in which none of the edges have a specified direction assigned to them.

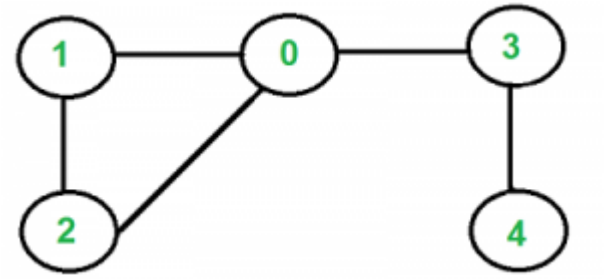


Figure 9.1: Example of Undirected Graph

Characteristics of an Undirected Graph:

- An undirected graph's edges are bidirectional by nature.
- Since the edges in an undirected graph have no orientation, there is no such thing as a "parent" or "child" vertex.
- Loops, or edges that join a vertex to itself, can be found in an undirected graph.
- The total number of edges that are linked to each vertex equals its degree.

Applications of Undirected Graph

- Undirected graphs are used to simulate social networks, in which nodes represent individuals and edges reflect the interactions between them.
- Traffic flow optimization: To simulate how cars move on road networks, undirected graphs are utilized in traffic flow optimization. Intersections and road segments are represented by the graph's vertices and edges, respectively, by the connections that link

them. Infrastructure planning and traffic flow optimization may both be done with the graph.

- Analyzing connections between web sites on the internet is possible using undirected graphs. An edge represents a link between web sites, while a vertex represents a single web page.

2. Directed graph

It is characterized as a exacting type of graph in which each edge has a direction assigned to it.

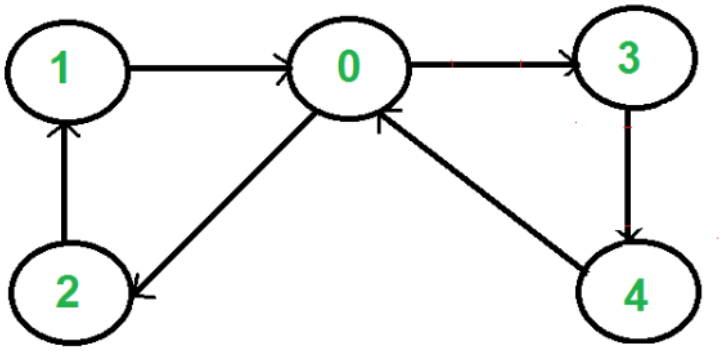


Figure 9.2: Example of Directed Graph

Characteristics of Directed Graph

- Directed edges: A directed graph shows a one-way connectivity between vertices by having edges that are connected to a direction.
- In a directed graph, there are two distinct degree measurements for each vertex: indegree and outdegree. A vertex's indegree is the number of edges that come into it, and its outdegree is the number of edges that leave it.
- Cycles: Paths that begin and terminate at the same vertex and have at least one edge are known as cycles in directed graphs. Understanding feedback loops and other graph patterns can be aided by cycles.
- The analysis of reachability between vertices may be done using paths in a directed graph, which follow the direction of the edges.

Applications of Directed Graph

Social networks: According to this concept, social networks are directed graphs with each individual being a vertex and interactions, such friendships or followers, being represented as

edges.

Roads, airports, and subway systems are examples of transportation networks. These systems may be represented as directed graphs, with vertices denoting places and edges denoting the relationships between them.

Networks of computers: Networks of computers, like the internet, may be seen as directed graphs, where vertices stand for hardware like computers or routers and edges indicate the relationships between them.

Project management: A directed graph with tasks as vertices and dependencies between them as edges may be used to model project management.

3. Weighted Graph

A graph with weights given to its edges—which might indicate cost, distance, or a variety of other relative measuring units—is called a weighted graph.

