ASSIGNMENT FINAL REPORT

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**	
Student Name	Nguyen Tuan Anh	**Student ID**	BH00667
Class	SE06203	**Assessor name**	Ta Quang Hieu

Student’s signature	Anh

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: 6	**Student names & codes**	**Final scores**	**Signatures**
	1. Nguyen Nang Trong– BH00676		Trong
	2. Hoang Tien Manh– BH00677		Manh
	3. Luu Duc Thuan– BH00690		Thuan
	4. Nguyen Tuan Anh– BH00667		Anh
Class	SE06203	**Assessor name**	Ta Quang Hieu

P5	P6	P7	P8	M3	M4	D3	D4

OBSERVATION RECORD

Student			
Description of activity undertaken			
Assessment & grading criteria			
How the activity meets the requirements of the criteria			
Student signature:		**Date:**	
Assessor signature:		**Date:**	
Assessor name:			

r Summative Feedback:		r Resubmission Feedback:
Grade:	Assessor Signature:	Date:
Internal Verifier’s Comments:		
Signature & Date:		

Table of Contents

I. Introduction

Discrete Mathematics serves as the foundational framework for numerous branches of computer science, engineering, and mathematics. Unlike continuous mathematics, which deals with objects that can vary smoothly, discrete mathematics focuses on distinct, separate values. It encompasses various topics such as logic, set theory, combinatorics, graph theory, and more, offering a powerful toolkit for problem-solving in diverse fields.

One of the fundamental aspects of discrete mathematics is its applicability in computer science. Algorithms, the backbone of computing, heavily rely on discrete structures and principles. Understanding discrete mathematics is crucial for designing efficient algorithms, analyzing their complexity, and ensuring the reliability and security of computational systems.

Moreover, discrete mathematics finds extensive use in cryptography, where it provides the theoretical foundation for encryption and decryption techniques. Concepts like modular arithmetic and number theory play a pivotal role in developing secure cryptographic algorithms, essential for safeguarding sensitive information in today's digital age.

In addition to its significance in computer science, discrete mathematics has profound implications in other disciplines as well. In engineering, it aids in modeling and analyzing discrete systems such as digital circuits and networks. In economics and social sciences, discrete mathematical techniques are employed for modeling decision-making processes, resource allocation, and social network analysis.

Furthermore, discrete mathematics offers elegant solutions to various real-world problems by utilizing principles of combinatorics and graph theory. Whether it's optimizing routes in transportation networks, scheduling tasks in project management, or analyzing biological networks, discrete mathematics provides powerful tools for modeling, analyzing, and solving such problems efficiently.

In this introductory report, we will delve into the core concepts of discrete mathematics, exploring its foundational principles, applications, and significance across different domains. Through this exploration, we aim to provide a comprehensive understanding of the role of discrete mathematics in shaping modern science and technology, and its indispensable contribution to problem-solving in diverse fields.

II. Content

LO1 Examine set theory and functions applicable to software engineering

Activity 1

P1 Perform algebraic set operations in a formulated mathematical problem.

Part I: Stick in mind that a < b represents the largest digits in your ID.

1. Let A and B be two non-empty finite sets. Assume that cardinalities of the sets A, B and A ∩ B are $\overline{9b}$, $\overline{2a}$ and a + b, respectively. Determine the cardinality of the set A ∪ B.

+ According to the formula for calculating the quantity of the union of two sets, we have:

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

$= \overline{9b} + \overline{2a} - (a + b)$

$= (90 + b) + (20 + a) - (a + b)$

= 90 + b + 20 + a – a – b

= 110

So |A ∪ B| is 110.

2. Suppose | A - B |= $\overline{3a}$ | A ∪ B |= $\overline{11b}$ and | A ∩ B |= $\overline{1a}$. Determine | B | .

Solution: We have $|A - B| = 38$, $|A \cup B| = 119$, $|A \cap B| = 18$. Then

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

$\Leftrightarrow 119 = (|A - B| + |A \cap B|) + |B| - |A \cap B|$

$\Leftrightarrow\ 119 = (38 + 18) + |B| - 18$

$\Leftrightarrow\ 119 = 56 + |B| - 18$

$\Leftrightarrow |B| = 81$

3. At a local market, there are $\overline{35b}$ customers. Suppose $\overline{11a}$ have purchased fruits, $\overline{9b}$ have purchased vegetables, $\overline{8a}$ have purchased bakery items, $\overline{4b}$ have purchased both fruits and vegetables, $\overline{3b}$ have purchased both vegetables and bakery items, $\overline{2a}$ have purchased both fruits and bakery items, and $\overline{1a}$ have purchased all three categories. How many customers have not purchased anything?

Suppose:

$|U| = 359$ (All Customers);

$|A| = 118$ (purchased fruits);

$|B| = 99$ (purchased vegetables) ;

$|C| = 88$ (purchased bakery items);

$|D| =$ (customers have not purchased anything)

$|A \cap B| = 49$ (purchased both fruits and vegetables);

$|B \cap C| = 39$ (purchased both vegetables and bakery items)

$|A \cap C| = 28$ (purchased both fruits and bakery items)

$|A \cap B \cap C| = 18$ (purchased all three categories)

Solution: We have

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

$\Leftrightarrow\ |A \cup B \cup C| = 118 + 99 + 88 - 49 - 28 - 39 + 18$

$\Leftrightarrow |A \cup B \cup C| = 207$

=> $|D| = |U| - |A \cup B \cup C|$

=> $|D| = 359 - 207 = 152$

P2 Determine the cardinality of a given bag (multiset).
Part II: Keep in mind that a < b represents the largest digits in your ID
1. List the bag of prime factors for each of the provided numbers.

a, $\overline{1a2}$

Assuming a = 6 then, we have:

162 = 2 * 9 * 9

= $2 * 9^2$

=> Bag of prime factors is {|2 : 1, 9 : 2|}

b, $\overline{2b0}$

Assuming b = 7 we will have:

270 = 2 * 3 * 3* 3 * 5

= 2 * 33 * 5

=> Bag of prime factors is {|2 : 1, 3 : 3, 5 : 1|}.

2. Find the cardinalities of

a) each of the aforementioned bags.
b) the intersection of the aforementioned bags.
c) the union of the aforementioned bags.
d) the difference of the aforementioned bags.

- In the two bags mentioned above, we have the following sets: A = {|2 : 1, 9 : 2|} and

B = {|2 : 1, 3 : 3, 5 : 1|}

a) each of the aforementioned bags.

|A| = 1 + 2 = 3

|B| = 1 + 3 + 1 = 5

b) The intersection of the aforementioned bags.

- We have A ∩ B = {|2 : 1|}

=> | A ∩ B | = 1

c) The union of the aforementioned bags.

- We have A ∪ B = {|2 : 1, 3 : 3, 5 : 1, 9 : 2|}

=> | A ∪ B | = 1 + 3 + 1 + 2 = 7

d) The difference of the aforementioned bags.

- We have A – B = {|2 : 0, 3 : 0, 5 : 0, 9 : 2|}

=> | A – B | = 2

- Besides, we have B – A = {|2 : 0, 3 : 3, 5 : 1, 9 : 0|}

=> | B – A | = 0 + 3 + 1 + 0 = 4

M1 Determine the inverse of a function using appropriate mathematical techniques.
Part III: Bear in mind that represents the largest digits in your ID.
1. Ascertain whether the given functions are invertible. If they are, identify the rule for the inverse function f^{-1}.

a, f: ℝ → ℝ with f (x) = bx + a.

Set a = 6, b = 7, we have: f(x) = 7x + 6

+ Task 1: f(x) = 7x + 6 is one – to – one function We have: 7x1 + 6 = 7x2 + 6

  => 7x1 = 7x2

  => x1 = x2

  => f(x) = 7x + 6 is one-to-one function. (1)

+ Task 2: f(x) = 7x + 6 is onto function

  We have: x = $\frac{y-6}{7}$

  => f (x) = f ($\frac{y-6}{7}$) = 7($\frac{y-6}{7}$) + 6

  => f (x) = f ($\frac{y-6}{7}$) = 7($\frac{y-6}{7}$) + 6

  => f (x) = y

  => f(x) = 7x + 6 is onto function. (2)

  Includes (1) and (2) => f(x) = 7x + 6 is invertible.

  We have: y = 7x + 6 <=> x = $\frac{y-6}{7}$

=> $f^{-1} = \frac{y-6}{7}$

b, f :[- b, +∞) → [0, +∞) with $f(x) = \sqrt{x+b}$.

Set b = 7:

f(x) = $\sqrt{x+7}$

We have: y = $\sqrt{x+7}$<=> y^2 = x + 7

=> x = y^2 – 7

=> f^{-1}.= y^2 – 7

2. Let f, g: ℝ → 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 ∘ f.

(g ∘ f) (x) = g(f(x)) = $\begin{cases} b(2x + a) - a, x < 0 \\ b(x^3 + b) - a, x \geq 0 \end{cases} \Rightarrow \begin{cases} 2bx + ab - a, x < 0 \\ bx^3 + b^2 - a, x \geq 0 \end{cases}$

Set a = 6, b = 7

$\Rightarrow \begin{cases} 14x + 42 - 6, x < 0 \\ 7x^3 + 49 - 6, x \geq 0 \end{cases}$

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

Part IV: Show that if A, B and C are sets, then $\overline{A \cup B \cup C} = \bar{A} \cap \bar{B} \cap \bar{C}$.

1. Demonstrate that each side is a subset of the other side.

Task 1: prove that: $\overline{A \cup B \cup C} \subseteq \bar{A} \cap \bar{B} \cap \bar{C}$.

Let x ∈ $\overline{A \cup B \cup C}$

By De Morgan' Law, we have: $\overline{A \cup B \cup C} = \bar{A} \cap \bar{B} \cap \bar{C}$.

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

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

Task 2: prove that $\bar{A} \cap \bar{B} \cap \bar{C} \subseteq \overline{A \cup B \cup C}$

Let y ∈ $\bar{A} \cap \bar{B} \cap \bar{C}$

By De Morgan' Law, we have: $\overline{A \cup B \cup C} = \bar{A} \cap \bar{B} \cap \bar{C}$.

Initially, we had y ∈ $\bar{A} \cap \bar{B} \cap \bar{C}$.

