ASSIGNMENT FINAL REPORT

Qualification	**Pearson BTEC Level 5 Higher National Diploma in Computing**		
Unit number and title	**Unit 18: Discrete Maths**		
Submission date	11 / 12 / 2025	**Date Received 1st Submission**	
Re-submission Date		**Date Received 2nd Submission**	
Student Name	NGUYEN VIET PHUC	**Student ID**	BD00671
Class	SE07202	**Assessor name**	Nguyen Thi Su

Plagiarism

Plagiarism is a particular form of cheating. Plagiarism must be avoided at all costs and students who break the rules, however innocently, may be penalised. It is your responsibility to ensure that you understand correct referencing practices. As a university level student, you are expected to use appropriate references throughout and keep carefully detailed notes of all your sources of materials for material you have used in your work, including any material downloaded from the Internet. Please consult the relevant unit lecturer or your course tutor if you need any further advice.

Student Declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I declare that the work submitted for assessment has been carried out without assistance other than that which is acceptable according to the rules of the specification. I certify I have clearly referenced any sources and any artificial intelligence (AI) tools used in the work. I understand that making a false declaration is a form of malpractice.

Student's signature	PHUC VIET

Grading grid

P1	P2	P3	P4	M1	M2	D1	D2

ASSIGNMENT GROUP WORK

Qualification	**Pearson BTEC Level 5 Higher National Diploma in Computing**		
Unit number and title	**Unit 18: Discrete Maths**		
Submission date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Group number:	**Student names & codes**	**Final scores**	**Signatures**
	Nguyen Viet Phuc - BD00671		Phuc Viet
	Do Duc An - BD00659		An
	Huynh Le Duc Thang - BD00339		Duc Thang
	Dao Duy Vien - BD00726		Duy Vien
Class	SE07202	**Assessor name**	Nguyen Thi Su

Plagiarism

Plagiarism is a particular form of cheating. Plagiarism must be avoided at all costs and students who break the rules, however innocently, may be penalised. It is your responsibility to ensure that you understand correct referencing practices. As a university level student, you are expected to use appropriate references throughout and keep carefully detailed notes of all your sources of materials for material you have used in your work, including any material downloaded from the Internet. Please consult the relevant unit lecturer or your course tutor if you need any further advice.

Student Declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I declare that the work submitted for assessment has been carried out without assistance other than that which is acceptable according to the rules of the specification. I certify I have clearly referenced any sources and any artificial intelligence (AI) tools used in the work. I understand that making a false declaration is a form of malpractice.

P5	P6	P7	P8	M3	M4	D3	D4

OBSERVATION RECORD

Student	**Nguyen Viet Phuc – BD00671**

Description of activity undertaken

M3 Simplify a Boolean equation using algebraic methods.

D3 Design a complex system using logic gates.

M4 Validate whether a given set with a binary operation is indeed a group

D4 Explore, with the aid of a prepared presentation, the application of group theory relevant to your given example.

Assessment & grading criteria

How the activity meets the requirements of the criteria

Student signature:	**Phuc Viet**	**Date:**	**10/12/2025**
Assessor signature:		**Date:**	
Assessor name:			

❒ Summative Feedback:	❒ Resubmission Feedback:

Grade:	Assessor Signature:	Date:

Internal Verifier's Comments:

Signature & Date:

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Based on the sample structure you provided and the actual content of your report file, here are the INTRODUCTION and CONCLUSION sections rewritten in a professional style listing the learning outcomes (LO) and criteria (P, M, D).

Discrete mathematics is a core and essential field, providing the foundation for logical thinking and crucial mathematical tools for Computer Science. This subject equips students with the ability to model, analyze, and solve complex problems in information technology, from database management and algorithm optimization to system security. This report aims to deepen understanding of how discrete mathematics shapes the IT landscape by exploring pillars such as set theory, graph theory, Boolean algebra, and abstract algebra.

This report includes the following chapters:

CHAPTER 1: EXAMINE SET THEORY AND FUNCTIONS APPLICABLE TO SOFTWARE ENGINEERING. (LO1)

- P1: Perform algebraic operations on sets in a formulated mathematical problem.
- P2: Determine the cardinality of a given multiset/bag.
- M1: Determine the inverse function of a function using appropriate mathematical techniques.
- D1: Construct corresponding proof principles to demonstrate properties of defined sets.
- CHAPTER 2: ANALYZE MATHEMATICAL STRUCTURES OF OBJECTS USING GRAPH THEORY. (LO2)
- P3: Model contextual problems using trees, both quantitatively and qualitatively.
- P4: Use Dijkstra's algorithm to find the shortest path spanning tree in a graph.
- M2: Evaluate the existence of Euler and Hamilton cycles in an undirected graph.
- D2: Construct a proof for the Five Color Theorem.

CHAPTER 3: INVESTIGATE SOLUTIONS TO PROBLEM SITUATIONS USING THE APPLICATION OF BOOLEAN ALGEBRA. (LO3)

- P5: Diagram a binary problem in the application of Boolean algebra.
- P6: Create a truth table and corresponding Boolean equation from a real-world application scenario.
- M3: Simplify Boolean equations using algebraic methods.
- D3: Design a complex system using logic gates (e.g., a divisibility test circuit for 3).

CHAPTER 4: EXPLORE APPLICABLE CONCEPTS WITHIN ABSTRACT ALGEBRA. (LO4)

- P7: Describe the distinguishing characteristics of different binary operations performed on the same set.
- P8: Determine the order of a group and the order of a subgroup in the given examples.
- M4: Verify whether a given set with a binary operation is actually a group.
- D4: Explore the application of group theory related to the given example (RSA algorithm) with the help of the prepared presentation.

CHAPTER 1 – LO1: EXAMINE SET THEORY AND FUNCTIONS APPLICABLE TO SOFTWARE ENGINEERING

1.1 (P1) Perform algebraic set operations in a formulated mathematical problem

1.1.1 Definition of Sets and Set Notation

A set is an unordered collection of objects, called elements or members of the set. (Cuemath , 2025)

❖ **Basic symbols:**

- Set name: Usually denoted by uppercase letters, for example: A,B,C.
- Elements: Usually denoted by lowercase letters, for example: a,b,c.
- Belonging/not belonging notation:

$$a \in A: read\ as\ "a\ is\ an\ element\ of\ set\ A".$$

$$a \notin A: read\ as\ "a\ is\ not\ an\ element\ of\ set\ A"$$

❖ **Ways to describe sets:**

- Roster Method: List all the elements of the entire set between the two curly braces {}.

  For example: The set V consisting of vowels in the English alphabet is written as V={a,e,i,o,u}.

- Set-Builder Notation: Specifies the properties that each element of a set must have.

  For example: the set O of odd positive integers less than 10 can be written as $0 = \{x\ |x\ is\ a\ positive\ integer\ and\ x < 10\ \}$

1.1.2 Basic Set Operations and Symbols

For now, in this section, I will focus on the basic definitions such as subsets, Cartesian products, as well as on each of the other basic concepts. Operations such as union, intersection, and difference will be discussed in more detail in the next lectures. However, based on general knowledge, I can briefly explain as follows:

- Union (∪): This union of two sets A and B is one of the sets that contains all elements that belong to A, or to B, or to both.
- Intersection (∩): The intersection of two sets A and B is the set that contains all elements that belong to both A and B.

- Difference (−): For each difference of sets A and B (denoted A−B), there is a set that contains elements that belong to A but not to B.

1.1.3 Venn Diagrams for Set Operations

"A Venn diagram consists of circles that represent different sets. Each circle overlaps with one or more other circles, demonstrating the common elements between the sets. The areas where the circles overlap represent the intersection of the sets, while the areas that do not overlap represent the differences."(Xmind , 2025)

- Universal Set (U): These are the sets that contain all the objects under consideration, usually represented by a rectangle.
- Subsets: These are usually represented by circles or other geometric figures inside the rectangle.
- Elements: Sometimes they are also represented by points inside circles.
- Example of subset representation: The diagram below illustrates that A is a subset of B $A \subseteq B$, meaning that every element of A is also an element of B.

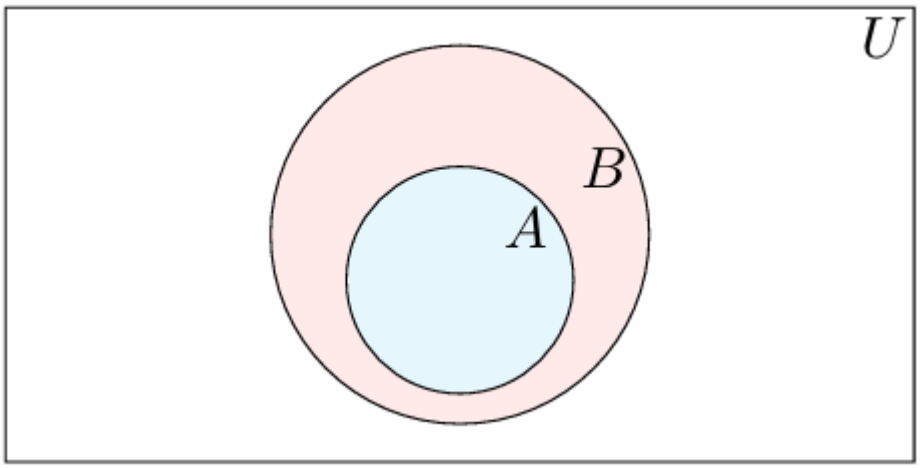


Figure 1. ***Venn diagramsh***

1.1.4 Cartesian Product of Sets

"Let A and B be sets. The Cartesian product of A and B, denoted as A×B, is the set of all ordered pairs (a,b) , where a ∈ A and b ∈ B. Hence, $A \times B = \{(a,b) \mid a \in A \text{ and } b \in B\}$" (Rosen, 2011)

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

- Important properties: The Cartesian product is not commutative, meaning $A \times B \neq B \times A$ unless $A = B$ or one of two sets is empty .
- Number of elements (force): If $|A| = m \; and \; |B| = n$, $then \; |A \times B| = mn$.
- Example: If $A = \{1,2\} and \; B = \{a, b, c\}$, $then$:

  $A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c), \}$.

1.1.5 Applications in Computing Context

- Set theory is the foundation for many areas of computer science. Here are some of the typical examples:
  - Databases: Querying a relational database like SQL is based on set operations. For example, JOIN is equivalent to each Cartesian product, UNION is the union, and INTERSECT is the intersection. Each table in the database can be viewed as a set of tuples.
  - Data Structures and Algorithms: With each data structure Set, HashSet, Map in every programming language (Java, Python, C++) is a direct realization of the mathematical concept of sets. They are used to store unique elements and from here to perform set operations very efficiently.
  - Formal Languages and Automata Theory: According to the alphabets of a language, it is defined as a finite set of characters. A language is defined as a set (possibly infinite) of strings created from that alphabet.
  - Artificial Intelligence and Machine Learning: In natural language processing, sets with vocabulary are one of the basic concepts. In classification algorithms, classes are also used for sets.
  - Computer Networks: For rule sets in firewalls, it is used to allow or block packets based on attributes such as source/destination IP address or port, this is a direct application of set theory.

1.1.6 Activity 1 – Part 1

- Exercise 1: I am student Nguyen Viet Phuc with registration number BD00671 , so a = 6 and b = 7.
  - We have:

$$|A| = \overline{9b} = 97$$

$$|B| = \overline{2a} = 26$$

$$|A \cap B| = a + b = 6 + 7 = 13$$

- We have recipe: $A \cup B = |A| + |B| - |A \cap B|$

$$= 97 + 26 - 13$$

$$= 110$$

⇨ Deduce the result: $A \cup B = 110$

❖ Exercise 2: I am student Nguyen Viet Phuc with registration number BD00671, so a = 6 and b = 7.

- We have :

$$|A - B| = \overline{3a} = 36$$

$$|A \cup B| = \overline{11b} = 117$$

$$|A \cap B| = \overline{1a} = 16$$

- Prove: $|B| = ?$
- Use the formula (1): $|A - B| = |A| - |A \cap B|$

$$36 = |A| - 16$$

=> $|A| = 52$

- Use the formula (2): $|A \cup B| = |A| + |B| - |A \cap B|$

$$117 = 52 + |B| - 16$$

⇨ So from that result: $|B| = 81$

❖ Exercise 3: I am student Nguyen Viet Phuc with registration number BD00671 , so a = 6 and b = 7.

Total number of customers in the market: $|U| = \overline{35b} = 357$

Number of customers buying fruit: $|F| = \overline{11a} = 116$

Number of customers buying vegetables: $|V| = \overline{9b} = 97$

Number of customers buying cakes: $|B| = \overline{8a} = 86$

Number of customers buying vegetables and fruits: $|V \cap F| = \overline{4b} = 47$

Number of customers buying vegetables and cakes: $|V \cap B| = \overline{3b} = 37$

Number of customers buying cakes and fruits: $|B \cap F| = \overline{2a} = 24$

Number of customers buying vegetables, cakes, fruits: $|V \cap F \cap B| = \overline{1a} = 14$

Find the number of customers who do not buy anything: $|U| - |F \cap V \cap B|$

Formula with 3 sets:

$$F \cup V \cup B = |F| + |V| + |B| - (|F \cap V| + |V \cap B| + |B \cap F|) + |F \cap V \cap B|$$

Analyze the pair but not all 3:

$$|F \cap V| = |F \cap V| - |F \cap V \cap B| = 47 - 14 = 33$$

$$|V \cap B| = |V \cap B| - |F \cap V \cap B| = 37 - 14 = 23$$

$$|B \cap F| = |B \cap F| - |F \cap V \cap B| = 24 - 14 = 10$$

Analysis of buying only each type:

$$F = |F| - (F \cap V + B \cap F + |F \cap V \cap B|)$$

$$= 116 - (33 + 10 + 14) = 116 - 57 = 59$$

$$V = |V| - (F \cap V + V \cap B + |F \cap V \cap B|)$$

$$= 97 - (33 + 22 + 14) = 97 - 70 = 27$$

$$B = |B| - (B \cap F + V \cap B + |F \cap V \cap B|)$$

$$= 86 - (10 + 23 + 14) = 86 - 47 = 39$$

The sum of the areas calculated above is respectively

$$|F|, |V|, |B|, |F \cap V|, |V \cap B|, |B \cap F|, |F \cap V \cap B|$$

$$59 + 27 + 39 + 23 + 10 + 14 = 205$$

Apply the comparison formula

$$F \cup V \cup B = |F| + |V| + |B| - (|F \cap V| + |V \cap B| + |B \cap F|) + |F \cap V \cap B|$$

$$F \cup V \cup B = 116 + 97 + 86 - (47 + 37 + 24) + 14 = 299 - 108 + 14 = 205$$

The result is the same as the sum of the calculated areas, from which we can deduce that the other number did not buy anything = $|U| - |F \cup V \cup B| = 357 - 205 = 152$

⇨ So there are about 152 customers who did not buy anything.