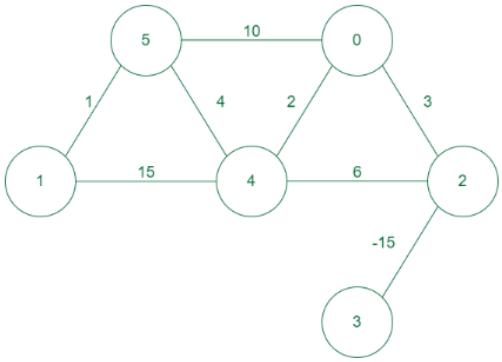


Figure 9.3: Example of Directed Graph

Applications of Weighted Graph:

- 2D matrix games: With 2D matrix games, you may determine the best path for the highest sum along the beginning and finishing points. You can find a lot of these game types online.
- Weighted graphs are used to determine the lowest spanning tree from a graph that shows the least amount of money needed to travel every node in the graph.
- Graphs representing constraints: Graphs are frequently used to depict restrictions among things. used in circuit design, artificial intelligence, scheduling, product creation, and asset allocation.

4. Unweighted graph

An unweighted graph is one in which there are no charges or weights attached to any of the edges. Alternatively, they just show that there is a link between two vertices.

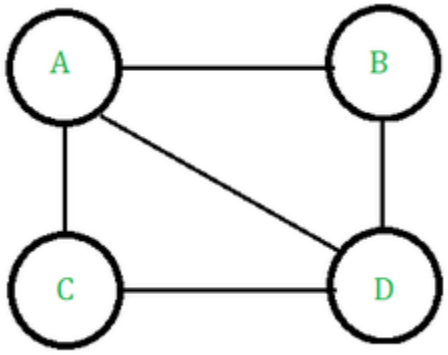


Figure 9.4: Example of Unweighted graph

Applications of Unweighted Graph:

- When presenting data that are unrelated in magnitude, unweighted graphs are utilized. Computation flow is modelled using unweighted graphs.
- An illustration of picture segmentation, in which edges stand in for adjacency connections and pixels for nodes.
- State space representation in AI decision-making and problem-solving procedures. depiction of networks of information, like the Internet.

Common Terminology:

- **Adjacent Vertices:** "Two vertices connected by an edge".
- **Degree of a Vertex:** Number of edges incident to the vertex. In a digraph, there are in-degree and out-degree.
- **Path:** "A sequence of vertices where each adjacent pair is connected by an edge".
- **Cycle:** "A path that starts and ends at the same vertex without repeating any edge".
- **Connected Graph:** There is a path between any pair of vertices.
- **Complete Graph:** "Every pair of distinct vertices is connected by a unique edge".
- **Subgraph:** "A graph formed from a subset of the vertices and edges of another graph".

Example 1: Use induction to demonstrate Euler's formula based on the graph's edge count.

Let P(n) be the assertion that $v-n+f=2$ holds for each planar graph with n edges. We shall demonstrate that for any $n\geq 0$, P(n) is true.

Base case: there is only one graph, one isolated vertex, with zero edges. Euler's formula is valid in this instance since v=1, f=1, and e=0.

Let P(k) hold for any arbitrarily chosen k≥0. Now imagine an arbitrary graph with v vertices, f faces, and k+1 edges. Regardless of this graph's appearance, we may subtract one edge to get a graph with k edges to which the inductive hypothesis can be applied.

There are two options. First, we may have an incident edge to a vertex of degree 1. Here, eliminate that vertex as well. Because the edge and vertex were removed without reducing the number of faces, the smaller graph will now meet v−1−k+f=2 under the induction hypothesis. As needed, v−(k+1)+f=2 is obtained by adding the edge and vertex back. In example number two, the edge we eliminate is incident to vertices with a degree larger than one. Eliminating the edge in this instance will result in one fewer face and the same number of vertices. Thus, based on the inductive premise, v−k+f−1=2. As needed, v−(k+1)+f=2 will result from adding the edge back.