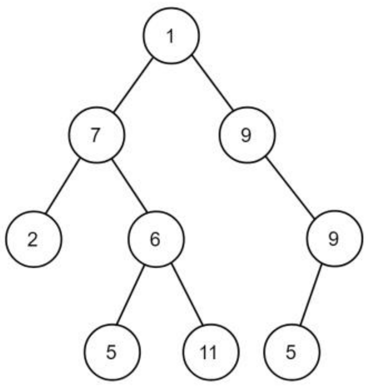
<=> y ∈ $\overline{A \cup B \cup C}$

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

2. Verify the equality using a membership table.

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

LO2 Analyse mathematical structures of objects using graph theory
Activity 2
P3 Model contextualised problems using trees, both quantitatively and qualitatively.
Part I: Discuss two notable instances of binary trees, providing both quantitative and qualitative analyses.
1. What is binary tree?



Binary tree

A binary tree is a fundamental data structure in computer science. It is a hierarchical structure that resembles a tree, where each node can have at most two children: a left child and a right child. Here's a breakdown of its key aspects:

• Structure:

+ Each node in the tree stores data.

+ A special node called the root sits at the top and has no parent.
+ Each child node is connected to its parent node by a single link.
+ No node can have more than two children.

• Properties:

+ Ordered: The order of children matters. The left child is always "lesser" than the right child in some way, depending on the data and the specific implementation.
+ Recursive: A binary tree can be defined recursively, meaning a tree is either empty or consists of a root node with two subtrees, which are themselves binary trees.

• Applications:

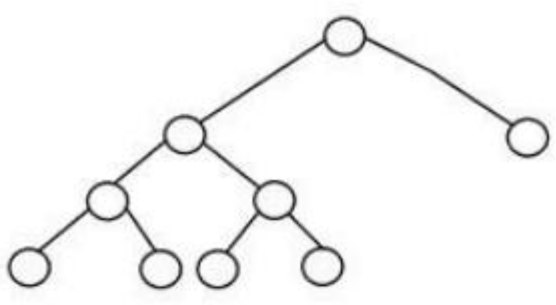
- Binary trees are used in various situations, including:
+ Binary search trees: Efficiently searching for data by exploiting the ordered nature of the tree.
+ Heaps: Priority queues where the element with the highest (or lowest) priority is accessed efficiently.
+ Expression trees: Representing mathematical expressions for evaluation.
+ Trie data structures: Implementing efficient operations like prefix searches used in spell checkers and autocompletion.Types of binary tree

2. Some common types of binary trees

2.1 Full Binary Tree

It is a special kind of a binary tree that has either zero children or two children. It means that all the nodes in that binary tree should either have two child nodes of its parent node or the parent node is itself the leaf node or the external node.

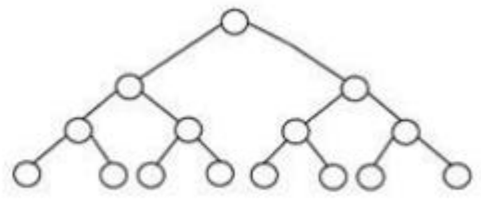
In other words, a full binary tree is a unique binary tree where every node except the external node has two children. When it holds a single child, such a binary tree will not be a full binary tree. Here, the quantity of leaf nodes is equal to the number of internal nodes plus one. The equation is like L=I+1, where L is the number of leaf nodes, and I is the number of internal nodes.

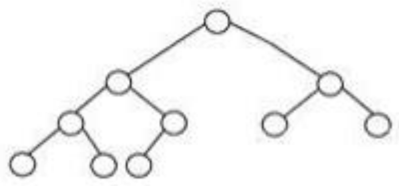


The structure of a full binary tree

2.2 Complete Binary Tree

A complete binary tree is another specific type of binary tree where all the tree levels are filled entirely with nodes, except the lowest level of the tree. Also, in the last or the lowest level of this binary tree, every node should possibly reside on the left side. Here is the structure of a complete binary tree:

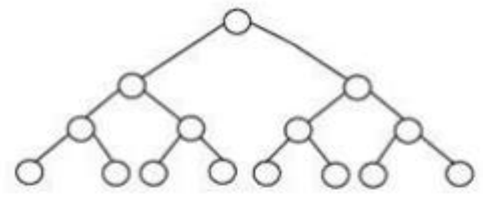


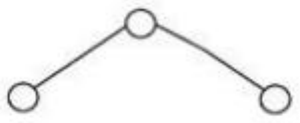


The structure of a complete binary tree

2.3 Perfect Binary Tree

A binary tree is said to be 'perfect' if all the internal nodes have strictly two children, and every external or leaf node is at the same level or same depth within a tree.A perfect binary tree having height 'h' has 2h – 1 node. Here is the structure of a perfect binary tree:

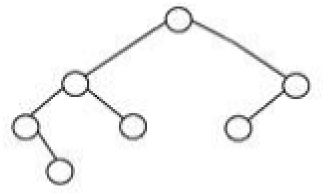




The structure of a perfect binary tree

2.4 Balanced Binary Tree

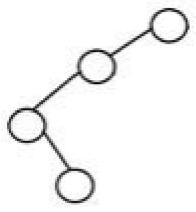
A binary tree is said to be 'balanced' if the tree height is O(logN), where 'N' is the number of nodes. In a balanced binary tree, the height of the left and the right subtrees of each node should vary by at most one. An AVL Tree and a Red-Black Tree are some common examples of data structure that can generate a balanced binary search tree. Here is an example of a balanced binary tree:



Balanced binary tree

2.5 Degenerate Binary Tree

A binary tree is said to be a degenerate binary tree or pathological binary tree if every internal node has only a single child. Such trees are similar to a linked list performance-wise. Here is an example of a degenerate binary tree:



Degenerate binary tree

2.6 Special Types of Binary Trees

Binary trees can also be grouped according to node values. The types of binary tree according to node structure include the following:

• Binary Search Tree

A binary search tree comes with the following properties:

+ In the left subtree of any node, you will find nodes with keys smaller than the node's key.
+ The right subtree of any node will include nodes with keys larger than the node's key.
+ The left, as well as the right subtree, will be types of binary search tree.

• AVL Tree

An AVL binary tree in DSA is self-balanced. In such a tree, the difference between the heights of the left and right subtrees for all nodes cannot be greater than one. So, the nodes in the right as well as left subtrees of the AVL tree will be one or less than that.

• Red Black Tree

If you want us to explain the binary tree and its types, the red-black tree will definitely find a mention. This kind of binary tree is self-balancing, with each node having an extra bit. The extra bit gets represented as either black or red.

The colors in a red black binary tree in data structure are useful for keeping the whole tree balanced during deletions and insertions. The balance of a red black tree won't be perfect. But these binary trees are perfect for bringing down the search time.

• B- Tree

A B- Tree is a type of self-balanced search tree in data structures. These binary trees support smooth access, deletion, and insertion of data items. B- trees are particularly common in file systems and databases.

Among the different types of binary tree, a B- tree helps with efficient storage and retrieval of large volumes of data. A fixed maximum degree or order is a key characteristic of a B- tree. This fixed value helps determine the total number of child nodes in a parent node.

The nodes present in a B- binary tree can include several keys and child nodes. The keys of a B- binary tree in algorithm design can help in indexing and locating data items.

• B+ Tree

A binary tree in data structure can also be classified as B+, which is one variant of the B- tree. Since a B+ tree comes with a fixed maximum degree, it enables efficient insertion, access, and deletion of data items. But a B+ binary tree includes all data items inside the leaf nodes.

The internal nodes of a B+ binary tree only include keys for locating and indexing data items. Due to this design, searches using a B+ tree will be a lot faster, and you will also be able to access data items sequentially. Moreover, the leaf nodes of a binary tree remain together in a linked list.

• Segment Tree

If you look into a binary tree and its types, you will come across one category called the segment or statistic tree. This type of binary tree is usually responsible for storing information related to different segments or intervals. With a segment tree, you will be able to perform querying of the stored segments in a specific point.

Among the different types of binary tree in data structure, you will realize that a segment tree is static. Therefore, you won't be able to modify the structure of a segment tree after it has been built.

P4 Use Dijkstra's algorithm to find a shortest path spanning tree in graph.

Part II: Stick in mind that a < b represents the largest digits in your ID

1. State Dijkstra's Algorithm in an undirected graph

1.1 What is Dijkstra's Algorithm?

- Dijkstra's algorithm is a popular algorithms for solving many single-source shortest path problems having non-negative edge weight in the graphs i.e., it is to find the shortest distance between two vertices on a graph.

- Dijkstra's algorithm can work on both directed graphs and undirected graphs as this algorithm is designed to work on any type of graph as long as it meets the requirements of having non-negative edge weights and being connected.

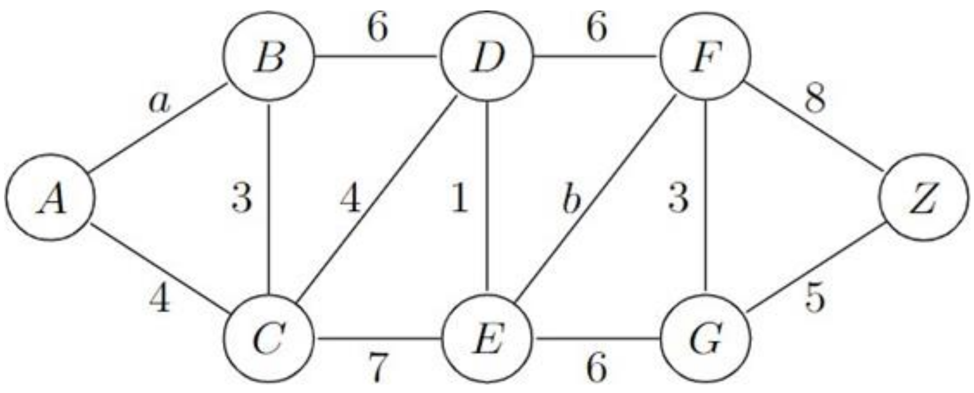
+ In a directed graph, each edge has a direction, indicating the direction of travel between the vertices connected by the edge. In this case, the algorithm follows the direction of the edges when searching for the shortest path.
+ In an undirected graph, the edges have no direction, and the algorithm can traverse both forward and backward along the edges when searching for the shortest path.

