Higher Nationals – Summative Assignment Feedback Form

Student Name/ID	MOHAMMED MAHROOF MOHAMMED AASHIK/E230667		
Unit Title	Unit 18 – Discrete Mathematics		
Assignment Number	1	**Assessor**	
Submission Date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	

Assessor Feedback

Grade:	**Assessor Signature:**	**Date:**
Resubmission Feedback:		
Grade:	**Assessor Signature:**	**Date:**
Internal Verifier's Comments:		
Signature & Date:		

* Please note that grade decisions are provisional. They are only confirmed once internal and external moderation has taken place and grades decisions have been agreed at the assessment board.

Important Points:

1. It is strictly prohibited to use textboxes to add texts in the assignments, except for the compulsory information. eg: Figures, tables of comparison etc. Adding text boxes in the body except for the before mentioned compulsory information will result in rejection of your work.
2. Avoid using page borders in your assignment body.
3. Carefully check the hand in date and the instructions given in the assignment. Late submissions will not be accepted.
4. Ensure that you give yourself enough time to complete the assignment by the due date.
5. Excuses of any nature will not be accepted for failure to hand in the work on time.
6. You must take responsibility for managing your own time effectively.
7. If you are unable to hand in your assignment on time and have valid reasons such as illness, you may apply (in writing) for an extension.
8. Failure to achieve at least PASS criteria will result in a REFERRAL grade .
9. Non-submission of work without valid reasons will lead to an automatic RE FERRAL. You will then be asked to complete an alternative assignment.
10. If you use other people's work or ideas in your assignment, reference them properly using HARVARD referencing system to avoid plagiarism. You have to provide both in-text citation and a reference list.
11. If you are proven to be guilty of plagiarism or any academic misconduct, your grade could be reduced to A REFERRAL or at worst you could be expelled from the course
12. Use word processing application spell check and grammar check function to help editing your assignment.
13. Use **footer function in the word processor to insert Your Name, Subject, Assignment No, and Page Number on each pag**e. This is useful if individual sheets become detached for any reason.

STUDENT ASSESSMENT SUBMISSION AND DECLARATION

When submitting evidence for assessment, each student must sign a declaration confirming that the work is their own.

Student name: MOHAMMED AASHIK	Assessor name:	
Issue date:	Submission date:	Submitted on:
Programme**:** HND in Computing		
Unit 18 – Discrete Mathematics		
Assignment number and title: 1 | Unit 18 – Discrete Mathematics		

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 penalized. 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.

Guidelines for incorporating AI-generated content into assignments:

The use of AI-generated tools to enhance intellectual development is permitted; nevertheless, submitted work must be original. It is not acceptable to pass off AI-generated work as your own.

Student Declaration

Student declaration I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

Student signature: E230667	Date:

Assignment Brief

Student Name/ID Number	MOHAMMED MAHROOF MOHAMMED AASHIK/E230667
Unit Number and Title	Unit 18 :Discrete Mathematics
Academic Year	2024/2025
Unit Tutor	
Assignment Title	Discrete mathematics in Computing
Issue Date	
Submission Date	

Submission Format

This assignment should be submitted at the end of your lesson, on the week stated at the front of this brief. The assignment can either be word-processed or completed in legible handwriting.

If the tasks are completed over multiple pages, ensure that your name and student number are present on each sheet of paper.

Unit Learning Outcomes

LO1 Examine set theory and functions applicable to software engineering.

LO2 Analyse mathematical structures of objects using graph theory.

LO3 Investigate solutions to problem situations using the application of Boolean algebra.

LO4	Explore applicable concepts within abstract algebra.

Learning Outcomes and Assessment Criteria

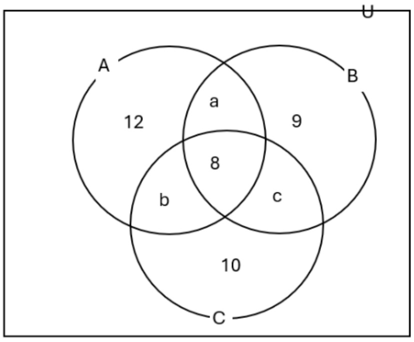
Pass	Merit	Distinction
LO1 Examine set theory and functions applicable to software engineering		**D1** Formulate corresponding proof principles to prove properties about defined sets.
P1 Perform algebraic set operations in a formulated mathematical problem. **P2** Determine the cardinality of a given bag (multiset).	**M1** Determine the inverse of a function using appropriate mathematical techniques.	
LO2 Analyse mathematical structures of objects using graph theory		**D2** Construct a proof of the Five Colour Theorem.
P3 Model contextualised problems using trees, both quantitatively and qualitatively. **P4** Use Dijkstra's algorithm to find a shortest path spanning tree in a graph.	**M2** Assess whether an Eulerian and Hamiltonian circuit exists in an undirected graph.	
L03 Investigate solutions to problem situations		

Assignment Activity

Activity 1

Part 1

1. Perform algebraic set operations in the following formulated mathematical problems.
   i. Let A and B be two non-empty finite sets. If the cardinalities of the sets A, B, and A ∪ B are 20, 35, and 45 respectively, find the cardinality of the set A ∩ B.
   ii. Given that n(U)=60, n(B)=20, n(A∩B) =8, and n(A−B) =17, find n(A∪B)?
   iii. If n(A)=40, n(B)=30, and n(C)=35, find n (A∪ B ∪ C).