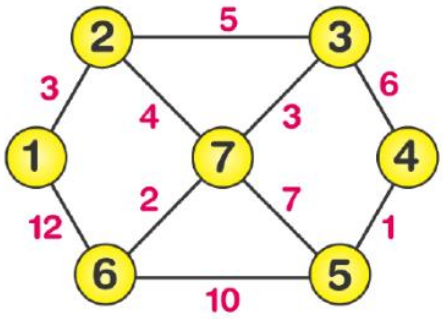
9.3 Adjacency Matrix

An easy labeled graph is represented by the adjacency matrix, also known as the connection matrix. It is a matrix with rows and columns that has 0 or 1 in the position of (Vi, Vj) depending on whether or not Vi and Vj are nearby. It is a condensed method of expressing the finite network made up of n vertices in a m x m matrix M. Adjacency matrix is also known as a vertex matrix at times, and its general definition is

$$\begin{Bmatrix} 1 & if\ P_i \rightarrow P_j \\ 0 & otherwise \end{Bmatrix}$$

An undirected graph's adjacency matrix is symmetric. According to this, the values in the ith row and jth column and the jth row and ith column are the same. Furthermore, matrix multiplication is an interesting fact. If there is a nonzero value in the ith row and jth column of the adjacency matrix after it has been multiplied by itself (a process known as matrix multiplication), a path with a length of two can be found from Vi to Vj. Even though a route has been constructed, it is not specified. The nonzero value denotes the quantity of unique pathways that are available.

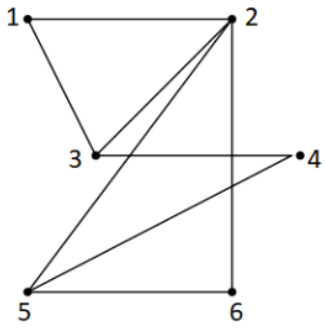
Example2: For the provided undirected weighted graph, note the adjacency matrix.



Solution

$$A = \begin{bmatrix} 0 & 3 & 0 & 0 & 0 & 12 & 0 \\ 3 & 0 & 5 & 0 & 0 & 0 & 4 \\ 0 & 5 & 0 & 6 & 0 & 0 & 3 \\ 0 & 0 & 6 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 10 & 7 \\ 12 & 0 & 0 & 0 & 10 & 0 & 2 \\ 0 & 4 & 3 & 0 & 7 & 2 & 0 \end{bmatrix}$$

Example 3: For the provided undirected weighted graph, note the adjacency matrix.



Solution

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

9.4 Graph Isomorphism

Graph isomorphism is a fundamental concept in graph theory that concerns whether two graphs are structurally identical.

Formally, two graphs

$G1 = (V1, E1))$ and $G2 = (V2, E2))$ are isomorphic "if there exists a bijection $f: V1 \rightarrow V2$ such that for any two vertices $u, v \in V1$ if and only if $(f(u), f(v)) \in E2$".

Here are some case study questions related to graph isomorphism:

Case Study 1: Finding Isomorphic Graphs

Scenario: You are given two graphs G1 and G2 with vertex sets {1,2,3,4} and edge sets {{1,2},{2,3},{3,4}} for G1, and {A,B,C,D} and edge sets {{A,B},{B,C},{C,D}} for G2.

Question : Determine if G1 and G2 are isomorphic.

Answer: To determine if G1 and G2 are isomorphic, we can check their structures:

- Both graphs have 4 vertices.
- Both have edges connecting consecutive vertices.

To find an isomorphism:

- Map 1→A,2→B,3→C,4→D1.

Verify:

- "{1,2} in G1 corresponds to {A,B} in G2",
- "{2,3} in G1 corresponds to {B,C}in G2",
- "{3,4} in G1 corresponds to {C,D} in G2".

Since we can establish a bijection between vertices that preserves edge connections, G1 and G2 are indeed isomorphic.

Case Study 2: Non-Isomorphic Graphs

Scenario: Consider two graphs G1 and G2 with vertex sets {1,2,3} and edge sets {{1,2},{2,3}} for G1, and {A,B,C} and edge sets {{A,B},{B,C}} for G2.