1.2 How to implement Dijkstra’s algorithm in an undirected graph?

- To implement Dijkstra's algorithm, we will follow these steps:

+ Mark the source node with a current distance of 0 and the rest with infinity.
+ Set the non-visited node with the smallest current distance as the current node.
+ For each neighbor, N of the current node adds the current distance of the adjacent node with the weight of the edge connecting 0->1. If it is smaller than the current distance of Node, set it as the new current distance of N.
+ Mark the current node 1 as visited.
+ Go to step 2 if there are any nodes are unvisited.

2.Apply Dijkstra’s algorithm to determine the shortest path length between vertices and in the provided weighted graph.

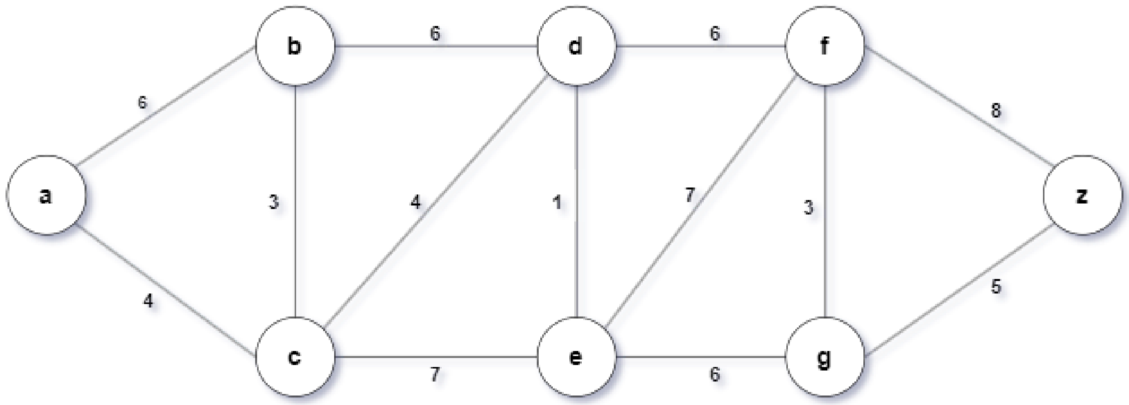


- Let U be the starting point and V be the points adjacent to U.

- Because length of AB = AC we have two cases: Start from A to B or Start from A to C

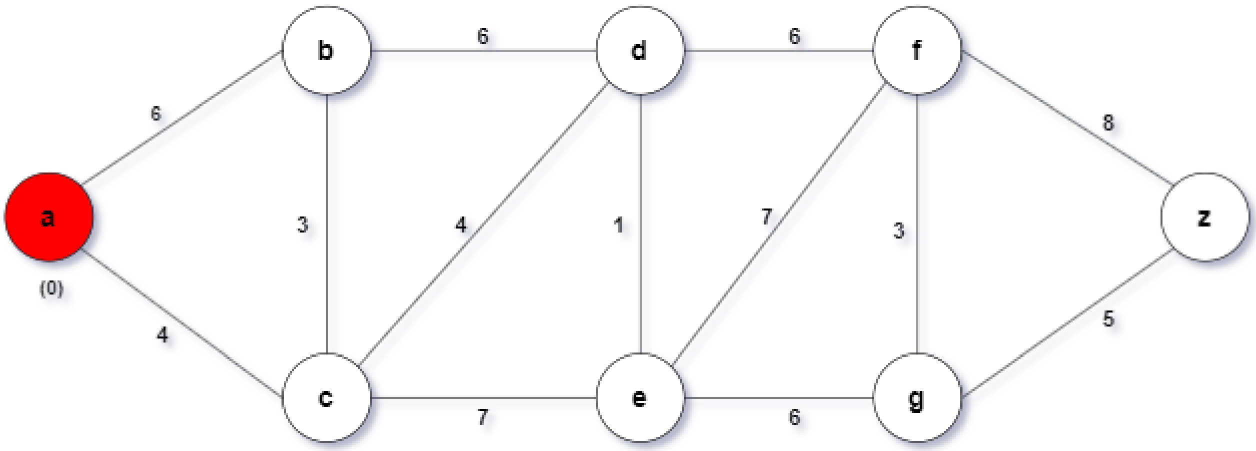
- Applying Dijsktra’ Algorithm with formula: deg(V) = Min (Deg(U) + Length of U & V), we have the following table:

- Set a = 6, b = 7, we have:

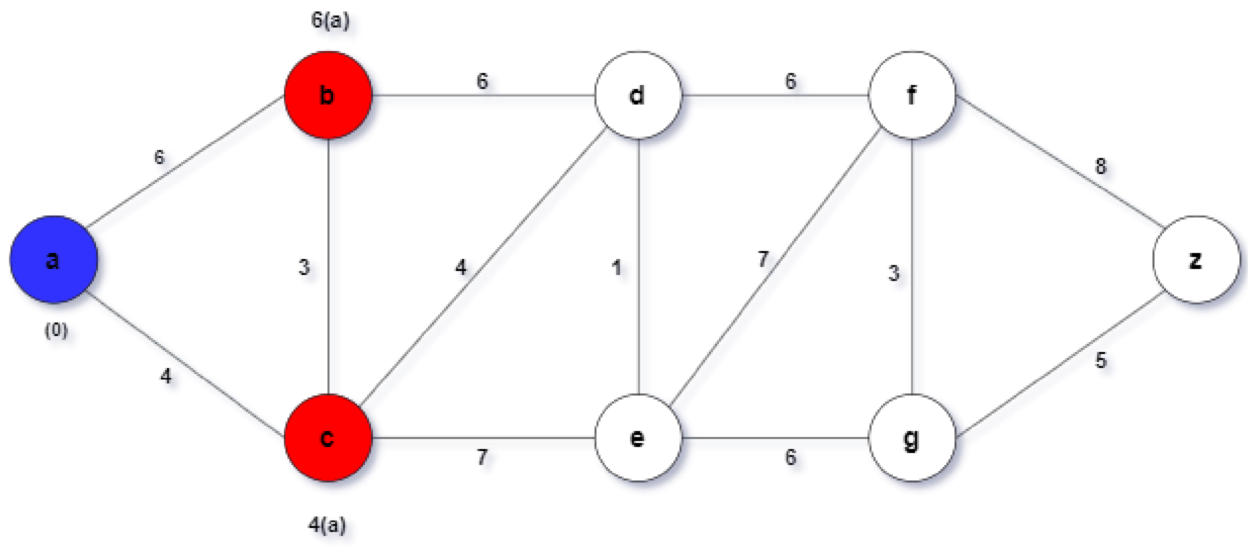


Solution:

➢ First, the labels are now initialized so that the label of a is 0 and all other labels are ∞.

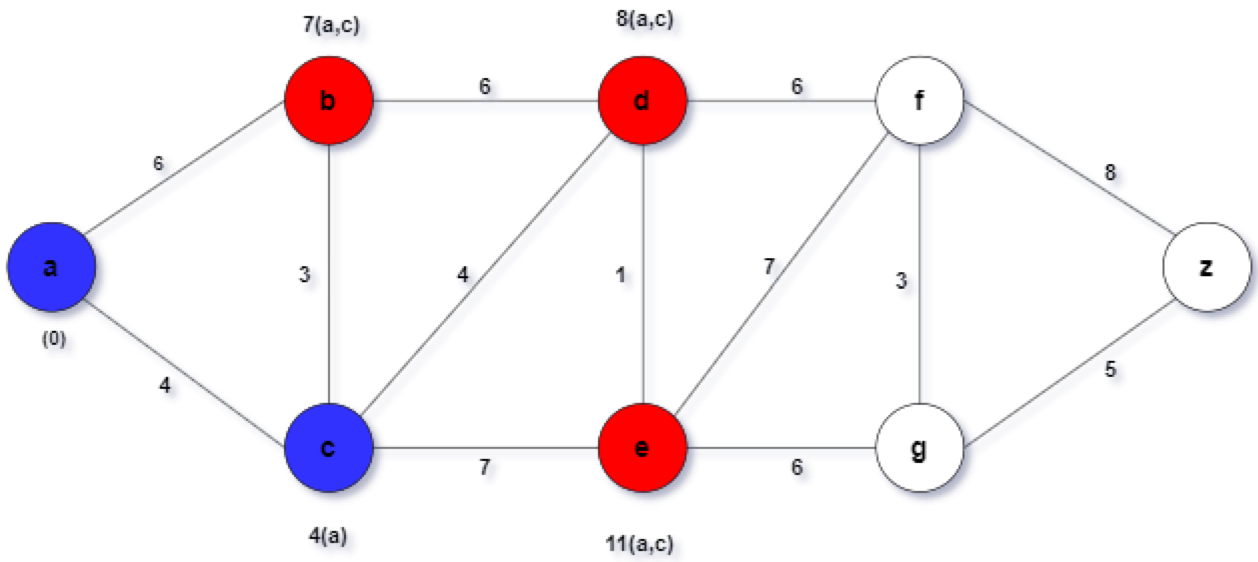


➢ Next, we start at A and there are two paths including A, B of length 6 and A, C of length 4.

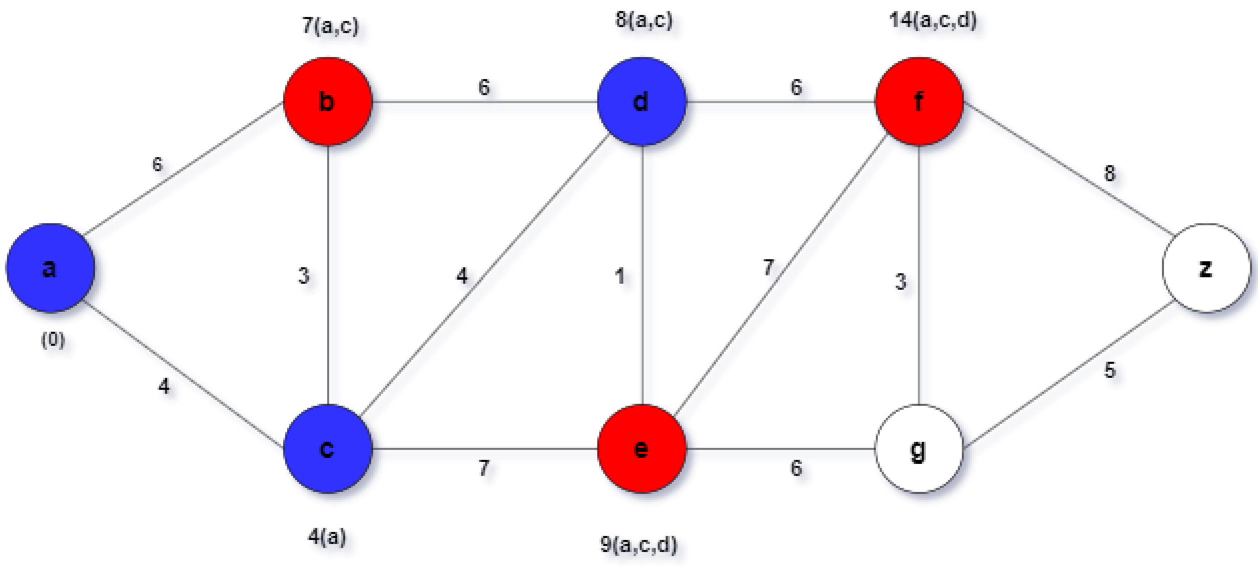


It follows that C is the first closest vertex to A, and the shortest path from A to C has length 4.