Part 2

1. Write the multisets (bags) of prime factors of given numbers.
   i. 320
   ii. 60
   iii. 500
2. Write the multiplicities of each element of multisets (bags) in Part 2-1(i, ii, iii) separately.
3. Determine the cardinalities of each multiset (bag) in Part 2-1(i, ii, iii)

Part 3

1. Determine whether the following functions are invertible or not and if a function is invertible, then find the rule of the inverse $\left(f^{-1}(x)\right)$, using appropriate mathematical techniques.
   i. $f: \Re \rightarrow \Re^{+}, f(x) = x^2$
   ii. $f: \Re^{+} \rightarrow \Re^{+}, f(x) = 5/x$
   iii. $f: \Re^{+} \rightarrow [2, \infty), f(x) = x^2 + 2$
   iv. $f: \Re - \{3\} \rightarrow \Re - \{1\}, f(x) = \frac{x-1}{x-3}$
   v. $f: [0, \pi] \rightarrow [3, -3]), f(x) = 3cos(x)$

Part 4

1. Formulate corresponding proof principles to prove the following properties about defined sets.
   i. $A = B \Leftrightarrow A \subseteq B$ and $\mathrm{B} \subseteq \mathrm{A}$.
   ii. De Morgan's Law by mathematical induction.
   iii. Distributive Laws for three non-empty finite sets A, B, and C.

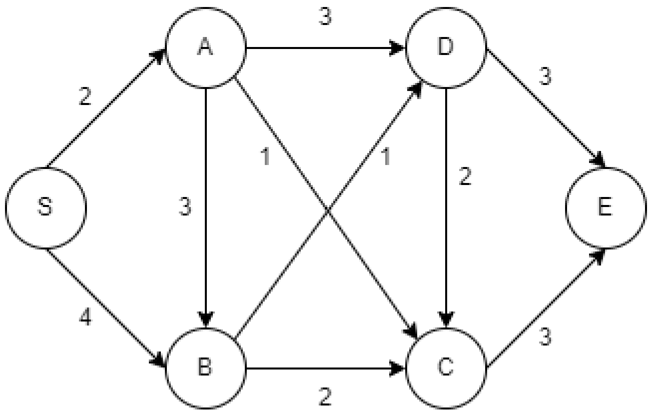
Activity 02

Part 1

1. Model two types of contextualized problems using binary trees both quantitatively and qualitatively.

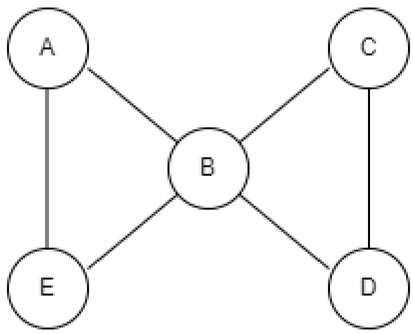
Part 2

1. Explain the Dijkstra’s algorithm for a directed weighted graph with all non-negative edge weights.
2. Use Dijkstra’s algorithm to find the shortest distance from the source vertex (S) to all other vertices, and draw the minimum spanning tree for the following weighted directed graph with vertices S, A, B, C, D, and E.

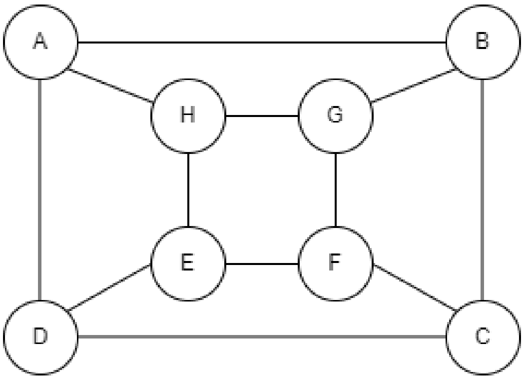


Part 3

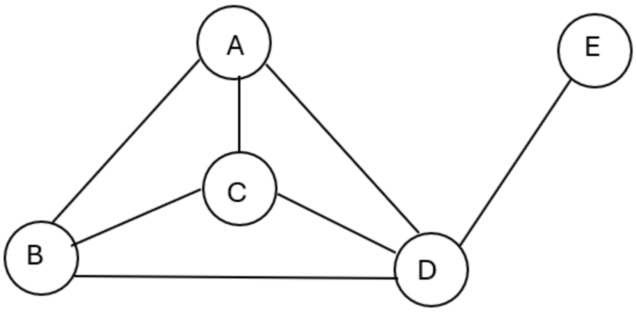
1. State whether the following undirected graphs contain an Eulerian cycle, a Hamiltonian cycle, or both. Provide explanations.
   i.



ii.



iii.



Part 4

1. Construct a proof of the five-color theorem for every planar graph.

Activity 03

Part 1

1. Diagram two real world binary problems in two different fields using applications of Boolean Algebra.

Part 2

1. Produce truth tables and its corresponding Boolean equation for the following scenarios.
   i. If the room temperature is above a certain threshold and the fan is not turned on, and someone is present in the room, then an alert should be activated.
   ii. If there is a power outage and you don't have a flashlight, then you will be in the dark.
2. Produce truth tables for given Boolean expressions.
   i. $\bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC + \bar{A}B\bar{C}$
   ii. $(A + \bar{B} + C)(A + B + C)(\bar{A} + B + \bar{C})$
   iii. $\bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\mathrm{CD}$

Part 3

1. Simplify the following Boolean expressions using algebraic methods.
   i. $A(A + B) + B(B + C) + C(C + A)$
   ii. $\bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + A\bar{B}C + ABC$
   iii. $(A + \bar{B} + \bar{C})(A + \bar{B} + C)(A + B + \bar{C})$
   iv. $\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