1.2 (P2) Determine the cardinality of a given bag (multiset)

1.2.1 Definition of bag

"In mathematics, a bag (or multiset, or mset) is a modification of the concept of a set that, unlike a set, allows for multiple instances for each of its elements. The number of instances given for each element is called the multiplicity of that element in the multiset." (Nt, 2025)

- Regarding the main difference with a set: In a set, each element is unique. In a bag, each element can appear many times.
- Notation: Bags are usually represented with curly braces {}, like for each set, but the context will indicate that it is a bag.

Example: At present $\{a, b, c\}$ is what will be called a set. And $\{a, a, b, c, c, c\}$ is a bag, and in this bag

- The multiple of a is 2
- The multiple of b is 1
- The multiple of c is 3

Another example is: Bag of fruits in a basket can be $\{apple\ ,\ orange\ , apple\ , banana\ , orange, apple\}$.

1.2.2 Operations on Bags

In discrete mathematics, a bag (or multiset) is a collection of elements that are allowed to repeat. Unlike a traditional set, where each element appears at most once, a bag keeps track of the multiplicity of each element — that is, the number of times an element appears.

- Union of Bags

  The union of two bags A and B , denoted as $A \cup B$, is a bag such that the multiplicity of each in the resulting bag is equal to the maximum of its multiplicities

  $$m_{A \cup B}(x) = \max(m_A(x), m_B(x))$$

  Example: Let A = $\{|p, q, q, r, r, r|\}$ = $\{|p : 1\ , q : 2\ , r : 3|\}$ and Let B = $\{|p, p, q, r, r, s, s|\}$ = $\{|p : 2\ , q : 1\ , s : 2|\}$

  The the union is: $A \cup B = \{|p, p, q, q, r, r, r, s, s|\}$ = $\{|p : 2\ , q : 2\ , r : 3\ , s : 2|\}$

  This mean that for each element , we take whichever bag has the higher count.

❖ Intersection of Bags

The intersection of two bags A and B, denoted as $A \cap B$, is a bag such that the multiplicity of each element is equal to the minimum of its multiplicities in A and B .

$$m_{A \cap B}(x) = \min(m_A(x), m_B(x))$$

Example: Give A = $\{|p, q, q, r, r, r|\}$ = $\{|p : 1, q : 2, r : 3|\}$ and B = $\{|p, p, q, r, r, s, s|\}$ = $\{|p : 2, q : 1, r : 2, s : 2|\}$

We have: $A \cap B = \{|p, q, r, r|\}$ = $\{|p : 1, q : 1, r : 2|\}$ only elements appearing in both bags are included, and their multiplicities are limited by the smaller count between A and B .

❖ **Difference of Bags**

The diference of two bags A and B, denoted as A – B, is a bag such that the multiplicity of each element is equal to the multiplicity in A minus the multiplicity in B , if the result is positive, otherwise, the multiplicity becomes zero.

$$m_{A-B}(x) = \max(m_A(x), m_B(x), 0)$$

Example: for the same bags $A = \{|p : 1, q : 2, r : 3|\}$ and B = $\{|p : 2, q : 1, r : 2, s : 2|\}$

We get: $A - B = \{|q, r|\} = \{|p : 0, q : 1, r : 1, s : 0|\}$

$B - A = \{|p, s, s|\} = \{|p : 1, q : 0, r : 0, s : 2|\}$

Note that : $A - B \neq B - A$

❖ Sum of Bags

The sum of two bags A and B, denoted as A + B is a bag such that the multiplicity of each element is equal to the sum of its multipliciies in A and B

$$m_{A+B}(x) = m_A(x) + m_B(x)$$

Example : For the same A = $\{|p : 1, q : 2, r : 3|\}$ and B = $\{|p : 2, q : 1, r : 2, s : 2|\}$ the result is

$$A + B = \{|p, p, p, q, q, q, r, r, r, r, r, s, s|\} = \{|p: 3, q: 3, r: 5, s: 2|\}$$

This operation effectively adds up the occurrences of each element from both bags.

1.2.3 Cardinality of Bags

Now the number of elements in a bag (also known as size or each count) is referring to the total number of elements in that bag, from here on it will take into account the number of each of these elements. With each of these it will mean that if an element appears multiple times in the bag, then each occurrence contributes to the total number of elements.

Formally, if A is a bag and $m_A(x)$ is the multiplicity of an element x in A, the cardinality of bag A is given by:

$$| A |= \sum_{x \in A} mA(x)$$

This is in contrast to an ordinary set , where each distinct element is counted only once .

Consider two bags: A = $\{|a , b , b, c, c, c|\} = \{|a : 1 , b : 2 , c : 3|\}$

$$A = \{|a , a , b, b, b, d|\} = \{|a : 2 , b : 3 , d : 1|\}$$

Then , the cardinality of each bag is :

$$| A |= 1 + 2 + 3 = 6$$

$$| B |= 2 + 3 + 1 = 6$$

Hence, both bags A and B contain six total elements , even though their element distributions differ .

1.2.4 Applications in Computing Context

Bags, also known as multi-sets, are widely used in Computer Science because they allow multiple occurrences of elements. This makes them suitable for representing sets of data in real-world situations, where repetition occurs.

- In database management systems, query results are often referred to as bags. When retrieved with data from a table without using the DISTINCT keyword, duplicate rows may appear in the results. This behavior directly reflects the concept of bags, where repeated data items are allowed, and makes sense.
- The access to each information and from here it will be processed by natural language, as well as Bag of Words models so that from here it will be possible to represent one of the documents as a bag of its words, considering the frequency of each word instead of its order. With such approaches

it will often be divided into tasks such as text classification as from here it will be extracted from the text.

- For analysis with each data and statistics, bags will be used so that the word is represented with data sets with each repeated value. As well as when it is counted with the occurrence of each item, transaction or from specific events, bags can store and manage data frequency effectively.
- In software development and algorithm design, bag operations such as union, intersection, and difference are used to compare data sets, detect duplicates, and merge information from multiple sources.
- In general, bags play an important role in computing because they provide a simple yet powerful way to represent data with repetition, a common feature in most information systems.

1.2.5 Activity 1 – Part 2

- Exercise 1

I am student Nguyen Viet Phuc with registration number BD00671, so a = 6, b = 7

a, We have: $\overline{1a2} = 162$

Analyze 162 into prime factors, we have: $162 = 2 \times 3^4$

The bag prime factor is:

$$A = \{|2: 1, 3: 4|\}$$

b, We have: $\overline{2b0} = 270$

Analyze 270 into prime factors, we have: $270 = 2 \times 3^3 \times 5$

The bag prime factor is:

$$A = \{|2: 1, 3: 3, 5: 1|\}$$

- Exercise 2

a, Find the cardinalities of each of the aforementioned bags

We have : $A = \{|2: 1, 3: 4|\}$ and $B = \{|2: 1, 3: 3, 5: 1|$

The cardinalities of A and B are :

$$|A| = 1 + 4 = 5$$

$$|B| = 1 + 3 + 1 = 5$$

b, Find the cardinalities of the intersection of the aforementioned bags

We have: $A = \{|2: 1, 3: 4|\}$ and $B = \{|2: 1, 3: 3, 5: 1|\}$

The intersection of sets A and B is:

$$A \cap B = \{|2: 1, 3: 3|\}$$

The cardinalities of $A \cap B$ is

$$|A \cap B| = 1 + 3 = 4$$

c, Find the cardinalities the union of the aforementioned bags.

We have: $A = \{|2: 1, 3: 4|\}$ and $B = \{|2: 1, 3: 3, 5: 1|\}$

The union of sets A and B is:

$$A \cup B = \{|2: 1, 3: 4, 5: 1|\}$$

The cardinalities of $A \cup B$ is

$$|A \cup B| = 1 + 4 + 1 = 6$$

d, Find the cardinalities the difference of the aforementioned bags.

We have: $A = \{|2: 1, 3: 4|\}$ and $B = \{|2: 1, 3: 3, 5: 1|\}$

The difference of sets A and B is:

$$A - B = \{|3: 1|\}$$

The cardinalities of $A - B$ is:

$$|A - B| = 1$$

1.3 (M1) Determine the inverse of a function using appropriate mathematical techniques

1.3.1 Definition of a Function

A function is a rule or mapping that assigns exactly one element of a set B (called the codomain) to each element of another set A (called the domain) .

If f is a function from A to B , we write $f : A \rightarrow B$ for each $a \in A$, the exists a unique $b \in B$ such that f(a) = b .

- Domain : the set of all possible input values
- Codomain : The set that contains all possible outputs .
- Range(Image) : The actual outputs produced by the function .

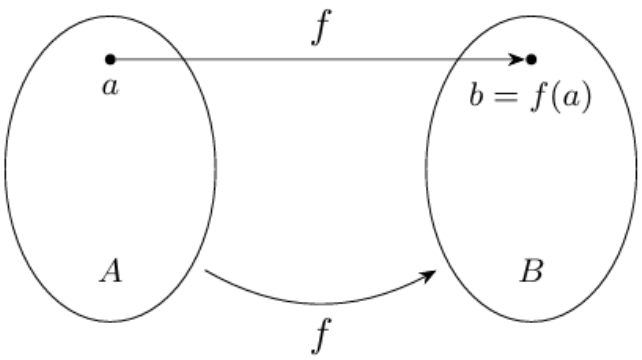


Figure 2 The Function f Maps A to B

Example : Let f(x) = x^2 where $x \in \mathbb{R}$ here,

- Domain = all real numbers,
- Codomain = all real numbers,
- Range = all non-negative real numbers $\{y \in \mathbb{R} \mid y \geq 0\}$.

1.3.2 Definition and Existence of Inverse Functions

The inverse of a function “reverses” the effect of the original function . If f map x to y , then the inverse function f^{-1} maps y back to x.

$$f(a) = b \; if \; and \; only \; if \; f^{-1}(b) = a$$

However, not every function has an inverse. For f to have an inverse, it must satisfy two important properites:

Injective (one – to – one): No two distinct elements in the domain map to the same element in the codomain . Formally, if $f\ (x1) = f(x2), then\ x1 = x2$.

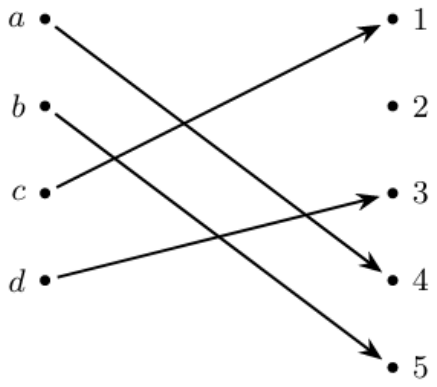


Figure 3 One – to – One Function

Surjective (Onto) : Every element in the codomain is the image of at least one element in the domain

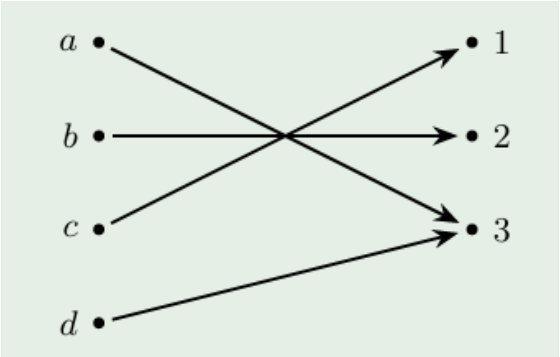


Figure 4 Onto functon

When a function is both injective and surjective , it is called bijective , and only bijective function possess inverses .

Example : $f\ (x) = 3x - 5$ This function is both one-to-one (each input gives a unique output) and onto (every real number can be obtained), so it has an inverse.

If a function fails to be one-to-one or onto, it cannot be inverted uniquely. For example, $f\ (x) =\ x^2$ is not invertible over all real numbers since $f(-2) = f(2) = 4$ violating the injectivity condition.

1.3.3 Finding the Inverse Algebraically

The inverse of a function can be determined algebraically by reversing the process of the original function the gemeral steps are:

Step 1 : Write the function as $y = f(x)$.

Step 2 : Swap the variables $x$and y.

Step 3 Solve the new equation for y.

Step 4 : Replace $y$with $f^{-1}(x)$.

- ❖ Example 1: Linear Function
  - Given $f\ (x) = \ 2x + \ 4$
    - ✓ Step 1: $y = \ 2x + \ 4$
    - ✓ Step 2: Swap x and y : $x = 2y + 4$
    - ✓ Step 3: Soleve for y

$$y = \frac{x - 4}{2}$$

- ✓ Step 4: Therefore

$$f^{-1}(x) = \frac{x - 4}{2}$$

- ✓ Verification

$$f(f^{-1}(x)) = 2\left(\frac{x - 4}{2}\right) + 4 = x$$

- ✓ Hence, the inverse is correct .

- ❖ **Example 2: Rational Function**

Give $\boldsymbol{f(x) = \frac{4x + 1}{3z - 2}}$

- ✓ Step 1: $y = \frac{4x+1}{3x-2}$
- ✓ Step 2: Swap x and y: $x = \frac{4y+1}{3y-2}$
- ✓ Step 3: Multiple both sides by $3y - 2$:

x (3y -2) = 4y + 1

- ✓ Simplify: $3xy - 2x = 4y + 1$
- ✓ Rearrange: $3xy - 4y = 2x + 1$
- ✓ Factor out y

$$y(3x - 4) = 2x + 1$$

- ✓ Solve for y

$$y = \frac{2x+1}{3x-4}$$

- ✓ Therefore $f^{-1}(x) = \frac{2x+1}{3x-4}$

❖ Example 3: Non-linear Function

- Given $f(x) = (x + 2)^3$
  - ✓ Step 1: $f(x) = (x + 2)^3$
  - ✓ Step 2: Swap x and y

$$x = (y + 2)^3$$

  - ✓ Step 3: Solve for y

$$y = \sqrt[3]{x} - 2$$

  - ✓ Thus $f^{-1}(x) = \sqrt[3]{x} - 2$

These steps reflect the general algebraic technique for finding inverse functions , ensuring that the function is bijective before inversion.