- Now, we need to find the second closest vertex to A by examining all paths that begin with the shortest path from the vertex C.

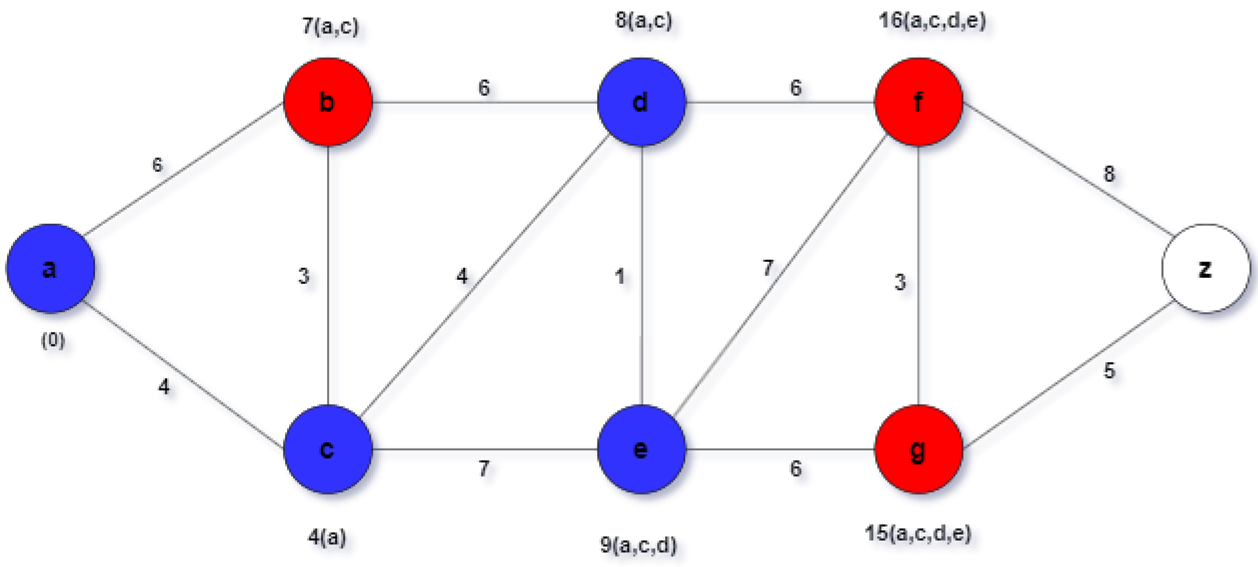


- To determine the third closest vertex to A, we need to checking all paths that begin with shortest path from the vertex D.

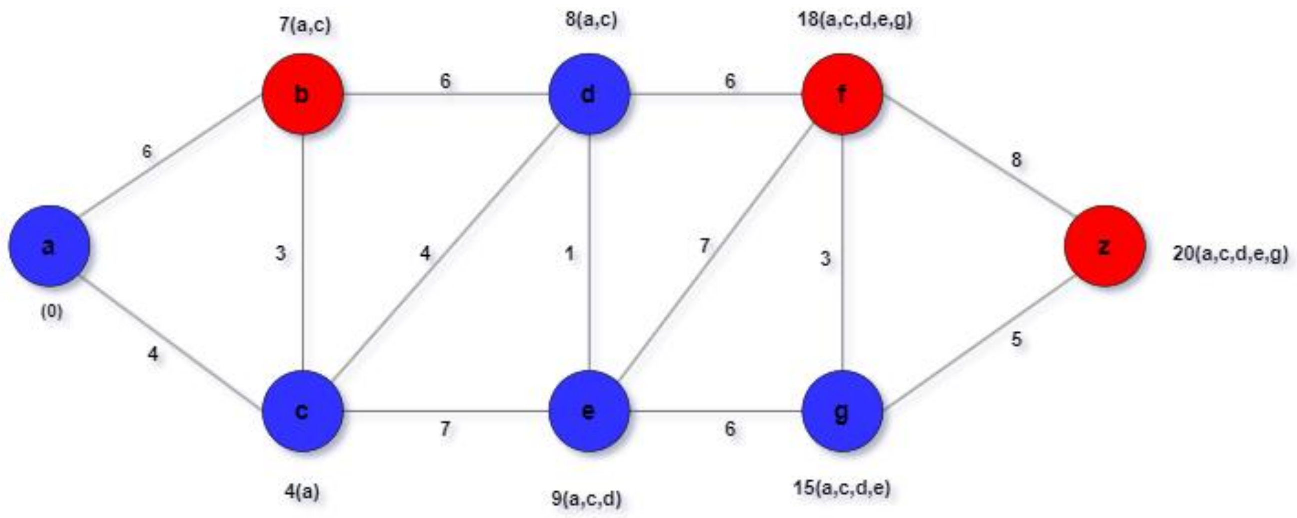


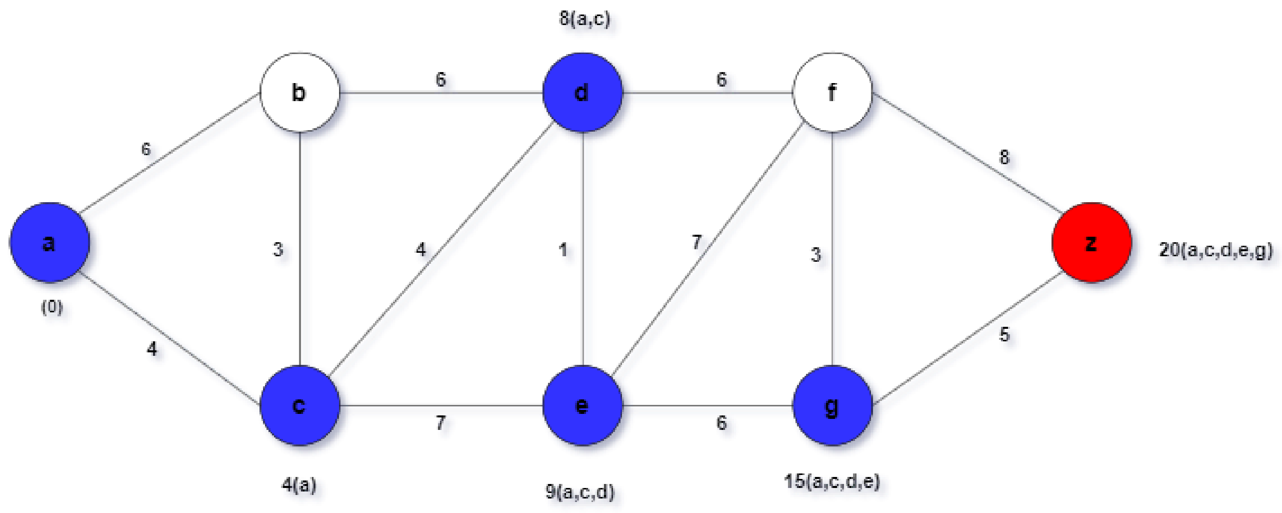
In this case, the third closest vertex to A is E, and the shortest path from A to E is A, C, D, E of length 9.

- To determine the fourth closest vertex to A, we need to checking all paths that begin with shortest path from the vertex E.



- Finally, to reach Z there is a unique path from G that is A, C, D, E, G, Z of length 20. It means that the fifth closest vertex to A is Z.

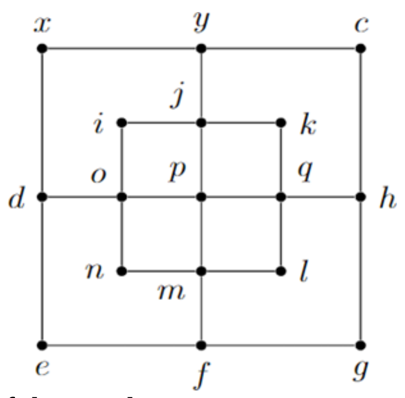




- We conclude that the shortest path between A and Z is A, C, D, E, G, Z of length 20.

M2 Assess whether a Eulerian and Hamiltonian circuit exists in an undirected graph.

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.



- Suppose a is the Hamiltonian path of the graph.

- We have the number of degrees of each vertex in the graph as follows:

+ Deg(x) = Deg(c) = Deg(g) = Deg(e) = 2.
+ Deg(y) = Deg(h) = Deg(f) = Deg(d) = 3.
+ Deg(i) = Deg(k) = Deg(l) = Deg(n) = 2.
+ Deg(j) = Deg(q) = Deg(m) = Deg(o) = Deg(p) = 4.
+ as a consequence of Dirac's theorem, if degree to deg(v) ≥ = $\frac{n-1}{2}$ every vertex of the graph, then A has a Hamiltonian path. So, in the graph above, all the steps are smaller $\frac{n-1}{2}$ = 8. So the above sufficient condition deduces that the above graph does not exist a Hamilton path. (1)
+ Next, we need to have a more certain conclusion that the above graph does not exist a Hamilton path.
+ The graph above has vertices with degree 2 including: x, c, e, g, i, k, l, n.
+ For example, we will give the starting point any vertex of x, c, g, e on the outside side and ending at any vertex of order 2 on the inside of the graph. This will survive at least one isolated peak. From there, the graph would not be able to survive the Hamilton path.
+ Next, we will give the starting point any vertex of i, k, l, n. If this is the case, 2 cases will occur: there will exist an isolated vertex or it will not be able to have a path to other vertices in the graph. Therefore, the above graph will not have a Hamiltonian cycle.