Question 2: Determine if G1 and G2 are isomorphic.

Answer: To determine if G1 and G2 are isomorphic, we analyze their structures:

- G1 has vertices {1,2,3} with edges {{1,2},{2,3}}.
- G2 has vertices {A,B,C} with edges {{A,B},{B,C}}.

For G1 and G2 to be isomorphic, there must exist a bijection between their vertices that preserves adjacency. However:

- G1 has vertex 3 connected to vertex 2,
- G2 has vertex C connected to vertex B.

No bijection can map 1→A , 2→B , 3→C (or any other mapping) that preserves the adjacency of G1 and G2.

Therefore, G1 and G2 are not isomorphic.

9.5 Summary

1. **Graphs and Their Components**:
    - **Vertices (Nodes)**: Represent objects or entities.
    - **Edges**: Represent relationships or connections between vertices.
    - **Degree of a Vertex**: Number of edges incident to a vertex.
    - **Paths**: Sequence of edges connecting vertices.
    - **Cycles**: Paths that start and end at the same vertex.
2. **Graph Representation**:
    - **Adjacency Matrix**: Represents connections between vertices using a matrix.
    - **Adjacency List**: Represents connections using lists or arrays.
    - **Edge List**: Represents connections using a list of edges.

9.6 Keywords

- Unweighted Graph
- Adjacent Vertices
- Subgraph
- Unweighted graph
- Weighted Graph
- Directed graph
- Graph theory

9.7 Self- Assessment Questions

1. Define the following terms in graph theory: vertex, edge, degree of a vertex, adjacency, path, cycle, connected graph. Provide examples where applicable.
2. Differentiate between directed and undirected graphs. Give an example scenario where each type of graph might be used.
3. Explain the difference between a simple graph and a multigraph. Provide examples of each.

9.8 Case Study

Imagine you are tasked with analyzing a social network represented as a graph of users (nodes) and friendships (edges). How would you identify influential users (nodes with high degree centrality)? What graph metrics or algorithms would you use to recommend new friendships to users?

9.9 References

1. "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
2. "Discrete Mathematics" by Richard Johnsonbaugh

Unit - 10

Tree

Learning Objective:

- In graph theory, the learning objectives related to trees focus on understanding the fundamental properties, structures, and algorithms associated with trees as a specific type of graph.
- Basic Concepts and Properties
- Understand the definition of a tree as a connected acyclic graph.
- Differentiate between trees, forests (disjoint union of trees), and general graphs.

Structure

10.1 Introduction

According to graph theory, a tree is a particular type of graph having a number of distinctive features. Any two vertices in an undirected network with precisely one path between them are called a tree. On the other hand, an acyclic linked graph is what a tree is.

Basic Terminology

- **Rooted Tree**: "Rooted tree are those tree in which one vertex is designated as the root. This gives a notion of hierarchy, with parent and child nodes".
- **Leaf**: A vertex with degree 1, except for the root if it's a single vertex.
- **Internal Node**: A node that has at least one child.
- **Height of a Tree**: "The length of the longest path from the root to a leaf".

Properties of Trees

1. **Uniqueness of Paths**: There is exactly one simple path between any two vertices.
2. **Edges and Vertices**: A tree with n vertices always has n−1 edges.
3. **Spanning Trees**: A subgraph of a graph with all of its vertices included is called a spanning tree. There is at least one spanning tree in every linked graph.

Mathematical Representation

A tree can be represented mathematically in several ways:

- **Adjacency List**: Each vertex has a list of adjacent vertices.
- **Adjacency Matrix**: A matrix A where A[i][j]=1 if there is an edge between vertices i and j, and 0 otherwise.
- **Edge List**: A list of all edges represented as pairs of vertices.

10.2 Types of Trees

Rooted vs unrooted tree: Rooted and unrooted trees are important classifications in graph theory.