1.3.4 Applications of Inverse Functions in Computing

Inverse functions have many practical applications in computer science, where the process often requires reversing a transformation or retrieving the original data from the transformed values.

- Encryption and Decryption: In cryptography, encryption is modeled by a function f that transforms plaintext into ciphertext. The decryption process is the inverse function f^{-1} , restoring the original message.
- Data Encoding and Normalization: When data are normalized or scaled using a transformation function, inverse functions are used to revert normalized data to their original range for interpretation and analysis.

- Computer Graphics and Transformations: Inverse functions are used in geometric transformations to reverse operations such as translation, rotation, or scaling, helping to restore original object positions in 2D and 3D environments.
- Algorithmic Reversibility: In algorithms involving mapping or state changes, inverse functions allow reversal of computations, which is crucial in debugging, reversible computing, and process tracking.
- Data Mapping and Network Routing: Inverse mappings help determine the original source or path from a given output, widely used in data retrieval and routing algorithms.

1.3.5 Activity 1 – Part III

My student ID is BD00671, therefore, according to the requirement that "a < b represents the largest digits in my ID," we have a = 6 and b = 7.

❖ Exercise 1

Ascertain whether the given functions are invertible. If they are, identify the rule for the inverse function f^{-1} .

a, $f : \mathbb{R} \to \mathbb{R}\ with\ f(x) = bx + a$

b, $f : [-b, +\infty] \to [0, +\infty]\ with\ f(x) = \sqrt{x + b}$

Solution:

a, we have: $f = \mathbb{R} \to \mathbb{R}\ with\ f(x) = bx + a = 7x + 6$

The function f has an inverse because it is a one-to-one correspondence.

Suppose that y is the image of x, so that:

$$y = 7x + 6$$

$$\Leftrightarrow 7x = y - 6$$

$$\Leftrightarrow x = \frac{y - 6}{7}$$

We concude that the inverse function of the given function f is $f^{-1}(y) = \frac{y-6}{7}$

b, We have:

$$f : [-7, +\infty) \to [0, +\infty)\ with\ f(x) = \sqrt{x + b} = \sqrt{x + 7}$$

The function f has an inverse because it is a one-to-one correspondence

Suppose that y is the image of x, so that:

$$y = \sqrt{x+7}$$

$$\Leftrightarrow y2 = x + 7$$

$$\Leftrightarrow x = y2 - 7$$

We concude that the inverse function of the given function f is $f^{-1}(y) = y^2 - 5$

❖ Exercise 2

Let $f, g: \mathbb{R} \to \mathbb{R}$ be defined as $f(x) = \begin{cases} 2x + a, x < 0 \\ x^3 + b, x \geq 0 \end{cases}$ and $g(x) = bx - a$. Find $g \circ f$

Solution :

We have $f(x) = \begin{cases} 2x + a, x < 0 \\ x^3 + b, x \geq 0 \end{cases} = \begin{cases} 2x + 6, x < 0 \\ x^3 + 7, x \geq 0 \end{cases}$

And $g(x) = bx - a = 7x - 6$

$$g \circ f(x) = g(f(x)) = \begin{cases} 7(2x+6) - 6, & x < 0 \\ 7(x^3+7) - 6, x \geq 0 \end{cases}$$

$$= \begin{cases} 14x + 36, x < 0 \\ 7x^3 + 43, x \geq 0 \end{cases}$$

1.4 (D1) Formulate corresponding proof principles to prove properties about defined sets

1.4.1 Introduction to Mathematical Proofs

A mathematical proof is a logical argument used to establish the truth of a given statement by applying infinitely valid reasoning and to the results it has proven of each previous one. Proofs are the foundations of discrete mathematics and every computing because they ensure the correctness and reliability of mathematical statements, algorithms and system designs.

With any of the methods it will be possible to prove it will be used depending on the nature of what will be claimed by each of those methods:

- Direct Proof: With each assumption of the premises being true and from this it will be used with each thing that can be logically deduced to reach the conclusion. Example if n is even , then n^2 *is even*.
- Proof by Contradiction: With each assumption being negative it will be concluded and from this we deduce in terms of contradiction.

- Proof by Contraposition: Prove $NOT\ Q\ \Rightarrow\ NOT\ P\ instead\ of\ proving\ P\ \Rightarrow\ Q.$
- Proof by Mathematical Induction: This will be used for statements involving individual natural numbers and from here on will involve proving the base case and from here on into the inductive steps.
- Proof by Exhaustion: With each examination of all of the possible cases, it is possible to verify the validity of a claim.

Proofs are essential in computer science to validate algorithms, verify program logic, and ensure that mathematical models behave as expected.

1.4.2 Set Identities

An identity set is an equation that contains set operations (such as union, intersection, or complement) that are always true for every set A, B, and C. Each of these identities forms the basis of algebra for reasoning about sets and relationships.

These identities are analogous to the logical equivalents used in propositional logic, where "AND" corresponds to intersection, and "OR" corresponds to union. Understanding and proving set identities provides the foundation for reasoning about relationships in databases, algorithms, and number systems.

Common Set Identities :

Law Type	Identity	Description
Identity Laws	$A\ \cup\ \emptyset = A, A\ \cap\ U = A$	Adding or intersecting with universal or empty sets.
Domination Laws	$\mathrm{A}\ \cup\ \mathrm{U}\ =\ \mathrm{U}, \mathrm{A}\ \cap\ \emptyset\ =\ \emptyset$	Extreme cases under union/intersection.
Idempotent Laws	$\mathrm{A}\ \cup\ \mathrm{A}\ =\ \mathrm{A}, \mathrm{A}\ \cap\ \mathrm{A}\ =\ \mathrm{A}$	Repetition doesn’t change the result.
Complement Laws	$\mathrm{A}\ \cup\ \mathrm{A}'\ =\ \mathrm{U}, \mathrm{A}\ \cap\ \mathrm{A}'\ =\ \emptyset$	A set and its complement cover all or none.
Commutative Laws	A ∪ B = B ∪ A, A ∩ B = B ∩ A	Order doesn’t affect results.
Associative Laws	A ∪ (B ∪ C) = (A ∪ B) ∪ C	Grouping of operations is irrelevant.

Distributive Laws	A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)	Similar to arithmetic distributivity.
De Morgan's Laws	(A ∪ B)′ = A′ ∩ B′; (A ∩ B)′ = A′ ∪ B′	Connect complements with union/intersection.
Absorption Laws	A ∪ (A ∩ B) = A; A ∩ (A ∪ B) = A	Combining set with part of itself doesn't change it.

These laws can be proved with methods of each algebra or from here with membership tables (to be introduced later). They are widely used in programming and numerical logic, for example, in simplifying Boolean expressions or database query conditions.

1.4.3 How to Prove Two Sets Are Equal

In order for us to prove that for approximately two sets they are equal — that is, A = B — we have to prove mutual inclusion:

- ✓ Show that every element of A is in B ($A \subseteq B$).
- ✓ Show that every element of B is in A ($B \subseteq A$)

This approach guarantees that the two sets conain exactly the same elements .

Example : Prove the distributive law

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

❖ Proof

- First inclusion

What we assume $x \in A \cap (B \cup C)$. By definition of intersection , $x \in A$ and by definition of union , $x \in B$ or in $x \in C$. From there x will have to be in $A \cap B$ or in $A \cap C$. Hence , $x \in (A \cap B) \cup (A \cap C)$. This shows that $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$.

- Sencond inclusion

Now assume $x \in (A \cap B) \cup (A \cap C)$. This means that x is eiher in $A \cap B$ or in $A \cap C$. In both cases , x $x \in A$ *and* $x \in (B \cup C)$, since being in $A \cap B$ *or* $A \cap C$ implies that x is in A and at least one of B or C . Thus $x \in A \cap (B \cup C)$, we have therefore proved $(A \cap B) \cup (A \cap C) \subseteq A \cap (B \cup C)$.

Since both inclusions are true and we can conclude that

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

With this simple yet powerful argument, we will show how we can move from element-based logic to set-based proofs.

Each statement about the membership of an element x corresponds to a logical operation — "and" represents intersection, "or" represents union.

This type of argument is thus directly analogous to direct proofs in propositional logic.

In the context of computer science, this argument forms the basis for verifying logical expressions, ensuring that different representations of a condition are equivalent — a crucial step in software correctness, database optimization, and logic circuit design.

1.4.4 Membership tables

While the above methods rely on arguments for individual elements, it gives a very practical and hence intuitive approach to prove equality with sets through membership tables. The membership table works very much like a truth table in propositional logic. This table lists all possible combinations of members of the relevant sets, where 1 denotes an element belonging to a set, and 0 denotes non-membership.

By comparing the results of two set expressions in all possible cases, we can determine whether the two expressions are equivalent. If both expressions give the same result for all possible combinations of members, then the equality is proven.

Let us use this method to verify the following equality:

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

This identity can be interpreted intuitively: The left-hand side represents elements that belong to either A or B but not both. The right-hand side also describes the same situation — all elements in the union that are not shared by both sets. Hence, we expect the two sides to be equivalent.Let us confirm this using a membership table.

A	B	$A - B$	$B - A$	$(\mathrm{A} - \mathrm{B}) \cup (\mathrm{B} - \mathrm{A})$	$\mathrm{A} \cup \mathrm{B}$	$\mathrm{A} \cap \mathrm{B}$	$(\mathrm{A} \cup \mathrm{B}) - (\mathrm{A} \cap \mathrm{B})$
1	1	0	0	0	1	1	0
1	0	1	0	1	1	0	1
0	1	0	1	1	1	0	1
0	0	0	0	0	0	0	0

Table 1 membership

1.4.5 Applications in Computing Context

With the techniques being proven and from here it will be named as the work that is collected of each deep connection with computer science. From there they will all have their practical application features including:

Algorithm verification: The work that it is all proven by algorithms so that from here it can be sure that the algorithms produce extremely accurate output results for all valid inputs. Examples of the work being proven are the final results and the accuracy of one of the algorithm arrangement work similar to proving a theory.

Simplify with each logic and each work the accuracy of the program: with the compatibility conditions of logic and as well as the De Morgan laws will all help it to extremely simplify each statement in all aspects of the conditions in the work that will be developed software. For example: !(A && B) is compatible with (!A || !B), and can also optimize structures to decide.

Database Query Prioritization: With SQL algorithms such as UNION, INTERSECT and EXCEPT, it will be compatible with the rationality, protocol and optimal efficiency of set operations. Let us understand the definitions of union operations allowed by database engineers to optimize complex problems to achieve higher efficiency.

Digital Circuit Design: Boolean Algebra is directly analogous to the theory of each set, from here it will be used to minimize logic gates in digital circuits, as well as further improve performance and reduce energy consumption.

Artificial Intelligence and Knowledge Representation: The fact that we reason by logic and as it is established, discuss each proposition (from the principles of proof) is one of the essential elements in

every rule-based AI system, which is allowed to work with machines that draw valid conclusions from the facts it already knows.

By holding principles that are all proven, with each expert in computing, it is not possible to reason formally, detect inconsistencies and build strong logic systems. The main discussion used to prove academic definitions is also the foundation for the correctness of programs, protocols and hardware systems.

1.4. 6 Activity 1 – Part IV

Show that if , AB Exercise 1 and Care sets, then $\overline{A \cup B \cup C} = \bar{A} \cap \bar{B} \cap \bar{C}$.

- Exercise 1: Demonstrate that each side is a subset of the other side .
  - Solution**:** We are asked to show that for any sets , A , B , and C .

$$\overline{A \cup B \cup C} = \bar{A} \cap \bar{B} \cap \bar{C} .$$

To prove this equality, we demonstrate that each side is a subset of the other.

- Prove that $\overline{A \cup B \cup C} \subseteq \bar{A} \cap \bar{B} \cap \bar{C}$.

  Let $x \in \overline{A \cup B \cup C}$.

$$=> x \notin A \cup B \cup C$$

$$=> x \notin A, x \notin B, x \notin C$$

$$=> x \in \bar{A}, x \in \bar{B}, x \in \bar{C}$$

$$=> x \in \bar{A} \cap \bar{B} \cap \bar{C} .$$

- Exercise 2: Verify the equality using a membership table .

  We now verify the same equality using a membership table .

  Here, 1 indicates that an element belongs to a set , and 0 means it does not.

A	B	C	$A \cup B \cup C$	$\overline{A \cup B \cup C}$	$\bar{A}$	$\bar{B}$	$\bar{C}$
1	1	1	1	0	0	0	0
1	1	0	1	0	0	0	1

1	0	1	1	0	0	1	0
1	0	0	1	0	0	0	0
0	1	1	1	0	1	0	0
0	1	0	1	0	1	0	1
0	0	1	1	0	1	1	0
0	0	0	0	1	1	1	1

Table 2 Membership table de morgan's

Observation : The last two columns $\overline{A \cup B \cup C} = \bar{A} \cap \bar{B} \cap \bar{C}$

CHAPTER 2 – LO2: ANALYSE MATHEMATICAL STRUCTURES OF OBJECTS USING GRAPH THEORY

2.1 (P3) Model contextualised problems using trees, both quantitatively and qualitatively

2.1.1 Definition and Types of Binary Trees

Definition

"A Binary Tree Data Structure is a hierarchical data structure in which each node has at most two children, referred to as the left child and the right child. It is commonly used in computer science for efficient storage and retrieval of data, with various operations such as insertion, deletion, and traversal." (GeeksforGeeks 2024)

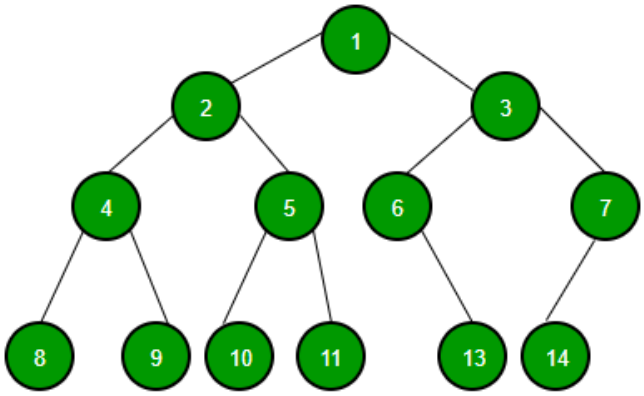


Figure 5 Binary Tree Data Structure

Some common types of binary trees

Full Binary Tree: A tree in which every node (except leaf nodes) has exactly two children.