- To prove this, we will assume the starting point of the Hamiltonian path in the graph above is x.

=> The edges and vertices connecting them will belong to the Hamilton path: xy, yc, ch, gh, gf, ed, ef, io, ij, kj, kq, lm, lq, nm, no.

=> The Hamilton road in will be constructed as follows: x → y → c → h → g →f → e → d → o → i → j → k → q → l → m → n. Thus, peak p becomes an isolated peak and cannot complete the Hamilton path. So this graph cannot survive the Hamiltonian path. Analogues for x like vertices are c, g, e. (2)

- Next, with the internal starting points being i. We will have the Hamilton road built as follows: i → j → k → q → l → m → n → o. At o, if we go to p, we will have to remove the edge od because according to the Hamiltonian path rules, from which it will be impossible to continue to external points such as x, y, c, h, g, f, e, d. Conversely, if we follow d, we will have to remove the edge op according to the Hamiltonian path rule, and p becomes an isolated vertex. Therefore, the graph would not exist a Hamiltonian path. Analogues for i like vertices are k, l,n. (3)

- Includes (1), (2) and (3) => In the graph above, no Hamiltonian path exists.

- With an Euler path, according to theorem 1, a connected graph has an Euler cycle if and only if every vertex of the graph has smooth degree. The graph above has up to 4 vertices with odd degrees.

- With an Euler path, according to theorem 1, a connected graph has an Euler cycle if and only if every vertex of the graph has smooth degree. The graph above has up to 4 vertices with odd degrees. In addition, according to theorem 2, a connected graph has an Euler path and no Euler cycle when the graph has exactly 2 unique vertices of odd degree.

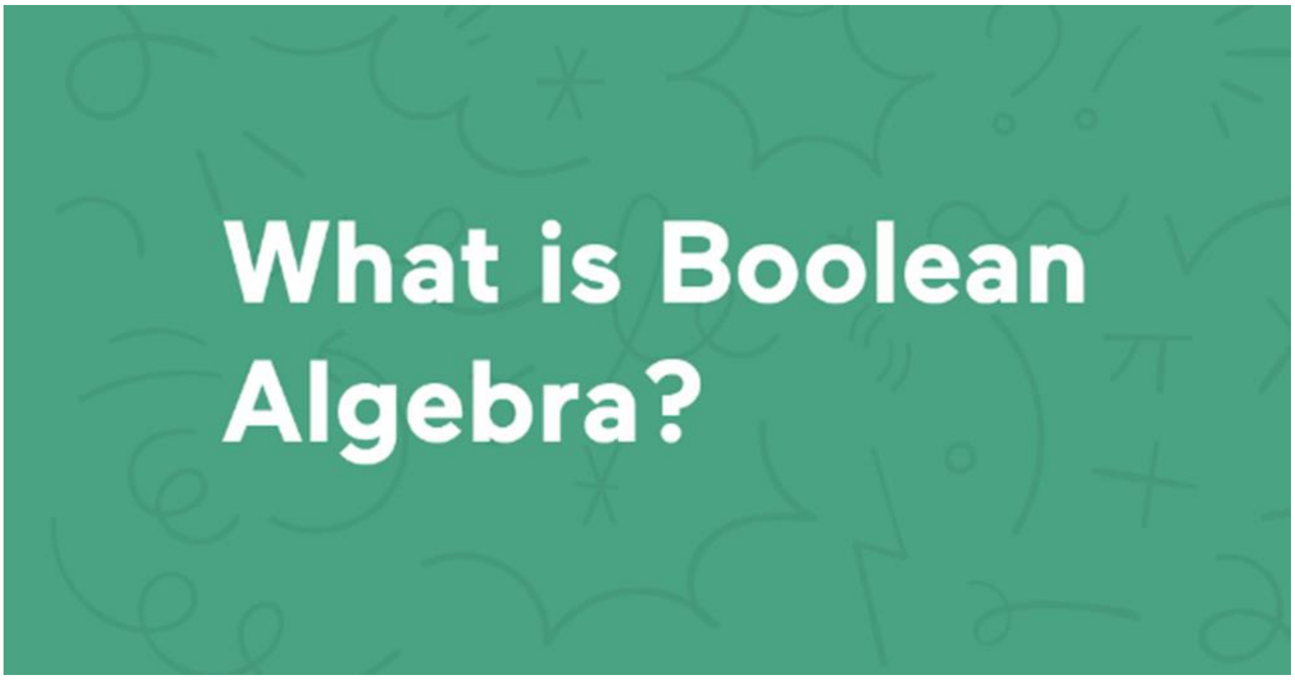
=> The graph above will not have an Euler path because the graph above has 4 vertices with odd degrees.

P5 Diagram a binary problem in the application of Boolean algebra.

Activity 1

Part I. Discuss the utilization of Boolean Algebra in addressing binary problems encountered in two diverse real-world domains and the practical applications therein.

1. What is Boolean Algebra?



- Define: Boolean algebra is a formal system of mathematics that is used to analyze and simplify digital logic circuits. The Boolean algebraic system was developed by 19th-century British mathematician George Boole. Boolean algebra is the study of truth values, it is closely related to logic. Many of the concepts in boolean algebra can be seen as special cases of more general logical principles.

The distinguishing factor of Boolean algebra is that it deals only with the study of binary variables. Most commonly Boolean variables are presented with the possible values of 1 ("True") or 0 ("False"). Variables can also have more complex interpretations, such as in set theory. Boolean algebra is also known as binary algebra

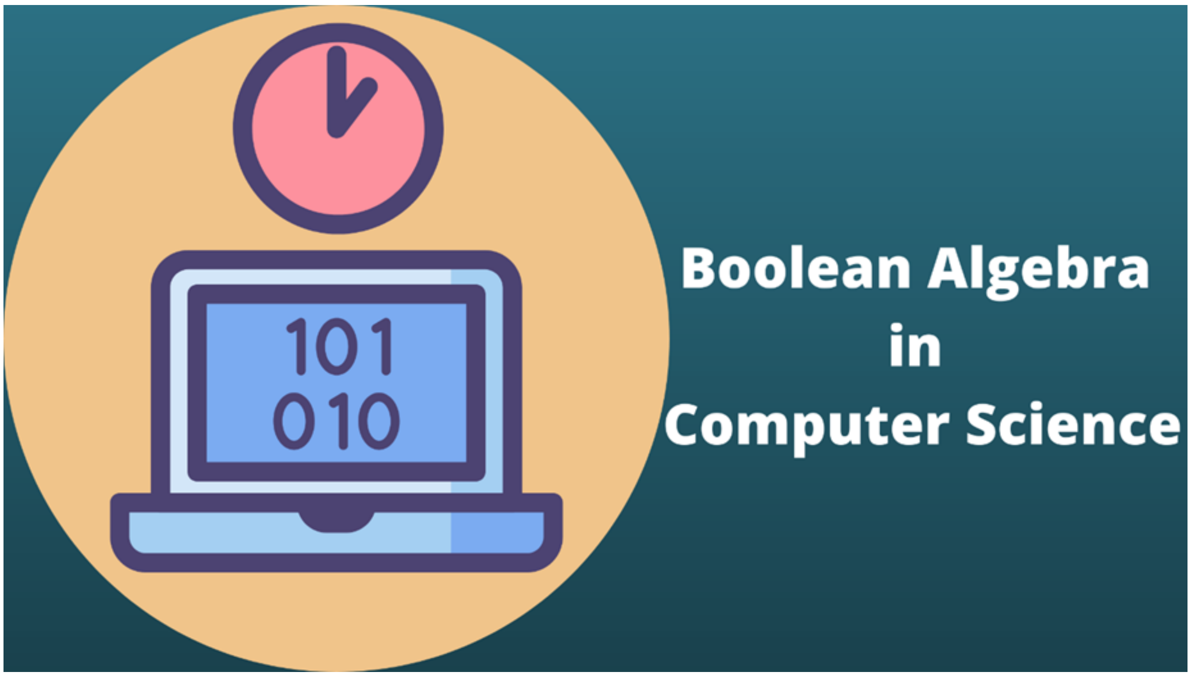
Boolean algebra is also used in set theory, where it provides a way to describe relationships between sets of objects. In general, boolean algebra is a powerful tool for modelling and manipulating data. Boolean algebra can be used to simplify logic circuits by reducing the number of gates required.

2. Applications of Boolean Algebra in real-world

2.1 Boolean Algebra in Finance

- Boolean algebra has applications in finance through mathematical modeling of market activities. For example, research into the pricing of stock options can be aided by the use of a binary tree to represent the range of possible outcomes in the underlying security. In this binomial options pricing model, where there are only two possible outcomes, the Boolean variable represents an increase or a decrease in the price of the security. This type of modeling is necessary because, in American options, which can be exercised anytime, the path of a security's price is just as important as its final price The binomial options pricing model requires the path of a security's price to be broken into a series of discrete time ranges.

2.2 Boolean Algebra in Computer Science



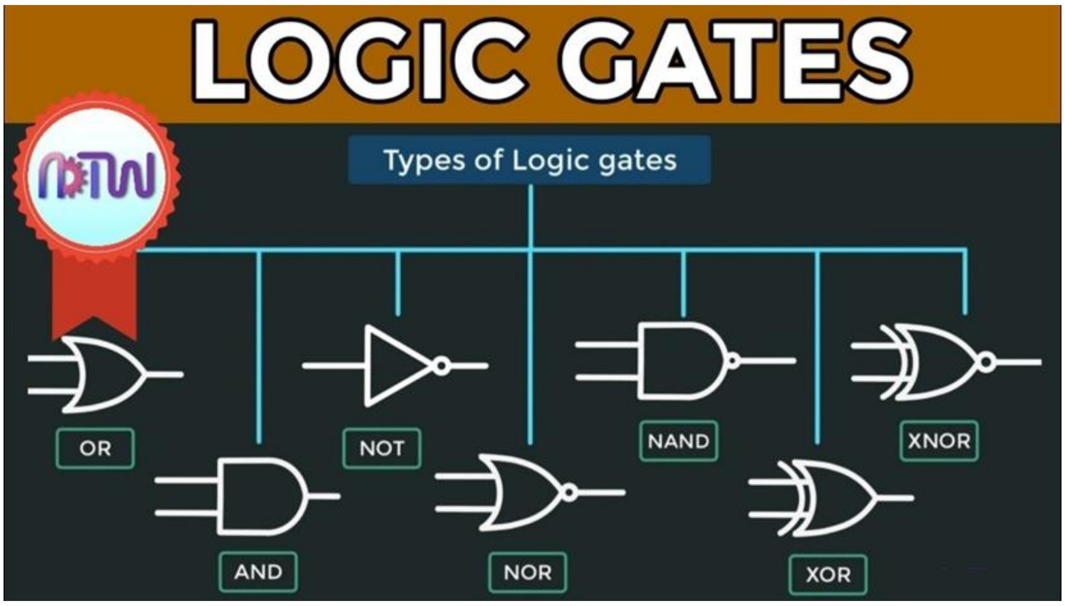
- Logic Gates and Circuits: Boolean algebra finds direct application in the design and analysis of logic gates and digital circuits. Logic gates, such as AND, OR, and NOT gates, implement Boolean functions, where inputs and outputs are binary values. Complex circuits and processors are built by combining these fundamental gates, enabling the creation of sophisticated computational systems.