Part 4

1. Consider the K-Maps given below. For each K- Map
   i. Construct both the Sum of Products (SOP) and Product of Sums (POS) Boolean expressions.
   ii. Design the logic circuit using either SOP or POS from part (i).
   You can select either SOP or POS, and design the circuit using AND, OR, and NOT gates.
   iii. Based on your choice in part (ii), redesign the circuit as follows:
      - If you selected SOP in part (ii), convert the logic circuit to NAND gates.
      - If you selected POS in part (ii), convert the logic circuit to NOR gates.

a)

AB/C	0	1
00	1	0
01	0	1
11	1	1
10	0	0

(b)

AB/CD	00	01	11	10
00	1	1	0	0
01	1	1	1	1
11	0	0	0	0
10	0	1	1	0

(c)

AB/C	0	1
00	1	1
01	1	0
11	0	1
10	1	1

Activity 04

Part 1

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

Part 2

1. Determine the operation tables for group G with orders 1, 2, 3 and 4 using the elements a, b, c, and e as the identity element in an appropriate way.

2. State the relation between the order of a group and the number of binary operations that can be defined on that set.
   i. How many binary operations can be defined on a set with 3 elements?
3. State the Lagrange's theorem of group theory.
   i. Discuss whether a group H with order 5 can be a subgroup of a group with order 20 or not. Clearly state the reasons.
4. Prepare a 10-slides presentation on an application of group theory in computer science.

Part 3

1. Validate whether the set $\mathbb{Z}$ is a group under the binary operation '*'defined as a*b=a+b+2 for any two integers a, b.

using the application of Boolean algebra		
P5 Diagram a binary problem in the application of Boolean	**M3** Simplify a Boolean equation using algebraic methods.	**D3** Design a complex system using logic gates.

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

Pass	Merit	Distinction
LO4 Explore applicable concepts within abstract algebra		**D4** Prepare a presentation that explains an application of group theory relevant to your course of study.
P7 Describe the distinguishing characteristics of different binary operations that are performed on the same set. **P8** Determine the order of a group and the order of a subgroup in the given examples	**M4** Validate whether a given set with a binary operation is indeed a group	

Acknowledgment

I would like to express my sincere gratitude to Mr. nalaka gallage , my esteemed lecturer in Discrete Mathematics. Her unwavering support, insightful guidance, and dedication to fostering a deep understanding of the subject have been instrumental in shaping my comprehension of discrete mathematics. This assignment on "Discrete Mathematics in Software Engineering Concepts" has been enriched through her valuable teachings, and I am truly appreciative of her contributions to my academic journey.

Thank you

Table of Contents

Activity 01

Part 1

1.

i. Let A and B be two non-empty finite sets. If the cardinalities of the sets A, B, and A ∪ B are 20, 35, and 45 respectively

A = 20, B = 35, A ∪ B = 45

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

$= 45 = 20 + 35 - n(A \cap B)$

$= n(A \cap B) = 55 - 45 = 10.$

Therefore, the cardinalities of the set $A \cap B$ is 10.

ii. $n(U) = 60$, $n(B) = 20$, $n(A \cap B) = 8$, and $n(A - B) = 17$, find $n(A \cup B)$?

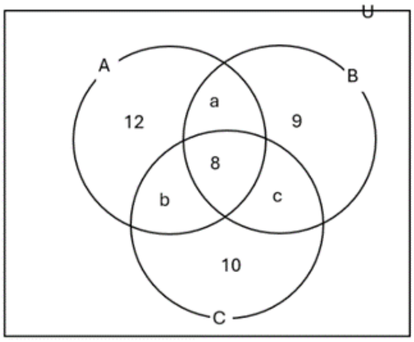
$n(A \cup B) = n(A-B) + n(B)$

$= 17 + 20$

$= 37$

Therefore, the cardinalities of the set $(A \cup B)$ is 37

iii. If $n(A)=40$, $n(B)=30$, and $n(C)=35$, find $n(A \cup B \cup C)$



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

$=n(A \cup B \cup C) = 40 + 30 + 35 - (8 + a) - (8 + c) - (8 + b) + 8$

$=n(A \cup B \cup C) = 89 - a - b - c \rightarrow 1$

By the diagram

$n(A) = 12 + a + b + 8 = 40$

$a = 20 - b \rightarrow 2$

$n(B) = 9 + a + c + 8 = 30$

$a + c = 13 \rightarrow 3$

$n(C) = 10 + b + c + 8 = 35$

$b + c = 17 \rightarrow 4$

By 3,

$= (20 - b) + c = 13$

$= b - c = 7 \rightarrow 5$

Add 4 + 5,

= b + c + b – c = 17 + 7

= 2b = 24

= b = 12

By 2,

= a = 20 -b

= a = 20 – 12

= a = 8

By 3,

= b + c = 17 = c = 7 -b

= c = 17 – 12

= c = 5

By 1,

= n (A∪B∪ C) = 89 – a – b – c

= n (A∪B∪ C) = 89 – 8 – 12 – 5

= n (A∪B∪ C) = 64

Therefore, n (A∪B∪ C) is 64.

Part 02

Question 01

i.

2	320
2	160
2	80
2	40
2	20
2	10
5	5
	1

$320 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 5$

Let the multiset of the prime factor of 320 is A.

A = [2, 2, 2, 2, 2, 2, 5]

ii.

2	60
2	30
3	15
5	5
	1

$60 = 2 \times 2 \times 3 \times 5$

Let the multiset of the prime factor of 60 is B.

B = [2, 2, 3, 5]

iii.