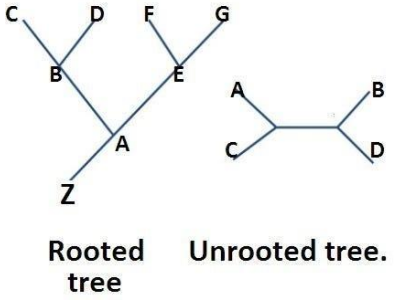


Figure 10.1: Rooted vs unrooted tree

All other nodes in a rooted tree are descendants of the one selected root node. Algorithms in computer science and hierarchical structures benefit from them.

An unrooted tree, which lacks a specified root node, can be used to describe chemical structures and evolutionary relationships as well as computer science procedures like graph isomorphism and clustering.

Ordered vs Unordered tree: In graph theory, there are two different categories of trees: ordered and unordered. The offspring of every node in an unordered tree are not arranged in any particular sequence, but those in an ordered tree are.

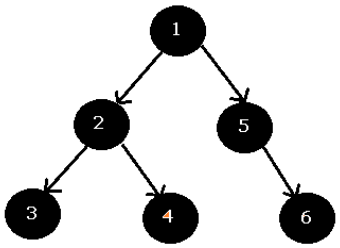


Figure 10.2 : Ordered Tree

Ordered trees are helpful in situations like mathematical expressions where the node order is important.

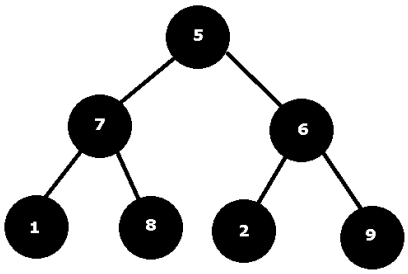


Figure 10..3: Unordered Tree

Unordered trees are helpful when describing sets or other scenarios where the order of nodes is irrelevant. By being aware of these variations, we may select the right structure for our needs and learn more about graph theory.

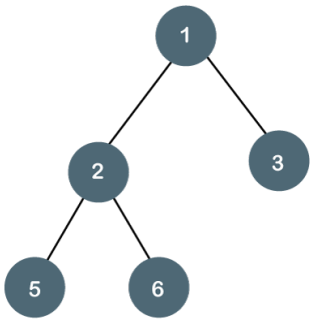


Figure 10.4: Binary tree

Multiway Trees: Trees that allow a node to have more than two offspring are known as multiway trees. Multiway trees include octrees, quad trees, and ternary trees, to name a few.

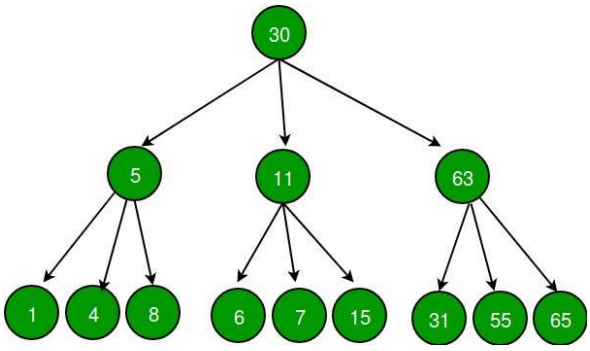


Figure 10.5: Ternary tree

Perfect vs. Balanced Trees: A binary tree that has two offspring on each internal node and leaves that are all at the same level is called a perfect binary tree. A balanced tree is one in which there is a maximum one-height difference between any node's subtrees. While all balanced trees are not perfect, all perfect trees are balanced.

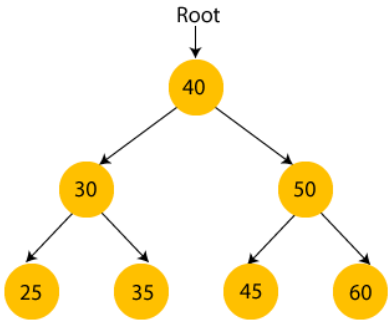


Figure 10.6 :Binary Search Tree

Binary search tree (BST) : An effective and straightforward data structure for quick searching, insertion, and deletion is a binary search tree (BST). The elements are arranged in such a way that keys smaller than the parent node's key are found in the left subtree, while keys larger than the parent are found in the right subtree. Although BSTs are often employed in computer science, performance problems can arise when they become imbalanced.