- Digital Signal Processing: Boolean algebra is applied in digital signal processing (DSP) to manipulate discrete signals efficiently. Boolean operations help define algorithms for filtering, transforming, and analyzing digital signals, contributing to applications like image processing, audio compression, and telecommunications.

- Cryptography and Information Security: Boolean algebra finds applications in cryptographic algorithms and information security. Boolean operations are utilized in designing secure hash functions, encryption schemes, and authentication protocols, contributing to the protection of digital information.

3. Logic Gates

3.1 What is Logic Gates?



- Logic gate is an electronic circuit designed by using electronic components like diodes, transistors, resistors, and more. As the name implies, a logic gate is designed to perform logical operations in digital systems like computers, communication systems, etc. Therefore, we can say that the building blocks of a digital circuit are logic gates, which execute numerous logical operations that are required by any digital circuit. A logic gate can take two or more inputs but only produce one output. The output of a logic gate depends on the combination of inputs and the logical operation that the logic gate performs.

- Logic gates use Boolean algebra to execute logical processes. Logic gates are found in nearly every digital gadget we use on a regular basis. Logic gates are used in the architecture of our telephones, laptops, tablets, and memory devices.

3.2 Application of Logic Gates in real-world

- The application of logic gate is numerous, but they are primarily defined by the mode of operations or the truth table of the logic gate. Safe thermostats, the push-button locks, the automatic water system and a number of other electronic gadgets all use basic logic gates.

- One of the key benefits of sophisticated methods is that the basic logic gate may be employed in a variety of configurations. Furthermore, there is no restriction on the number of gates that may be employed in a single device. It may, however, be decreased owing to physical constraints on the device. Logic gate arrays can be found in digital integrated circuits.

- In computers, logic gates transform 1s and 0s from input wires. The numerals zero and one are not actual numbers. They represent layers of logic. Furthermore, they are essential components of every digital computer. The inputs are turned into logic gates based on their states.

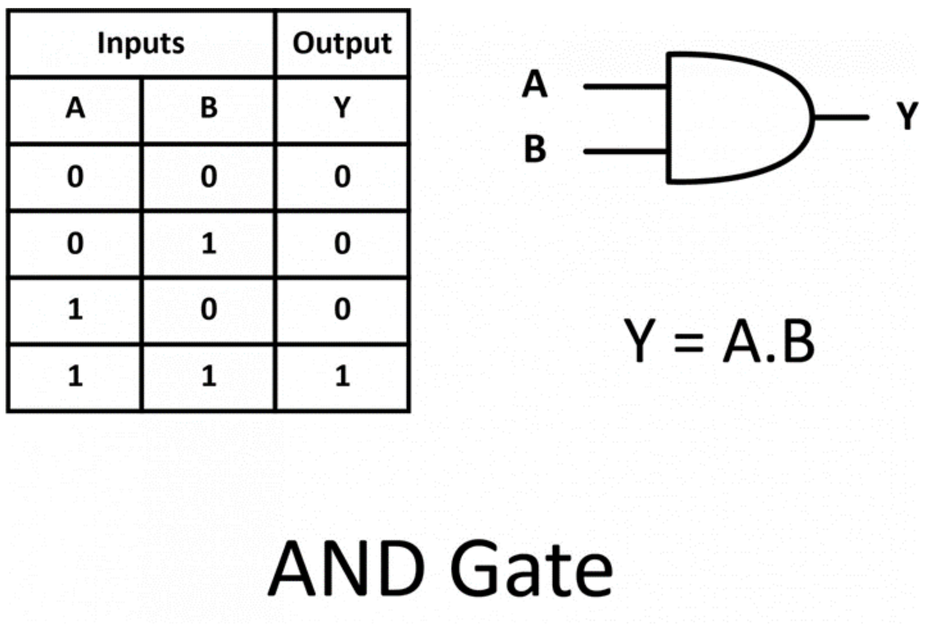
- A logic gate is an electrical device whose input and output voltages have a logical connection. There are three fundamental gates: OR, AND, and NOT. Logic gates are made using semiconductor devices. Each fundamental logic gate has its own symbol and truth table. The truth table contains a list of all possible inputs and outputs.

3.3 Types of Logic Gates

Logic gate is a digital gate that allows data to be manipulated. Logic gates, use logic to determine whether or not to pass a signal.

Logic gates, on the other hand, govern the flow of information based on a set of rules. The logic gates can be classified into the following major types:

3.3.1 AND Gate



Inputs		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

In digital electronics, the AND gate is one of the basic logic gate that performs the logical multiplication of inputs applied to it. It generates a high or logic 1 output, only when all the inputs applied to it are high or logic 1. Otherwise, the output of the AND gate is low or logic 0.

The AND gate has the following two main properties:

- The AND gate can receive two or more input values at once.
- When all of the inputs are logic 1, this gate produces logic 1.

For two-input AND gate, the Boolean expression is given by:

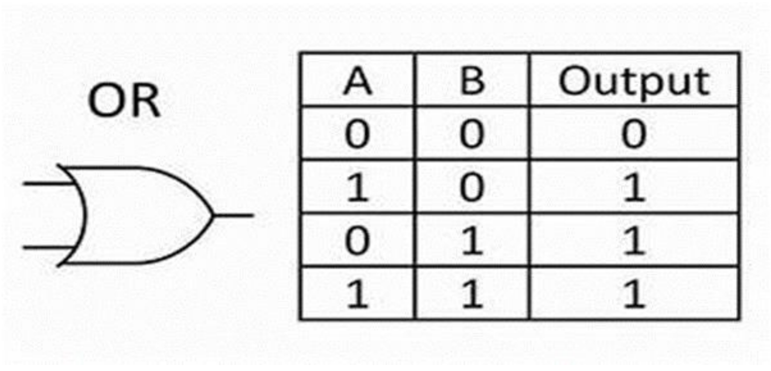
Z = A.B

Where, A and B are inputs to the AND gate, while Z denotes the output of the AND gate. - We can extend this expression to any number of input variables, such as:

Z = A.B.C.D.....

3.3.2 OR Gate

- The OR gate is a digital logic gate that implements logical disjunction. The OR gate outputs "true" if any of its inputs are "true" otherwise it outputs "false". The input and output states are normally represented by different voltage levels.



A	B	Output
0	0	0
1	0	1
0	1	1
1	1	1

In digital electronics, there is a form of simple logic gate that gives a low or logic 0 output when all inputs are low or logic 0. For all other input combinations, the OR gate's output is high, or logic 1. This logic gate is known as an OR gate. An OR gate can have several inputs but only one output.

The OR gate is mostly used for logical sum operations and has two properties:

- It supports multiple input lines.
- When all of the OR gate's inputs are low or logic 0, its output is also low or logic zero.

The operation of an OR gate can be mathematically described through a mathematical expression called Boolean expression of the OR gate.

- The boolean expression for a two input OR gate is given by:

Z = A + B

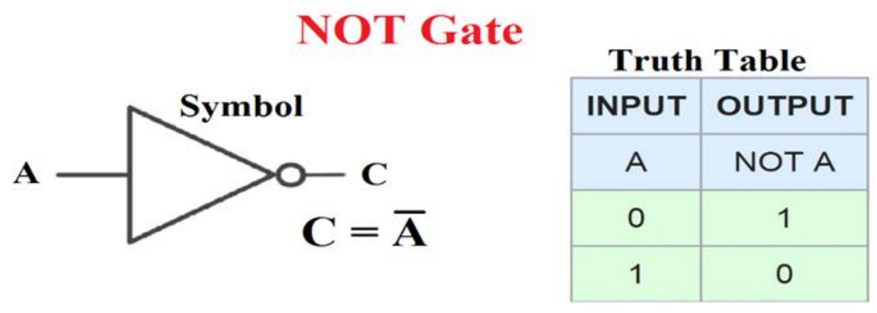
- The boolean expression for a three-input OR gate is:

Z = A + B + C

Here, A, B, and C are inputs and Z is the output variables. We can extend this boolean expression to any number of input variables.

3.3.3 NOT GATE

In digital electronics, the NOT gate is another basic logic gate used to perform the addition of the input signal applied to it. It requires only one input and one output. The output of the NOT gate is the complement of the input applied to it. Therefore, if we apply a low or logic 0 output to the NOT gate, it will give a high or logic 1 output and vice versa. The NOT gate is also known as the inverting gate because it performs the inversion operation.



INPUT	OUTPUT
A	NOT A
0	1
1	0

Properties of NOT Gate:

- The output of NOT gate is the complement or inverse of the input applied to it.
- NOT gate takes only one output.

The logic operation of NOT gate is described by its boolean expression given below:

Z = $\boldsymbol{\overline{A}}$

The bar over the input variable A represents the inversion operation

P6 Produce a truth table and its corresponding Boolean equation from an applicable scenario.

Part II.

1. Develop the truth table and derive the corresponding Boolean equation for each of the following scenarios

a) "In a secure facility, if the access card is swiped OR the correct PIN is entered AND the security system is NOT in maintenance mode, then the door should unlock."

Let:

- A represent "Access card is swiped."
- P represent "Correct PIN is entered."
- M represent "Security system is in maintenance mode."
- D represent "Door unlocks."