Complete Binary Tree: A tree in which all levels, except possibly the last level, are completely filled, and from which all nodes at the last level are pushed to the left.

Balanced Binary Tree: A tree in which, at any node, the difference in total height between the left subtree and the right subtree does not exceed a certain value.

2.1.2 Case 1: Binary Search Tree (BST)

Contextualized Problem: In many software applications, we have large collections of data (e.g., phonebooks, dictionaries, user databases) and from there we will perform searches, additions, and deletions very quickly. If we store the data in an unordered list (array), the search will take O(n) time, which is very slow for large amounts of data.

Modeling: Binary Search Tree (BST) models this problem by storing data in a structured rule

- Every node will contain a key (e.g. name, ID, number).
- At any node (let's call it P), all the keys in the left subtree of P will be smaller than the key of P.
- All the keys in the right subtree of P will be larger than the key of P.

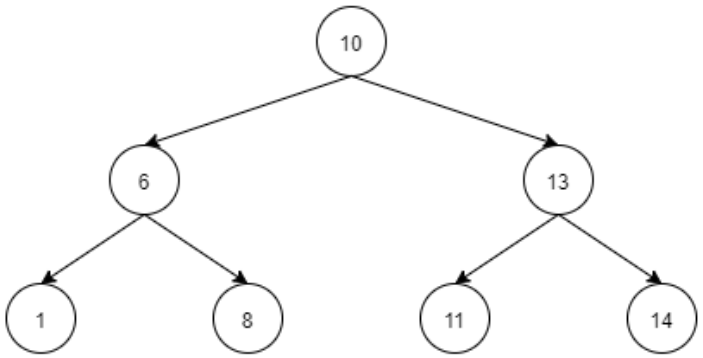


Figure 6 Binary Search Tree – BST

Qualitative Analysis:

- Structure: This is one of the ordered binary trees. The very specific structure of each tree depends on the order in which the elements are added.
- Meaning: Each of these structures allows "divide and conquer". When searching for a value in, we start at the root. If the value to be searched is smaller than the current node, we know for sure that it (if any) must be in the left subtree; otherwise, it must be in the right subtree. We remove half of the irrelevant nodes at each step.
- Advantages: Allows for very fast searching, adding, and deleting (in the average case). Data is also maintained in an ordered state (can traverse in-order to get a sorted list).

Quantitative Analysis: We will then be performing the performance analysis (time complexity) of each operation based on n (number of nodes) and h (height of the tree).

2.1.3 Case 2: Expression Tree

Contextualized Problem: With compilers and typical calculators, how can one represent and from here calculate one of the mathematical expressions (e.g., (5 + 3) * (10 - 4)) in a precise way, following the correct order of precedence of the operations?

Modeling: The expression tree models this mathematical expression:

- Leaf nodes: Are operands, i.e. numbers or variables (e.g. 5, 3, 10, 4).
- Internal nodes: Are operators (eg: +, *, -).
- The structure of the tree represents the order of precedence. The operation at the root node is the operation that is performed last.

For example: The expression (5 + 3) * (10 - 4) will have a root node of *. The left subtree is the expression tree for (5 + 3), and the right subtree is the expression tree for (10 - 4).

Qualitative Analysis:

- Structure: a binary tree where leaf nodes are operands and internal nodes are operators.
- Meaning: With each of these structures, it will always remove any ambiguity about the order of precedence. No need to use parentheses.
- Advantage: It is easy to compute the value of the expression using post-order traversal. We can also traverse the tree in different orders to get the expression in prefix, infix, or postfix form.

Quantitative Analysis :

- Space Complexity: O(n) – Memory required to store n nodes (both operands and operators).
- Parsing: O(n). The original expression (in string form) needs to be iterated through once to build the tree.
- Evaluation: O(n) To evaluate a value, we need to traverse the tree in post-order. The algorithm will visit each node exactly once. When visiting an operator node, we take the calculated result from its two child nodes and perform the operation.

2.2 (P4) Use Dijkstra's algorithm to find a shortest path spanning tree in graph

2.2.1 Introduction Dijkstra's Algorithm

"Dijkstra's algorithm is used to find the shortest path between the two mentioned vertices of a graph by applying the Greedy Algorithm as the basis of principle."(Kapoor, A. 2023)

- With algorithms that are working by being able to maintain one of the sets for visited and unvisited vertices. Starting from the root vertex, the algorithm continuously selects the vertex with the smallest exploration distance, from which it will be updated with the distances of its neighbors, and from here it will continue until all vertices are processed or the goal is reached.
- With directed graphs, each edge has a direction, representing the direction of travel between each vertex connected by that edge. In these cases, the algorithm will usually follow the direction of each edge when searching for the shortest path.
- In undirected graphs, edges have no direction, and the algorithm can traverse both forward and backward along edges when searching for the shortest path.

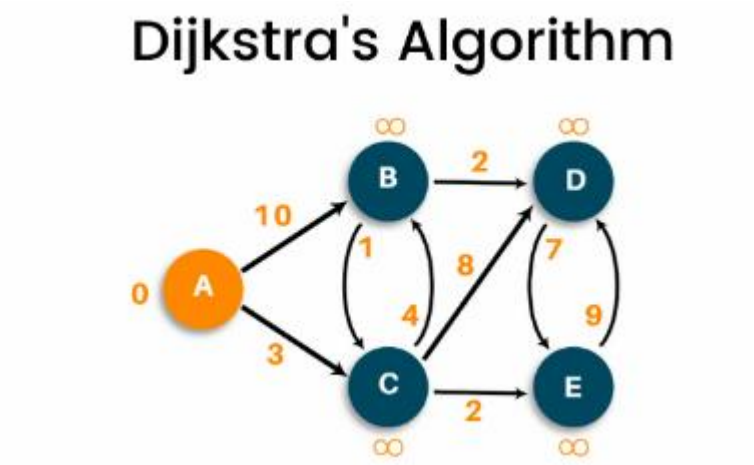


Figure 7 Dijkstra's Algrothim

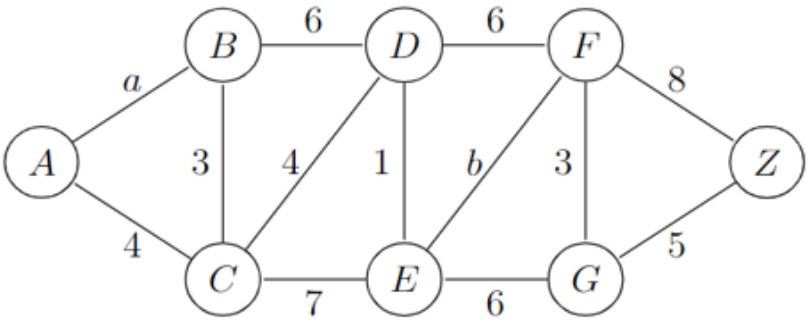
2.2.2 Step-by-step Algorithm

- Mark the source nodes with each current distance as 0 and for the remaining nodes with infinity.
- Set the unvisited node with the smallest current distance as the current node from here on.

- For the neighbors, N of the current node plus the current distance of the adjacent node plus the weight of the edge connecting 0->1. If these distances are less than the current distance of the node, set it as the new current distance as N.
- Mark the current node 1 as visited.
- Go to step 2 if there are any unvisited nodes.

2.2.3 Activity 2 – Part II

Apply Dijkstra's algorithm to determine the shortest path length between vertices A *and* Z *in the provided weighted graph.*



I am student Nguyen Viet Phuc, registration number BD00671, with a = 6 and b = 7.

Graph: undirected weighted, vertices A, B, C, D, E, F, G, Z

Edges:

Edge	Weight	Edge	Weight
A–B	6	D–F	6
A–C	4	E–F	7
B–C	3	E–G	6
B–D	6	F–G	3
C–D	4	F–Z	8

C–E	7	G–Z	5
D–E	1		

Table 3 Edges

- Dijkstra's Algorithm Table

G	V(i)	A	B	C	D	E	F	G	Z
{A,B,C,D,E,F,G,Z}	-	0	∞	∞	∞	∞	∞	∞	∞
{B,C,D,E,F,G,Z}	A	0	6	4	∞	∞	∞	∞	∞
{C,D,E,F,G,Z}	C	0	6	4	8	11	∞	∞	∞
{D,E,F,G,Z}	B	0	6	4	8	11	∞	∞	∞
{E,F,G,Z}	D	0	6	4	8	9	14	∞	∞
{F,G,Z}	E	0	6	4	8	9	14	15	∞
{G,Z}	F	0	6	4	8	9	14	15	22
{Z}	G	0	6	4	8	9	14	15	20
{}	Z	0	6	4	8	9	14	15	20

Table 4 Dijkstra's algorithm

- Shortest Path Results

Vertex	Shortest Distance from A	Predecessor
A	0	–
B	6	A
C	4	A

D	8	C
E	9	D
F	14	D
G	15	E
Z	20	G

Table 5 Shortest path

Conclusion: shortest path length $A \rightarrow C \rightarrow D \rightarrow E \rightarrow G \rightarrow Z = 20$

2.3 (M2) Assess whether a Eulerian and Hamiltonian circuit exists in an undirected graph

2.3.1 Eulerian Circuit

"An Euler circuit is a circuit that uses every edge in a graph with no repeats. Being a circuit, it must start and end at the same vertex."(courses.lumenlearning 2025)

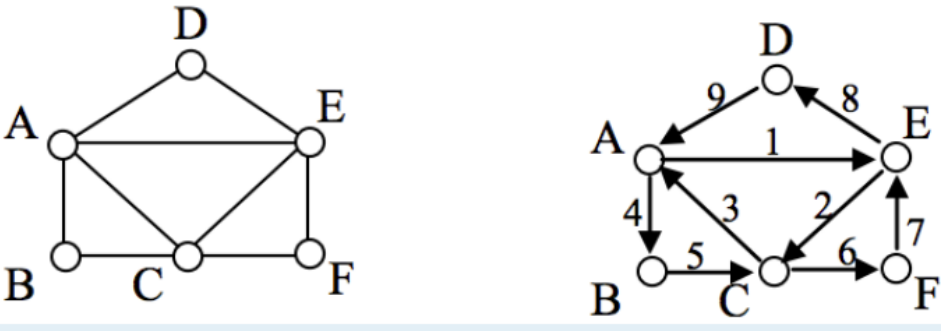


Figure 8 Eulerian Circuit

With the following conditions: Any undirected graph has an Euler cycle if and only if

- The graph is connected, and each has all vertices of even degree.
- If the graph is connected but has exactly two vertices of odd degree, then it does not have an Euler cycle but only an Euler path.

2.3.2 Hamiltonian Circuit

"A Hamiltonian circuit is a circuit that visits every vertex once with no repeats. Being a circuit, it must start and end at the same vertex."(courses.lumenlearning , 2025)

❖ Characteristics:

- Without Euler's exact conditions,
- it is still possible to rely on Dirac's theorems:
- If the graph has n≥3 vertices and all vertices have degree ≥ n/2, then the graph has a Hamiltonian circuit.
- sually they will be tested by knowing how to draw it or from here they will also be able to list possible paths.

❖ Short comparison:

Type	Describe	Conditions of existence
Hamiltonian Circuit	Go through every vertex, return to the starting point	There are no fixed rules (usually tested by trial and error)
Hamiltonian Path	Go through every vertex, without returning to the starting point	Exists in many connected graphs

Table 6 Short comparison

2.3.3 Activity 2 Part III

Does the following graph have a Hamilton path? If so, find such a path. If it does not, give an argument to show why no such path exists.

❖ Graph description**:** graph has vertices

$$x, y, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q$$

arranged in two nested squares, connected by horizontal and vertical edges.

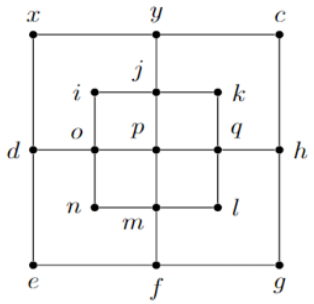


Figure 9 Graph description

❖ Eulerian Test

- Counting the degrees of the vertices will show that many vertices have odd degrees (like d, h, o, q,...).
- Therefore, with graphs that do not have Euler cycles, there may be Euler paths because the graph is still connected.
- Conclusion: These graphs will not have Euler cycles.

❖ Check the Hamiltonian

When examining the paths, we can find a Hamiltonian path that passes through all vertices exactly once as follows:

$$e - n - o - i - j - k - q - p - m - l - h - d - x - y - c - g - f$$

With paths like this it will go through all vertices but it will not return to the starting point,it is a Hamiltonian path, not a Hamiltonian circuit.

2.4 (D2) Construct a proof of the Five Color Theorem

2.4.1 Introduction

The formal statement of the theorem is:Any planar graph G can be colored with five colors. It is worth noting that while the Four Color Theorem (which states that four colors are sufficient) is extremely difficult to prove, the proof for the Five Color Theorem is relatively simple and elementary.

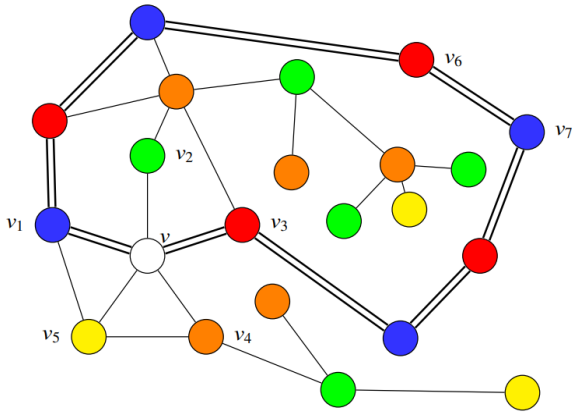


Figure 10 File color theorem

2.4.2 Theorem Statement

❖ Key concepts for proof

In order for us to be able to prove each of the five color theorems, we will need an important foundational theorem that is not directly related to coloring, but to the structure of planar graphs.