AVL trees : Binary search trees known as AVL trees keep their equilibrium by making sure that each node's left and right subtrees have height differences of no more than one. They were created by Adelson-Velsky and Landis, and they are frequently utilized for quick searching and insertion in databases and compilers.

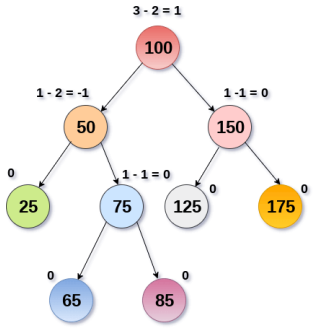


Figure 10.7: AVL-tree

B-trees: A self-balancing search tree that allows for more than two offspring per node is called a B-tree. File systems, databases, and other applications where huge volumes of data need to be effectively stored and retrieved employ them because they are designed for access from external storage.

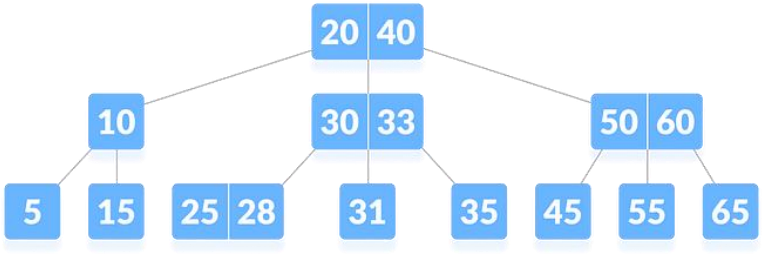


Figure 10.7: B-tree

Red-Black : Another kind of self-balancing binary search tree is the red-black tree, which makes sure that no path from the root to a leaf is more than twice the length of any other path. They provide quick searching and insertion in databases, compilers, and computer networks.

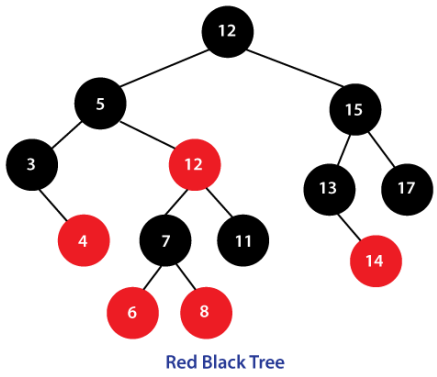


Figure 10.8: Red Black tree

10.3 Spanning Tree

Spanning tree are those tree which are connected graph G is a subgraph that includes all the vertices of G and is a tree (a connected acyclic graph).

Definition: A spanning tree of a graph G is a subgraph that is a tree and includes all the vertices of G.

Properties:

- A spanning tree of G has |V|−1 edges, where |V| is the number of vertices in G.
- It is connected, “meaning there is a path between any pair of vertices in the spanning tree”.
- It is acyclic, meaning there are no cycles in the spanning tree.

Importance:

- Spanning trees are fundamental in network design and optimization problems.
- They can be used to find efficient communication routes or to ensure all nodes in a network are interconnected.

Construction:

- **Depth-First Search (DFS)** and **Breadth-First Search (BFS)** are commonly used algorithms to construct spanning trees.

Applications:

- Network design: Ensuring connectivity while minimizing cost (e.g., cable layout).
- Circuit design: Ensuring all components are connected with the minimum number of links.

- Clustering and hierarchical clustering algorithms.

Types of Spanning Trees:

- "The spanning tree with the shortest feasible sum of edge weights is known as the Minimum Spanning Tree (MST)".
- "The spanning tree with the highest possible sum of edge weights is known as the maximum spanning tree".