The statement "In a secure facility, if the access card is swiped OR the correct PIN is entered AND the security system is NOT in maintenance mode, then the door should unlock" can be expressed as:

(A OR P) AND (NOT M) → D.

The truth table for this scenario would be:

A	P	M	NOT M	A OR P	(A OR P) AND (NOT M)	D
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	1	0	1	1	1	1
0	1	1	0	1	0	0
1	0	0	1	1	1	1
1	0	1	0	1	0	0
1	1	0	1	1	1	1
1	1	1	0	1	0	0

➢ Row 1 (A = 0, P = 0, M = 0):

- A = 0, P = 0, M = 0: he access card, PIN, and maintenance mode are all inactive.
- A or P = 0 because both variables are 0.
- NOT M = 1 because M = 0
- (A OR P) AND (NOT M) = 0 x 1 = 0

=> Therefore, D = 0: The door is locked

➢ Row 2 (A = 0, P = 0, M = 1):

- A = 0, P = 0, M = 1.
- A OR P = 0 because both variables are 0.
- NOT M = 0 because M = 1.
- (A OR P) AND NOT M = 0 × 0 = 0.

=> Because (A OR P) AND NOT M = 0, so D = 0

➢ Row 3 (A = 0, P = 1, M = 0):

- A = 0: The access card is inactive.
- P = 1: The correct PIN is entered.
- M = 0: The system is not in maintenance mode.

- A OR P = 1 because P = 1.
- NOT M = 1 because M = 0.
- (A OR P) AND NOT M = 1 × 1 = 1.

  => Therefore, D = 1: The door unlocks.

➢ Row 4 (A = 0, P = 1, M = 1):

- A = 0, P = 1, M = 1: Similar to row 3, but the system is in maintenance mode.
- (A OR P) AND NOT M = 1 × 0 = 0.

  => Therefore, D = 0: The door remains locked.

➢ Row 5 (A = 1, P = 0, M = 0):

- A = 1: The access card is active.
- P = 0: The PIN is not entered correctly.
- M = 0: The system is not in maintenance mode.
- A OR P = 1 because A = 1.
- NOT M = 1 because M = 0.
- (A OR P) AND NOT M = 1 × 1 = 1.

  => Therefore, D = 1: The door unlocks.

➢ Row 6 (A = 1, P = 0, M = 1):

- A = 1, P = 0, M = 1: Similar to row 5, but the system is in maintenance mode.

  => (A OR P) AND NOT M = 1 × 0 = 0.

  => Therefore, D = 0: The door remains locked.

➢ Row 7 (A = 1, P = 1, M = 0):

- A = 1: The access card is active.
- P = 1: The correct PIN is entered.
- M = 0: The system is not in maintenance mode.
- A OR P = 1 because both A and P are 1.
- NOT M = 1 because M = 0.

- (A OR P) AND NOT M = 1 × 1 = 1.

  => Therefore, D = 1: The door unlocks.

- Row 8 (A = 1, P = 1, M = 1):

- A = 1 , P = 1, M = 1: Similar to row 7, but the system is in maintenance mode.

- (A OR P) AND NOT M = 1 × 0 = 0.

  => Therefore, D = 0: The door remains locked.

- From the truth table, we can derive the Boolean equation for the scenario:

  => $D = A \times \overline{M} + P \times \overline{M}$

  b. "For a computer to successfully log in, either a valid username and password combination must be entered OR a security token must

  be provided, but not both."

- U represents "Valid username and password combination is entered."

- P represents "Security token is provided."

- L represents "Computer successfully logs in."

- The expression provided for the scenario is:

  (U AND NOT P) OR (NOT U AND P) → L

  This expression signifies that the computer successfully logs in if either a valid username and password combination is entered but not a security token, or if a security token is provided but not a valid username and password combination.

- Now, let's interpret the truth table:

U	**P**	**(U AND NOT P) OR (NOT U AND P)**	**L**
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

- When both U and P are 0, implying no valid username-password combination and no security token, the computer does not log in (L = 0).
- When U is 0 and P is 1, meaning no valid username-password combination but a security token is provided, the computer successfully logs in (L = 1).
- When U is 1 and P is 0, indicating a valid username-password combination but no security token, the computer successfully logs in (L = 1).
- When both U and P are 1, implying both a valid username-password combination and a security token are provided, the computer does not log in (L = 0).

From the truth table, we can derive the Boolean equation for L:

$\boldsymbol{L = (U \times \overline{P}) + (\overline{U} \times P)}$

2. Generate a truth table for the provided Boolean expression: $\boldsymbol{xyz + x\overline{y}\overline{z} + \overline{x}y\overline{z} + \overline{x}\overline{y}z}$

- ➢ **The given Boolean expression is:** $\boldsymbol{xyz + x\overline{y}\overline{z} + \overline{x}y\overline{z} + \overline{x}\overline{y}z}$
- ➢ **This expression consists of four terms:**

- $\boldsymbol{xyz}$
- $\boldsymbol{x\overline{y}\overline{z}}$
- $\boldsymbol{\overline{x}y\overline{z}}$
- $\boldsymbol{\overline{x}\overline{y}z}$

Now, let's understand how each term contributes to the overall expression:

- $\boldsymbol{xyz}$**:** This term is true (1) only when all three variables x, y and z are true (1). In the truth table, this term evaluates to 1 only in the last row where x = 1, y = 1 and z = 1.
- $\boldsymbol{x\overline{y}\overline{z}}$**:** This term is true (1) when x is true (1) and both y and z are false (0). It's essentially the negation of y and z while x is true. In the truth table, this term evaluates to 1 only in the sixth row where x = 1, y = 0, and z = 1
- $\boldsymbol{\overline{x}y\overline{z}}$**:** This term is true (1) when y is true (1) and both x and z are false (0). It's essentially the negation of x and z while y is true. In the truth table, this term evaluates to 1 only in the seventh row where x = 0, y = 1 and z = 1
- $\boldsymbol{\overline{x}\overline{y}z}$**:** This term is true (1) when z is true (1) and both x and y are false (0). It's essentially the negation of x and y while z is true. In the ruth table, this term evaluates to 1 in the fourth and last rows where x = 0, y = 1 and z = 1.

Now, let's look at the truth table:

x	y	z	xyz	$x\bar{y}\,\bar{z}$	$\bar{x}y\bar{z}$	$\bar{x}\bar{y}z$	$xyz + x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}z$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1
0	1	0	0	0	1	0	1
0	1	1	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0

In this truth table:

- Each row represents a unique combination of x, y and z
- The columns represent the values of the individual terms of the Boolean expression.
- The last column represents the result of the entire Boolean expression

Part III. Simplify the following Boolean expressions.

1. $x(x + y) + y(y + z) + z(z + x)$

(Applying Distributive Law)

The distributive theorem in Boolean algebra is: $x(y + z) = xy + xz$

= $(xx + xy) + (yy + yz) + (zz + zx)$

Using the absorption theorem: $x + xy = x$

= $x(1 + y) + y(1 + z) + z(1 + x)$

= $x + y + z$

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

Applying the Distributive Law: $(x + \bar{y})(y + z)$

=> $xy + xz + \bar{y}y + \bar{y}z + xz + x\bar{x} + yz + y\bar{x}$

Applying the Zero attribute: $\bar{x}x = 0, \bar{y}y = 0$

=>$xy + xz + 0 + \bar{y}z + xz + 0 + yz + y\bar{x}$

=> $xy + xz + \bar{y}z + xz + yz + y\bar{x}$

Applying the Idempotent Law and Combine the two xz, zx term into one:

=> $xy + xz + yz + \bar{y}z + y\bar{x}$

Applying Distributive laws:

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

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

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

Because $x + 1 = 1$

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

Applying Distributive laws:

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

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

=> $y.1 + z$

=> $y + z$

Sumary: The above Boolean expression is simplified to: $y + z$

3. $(\mathbf{x} + \mathbf{y})(\mathbf{xz} + \bar{\mathbf{x}}\mathbf{z}) + \mathbf{zx} + \mathbf{x}$

Applying the distribution law:

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

= $x\bar{x}z + y\bar{x}z + xxz + yxz + zx + x$

Using properties of Boolean algebra applying these properties: $xxz = xz, x\bar{x} = 0$

= $0 + y\bar{x}z + xz + yxz + zx + x$

= $y\bar{x}z + xz + yxz + zx + x$

Group of similar terms:

= $y\bar{x}z + xz + (1 + y) + zx + x$

Applying Domination laws: $(1 + y) = 1$

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

Applying Idempotent laws: $xz + zx = xz$

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

Applying the distribution law:

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

Applying Domination laws: $(z + 1) = 1$

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

Sumary: $y\bar{x}z + x$

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

Applying the distribution law:

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

Simplify elements using properties Zero property, Idempotent laws:

= $\mathbf{0} + \bar{x}y + x + x\bar{y} + yx + \mathbf{0}$

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

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

Applying Absorption laws:

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

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

= $x + \bar{x}y$

Sumary: $x + \bar{x}y$

Activity 2

P7 Describe the distinguishing characteristics of different binary operations that are performed on the same set.

Part 1

1. Explain the attributes of various binary operations executed within a common set.

Binary Operations:

- A binary operation is defined for any set (S) such that it takes **two elements** from (S) as inputand produces a **single element** in (S) as output.

- These operations are performed on two inputs (known as **operands**).
- Examples of fundamental binary operations **include addition, subtraction, multiplication**,and **division.**

Properties of Binary Operations:

- Binary operations exhibit several important properties:

+ **Closure Property:** The result of a binary operation on elements from a set remains withinthe same set. Formally: $[x , y \in X \Longrightarrow x * y \in X]$
+ **Associativity**: The order of applying a binary operation does not affect the result. Forelements (x), (y), and (z): [(x * y) * z = x * (y * z)]
+ **Commutativity:** The order of operands does not affect the result. For elements (x) and (y):[x * y = y * x]
+ **Identity Element**: There exists an element (e) such that (x * e = e * x = x) for all (x). Thiselement is called the identity element.
+ **Inverse Element**: For each element (x), there exists an element (y) such that (x * y = y * x = e), where (e) is the identity element.

Types of Binary Operations:

- Binary operations can be categorized based on the set they operate on:

+ **Binary Addition:** Defined on a set (A) with elements (x) and (y).
+ **Binary Subtraction:** Similar to addition but with subtraction.
+ **Binary Multiplication:** Involves multiplying two elements.
+ **Binary Division:** Involves dividing one element by another.