2	500
2	250

5	125
5	25
5	5
	1

$500 = 2 \times 2 \times 5 \times 5 \times 5$

Let the multiset of the prime factor of 500 is C.

C = [2, 2, 5, 5, 5]

Question 02

I. A = [2, 2, 2, 2, 2, 2, 5]

$\mu_A(2) = 6$

$\mu_A(5) = 1$

II. B = [2, 2, 3, 5]

$\mu_B(2) = 2$

$\mu_B(3) = 1$

$\mu_B(5) = 1$

III. C = [2,2, 5, 5, 5]

$\mu_C(2) = 2$

$\mu_C(5) = 3$

Question 03

I. $A = [\ 2, 2{,}2, 2, 2, 2, 5]$

$n(A) = \mu_A(2) + \mu_A(5)$

$= n(A) = 6 + 1$

$= n(A) = 7$

II. $B = [2,\ 2, 3, 5]$

$n(B) = \mu_B(2) + \mu_B(3) + \mu_B(5)$

$= n(B) = 2 + 1 + 1$

$= n(B) = 4$

III. $C = [2{,}2, 5, 5, 5]$

$n(A) = \mu_C(2) + \mu_C(5)$

$= n(A) = 2 + 3$

$= n(A) = 5$

<u>Part 3</u>

1)

i. $f: R \rightarrow R^+;\ f(x) = x^2$

$f(x) = x^2$

Check whether the above function is one to one. $\longrightarrow$ (1)

$f(x) = x^2$ is one to one if and only if $f(x^1) = f(x^2)$

$\Rightarrow x^1 = x^2$

Let, $f(x_1) = f(x_2)$

$= x_1^2 = x_2^2$

$= x_1 = \sqrt{x}2$

$= x_1 = (+x_2)$ or $x_1 = (-x_2)$

Therefore, $f(x_1) = f(x_2) \nRightarrow x_1 = x_2$

Therefore, (x) is not one to one $\longrightarrow$(2)

y (1) & (2),

Since there are multiple values of $x1$ for a given $x2$, the function is not one to one f is not One to One correspondent among $\mathfrak{R} \rightarrow \mathfrak{R}+$. Thus, f is not invertible.

ii. $f: \mathfrak{R}^+ \rightarrow \mathfrak{R}^+$, $f(x) = 5/x$

If we draw a segment of a horizontal line (domain of function $\mathfrak{R}$) in the range ($\mathfrak{R}^+ \leftarrow$range of the function), for example, y=3, and the line intersects the graph, it is ONTO.

Referring to the graph, when we draw a horizontal line segment (domain of function $\mathfrak{R}$) in the range ($\mathfrak{R}^+$ range of the function) anywhere, it intersects the function's graph only once, making it ONE TO ONE (1-1).

The function is invertible, and there is an inverse function, as seen in the above image.

$$\boldsymbol{f(x) = \frac{5}{x}}$$

$$\boldsymbol{y = 5/x}$$

*x

$$\boldsymbol{xy = \frac{5}{x} * x}$$

$$\boldsymbol{\frac{xy}{y} = \frac{5}{y}}$$

$$\boldsymbol{x = \frac{5}{y}}$$

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

=$\boldsymbol{f^{-1}: \Re^{+} \rightarrow \Re^{+}}$

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

iii. $f: \Re^{+} \rightarrow [2, \infty)$, $f(x) = x^2 + 2$

When we create a horizontal line segment (domain of function R) in the range(2,∞ range of the function) assume that horizontal line is y=3, refer the graph , it crosses the graph, then its ONTO

Referring to the graph, when we draw a horizontal line segment (domain of function R in the range (2,∞range of the function) anyplace, it meets precisely at 1 point in the function's graph, making it ONE TO ONE (1-1).

Referring to the above figure, the function is invertible, inverse function does exist

$f(x) = x^2 + 2$ $y = x^2 + 2$	$\boldsymbol{f: \Re^{+} \rightarrow [2, \infty)},$ $f^{-1}(\boldsymbol{x}) = \boldsymbol{\sqrt{x-2}}$

(-2) $y - 2 = x^2 \cancel{+ 2 - 2}$ $y - 2 = x^2$ $x^2 = y - 2$ $x = \pm\sqrt{y - 2}$ $f^{-1}(\boldsymbol{x}) = \sqrt{\boldsymbol{x - 2}}$	

iv. $f: \Re - \{3\} \to \Re - \{1\}, f(x) = \frac{x-1}{x-3}$

When a horizontal line segment (domain of function $\Re - \{3\}$) is drawn in the range ($\Re - \{1\}$range of the function), for example, y=(-5), consult the graph; if it crosses the graph just once, it is ONTO.

Referencing the graph, we can see that when we draw a horizontal line segment (domain of function $\Re - \{3\}$) in the range ($\Re - \{1\}$ range of the function) anyplace, it crosses precisely at 1 point in the function's graph, making it ONE TO ONE (1-1).

$$\boldsymbol{f(x) = \frac{x-1}{x-3}}$$

$$\boldsymbol{y = \frac{x-1}{x-3}}$$

*(x-3)

$$\boldsymbol{y(x-3) = \frac{x-1}{x-3}} * (x-3)$$

$$\boldsymbol{yx - 3y = (x-1)}$$

(-x,3y)

$$\boldsymbol{yx - 3y - x + 3y = (x-1) - x + 3y}$$

$$\boldsymbol{yx - x = 3y - 1}$$

$$\boldsymbol{x(y-1) = 3y - 1}$$

/(y-1)

$$\boldsymbol{\frac{x(y-1)}{y-1} = \frac{3y-1}{y-1}}$$

$$\boldsymbol{x = \frac{(3y-1)}{(y-1)}}$$

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

$$\boldsymbol{f^{-1}: \Re - \{3\} \to \Re - \{1\},}$$

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

v. $f\colon [0,\pi] \to [3,-3])$, $f(x) = 3cos(x)$

When a horizontal line segment (domain of function 0,π) is drawn in the range (-3,3 range of the function), let's say that the horizontal line is y=1. Look at the graph; if it crosses it only once, it is ONTO.