10.4 Summary

Trees in graph theory provide a structured and efficient way to model hierarchical relationships, connectivity patterns, and decision-making processes. They serve as foundational structures in computer science, mathematics, and various fields where organizing and navigating through data efficiently is crucial. Mastering the concepts and algorithms related to trees enhances problem-solving skills and facilitates the design of efficient algorithms for practical applications.

10.5 Keywords

- Tree
- Depth-First Search
- Minimum Spanning Tree
- Spanning tree

10.6 Self- Assessment Questions

1. What is a tree in graph theory?
2. State the minimum number of edges in a tree with n vertices.
3. Differentiate between a tree and a forest.
4. Explain how Depth-First Search (DFS) can be used to determine if a graph is a tree.
5. Name a practical application of trees in computer science.
6. What is a spanning tree of a graph?
7. State one algorithm used to find a Minimum Spanning Tree (MST) in a graph.

10.7 Case Study

A telecommunications company plans to establish a new network infrastructure in a city. Describe how you would use Minimum Spanning Trees (MSTs) to determine the optimal placement of communication hubs to ensure maximum connectivity with minimal cost.

10.8 References

1. "Discrete Mathematics and Its Applications" by Kenneth H. Rosen
2. "Discrete Mathematics" by Richard Johnsonbaugh

Unit -11

Inference Theory

Learning Objective:

- Inference theory typically refers to a branch of logic or philosophy that deals with the process of reasoning and drawing conclusions from premises
- Define logical inference and its role in deductive reasoning.
- Differentiate between deductive and inductive reasoning.
- Study formal systems of logic, including propositional and predicate logic.
- Learn basic rules of inference (modus ponens, modus tollens, etc.) and their application in deriving conclusions from premises.

Structure

11.1 Introduction

Rules of Inference: All mathematical theorems, as well as those in other fields, are backed up by supporting evidence. All these proofs are essentially a series of arguments that provide solid proof that the theory is true. In order to infer new claims and finally demonstrate the theorem's validity, the arguments are connected using Rules of Inference.

Argument :A series of premises and claims that conclude with a conclusion is called an argument.

Validity: A logical argument is considered valid only if it is structured in a way that precludes the possibility of the premises being true and the conclusion being untrue.

Fallacy: An erroneous line of thought or error that produces unsound thinking.
Simple arguments can be the foundation for more complex, legitimate arguments. It is crucial to use some straightforward arguments that have been proven to be correct. We refer to these arguments as Rules of Inference. The most popular Rules of Inference are shown in the table below:

Rules of Inference	Tautology
$p, p \to q, \therefore q$	$(p \wedge (p \to q)) \to q$
¬q, p → q, ∴ ¬p	(¬q ∧ (p → q)) → ¬p
p → q, q → r, ∴ p → r	((p → q) ∧ (q → r)) → (p → r)
¬p, p ∨ q, ∴ q	(¬p ∧ (p ∨ q)) → q
p, ∴ (p ∨ q)	p → (p ∨ q)
(p ∧ q) → r, ∴ p → (q → r)	((p ∧ q) → r) → (p → (q → r))
p ∨ q, ¬p ∨ r, ∴ q ∨ r	((p ∨ q) ∧ (¬p ∨ r)) → (q ∨ r)

Table 11.1: Rules of Inference

11.2 Logical Connectivities

In discrete mathematics, especially in the context of logic and set theory, "logical connectives" refer to the fundamental operations used to combine propositions (statements) to form more complex statements. These connectives are crucial for constructing logical expressions and reasoning about their truth values. Here are the primary logical connectives and their roles:

1. **Negation (¬)**: Also known as "not." It reverses the truth value of a proposition.
2. **Conjunction (∧)**: Also known as "and." It connects “two propositions and is true only when both propositions are true”.
3. **Disjunction (∨)**: Also known as "or." It connects “two propositions and is true if at least one of the propositions is true”.
4. **Implication (⇒)**: Also known as "if...then." It represents a conditional statement where p⇒qp means "if p then q."
5. **Biconditional (⇔)**: Also known as "if and only if." It connects two propositions and is true if both propositions have the same truth value.