- The basis of the proof: Let G be a planar, connected graph. From here, there must exist one of the vertices v in G, from which it is possible to have the degree of v, denoted by d(v), such that d(v) <= 5 (In other words, every planar graph always has at least one vertex with 5 or fewer neighbors).
- Kempe Chain: This is a tool used to handle the difficult case of the problem. A Kempe chain G' is a maximal, bi-colored, connected subgraph of G. (For example, a Blue-Red chain is both vertices and edges belonging to a continuous path using only two colors, blue and red).

❖ Proof Content

Now we will be able to use the methods from here to prove by induction on the number of vertices V of the graphs G.

- Base Case

Now, since all graphs G have 5 vertices or less (V <= 5), it is obvious that we can color G with a range of those five colors (each vertex will have a different color).

- Inductive Step

Induction Conjecture: Suppose that every planar graph G with fewer vertices than G can be 5-colored.

✓ Proof step, now, we consider graph G.

- We know that G must have one of the vertices v with degree $d(v) \leq 5$
- Temporarily, it will be deleting vertex v (and from the edges connected to it) then it will be from G so from here we have created a new graph G. Since G will have less vertices than G, from then on, by the hypothesis with each induction, G can be colored with 5 colors.
- Now, we move vertex v back to G to restore the graph G. We need to find a color for v from the 5 available colors. We have all the neighboring vertices of v.

- We have two cases for vertex v:

✓ Case 1: d(v) < 5 or for the neighborhoods of v it will use 4 colors or less

- If v has degree less than 5 (i.e. v has 4 or fewer neighbors), or if v's 5 neighbors (v_1, ..., v_5) use only 4 (or fewer) colors out of 5.
- In both cases, it will always have at least one color out of 5, and henceforth it will not be used by v's neighbors.
- Therefore, we can use that remaining color to color v. The coloring is complete.

✓ Case 2: d(v) = 5 and all 5 neighbors use 5 different colors

Here is the only complicated case: v has exactly 5 neighbors v_1, v_2, v_3, v_4, v_5, and they are colored with 5 different colors (e.g. v_1=Blue, v_2=Green, v_3=Red, v_4=Orange, v_5=Yellow).

- We can't just color v right away. We have to use Kempe Chains to change the color of some of the neighboring vertices.
- Step 2a: Consider v_1 (Blue) and v_3 (Red).
  - Find the Kempe Blue-Red chain starting from v_1.
  - Possibility: If these Blue-Red chains are not connected to v_3 , then from here we can reverse the color on the entire chain (every Blue vertex becomes Red, every Red vertex becomes Blue).
  - Since v_3 will not be in the chain, with each of its colors it will not change (still Red). But v_1 now also becomes Red.

  Result: v_1 and v_3 will both be Red. The vertex v now has no Blue neighbors. From here we can color v Blue.

- Step 2b: Consider v_1 (Blue) and v_3 (Red) (continued).
  - Probability: If the Blue-Red strings all connect v_1 and v_3
  - This means that there exists a path P consisting of only Blue and Red vertices connecting v_1 and v_3.
  - This path P, together with the edges $\overline{vv1}$ and $\overline{vv3}$, forms a closed loop.
  - According to the Jordan Curve Theorem, this loop P will be evenly distributed across the plane into an "inside" and an "outside" region.
- Step 2c: Consider v_2 (Green) and v_4 (Orange).
  - Since the loop P (formed by v, v_1, v_3 and the Blue-Red chain) intersects the plane, v_2 (Green) must lie "inside" P and v_4 (Orange) must lie "outside" P (or vice versa).
  - Therefore, there cannot be a Green-Orange Kempe chain connecting v_2 and v_4, because such a chain would have to cut through the loop P, which violates the planarity of the graph (edges cannot cross each other).
  - Since v_2 and v_4 are not connected by a Green-Orange chain, we can apply the logic: We reverse the colors on the Green-Orange chain starting from v_2.
  - Result: v_2 (which was Green) now becomes Orange. Vertex v now has no Green neighbors. We can color v Green24.

CHAPTER 3 – LO3: INVESTIGATE SOLUTIONS TO PROBLEM SITUATIONS USING THE APPLICATION OF BOOLEAN ALGEBRA

3.1 (P5) Diagram a binary problem in the application of Boolean algebra

3.1.1 **Introduction to Boolean Algebra in Solving Binary Problems**

Boolean Algebra is crucial in solving binary problems in diverse real-world domains.

In Computer Science: It plays a fundamental role in digital circuit design, where binary logic gates use Boolean operations for processing and controlling information flow.

In Software Programming: Boolean algebra is employed in decision-making processes, conditionals, and algorithms, contributing to efficient problem-solving.

The Binary System is a base-2 numeral system that uses two digits, 0 and 1, called bits4. Electronic circuits and CPUs operate fundamentally on these binary signals: 0 (representing LOW voltage) and 1 (representing HIGH voltage).

3.1.2 Basic Logic Gates

To diagram a binary problem, we use logic gates. Logic gates are electronic devices that perform logical operations on one or more binary inputs to produce a binary output.

- **NOT Gate (Inverter):**
  - Accepts one Boolean variable as input and produces the complement of this value as its output.
  - Boolean expression: $B = \bar{A}$ ($or\ B = NOT\ A$).
  - If the input is 1, the output is 0, and vice versa.
- **AND Gate:**
  - The output of an AND gate is 1 only if all of its inputs are 1; otherwise, the output is 0.
  - This represents the Boolean product.
  - Boolean expression: $C = A \cdot B(or\ xy)$.

❖ **OR Gate:**

- The output of an OR gate is 1 if at least one of its inputs is 1.
- This represents the Boolean sum.
- Boolean expression: C = A + B.
- Additionally, gates such as NAND, NOR, and XOR are widely used in manufacturing because they are cost-effective, stable, and use fewer transistors.

❖ **Application Domains**

a. Hardware Domain (Digital Circuit Design)

- In digital circuit design, complex Boolean expressions are translated into schematic logic circuits to perform specific tasks.
- Problem: A logical method is required to control signal flow and perform calculations based on voltage states (0 and 1).
- Solution: Logic gates (AND, OR, NOT, etc.) serve as the physical manifestations of Boolean operations. For example, CPUs use them within the Arithmetic Logic Unit (ALU) for computation.

b. Software Domain (Programming & Decision Making)

- In programming, software programs must constantly evaluate scenarios to result in a TRUE or FALSE outcome.
- Problem: Algorithms need a way to determine which path of execution to take based on dynamic conditions.
- Solution: Developers use algebraic operators (AND &&, OR ||, NOT !) to combine conditions. For example: Authentication systems verify users (Username_Correct AND Password_Correct).

3.1.3 Activity 1 Part I

Diagram a binary problem in Boolean algebra, illustrating number representation and logic gate usage in two diverse real-world domains and the practical applications therein.

Domain 1 – Hardware (Digital Circuit Design / Computer Architecture)

❖ **Binary Problem:**

- CPUs and electronic circuits operate fundamentally on binary signals: 0 (LOW voltage) and 1 (HIGH voltage).
- A logical method is required to control signal flow, make hardware-level decisions, and perform calculations based on these voltage states.

❖ **How Boolean Algebra solves it:**

- Physical Implementation: Logic gates (such as AND, OR, NOT, NAND, NOR, XOR) are the physical manifestations of Boolean operations.
- Circuit Design: Complex Boolean expressions are translated into schematic logic circuits to perform specific tasks.
  - ✓ *Example:* An AND gate uses transistors to ensure current flows only if both input terminals are HIGH (1).
  - ✓ *Note:* The NAND gate is widely used in manufacturing because it is cost-effective, stable, and uses fewer transistors to implement other logic functions.

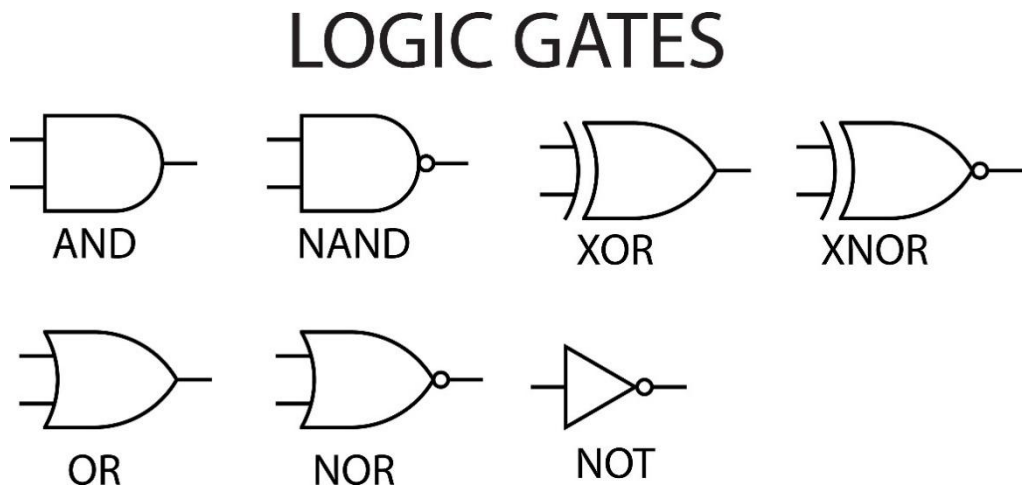


Figure 11 LOGIC GATES

❖ **Practical Applications:**

- CPU Design: It is the core of the Arithmetic Logic Unit (ALU) and Control Units.
- Computational Circuits: Used in adders (for math), multiplexers (for signal routing), and decoders.

- Memory Storage: SRAM and DRAM memory cells operate based on bistable logic circuits (flip-flops).
- Digital Electronics: Found in everyday devices like smartphones, routers, and IoT sensors.

Domain 2 – Software (Programming, Algorithms, & Decision Making)

❖ **Binary Problem:**

- Decision Making: Software programs must constantly evaluate scenarios to result in a TRUE or FALSE outcome.
- Flow Control: Algorithms need a way to determine which path of execution to take based on dynamic conditions.
- How Boolean Algebra solves it: Logical Expressions: In programming, conditional statements are essentially Boolean expressions.
  - ✓ Example: if (age > 18 && isRegistered) { ... }
- Operators: Developers use algebraic operators to combine conditions:
  - ✓ AND (&&): Both conditions must be true.
  - ✓ OR (||): At least one condition must be true.
  - ✓ NOT (!): Inverts the truth value.
- Usage Contexts: Boolean logic controls branching (if/else), loops (while/for), and is essential in search and sort algorithms.

❖ **Practical Applications:**

- Authentication: Verifying users (e.g., Username_Correct AND Password_Correct).
- Game Development: AI behavior trees and triggers (e.g., IF Player_In_Range THEN Attack).
- Optimization: Using Bitwise operations and bitmasking for high-performance computing.
- Embedded Systems: Controlling hardware states in Arduino or PLC programming (e.g., turning a sensor on/off).

3.2 (P6) Produce a truth table and its corresponding Boolean equation from an applicable scenario

3.2.1 Truth table and booleand Equation

❖ **Truth Table**

- Definition: The Truth Table is a systemizing tool that lists all possible combinations of input values (0 or 1) and the corresponding output value of the Boolean Function (F).
- Purpose: It is the first step in converting a natural language problem into a mathematical model, ensuring every logical condition is considered.
- Rule: For n input variables, there will be 2^n rows in the truth table.

p	q	p ^ q
T	T	T
T	F	F
F	T	F
F	F	F

Figure 12 Truth table

❖ **Boolean Equation**

- A Boolean expression is an algebraic expression that uses input variables and Boolean operations (· (AND), + (OR), and overbar (NOT)) to precisely represent the Boolean function identified in the Truth Table.
- Sum-of-Products (SOP) Form (or Disjunctive Normal Form):
  - ✓ Any Boolean function can be represented in SOP form[8]. This is the Boolean sum (OR operation) of distinct minterms.
  - ✓ A Minterm is a Boolean product of n literals (the variables or their complements). A variable is left uncomplemented if its value is 1 and complemented if its value is 0 in that row.
  - ✓ The SOP Equation is formed by taking the Boolean sum of the Minterms corresponding to the rows where the function output (F) is 1.

3.2.2 Activity 1 Part II.

Produce a truth table and corresponding Boolean equation from a given scenario by breaking it down into components and re-expressing it logically.

Develop the truth table and derive the corresponding Boolean equation for the following scenario.

- **Scenario:** "In a smart car system, the dashboard warning light turns ON if the Engine is running AND either the Driver's Door is open OR the Seatbelt is NOT fastened."
- **Solution**: Let's break down the given scenario into its components and create a truth table:
  - E represents "Engine is running."
  - D represents "Driver's Door is open."
  - S represents "Seatbelt is fastened" (Note: The scenario says *NOT* fastened, so we will treat 'Not Fastened' as logic 0, or use NOT S).
    - ✓ Let's simplify for the boolean variable: Let B represent "Seatbelt is unfastened" (1 = Unfastened, 0 = Fastened).
  - W represents "Warning light turns ON."

    The statement can be expressed as: $E\ AND\ (\ D\ OR\ \ B) \rightarrow\ W$

- The truth table is:

E	D	B	D OR B	E AND (D OR B)	W
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	1

1	1	0	1	1	1
1	1	1	1	1	1

Table 7 Truth table

(Note: The warning light only turns on (1) when the Engine is 1 AND at least one of the unsafe conditions D or B is 1).

The Boolean equation corresponding to the given scenario is:

$$W=E\cdot(D+B)$$

Or in expanded Sum-of-Products form:

$$W=E\cdot\overline{D}\cdot B+E\cdot D\cdot\overline{B}+E\cdot D\cdot B.$$

Explan:This equation represents the logical conditions for the car's safety system. The warning light (W) activates only when the engine is active (E=1) and there is a safety breach: either the door is open (D=1) or the seatbelt is unfastened (B=1).