Referring to the graph, we can see that when we draw a horizontal line segment (domain of the function 0, π) in the range (-3,3 range of the function) anyplace, it crosses precisely at 1 point, making it ONE TO ONE (1-1).

The function is invertible, and there is an inverse function, as seen in the above image.

$$f(x) = 3cos(x)$$

$$y = 3cos(x)$$

/3

$$\frac{y}{3} = \frac{\cancel{3}cos(x)}{\cancel{3}}$$

$$\frac{y}{3} = cos(x)$$

$$x = cos^{-1}\left(\frac{y}{3}\right)$$

$$f^{-1}x = cos^{-1}\left(\frac{x}{3}\right)$$

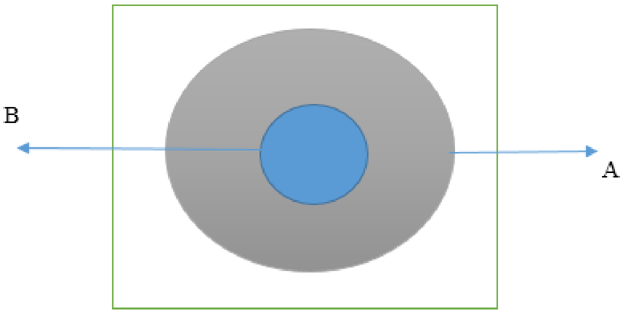
$$f^{-1}: [0, \pi] \to [3, -3]),\ \ f^{-1}x = cos^{-1}\left(\frac{x}{3}\right)$$

Part 4

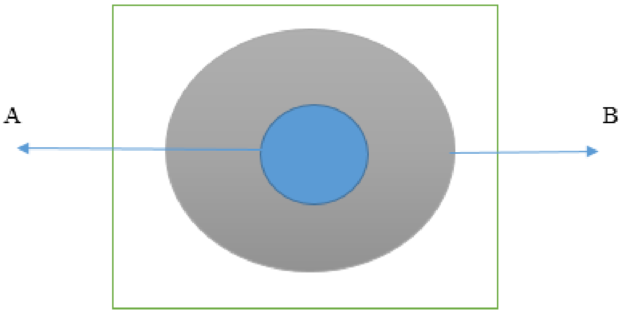
1.

i. $\boldsymbol{A} = \boldsymbol{B} \Leftrightarrow \boldsymbol{A} \subseteq \boldsymbol{B}$ ***and*** $\boldsymbol{B} \subseteq \boldsymbol{A}$.

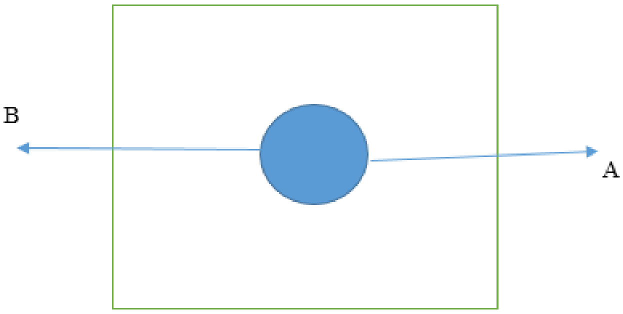
The statement $\boldsymbol{A} = \boldsymbol{B} \Leftrightarrow \boldsymbol{A} \subseteq \boldsymbol{B}$ and $\boldsymbol{B} \subseteq \boldsymbol{A}$ represents the concept of set equality and provides insight into the relationship between two sets, $\boldsymbol{A}$ and $\boldsymbol{B}$. To prove this equivalence, need to establish two key implications: one from left to right (⇒) and the other from right to left (⇐). Starting with the left-to-right implication (⇒), assuming $\boldsymbol{A} = \boldsymbol{B}$ means that every element in $\boldsymbol{A}$ is also in $\boldsymbol{B}$, and vice versa, since sets only contain distinct elements. Therefore, if $\boldsymbol{A} = \boldsymbol{B}$, it naturally follows that $\boldsymbol{A}$ is a subset of $\boldsymbol{B}$ ($\boldsymbol{A} \subseteq \boldsymbol{B}$) and $\boldsymbol{B}$ is a subset of $\boldsymbol{A}$ ($\boldsymbol{B} \subseteq \boldsymbol{A}$). This is a straightforward application of the definition of set equality, which implies mutual inclusion of elements in both sets. Conversely, considering the right-to-left implication (⇐), which asserts that if $\boldsymbol{A} \subseteq \boldsymbol{B}$ and $\boldsymbol{B} \subseteq \boldsymbol{A}$, then $\boldsymbol{A} = \boldsymbol{B}$, we can see that every element of $\boldsymbol{A}$ is in $\boldsymbol{B}$, and every element of $\boldsymbol{B}$ is in $\boldsymbol{A}$. Combining these two statements, it becomes evident that $\boldsymbol{A}$ and $\boldsymbol{B}$ contain precisely the same elements, making them equal. This two-way implication demonstrates that $\boldsymbol{A} = \boldsymbol{B}$ if and only if $\boldsymbol{A}$ is a subset of $\boldsymbol{B}$ and $\boldsymbol{B}$ is a subset of $\boldsymbol{A}$. It is a fundamental principle in set theory, known as the Axiom of Extensionality, which essentially states that two sets are equal if they have the same elements. To illustrate the significance of this equivalence, consider a scenario where the author wants to prove that two sets A and B are equal. By showing that $\boldsymbol{A} \subseteq \boldsymbol{B}$ and $\boldsymbol{B} \subseteq \boldsymbol{A}$, it is established that the elements of both sets are identical, thus proving the equality of $\boldsymbol{A}$ and $\boldsymbol{B}$. This principle simplifies set equality proofs and underlines the core idea that sets are characterized by their constituent elements. Ultimately, it highlights the power of mutual inclusion as the defining criterion for set equality, offering a foundational concept in set theory and mathematics. This insight into the relationship between sets is a fundamental concept in mathematics, elucidated by the author.