These logical connectives form the basis for constructing logical expressions, which are evaluated based on the truth values of their constituent propositions. They are fundamental in formalizing arguments, reasoning about conditions, and defining relationships in various domains of discrete mathematics, computer science, and philosophy.

In formal logic, these connectives are often represented using symbols (¬, ∧, ∨, ⇒, ⇔) and are used to build complex logical expressions from simple propositions, enabling precise analysis and deduction in a structured manner.

11.3 Truth Table

A truth table is an essential tool in logic that displays a compound statement's truth values for every conceivable combination of the truth values of the individual propositions that make up the proposition (variables). It aids in figuring out the truth value of intricate logical statements and comprehending how they behave in various scenarios.

Components of a Truth Table

1. **Variables**: Represented by p,q,r,…, these are propositions or statements that can be either true (T) or false (F).
2. **Columns**: Each column in the truth table corresponds to a variable or a compound proposition.

3. **Rows**: Each row represents a unique combination of truth values for the variables.

Steps to Construct a Truth Table

1. **Identify Variables**: List all variables involved in the logical expression.
2. **Determine Possible Combinations**: For n variables, there are 2^n possible combinations of truth values (since each variable can independently be true or false).
3. **Construct Columns**: Create columns in the truth table for each variable and for the compound proposition(s) being evaluated.
4. **Assign Truth Values**: Fill in each row with all possible combinations of truth values for the variables.

Example 1: Let's construct a simple truth table for the logical expression $P \wedge Q$

Solution

P	Q	pΛq
T	T	T
T	F	F
F	T	F
F	F	F

Example 2: Let's construct a simple truth table for the logical expression $P \vee Q$

P	Q	Pvq
T	T	T
T	F	T
F	T	T
F	F	F

Example 3: Let's construct a simple truth table for the logical expression $P \rightarrow Q$

P	Q	p→q
T	T	T
T	F	F
F	T	T
F	F	T

Example 4: Let's construct a simple truth table for the logical expression $(p \vee \sim q) \rightarrow \sim r$

Solution

p	Q	r	~q	pv~q	~r	(pv~q) →~r
T	T	T	F	T	F	F
T	T	F	F	T	T	T
T	F	T	T	T	F	F
T	F	F	T	T	T	T
F	T	T	F	F	F	T
F	T	F	F	F	T	T
F	F	T	T	T	F	F
F	F	F	T	T	T	T

Example 4: Let's construct a simple truth table for the logical expression $p \rightarrow (q \rightarrow r)$

Solution

P	q	R	p → q	(p → q) → r	q → r	p → (q → r)
T	T	T	T	T	T	T
T	T	F	T	F	F	F
T	F	T	F	T	T	T
T	F	F	F	T	T	T
F	T	T	T	T	T	T
F	T	F	T	F	F	T
F	F	T	T	T	T	T
F	F	F	T	F	T	T

11.4 Summary

Inference theory encompasses a range of topics from basic logic and reasoning skills to formal systems of deduction and their practical applications. Mastering inference theory involves understanding logical principles, developing analytical skills, and applying them to solve problems and evaluate arguments effectively across different disciplines.

11.5 Keywords

1. Function
2. Relations
3. Graph Theory

11.6 Self- Assessment Questions

1. How does modus ponens work in logical inference?
2. Give an example of an inductive reasoning process.
3. What role does inference theory play in artificial intelligence?

4. How can logical reasoning skills benefit everyday decision-making?
5. Differentiate between deductive and inductive reasoning.
6. Name two basic rules of inference in propositional logic.
7. Explain the concept of logical validity.

11.7 Case Study

A lawyer needs to construct a persuasive argument for a court case. Discuss how principles of inference theory, such as logical validity and soundness, can be applied to formulate and defend legal arguments effectively.

11.8 References

1. Discrete Mathematics and Its Applications" by Kenneth H. Rosen
2. "Discrete Mathematics" by Richard Johnsonbaugh