- **Generate a truth table for the provided Boolean expression:**

$$F(x,y,z)=x\cdot\bar{y}+y\cdot z$$

Solution:

x	y	z	$\bar{y}$ (NOT y)	x·$\bar{y}$	y·z	Output F
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1

1	0	1	1	1	0	1
1	1	0	0	0	0	0
1	1	1	0	0	1	1

Table 8 Truth table for the provided Boolean

3.3 (M3) Simplify a Boolean equation using algebraic methods

3.3.1 Introduction

A Boolean algebra is a mathematical structure that is similar to a Boolean ring, but that is defined using the meet and join operators instead of the usual addition and multiplication operators. **(Weisstein, E.W , 2025)**

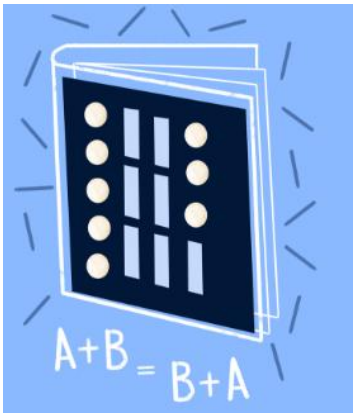


*Figure 13**Boolean algebra***

In every digital system, Boolean simplification is essential because it reduces the number of logic gates needed, which reduces hardware costs, circuit complexity, power consumption, and latency. Therefore, Boolean simplification is an important step before designing an optimal logic circuit

3.3.2 Fundamental Boolean Laws and Properties

Let A, B, C be Boolean variables. For each of the following simplification procedures, the following Boolean laws will apply:

- **Identity Laws**

These rules state that for each combination of a Boolean variable with a neutral element, the variable will not change.

$$A + 0 = A$$

$$A \cdot 1 = A$$

Adding 0 doesn't change anything, and multiplying by 1 doesn't change anything.

❖ **Null / Dominance Laws**

These laws will often describe extreme conditions where the outcome is completely determined regardless of the value of the variable.

$$A + 1 = 1$$

$$A \cdot 0 = 0$$

- If OR with 1, the result will always be 1.
- If AND with 0, the result will always be 0.

❖ **Idempotent Laws**

Repeating with the same values in an OR or AND operation from there will produce the same value.

$$A + A = A$$

$$A \cdot A = A$$

This would all reflect logical redundancy.

❖ **Complement Laws**

A variable with its complement combined with each of its elements will produce an absolute result of either True or False.

$$A + \bar{A} = 1$$

$$A \cdot \bar{A} = 0$$

These laws will be necessary so that we can eliminate provisions that are unfair or redundant.

❖ **Commutative Laws**

The order of each variable in the OR or AND operations does not affect the result.

$$A + B = B + A$$

$$A \cdot B = B \cdot A$$

This will then allow for free rearrangement of terms during simplification.

❖ **Associative Laws**

The grouping of terms does not affect the result.

$$A + (B + C) = (A + B) + C$$

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

These rules will often allow expressions to be expanded or grouped without changing the logic.

❖ **Distributive Laws**

These laws allow variables to be expanded or factored.

$$A(B + C) = AB + AC$$

$$A + BC = (A + B)(A + C)$$

The second form is extremely powerful for reductions using factoring techniques.

❖ **Absorption Laws**

Absorption is about eliminating each redundant term because it is already implied by other terms.

$$A + AB = A$$

$$A(A + B) = A$$

These rules often produce the greatest simplifications in any given algebraic operation.

❖ **De Morgan's Theorems**

These theorems, with their conversion of OR to AND (and vice versa) on negation, will often allow simplifications for otherwise very complex additions.

$$\overline{AB} = \bar{A} + \bar{B}$$

They are particularly useful when dealing with complemented groups or expressions inside parentheses.

3.3.3 General Approach to Algebraic Simplification

Simplifying Boolean expressions is not simply a mechanical process; it may require the application of laws to a system of equations in a logical order. The following structured approach is often used when simplifying Boolean expressions:

- **Expression Expansion:** With each application of each distributive law so that it will be analyzed for the terms that are nested or from which it will be composed. With such things it will be easy to get with the forms with each hidden pattern.
- **Combining similar terms:** With each application of each commutative law and from which it will be combined to group with variables or each term of the similar surface it will be able to combine or eliminate.
- **Factor expression:** With the extraction of common factors by can be used with the laws of distribution to find the opportunities of absorption or further reduction.
- **Removing redundant components:** Can be applied with the laws that are not homogeneous, and also as idempotent so that from here it will be eliminated with each unnecessary element.
- **Applying the absorption laws:** It is using these laws to reduce each expression to simpler forms in a very efficient way. This will often produce the greatest simplification.
- **Applying the De Morgan Theorem (when the complement appears):** With it will be converted of the groups being added to each infinitely simpler form then from here it can all be combined or also from here will be reduced.
- **Repeating with each reduction cycle:** Can be continued and also from here will apply with the laws until it will not be able to reduce any more and the expression reaches the minimum form.

3.3.4 Activity 1 Part III

Simplify a Boolean equation using algebraic methods.

- Simplify the following Boolean expressions.

a, $\boldsymbol{x(x+y)+y(y+z)+z(z+x)}$

- Step 1 — Apply Distributive Law

$$x(x+y) = xx + xy$$

$$y(y+z) = yy + yz$$

$$z(z+x) = zz + zx$$

Substituting these:

$$F = xx + xy + yy + yz + zz + zx$$

- Step 2: Apply the Idempotent Law $AA = A$

$$xx = x, yy = y, zz = z$$

Therefore:

$$F = x + xy + y + yz + z + zx$$

- Step 3: Apply the Absorption Law $A + AB = A$

$$x + xy = x$$

$$y + yz = y$$

$$z + zx = z$$

Thus the entire expression becomes:

$$F = x + y + z$$

Hence, the simplified Boolean expression is:

$$F = x + y + z$$

b, $(\boldsymbol{x} + \bar{\boldsymbol{y}})(\boldsymbol{y} + \boldsymbol{z}) + (\boldsymbol{x} + \boldsymbol{y})(\boldsymbol{z} + \bar{\boldsymbol{x}})$

- Step 1 — Expand the first product by distributive law:

$$(x + \bar{y})(y + z) + (x + y)(z + \bar{x})$$

(Use $A(B + C) = AB + AC$ with $A = x + \bar{y}$.)

So

$$F = (y + z)x + (y + z)y + (x + y)(z + \bar{x})$$

- Step 2 — Distribute inside the first two terms:

$$(y + z)x = yx + zx$$

$$(y + z)\bar{y} = y\bar{y} + z\bar{y}$$

Thus

$$F = yx + zx + y\bar{y} + z\bar{y} + (x + y)(z + \bar{x})$$

- Step 3 — Apply complement law $A\bar{A} = 0$:

$$y\bar{y} = 0$$

So

$$F = yx + zx + 0 + z\bar{y} + (x + y)(z + \bar{x})$$

$$\Rightarrow F = yx + zx + z\bar{y} + (x + y)(z + \bar{x})$$

- Step 4 — Expand the second big product $(x + y)(z + \bar{x})$:

$$(x + y)(z + \bar{x}) = (x + y)z + (x + y)\bar{x}$$

$$= xz + yz + x\bar{x} + y\bar{x}$$

(Distribution twice.)

- Step 5 — Substitute and simplify known complements $x\bar{x} = 0$:

Now

$$F = yx + zx + zy + xz + yz + 0 + yx$$

$$\Rightarrow F = yx + zx + z\bar{y} + xz + yz + y\bar{x}$$

- **Step 6 — Combine like terms and use idempotent/commutative laws**

Note yx and xz etc. We can group terms containing y:

Group yx and $y\bar{x}$:

$$yx + y\bar{x} = y(x + \bar{x})$$

$$(Use\ factorization\ AB + A\bar{B} = A(B + \bar{B}) or\ distributive\ law.)$$

But $x + \bar{x} = 1$*(complement law), so*

$$y(x + x) = y \cdot 1 = y.$$

So those two terms reduce to y. Substitute:

Remaining terms are zx + xz(which are same as xz), z$\bar{y}$, and yz.

So after this reduction:

$$F = y + zx + z\bar{y} + yz$$

- **Step 7 — Observe $zx + xz = zx$ (idempotent/commutative) so keep zx. Now combine terms with z:**

  Group $zx + z\bar{y} + yz$ by factoring z where useful:

  $$zx + zy + yz = z(x + y + y)$$

  $$(\text{Use distributive law } \mathrm{z(x + \bar{y}) + yz = z(x + \bar{y}) + zy = z(x + \bar{y} + y).})$$

  But $\bar{y} + y = 1$(complement law), hence

  $$x + \bar{y} + y = x + 1 = 1$$

  (since $A + 1 = 1$).

  So

  $$z(x + \bar{y} + y) = z \cdot 1 = z.$$

  (since $A + 1 = 1$).

  So

  $$z(x + \bar{y} + y) = z \cdot 1 = z.$$

- **Step 8 — Substitute back:**

  $$F = z + y$$

  which is the simplified form.

c, $(\boldsymbol{x + y})(\boldsymbol{xz + x\bar{z}}) + \boldsymbol{zx} + \boldsymbol{x}$

Solution (step–by–step with Boolean algebra laws)

- **Step 1 — Factor out x:**

  $$xz + xz = x(z + z)(Distributive\ Law)$$

- **Step 2 — Apply complement law:**

  $$z + z = 1(Complement\ Law)$$

- **Step 3 — Apply identity law:**

  $$x \cdot 1 = x(Identity\ Law)$$

So:

$$xz + x\bar{z} = x$$

- **Step 4 — Substitute back into the original expression:**

$$F = (x + y)x + zx + x$$

- **Step 5 — Distribute:**

$$(x + y)x = xx + yx$$

- **Step 6 — Apply idempotent law $xx = x$:**

$$(x + y)x = x + yx$$

Thus:

$$F = x + yx + zx + x$$

- **Step 7 — Apply absorption law $A + AB = A$:**

$$x + yx = x$$

So:

$$F = x + zx + x = x + zx$$

- **Step 8 — Apply absorption again:**

$$x + zx = x$$

Final Answer

$$F = x$$

d, $\bar{x}(x + y) + (x + y)(x + \bar{y})$

Solution (step–by–step with Boolean laws)

- **Step 1: Expand $\bar{x}(x + y)$**

$$\bar{x}(x + y) = \bar{x}x + \bar{x}y = 0 + \bar{x}y = \bar{x}y$$

So $\bar{x} * x = 0$ according to the Law of Compensation

- **Step 2 Develop** $(x + y)(x + \bar{y})$:

$$x(x + \bar{y}) + y(x + \bar{y})$$

$$= x + x\bar{y} + xy + y\bar{y}$$

- **Step 3 — Expand the product $y(x+y)$using distributive law:**

$$\bar{x}y + x + x\bar{y} + xy$$

- **Step 4 Simplify terms**

$$y\bar{y} = 0$$

$$x\bar{y} + xy = x(\bar{y} + y) = x(1) = x$$

So the whole second part becomes:

$$(x + y)(x + \bar{y}) = x$$

- **Step 5 Combine both parts:**

$$\bar{x}y + x = x + \bar{x}y$$

$$\Rightarrow F = x + \bar{x}y$$

3.4 (D3) Design a complex system using logic gates

3.4.1 Introduction to Group Theory in Cryptography

❖ **Boolean Function Determination**

In digital systems, Binary Decimal Encoding (BCD) is an encoding method that uses a sequence of 4 bits to represent a single decimal digit (from 0 to 9).

- The binary input variables are named w, x, y, z with corresponding weights of 8, 4, 2, and 1.
- Valid range: Only binary combinations corresponding to decimal numbers from 0 to 9 are considered valid.
- Combinations from 10 to 15 (1010_2 *to* 1111_2) are considered "Don't Care" states (not occurring in the 1-digit BCD standard) and can be used to simplify circuits.

❖ **Problem Statement**

Objective: Design a combinational logic circuit using basic logic gates to detect numbers divisible by 3.

Input: A 4-bit BCD code (w, x, y, z) representing a decimal digit ($0 \le N \le 9$).

Condition: The numbers that satisfy the condition of being divisible by 3 within the range of 0-9 are: 0, 3, 6, 9.

3.4.2 Activity

❖ **Analysis and Boolean Function Formulation**

To optimize hardware design and reduce the number of logic gates, instead of using full minterms (4 variables), we will use simplified terms based on the characteristics of BCD code (eliminating unnecessary variables due to input constraints up to 9).

The simplified logical analysis for the numbers 0, 3, 6, and 9 is as follows:

- Digit 0 (0000): All bits must be 0.

$$\rightarrow \bar{x}\,\bar{y}\,\bar{z}\,\bar{w}$$

- Digit 3 (0011): Normally expressed as $\bar{w}\bar{x}yz$. owever, in BCD, if y = 1, z = 1, and x = 0, the value can only be 3 (the case w = 1 would correspond to 11, which is invalid).

$$\rightarrow\ yz\bar{x}$$

- Digit 6 (0110): Normally expressed as $\bar{w}xy\bar{z}$. Similarly, if x = 1, y = 1, and z = 0, the value can only be 6 the case w= 1 would correspond to 14, which is invalid).

$$\rightarrow\ xy\bar{z}$$

- Digit 9 (1001): Normally expressed as $w\bar{x}\bar{y}z$. Only the conditions w = 1 (number ≥ 8) and z = 1 (odd number) are sufficient to identify 9 (the cases 11, 13, and 15 are invalid).

$$\rightarrow wz$$

- Final simplified Boolean expression:

  By OR-ing all simplified terms, the resulting Boolean function is:

$$F = \bar{x}\bar{y}\bar{z}\bar{w} + \bar{x}\,y\,z\ + xy\bar{z} + wz$$

❖ Truth Table for Divisibility-by-3 Function

The truth table below describes the system behavior for BCD inputs and the corresponding simplified terms.

Decimal	Binary Inputs (w x y z)	Output (F)	Simplified Term
0	0 0 0 0	1	$\bar{x}\bar{y}\bar{z}\bar{w}$
1	0 0 0 1	0	–

2	0 0 1 0	0	–
3	0 0 1 1	1	$yz\bar{x}$
4	0 1 0 0	0	–
5	0 1 0 1	0	–
6	0 1 1 0	1	$xy\bar{z}$
7	0 1 1 1	0	–
8	1 0 0 0	0	–
9	1 0 0 1	1	wz

Table 9 Truth Table for Divisibility-by-3

❖ **Circuit Design Description**

Based on the simplified Boolean expression

$$F = \bar{x}\bar{y}\bar{z}\bar{w} + \bar{x}\,y\,z\, + xy\bar{z} + wz$$

the circuit design consists of the following components:

- Inverter Stage (NOT Gates)

  NOT gates are used to generate the complemented signals required, such as $\bar{x}$, $\bar{z}$, and the combined complements for $\bar{x}\bar{y}\bar{z}\bar{w}$.

- Product Stage (AND Gates)

  ✓ One 4-input AND gate (or equivalent combination using NAND/NOR gates) to implement $\bar{x}\bar{y}\bar{z}\bar{w}$(detecting digit 0).