$B \subseteq A$



$\boldsymbol{A \subseteq B}$



ii. **De Morgan’s Law by mathematical induction.**

De Morgan's Law states that how mathematical statements and concepts are related through their opposites. In set theory, De Morgan's Laws describe the complement of the union of two sets is always equals to the intersection of their complements. And the complement of the intersection of two sets is always equal to the union of their complements.

De Morgan's Laws Statement and Proof

1) De Morgan's law of union

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

2) De Morgan's law of intersection

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

De Morgan's law of union

$= (A \cup B)' = A' \cap B'$ ……… **(1)**

Where complement of a set is given as

$A' = \{ x : x \in U \text{ and } x \notin A \}$

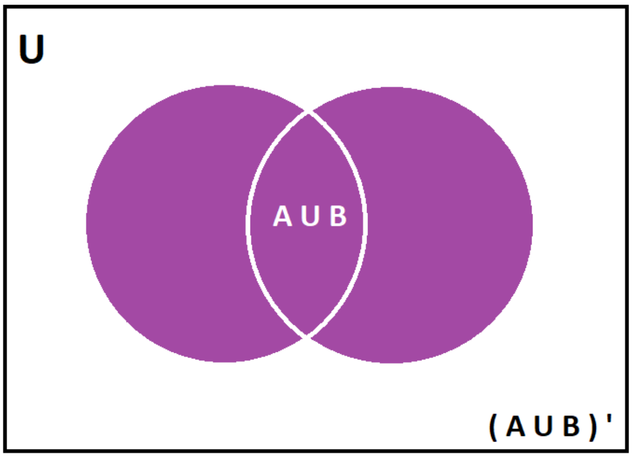
Where,

A' = It denotes the complement

We can easily understand this concept with the help of Venn Diagrams.

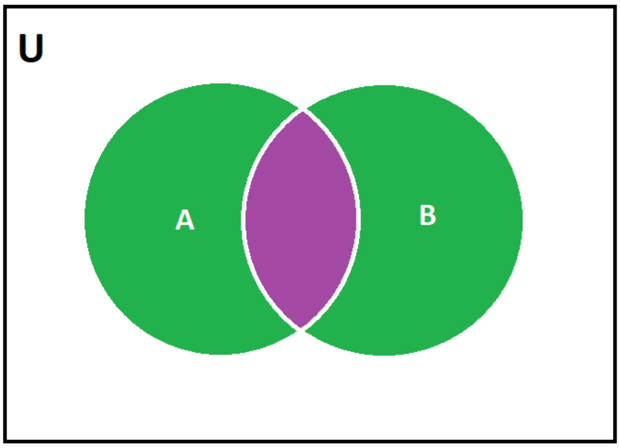
The left-hand side of the first equation produces the complement of both sets A and B. It means the union of set A and B is the set of all elements that lie either in Set A or in Set B. The given diagram depicts the Venn diagram of set A and Set B.

LHS



: (A∪B)'

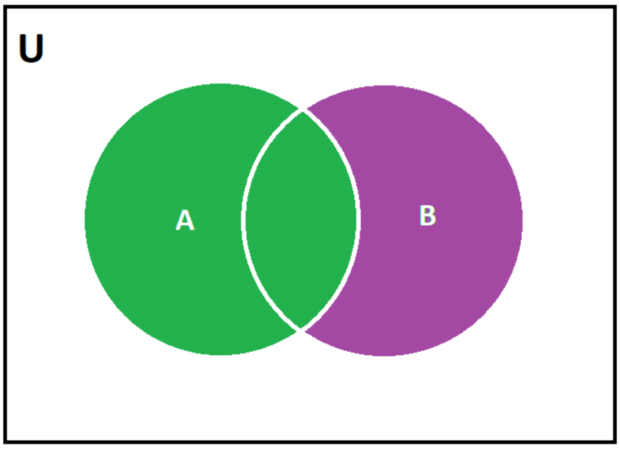
LHS



In the above diagram, the highlighted purple color area represents the A. The complement of **(A)'** is a set of all those elements except the highlighted blue color area. The given diagram depicts the Venn diagram of the complement of A and B.

Similarly, the right-hand side of the first equation can be represented by the given Venn diagram. The given diagram depicts the complement of A.

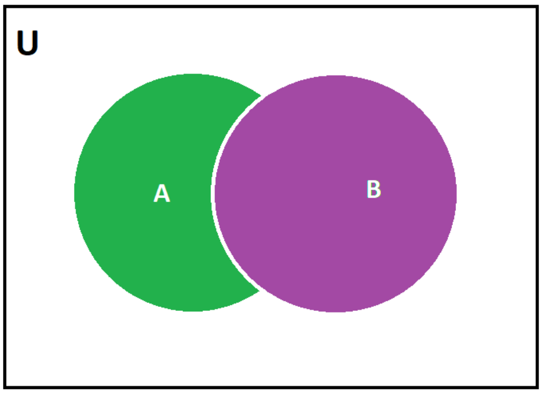
RHS



The green part represents Set A, and the purple part represents its complement that is A.'