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

1. Subtraction on the set of natural numbers N

The subtraction operation is defined as $- : N \times N \rightarrow N$ where (x, y) ↦ x – y.

To check if this qualifies as a binary operation, we need to see if for any x , y ϵ N,

the result x – y is also in N

- Example: Let x = 3 and y = 5. Then x – y = 3 – 5 = - 2, which is not a natural number.

Since subtraction of two natural numbers does not always result in a natural number, this operation is not a binary operation on the set of natunal numbers N.

2. Exponential operation on the set of integers Z

The exponential operation is defined as $\wedge : Z \times Z \rightarrow Z$ where $(x , y) \longmapsto x^y$.

To check if this qualifies as a binary operation, we need to see if for any $x , y \in Z$, the result x^y is also in Z.

- Example: Let $x = 2$ and $y = -1$. Then $x^y = 2^{-1} = \frac{1}{2}$, which is not an integer.
- Another Example: Let x = -2 and y = $\frac{1}{2}$. Then $x^y = (-2)^{\frac{1}{2}}$, which is not defined in the set of integers.

Since the result of raising one integer to the power of another is not always an integer, this operation is not a binary operation on the set of integers Z.

P8 Determine the order of a group and the order of a subgroup in given examples.
Part 2

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

- Solutions:

+ Applying the theorem and properties of groups, we will obtain Cayley tables in the following order.

+ To do this exercise, we need to satisfy the condition that: on all rows and columns, all elementsare displayed and there are no repeating elements on a row or column.

+ Set ∘ = * accordingly to create the table

+ Order 1:

- A group of order 1 has only one element, which must be the identity element. Let's denote it by e.
- In this case, there is only one element in the group, which must be the identity element. Since there's only one element, the operation table will simply have one entry.

*	e
e	e

- Explanation:

- This is the simplest group with only one element, e, which is the identity element.
- The only possible operation is e * e = e, as there are no other elements.

+ Order 2:

- A group of order 2 has two elements, where one is the identity element. Let's denote the elements by e (identity) and a.
- Here, we have one identity element and one other element. Since every element must have an inverse, the non-identity element must be its own inverse. Therefore, the operation table will reflect this property.

*	e	a
e	e	a
a	a	e

- Explanation:

- e is the identity element, so it doesn't change any element: e * e = e , e * a = a, and a * e = a .
- a is the only non-identity element and a * a = e (since a is its own inverse in this group of order 2).

+ Order 3:

- A group of order 3 has three elements, where one is the identity element. Let's denote the elements by e (identity), a, and b.
- In this case, we have one identity element and two other elements. Each element must appearonce in every row and column, satisfying the closure property.

*	e	a	b
e	e	a	b
a	a	b	e
b	b	e	a

- Explanation:

- e is the identity element, so it does not change any element: e * e = e , e * a = a, and e * b = b.
- a * a = b and a * b = e (since a and b are elements of order 3, meaning a^3 = e and b^3 = e).
- b * b = a and b * a = e, adhering to the rule b^3 = e and a^3 = e.

+ Order 4:

- This group has four elements, all of which are their own inverses. Denote the elements by e (identity), a, b, and c, with the properties $a^2 = e,\ b^2 = e, c^2$ = e, and ab = c, ac = b, bc = a.

- In this case, we have one identity element and three other elements. Each element mustappear once in every row and column, satisfying the closure property.

*	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:
- e is the identity element, so it does not change any element.
- The elements a, b, and c all have order 2, meaning a * a = e, b * b = e, and c * c = e.
- The operations between different elements are: a * b = c, a * c = b, b * c = a.

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

- Solution:
+ Lagrange's theorem:

- Lagrange theorem states that the order of the subgroup K is the divisor of the order of thegroup G.

- If G is a group of finite order m, then the order of any a∈G divides the order of G and inparticular am = e.

- If the order of finite group G is a prime order, then it has no proper subgroups.
- A group of prime order (the order has only two divisors) is a cyclic group.

+ Explain:

- Suppose we have a group K with order 4. The possible subgroups of a group of order 4 are of orders 1, 2, and 4.

- Next, we have group G with order 9. According to Lagrange's theorem, the order of anysubgroup of G must divide 9. Therefore, the possible orders of subgroups of G are 1, 3, and 9.

- According to Lagrange's Theorem, the order of the subgroup H is a divisor of the order of the group G. According to the problem, we clearly see that 4 is not a divisor of 9.

  => a group K with order 4 cannot be a subgroup of a group G with order 9

Part 3

1: Check whether the set $S = R \setminus \{-1\}$ is a group under the binary operation $*$ defined as:

$a * b = a + b + ab$

To determine if $(S,*)$ forms a group, we need to check the following four properties:

a) Closure: For all $a, b \in S$, $a * b \in S$.
b) Associativity: For all $a, b, c \in S$, $(a * b) * c = a * (b * c)$.
c) Identity element: There exists an element $e \in S$ such that for all $a \in S$, $a * e = e * a = a$.
d) Inverse element: For each $a \in S$, there exists an element $b \in S$ such that $a * b = b * a = e$ (where e is the identity element).

a) Closure

We need to check if $a * b \in S$ for all $a, b \in S$.

Given $a, b \in S$, we have:

$a * b = a + b + ab$

Since $a, b \in R \setminus \{-1\}$, the expression $a + b + ab$ will also be a real number. However, we need to ensure that $a * b \neq -1$.

Let's assume $a * b = -1$:

$a + b + ab = -1$

$ab + a + b = -1$

$ab + a + b + 1 = 0$

$(a + 1)(b + 1) = 0$

For this to be true, either $a = -1$ or $b = -1$. Since $a, b \in S = R\setminus\{-1\}$, a and b cannot be -1. Thus, $a * b \neq -1$ and closure holds.

b) Associativity

We need to verify if $(a * b) * c = a * (b * c)$.

Let's compute both sides:

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

Using the definition of $*$:

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

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

Now, compute $a * (b * c)$:

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

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

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

Both sides are equal, so associativity holds.

c) Identity element

We need to find an element $e \in S$ such that for all $a \in S$, $a * e = e * a = a$.

Let's set $a * e = a$:

$a + e + ae = a$

$e + ae = 0$

$e (1 + a) = 0$

Since $a \neq -1$, $1 + a \neq 0$. Thus, e=0. We need to check if $e = 0$ works:

$a * 0 = a + 0 + a(0) = a$

$0 * a = 0 + a + 0 (a) = a$

So, $e = 0$ is the identity element.

d) Inverse element

We need to find b ∈ S such that $a * b = b * a = e$.

Since e = 0 :

$a * b = 0$

$a + b + ab = 0$

$b(1 + a) = -a$

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

For b to be in S , $b \neq -1$. Hence, the inverse exists.

Conclusion

All group properties are satisfied, so $(S,*)$ is a group.

2: A binary operation on a set S is a function that combines any two elements of S to form another element of S. Formally, if S is a set, a binary operation $*$ on S is a function $* : S \times S \to S$.

To determine the total number of binary operations that can be defined on a set S with n elements, we need to consider that each pair of elements in S can be mapped to any element in S.

Number of pairs: Since $*$ is defined on S×S, the total number of pairs (a , b) where a , b ∈ S is n^2.

Mapping pairs to elements: Each of these n^2 pairs can be mapped to any of the n elements in S. Therefore, there are n choices for the image of each pair under the binary operation.

Total number of binary operations: Since each pair has n choices and there are n^2 pairs, the total number of binary operations on S is n^2 .

Application to a Set with 3 Elements

Let's apply this to a set containing 3 elements, say S={a,b,c}.

Number of elements (n): Here, n=3.

Number of pairs (n^2): The number of pairs is $3^2 = 9$.

Choices for each pair: Each pair can be mapped to any of the 3 elements.

Thus, the total number of binary operations on S is:

$3^{3^2} = 3^9$

Calculating

$3^9 = 19683$

So, the total number of binary operations that can be defined on a set containing 3 elements is 19683.

III. Conclusion

In conclusion, the intricate interplay between Boolean Algebra, Group Theory, and Lagrange's Theorem not only highlights their shared principles but also underscores the profound unity and interconnectedness of mathematical concepts. These disciplines, each with its own unique focus and methodology, converge to provide a comprehensive framework for understanding the fundamental structures and relationships that underpin mathematical reasoning. From the formal manipulation of logical expressions to the analysis of algebraic structures and group symmetries, Boolean Algebra, Group Theory, and Lagrange's Theorem offer powerful tools for exploring and reasoning about complex systems across diverse domains.

Moreover, the exploration of their interconnections unveils new perspectives and avenues for interdisciplinary research and innovation. By bridging disciplinary boundaries and leveraging insights from one field to enrich another, mathematicians and scientists can unlock new frontiers of knowledge and address complex challenges in mathematics, computer science, cryptography, physics, and beyond. The symbiotic relationship between these disciplines serves as a testament to the enduring elegance and universality of mathematical thought, inspiring future generations of researchers to continue unraveling the mysteries of the mathematical universe.

As we navigate the ever-expanding landscape of mathematical abstraction, the synergy between Boolean Algebra, Group Theory, and Lagrange's Theorem serves as a beacon of enlightenment, guiding us towards deeper understanding and discovery. By embracing the interconnectedness of mathematical concepts and fostering collaboration across disciplines, we can harness the full potential of mathematics to advance human knowledge and tackle the complexities of the modern world. In this journey of exploration and discovery, the unity and harmony embodied by Boolean Algebra, Group Theory, and Lagrange's Theorem inspire us to push the boundaries of mathematical inquiry and pave the way for a brighter future filled with innovation and insight.

IV. References

Group slides. Available at: asm .pdf

GeeksforGeeks. (n.d.). *Binary Tree Data Structure*. [online] Available at: https://www.geeksforgeeks.org/binary-tree-data-structure/.

GeeksforGeeks. (2023). *What is Binary Tree?* [online] Available at: https://www.geeksforgeeks.org/what-is-binary-tree/.

GeeksforGeeks. (2022). *Complete Binary Tree*. [online] Available at: https://www.geeksforgeeks.org/complete-binary-tree/.

GeeksforGeeks. (2022). *Balanced Binary Tree*. [online] Available at: https://www.geeksforgeeks.org/balanced-binary-tree/.