  ✓ One 3-input AND gate to implement $yz\bar{x}$(detecting digit 3).

  ✓ One 3-input AND gate to implement $xy\bar{z}$(detecting digit 6).

  ✓ One 2-input AND gate to implement wz(detecting digit 9).

- Sum Stage (OR Gate)

One 4-input OR gate is used to combine the outputs of all AND gates. If any simplified term evaluates to true (i.e., the input is 0, 3, 6, or 9), the output F is set to logic high (1).

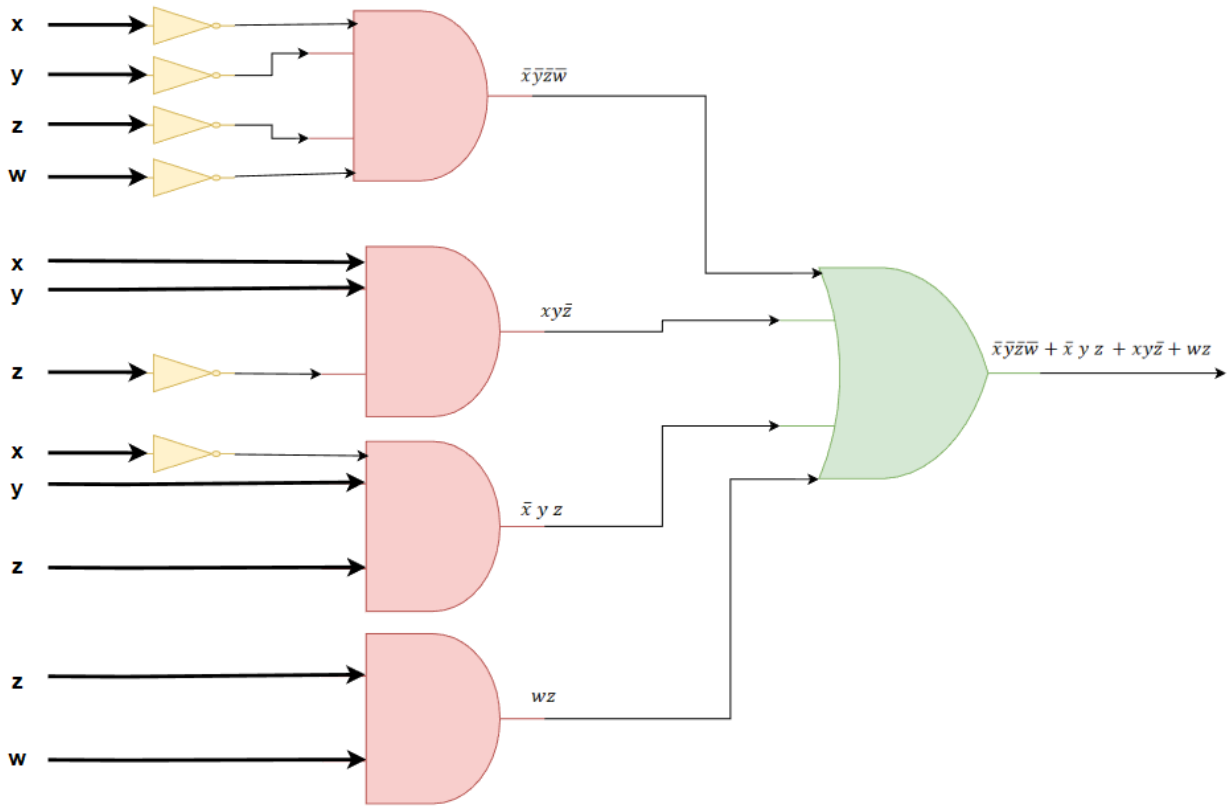


Figure 14 Circuit design

CHAPTER 4 – LO4: EXPLORE APPLICABLE CONCEPTS WITHIN ABSTRACT huALGEBRA

4.1 (P7) Describe the distinguishing characteristics of different binary operations that are performed on the same set

4.1.2 Binary operation

A binary operation on a set $S$is defined as a function that combines any two elements of $S$to produce another element also belonging to S(Gallian, 2017). Although multiple binary operations may be defined on the same underlying set, each operation can exhibit different structural properties. These properties are essential for distinguishing how one operation behaves relative to another and for determining the algebraic structures that emerge from these operations.

- **Closure**

Closure refers to the requirement that applying the operation to any two elements of the set results in another element of the same set. According to Fraleigh (2002), closure is the foundational condition that determines whether a given rule is a valid binary operation. Different operations defined on the same set may differ in this property. For example, addition on $\mathbb{Z}$is closed, while exponentiation on $\mathbb{Z}$is not always closed.

- **Associativity**

Associativity determines whether the grouping of elements affects the outcome of the operation. An operation $*$is associative if

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

for all $a, b, c \in S$.

As noted by Dummit and Foote (2004), associativity significantly influences the type of algebraic structure the operation may form. For example, addition is associative on $\mathbb{Z}$, whereas subtraction is not, even though both operations are defined on the same set.

- **Commutativity**

Commutativity describes whether the order of operands impacts the result of the operation. An operation $*$is commutative if

$$a * b = b * a$$

for all $a, b \in S$.

Gallian (2017) emphasizes that commutativity distinguishes many classical operations. For instance, multiplication of integers is commutative, while subtraction is not, although both operate on the set $\mathbb{Z}$.

❖ **Identity Element**

An identity element is an element $e \in S$that satisfies

$$a * e = e * a = a$$

for every element $a \in S$.

Fraleigh (2002) highlights that the existence of an identity element is operation-dependent. On the same set $\mathbb{Z}$, the addition operation has an identity (0), while the subtraction operation does not.

❖ **Inverse Element**

Given the existence of an identity, an inverse element for $a \in S$is an element $b \in S$such that

$$a * b = b * a = e.$$

Dummit and Foote (2004) note that different operations on the same set may differ in whether inverses exist for all elements. For example, under addition on $\mathbb{Z}$, every element has an inverse (its negative), while under multiplication, only 1 and −1 have inverses in $\mathbb{Z}$.

❖ **Distributivity**

Distributivity describes the interaction between two binary operations. Operation $*$is said to distribute over operation $\circ if$

$$a * (b \circ c) = (a * b) \circ (a * c).$$

As explained by Gallian (2017), distributivity is a critical property in algebraic structures such as rings. On $\mathbb{Z}$, multiplication distributes over addition, but addition does not distribute over multiplication.

❖ **Idempotence**

An operation $*$is idempotent if

$$a * a = a$$

for all $a \in S$.

Fraleigh (2002) points out that idempotence serves as an important distinguishing feature, particularly in lattice theory. While operations like "min" and "max" on ordered sets are idempotent, conventional operations such as addition and multiplication on $\mathbb{Z}$are not.

Binary operations defined on the same set may vary significantly in their structural characteristics. By examining closure, associativity, commutativity, identity, inverse, distributivity, and idempotence, it becomes possible to distinguish their algebraic behavior in a rigorous and meaningful way. These properties determine how operations interact with elements of the set and shape the resulting algebraic structures, including semigroups, monoids, groups, and rings.

4.1.2 Activity 2 Part I

Check whether the operations applied to pertinent sets qualify as binary operations.

a) Subtraction on set of natural numbers.

$$-: \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$
$$(x, y) \mapsto x - y$$

Solution:

The operation of subtraction is not a binary operation on the set of natural numbers ($\mathbb{N}$) because it lacks the closure property.

Choose two elements $x, y \in \mathbb{N}$, for example, let x = 3 and y = 5. Both $3 \in \mathbb{N}$ and $5 \in \mathbb{N}$

Apply the operation: $x - y = 3 - 5 = -2$

The result, -2 is a negative integer and is not an element of the set of natural numbers $\mathbb{N}$.

Since we found at least one pair of elements in $\mathbb{N}$ whose difference is not in $\mathbb{N}$, the subtraction operation does not map $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$, and thus, it does not qualify as a binary operation on $\mathbb{N}$.

b) Exponential operation on set integers.

$$^{:\mathbb{Z}} \times \mathbb{Z} \to \mathbb{Z}$$
$$(x, y) \mapsto x^y$$

Solution:

The exponential operation is not a binary operation on the set of integers $\mathbb{Z}$ because it lacks the closure property.

- Counterexample 1 (Negative Exponent):
  - Choose two elements $x, y \in \mathbb{Z}$, for example, let $x = 2$ and $y = -3$. Both $2 \in \mathbb{Z}$ and $-3 \in \mathbb{Z}$.
  - Apply the operation: $x^y = 2^{-3}$.
  - $2^{-3} = \frac{1}{2^3} = \frac{1}{8}$
  - The result, $\frac{1}{8}$, is a fractional number and is not an element of the set of integers $\mathbb{Z}$.
- Counterexample 2 (Zero Base, Negative Exponent):
  - Choose $x = 0$ and $y = -1$.
  - $x^y = 0^{-1} = \frac{1}{0}$, which is undefined. An undefined result cannot be in $\mathbb{Z}$.

Since we found pairs of integers whose result under exponentiation is either a non-integer fraction or undefined, the operation does not qualify as a binary operation on $\mathbb{Z}$

4.2 (P8) Determine the order of a group and the order of a subgroup in given examples

4.2.1 Basic Concepts of Groups

A Group is a fundamental algebraic structure in discrete mathematics, abstract algebra, and computer science. A group is typically denoted as $(G,\circ)$, where:

- G is a set of elements.
- $\circ$ is a binary operation performed on G.

For $(G,\circ)$ to be called a group, it must satisfy four conditions: closure, associativity, existence of an identity element, and existence of inverse elements.

4.2.2 Closure

Closure is the first axiom a set G and an operation $\circ$ must satisfy to form a group.If a and b are any two elements in the set G, then the result of the operation $a \circ b$ must also be an element of G. In other words, the set G is closed under the operation $\circ$.

Mathematically, for all $a, b \in G$, we must have $a \circ b \in G$.

4.2.3 Associativity

Associativity is the second group axiom, defining the rule for grouping elements in an operation. For any three elements $a, b, and\ c$ in the set G, the order in which the operations are performed does not change the final result. That is, applying the operation to $a\ and\ b$ first, then to c, yields the same result as applying it to $b\ and\ c$ first, then to a.

Mathematically, for all $a, b, c \in G$, we must have $(a \circ b) \circ c = a \circ (b \circ c)$.

4.2.4 Identity element

The identity element is a unique element, often denoted by e, that is a member of the set G.

When the operation $\circ$ is applied to any element $a\ in\ G$ and the identity element e, the result is always the element a itself. The identity element acts as a neutral element for the operation.

Mathematically, there exists an element $e \in G$ such that for all $a \in G$, $a \circ e = e \circ a = a$.

4.2.5 Inverse element

The inverse element states that for every element in the set G, there must be a corresponding element, its inverse, also in G. For every element $a \in G$, there exists an element $a^{-1} \in G$ such that when the operation $\circ$ is applied to a and a^{-1}, the result is the identity element e.

Mathematically, for all $a \in G$, there exists an element $a^{-1} \in G$ such that $a \circ a^{-1} = a^{-1} \circ a = e$.

4.2.6 Order of a Group

The order of a group $(G,\circ)$, denoted by $|\ G\ |$, is the number of elements in the set G.

If G contains a finite number of elements, the group is called a finite group, and its order is a specific natural number.

If G contains an infinite number of elements, the group is called an infinite group, and its order is said to be ∞.

Example 1: Finite Group

Consider the set $G = \{0, 1, 2, 3\}$ and the operation $\oplus$(addition modulo 4). The group is ({0, 1, 2, 3}).

The elements of the set are $0, 1, 2, 3$.

The total number of elements in $G\ is\ 4$.

Therefore, the order of the group is $|G| = 4$.

Note: This group is often denoted as $\mathbb{Z}_4$(the integers modulo 4).

Example 2: Infinite Group

Consider the set of all integers, $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$, and the operation $+$ (standard addition). The group is $(\mathbb{Z}, +)$.

The set $\mathbb{Z}$ contains an infinite number of elements.

Therefore, the order of the group is $|\mathbb{Z}| = \infty$.

4.2.7 Order of a Subgroup

A subgroup H of a group G is a subset of G that is itself a group under the same operation as G.

The order of a subgroup H, denoted by $|H|$, is simply the number of elements in the set H.

If H is a finite set, $|H|$ is a natural number.

If H is an infinite set, $|H| = \infty$.

By Lagrange's Theorem, for any finite group G, the order of any subgroup $H\ of\ G$ must be a divisor of the order of G (i.e., $|H|$ divides $|G|$).

Example 3: Subgroup of Even Integers

Let:

$$G = (\mathbb{Z}, +)$$

$$H = 2\mathbb{Z} = \{\dots, -4, -2, 0, 2, 4, \dots\}$$

Both sets contain infinitely many elements:

$$|\ G\ | = \infty, |\ H\ | = \infty$$

Example 4: Finite Subgroup of a Finite Group

Consider the group of integers modulo 6 under addition, $G = (\mathbb{Z}_6, \oplus)$, where $\mathbb{Z}_6 = \{0,1,2,3,4,5\}$. The order of the group is $|G| = 6$.

Let H be the cyclic subgroup generated by the element 2, denoted as $< 2 >$:

$$H =< 2 >= \{n \circ 2 \ (mod\ 6)\ n\}$$

The elements of H are calculated by repeatedly adding 2:

$$1 \circ 2 = 2$$

$$2 \circ 2 = 4$$

$3 \circ 2 = 6 = 0$ (mod 6) (back to the identity)

$4 \circ 2 = 8 = 2$ (mod 6) (repeats)

The distinct elements in the subgroup are $H = \{0, 2, 4\}$.

Check Subgroup Conditions: H is a subset of G, and it forms a group under$\oplus$.

Determine Order: The total number of elements in H is 3.

Therefore, the order of the subgroup is $|H| = 3$. (Note that 3 is a divisor of 6, which is consistent with Lagrange's Theorem).

4.2.8 Activity 2 – Part II

Construct the operation tables for group G with orders 1, 2, 3, and 4, utilizing the elements a, b, c and e as the identity element in a suitable manner.

Solution

The operation table (or Cayley table) displays the result of the binary operation between every pair of elements in a finite group G. Since these groups are small, they are generally cyclic or can be constructed based on the properties of a group (closure, identity element e, and inverses).