Similarly, the component of B can be represented as

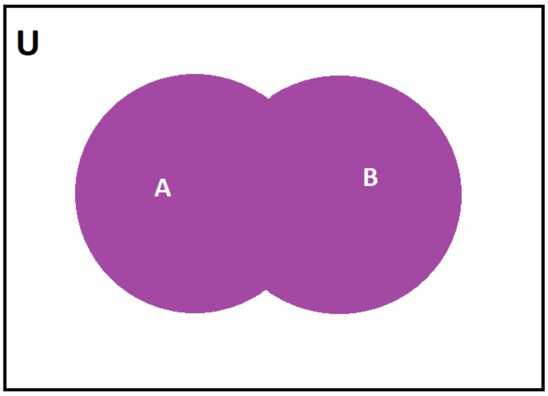
RHS



The purple part represents the set B, and the green part indicates its complement: B'.

Now, we will be combined both the image 3 and 4 to one another, we get the given Venn Diagram

RHS



Therefore,

L.H.S = R.H.S

Mathematically,

$A \cup B$ = Either A or B

$(A \cup B)'$ = Neither A nor B

A' = Not lies in A

B' Not lies in B

$A' \cap B'$ = Not in A and not in B.

Let $J = (A \cup B)'$ and $K = A' \cap B'$.

Let's s be an arbitrary element of J then $s \in J = s \in (A \cup B)'$.

$= s \notin (A \cup B)$

$= s \notin A$ and $s \notin B$

$= s \in A'$ and $s \in B'$.

$= s \in A' \cap B'$.

$= s \in K$

Therefore, $J \subset K$ **………(i)**

Again, let t be an arbitrary element of K then t ∈ K = t ∈ A' ∩ B.'

= t ∈ A' and t ∈ B.'

= t ∉ A and t ∉ B

= t ∉ (A U B)

= t ∈ (A U B).'

= t ∈ J

Therefore, K ⊂ J **………… (ii)**

Now combine (i) and (ii) we get; J = K i.e. (A U B)' = A' ∩ B'

Therefore, by applying Venn Diagrams and Analyzing De Morgan's Laws,

we have proved that (A)' = A' ∩B.'

De Morgan's theorem describes that the product of the complement of all the terms is equal to the summation of each individual term's component.

Proof of De Morgan's law:

(A ∩ B)' = A' U B.'

Let P = (A ∩ B)' and Q = A' U B'

Let's s be an arbitrary element of M then s ∈ P = s ∈ (A ∩ B).'

= s ∉ (A ∩ B)

= s ∉ A or x ∉ B

= s ∈ A' or s ∈ B.'

= s ∈ A' U B.'

= s ∈ N

Therefore, P ⊂ Q **…………(i)**

Again, let t be an arbitrary element of Q then t ∈ N = t ∈ A' U B.'

= t ∈ A' or t ∈ B.'

= t ∉ A or t ∉ B

= t ∉ (A ∩ B)

= t ∈ (A ∩ B).'

= t ∈ M

Therefore, Q ⊂ P**…………(ii)**

Now combine (i) and (ii) we get; P = Q i.e. (A ∩ B)' = A' U B'

(javatpoint, 2023)

By mathematical induction prove that De morgans Law is true for any number of sets.

$(A \cup B)^c = A^c \cap B^c$

When the number of sets (n) is 2, the de morgan's law is true. let's check whether it is true for n=3

$(A \cup B)^c$ = $((A \cup B) \cup C)^c$; Associative law

= (A ∪ B) ∩ C ; De morgan's law for n=2

= (A ∩ B) ∩ C ; De morgan's law for n=2

= $A^c \cap B^c \cap$ C ; Associative law

therefore, De Morgan's law is true for n=3

Assume that de Morgan's law is true for n=k (k ∈ N, K ≠ 1)

$(A_1 \cup A_2 \cup A_3 \cup \ldots\ldots \cup A_K)^c = A_1^C \cap {}_2^C \cap A_3^C \cap A_K^C$

let's check whether it is true for n=k+1

$(A_1 \cup A_2 \cup A_3 \cup \ldots\ldots \cup A_K \cup A_{K+1})^c = (A_1 \cup A_2 \cup \ldots\ldots \cup A_K) \cup A_{K+1}$

$\underline{(A \cap B)^c = A^c \cup B^c}$

When the number of sets (n) is 2, the de organ's law is true. let's check whether it is true for n=3

$(A \cap B)^c$ = $((A \cap B) \cap C)^c$; Associative law

= $(A \cap B) \cup C$; De morgan's law for n=2

= $(A \cup B) \cup C$; De morgan's law for n=2

= $A^c \cup B^c \cup C$; Associative law

therefore, De Morgan's law is true for n=3

Assume that de Morgan's law is true for n=k ($k \in N$, $K \neq 1$)

$(A_1 \cap A_2 \cap A_3 \cap . . . . . . \cap A_K)^c = A_1^C \cup {}_2 \cup A^C{}_3 \cup A^C{}_K$

let's check whether it is true for n=k+1

$(A_1 \cap A_2 \cap A_3 \cap . . . . . . \cap A_K \cap A_{K+1})^c = (A_1 \cap A_2 \cap . . . . . . \cap A_K) \cap A_{K+1}$

These two De Morgan's Laws are essential tools in set theory and have applications in various areas of mathematics and computer science, particularly in Boolean algebra, logic, and data analysis. They allow to manipulate and simplify complex set expressions by transforming union and intersection operations into complement and complement into union and intersection operations, making them valuable in solving problems involving sets and logic.