❖ **Group of Order 1: $G_1 = \{e\}$**

This is the trivial group, containing only the identity element e

∘	e
e	e

❖ **Group of Order 2: $G_2 = \{e, a\}$**

The identity is e. Since every element must have an inverse, and $a \neq e$, a must be its own inverse (i.e., $a \circ a = e$).

∘	e	a
e	e	a
a	a	e

❖ **Group of Order 3: $G_3 = \{e, a, b\}$**

Any group of prime order is cyclic. Let a be the generator.

∘	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

❖ **Explanation**:

- a∘a must be the only remaining element, b.
- a ∘ b must be the identity e (since G_3 is cyclic, $a^3 = e$).
- b ∘ a: Since the group is cyclic (and thus Abelian), b ∘ a = a ∘ b = e.
- b ∘ b: Since $b = a^2, b \circ b = a^4 = a^3 \circ a = e \circ a = a$.

❖ **Group of Order 4:** $\boldsymbol{G_4 = \{e, a, b, c\}}$

There are two distinct group structures of order 4 (up to isomorphism):

The Cyclic Group $\mathbb{Z}_4$(e.g., generated by a)

The Klein Four-Group V_4 (every non-identity element is its own inverse).

Case 1: Cyclic Group $\mathbb{Z}_4$

$\boldsymbol{G_4} = \{e, a, a^2, a^3\}$. Let $b = a^2\ and\ c = a^3$. The generator a has order 4, so $a^4 = e$.

∘	e	a	b	c
e	e	a	b	c
a	a	b	c	e
b	b	c	e	a
c	c	e	a	b

Case 2: Klein Four-Group V_4

In this group, every non-identity element has order 2, meaning $a \circ a = e, b \circ b = e, and\ c \circ c = e$

∘	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

⇨ Explanation: Due to closure, the product of any two distinct non-identity elements must be the third non-identity element. For instance:

- a ∘ b cannot be a (since b ≠ e).

- $a \circ b$ cannot be b (since $a \neq e$).
- $a \circ b$ cannot be e (since b is the inverse of b, and $a \neq b$).
- Therefore, $a \circ b = c$.

4.2.9 State the Lagrange's theorem of group theory. Using this theorem to discuss whether a group K with order 4 can be a subgroup of a group G with order 9 or not. Provide a clear exposition of the reasons.

Solution

a) Statement of Lagrange's Theorem

Lagrange's Theorem states that for any finite group G, the order (the number of elements) of any subgroup H of G must be a divisor of the order of G.

Mathematically, if H is a subgroup of a finite group G, then:

$$|H| \text{ divides } |G|$$

Where $|G|$ is the order of group G and $|H|$ is the order of subgroup H.

b) Discussion using Lagrange's Theorem

We are asked to discuss whether a group K with order 4 (i.e., $|K|=4$) can be a subgroup of a group G with order 9 (i.e., $|G|=9$).

Conclusion: No, the group K cannot be a subgroup of group G.

c) Clear Exposition of the Reasons:

❖ **Identify the Orders:**

- The order of the potential subgroup K is $|K| = 4$.
- The order of the group G is $|G| = 9$.
- Apply Lagrange's Theorem:
- If K were a subgroup of G, then according to Lagrange's Theorem, the order of K ($|K|=4$) must divide the order of G ($|G|=9$).

- Check for Divisibility:
- We check if 4 divides 9.
- $\frac{9}{4} = 2.25$, which is not an integer.
- Since 4 is not a divisor of 9, the condition required by Lagrange's Theorem is violated.
- Therefore, because the order of group K (4) does not divide the order of group G (9), K cannot exist as a subgroup of G.

4.3 (M4) Validate whether a given set with a binary operation is indeed a group

4.3.1 Introduction

In abstract algebra, determining whether a binary operation will transform a set into a group is a key step in analyzing the structure of mathematical systems. Each of these tests helps us better understand the ways in which individual elements in a set interact with each other, and allows us to apply the powerful properties of group theory to many fields, including computer science, cryptography, and mathematical modeling.

In this section, we will look at the set

$$S = \mathbb{R} \setminus \{-1\}$$

together with the operation defined by

$$a * b = a + b + ab.$$

The task is to check whether (S,$*$) satisfies all four group axioms, including: closure, associativity, identity element and inverse element.

4.3.2 The Connection Between the Order of a Group and the Quantity of Binary Operations

- ❖ Order of a Group
  - The order of a group G, denoted $|G|$, is the number of elements in the group.
  - If a finite group has n elements, then there exist n^2 pairs (a, b) that can all be realized with binary operations.
- ❖ Binary Operation on a Finite Group

- One of the binary operations on the set S assigns to each pair (a, b) one of the new elements in S itself.
- For each finite group, the entire result of the operation can be represented in a Cayley table of size $n * n$.
- Cayley tables must reflect four group axioms: with results in the set (closure), associative operations, existence of identity elements, every element has an inverse.

❖ Implications

- Larger groups give more complex tables of operations.
- Each group of prime order has infinitely simpler structures, always cyclic.
- Understanding the relationship between the number of elements and the number of operations will help us to evaluate the complexity of the algebraic structure.

❖ Problem Solving

Check whether the set $S = \mathbb{R} \setminus \{-1\}$ is one of the groups under binary operation $*$ defined by $a * b = a + b + ab$ for all $a, b \in S$

- **Closure**

  Take any $a, b \in S$ We need to prove that $a * b \neq -1$.

  Suppose the opposite

  $$a + b + ab = -1.$$

  Change:

  $$ab + a + b + 1 = 0 \Leftrightarrow (a + 1)(b + 1) = 0.$$

  Because $a, b \neq -1$, so $a + 1 \neq 0$ and $b + 1 \neq 0$.

  Therefore the above equation cannot occur.

⇨ $a * b \neq 1, so\ a * b \in S$.

⇨ The closure is satisfied.

- **Associativity**

  We prove $(a * b) * c = a * (b * c)$.

  Caculater $(a * b) * c$:

$$(a * b) * c = (a + b + ab) + c + (a + b + ab)c$$

$$= a + b + c + ab + ac + bc + abc.$$

Caculater $a * (b * c)$:

$$a * (b * c) = a + (b + c + bc) + a(b + c + bc)$$

$$= a + b + c + bc + ab + ac + abc.$$

The two expressions are identical

⇨ Associativity is satisfired.

- **Inverse Element**

Prize $a * b = 0$

$$a + b + ab = 0 \Leftrightarrow b(1 + a) = -a.$$

Therefore

$$b = \frac{-a}{1 + a}.$$

So $a \neq -1$, should be sample $1 + a \neq 0$, valid expression

Check if $b = -1$ occurs.

$$\frac{-a}{1 + a} = -1 \Leftrightarrow -a = -1 - a \Leftrightarrow 0 = -1, contradiction$$

$Every\ element\ has\ an\ inverse\ in\ S$

⇨ Summary : Because of math $a * b = a + b + ab$ on the set $S = \mathbb{R} \setminus \{-1\}$ Fully satisfying the four group axioms (Closure, Associativity, Identity, Inverse), we conclude that:

$$(S *)\ is\ a\ group.$$

$Besides$: This is an abelian group because $a * b = b * a$. Group $(S,*)$ isomorphic to the multiplication group $\mathbb{R}\setminus\{0\}, °)$ through mapping $\varphi(x) = x + 1$. Since S is infinite, the group has infinitely many elements and cannot be represented by a finite Cayley table.

4.4 (D4) Explore, with the aid of a prepared presentation, the application of group theory relevant to your given example

4.4.1 Introduction

In the modern digital age, secure communication has become an essential component in almost every aspect of daily life, from online banking to the exchange of confidential information over the internet. Ensuring that data is transmitted securely requires not only simple arithmetic operations but instead, it is based on advanced mathematical structures from abstract algebra. Group theory, a core branch of discrete mathematics, provides the theoretical foundation for some of the cryptographic systems currently in use around the world.

This paper explores the application of Group Theory in Public Key Cryptography, with a particular focus on the RSA algorithm. The goal is to demonstrate that group-theoretic concepts, such as modular arithmetic, finite groups, and Euler's theorem, are not just abstract mathematical ideas, but are directly responsible for maintaining global information security. By analyzing RSA through the lens of algebraic structures, this paper highlights the important connection between abstract algebra and real-world cryptographic practices.

4.4.2 Mathematical Background

❖ Modular Arithmetic

Modular arithmetic is the mathematical framework on which modern cryptographic systems are built. The notation indicates that integers *a* and *b* leave the same remainder when divided by *n*. This arithmetic behaves well under addition, multiplication, and exponentiation, making it suitable for secure computations.

$$a \equiv b(mod n)$$

❖ The Multiplicative Group of Integers Modulo n, $\mathbb{Z}_n^*$

The set

$$Zn* = \{a \in Z \mid 1 \leq a < n,\ gcd(a,n) = 1\}$$

contains all integers less than *n* that are relatively prime to it. Under multiplication modulo *n*, this set forms a group because:

- It is closed under multiplication mod *n*.
- The operation is associative.

- It contains an identity element (1).
- Every element has a multiplicative inverse modulo *n*.

Thus, $\mathbb{Z}_n^*$ is a finite group whose order is given by Euler's phi function:

$$| Zn * |= \phi(n)$$

❖ **Euler's Theorem and Group Theory**

Euler's theorem states:

$$a^{\phi(n)\equiv}1 \ (modn) \ \ for\ any\ a \in Zn*$$

This result is a direct consequence of Lagrange's Theorem, which states that the order of any element in a finite group divides the order of the group. This theoretical fact is the key to the RSA decryption process and ensures that exponentiation cycles predictably within the group.

4.4.3 RSA Algorithm Explained Using Group Theory

RSA is one of the most widely used public-key cryptographic systems. Its security and functionality rely heavily on the group-theoretic properties outlined above.

❖ **Key Generation**

- Choose two large prime numbers:

$$p\ and\ q$$

- Compute

$$n = pq$$

The group of interest is now $\mathbb{Z}_n^*$.

- Compute the size of the group:

$$\phi(n) = (p-1)(q-1)$$

- Select the public exponent *e* such that:

$$1 < e < \phi(n), \qquad gcd(e, \phi(n)) = 1$$

- Compute *d* such that:

$$ed \equiv 1 \pmod{\phi(n)}$$

Here, d is the multiplicative inverse of e within the group of integers modulo $\phi(n)$.The pair $(e, n)$is the public key, and $(d, n)$is the private key.

❖ **Encryption**

For a message m, where $m < n$, the ciphertext is:

$$c = m^e \ (mod\ n)$$

This operation is exponentiation within the group modulo n.

❖ **Decryption**

The receiver recovers the message using:

$$m = c^n \ (mod\ n)$$

Why does this work?

Because:

$$c^n = (m^e)^d = \ m^{ed} \equiv m^{1+k\phi(n)} \equiv m \ (mod\ n)$$

based on Euler's theorem and the cyclic structure of the group.

Thus, group theory guarantees that decryption reverses encryption correctly and securely.

❖ **Why Group Theory Ensures Security**

RSA is secure because:

- Multiplying large primes $p$and $q$to obtain *n* is easy.
- Factoring n back into $p$and $q$is computationally infeasible for large values (a hard number-theoretic problem).
- Without knowing $\phi(n)$, an attacker cannot compute the inverse exponent *d*.
- All security relies on group-theoretic properties such as inverses, modular exponentiation cycles, and the difficulty of reversing group operations without structural knowledge.

⇨ Thus, the RSA algorithm is an example of a trapdoor function—easy to compute in one direction but extremely hard to reverse without private group information.

CONCLUSION

Summary of main content: This report has completed the survey and application of the four main pillars of Discrete Mathematics: Set Theory & Functions (LO1), Graph Theory (LO2), Boolean Algebra (LO3), and Abstract Algebra (LO4). Contents ranging from data modeling, finding the shortest path (Dijkstra), designing digital logic circuits, to analyzing RSA security have all been covered in detail.

Final results: All problems posed have been thoroughly solved: finding the optimal path in the graph, successfully designing a divisibility test circuit with a minimum number of gates, and proving group axioms in algebraic structures. The results confirm that Discrete Mathematics is an indispensable foundation for building efficient algorithms and secure systems in Information Technology.

EVALUATION

- Accuracy: The calculation results in the truth table, Dijkstra's algorithm, and Boolean algebraic transformation steps all ensure high accuracy, strictly adhering to the stated mathematical theorems and axioms.
- Strengths: The logic circuit design solution has been optimized through Boolean function simplification, saving hardware resources. The application of group theory to RSA demonstrates in-depth thinking about data security.
- Weaknesses: The graph and number group models used in this paper are still on a small scale (e.g., graphs with few vertices, low-level groups) to serve manual calculations, not fully reflecting the complexity of real-world "Big Data" systems.
- Opportunities & Challenges: This approach provides a solid foundation for developing more complex algorithms; however, scaling up will require the support of automated computing tools to maintain performance.

REFERENCES

Cuemath (2025). Sets - Types, Symbols, Properties, Examples | Sets in Math. [online] Cuemath. Available at: https://www.cuemath.com/algebra/sets/.

Xmind , 2025 . What Is a Venn Diagram: Definition, Applications, and Examples. [online] Available at: https://xmind.com/blog/what-is-a-venn-diagram.

Wikipedia , 2022 . Multiset. [online] Available at: https://en.wikipedia.org/wiki/Multiset.

Kapoor, A. (2023). What is Dijkstra's Algorithm? Here's How to Implement It with Example? [online] Simplilearn.com. Available at: https://www.simplilearn.com/tutorials/cyber-security-tutorial/what-is-dijkstras-algorithm.

courses.lumenlearning(2025). Hamiltonian Circuits | Mathematics for the Liberal Arts Corequisite. A vailable at: https://courses.lumenlearning.com/mathforliberalartscorequisite/chapter/hamiltonian-circuits/.

Rosen, K. H., 2011. *Discrete Mathematics and Its Application.* 7th McGraw-Hill.

Sciencedirect(2025) Binary Coded Decimal - an overview | ScienceDirect Topics. [online] Available at: https://www.sciencedirect.com/topics/computer-science/binary-coded-decimal.

Dummit, D. S., & Foote, R. M. (2004). *Abstract Algebra* (3rd ed.). Wiley.

Fraleigh, J. B. (2002). *A First Course in Abstract Algebra* (7th ed.). Addison-Wesley.

Gallian, J. A. (2017). *Contemporary Abstract Algebra* (9th ed.). Cengage Learning.