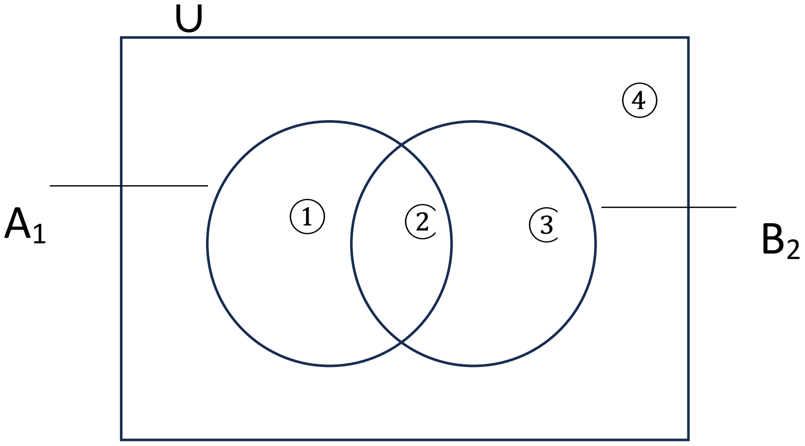
$(A_1 \cap A_2 \cap A_3 \ldots A_n)^c = A_1^c \cup A_2^c \cup A3c \ldots A_n^c$

Base case: when n=2; $(A_1 \cap A_2)^c = A_1^c \cup A_2^c$

For any set of A_1 and A_2, $(A_1 \cap A_2)^c = A_1^c \cup A_2^c$

Proof of Venn diagram

LHS:



$(A_1 \cap A_2)^c$

$(1 + 2) \cap (\cancel{2} + 3)$

$(2)^c$

$1 + 3 + 4$

RHS:

$A_1^c \cup A_2^c$

$(1 + 2)^c \cup (2 + 3)^c$

$(2 + 4) \cup (1 + \cancel{4})$

$1 + 4 + 3$

∴ LHS = RHS

Assumption

De Morgan's law when n=k and k>2

$(A_1 \cap A_2 \cap A_3 \ldots \cap A_k)^c = A_1^c \cup A_2^c \cup A_3^c \ldots \cup A_k^c \rightarrow 1$

Induction step

We need to prove that, when n=k+1, De Morgan's law is true using above

$(A_1 \cap A_2 \cap A_3 \ldots \cap A_k \cap A_{k+1})^c = A_1^c \cup A_2^c \cup A_3^c \ldots \cup A_k^c \cup A_{k+1}^c \rightarrow$ (2)

Define $B = (A_1 \cap A_2 \cap A_3 \ldots \cap A_k)$

Substituting B in the equation (2) to the left side the we have

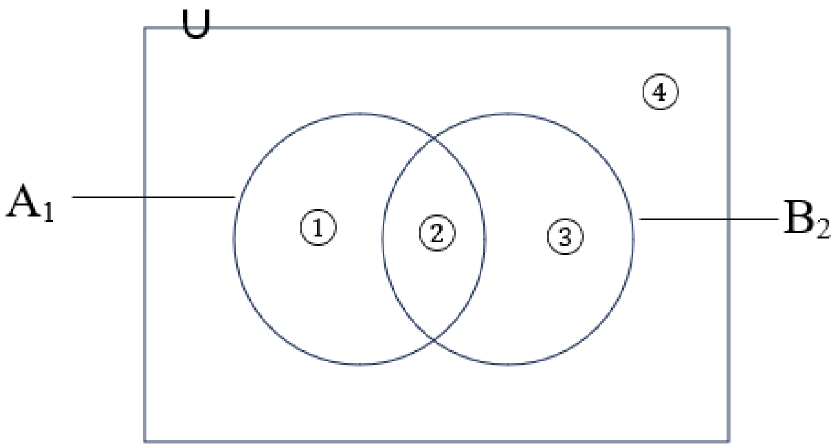
$(A_1 \cap A_2 \cap A_3 \ldots \cap A_k \cap A_{k+1})^c = (B \cap A_{k+1})^c = B^c \cup A_{k+1}^c$ **when n=2 De Morgan law is true**

$= (A_1 \cap A_2 \cap A_3 \ldots \cap A_k)^c \cup A_{k+1}^c$

$= A_1^c \cup A_2^c \cup A_3^c \ldots \cup A_k^c \cup A_{k+1}^c$

By the mathematical induction when n=k+1 De Morgan's law is true

Then we can prove that De Morgan law is true for all n Є N



LHS:

$(A_1 \cup A_2)^c$

$(1+2) \cup (\cancel{2}+3)$

$(1+2+3)^c$

4

RHS:

$A_1^c \cap A_2^c$

$(1+2)^c \cap (2+3)^c$

$(3+4) \cap (1+\cancel{4})$

4

∴ LHS = RHS

Generalized De Morgan’s law for n number of sets:

$(A_1 \cup A_2 \cup A_3 \ldots A_n)^c = A_1^c \cap A_2^c \cap A3c \ldots A_n^c$

Base case: when n=2; $(A_1 \cup A_2)^c = A_1^c \cap A_2^c$

For any set of A_1 and A_2, $(A_1 \cup A_2)^c = A_1^c \cap A_2^c$

Assumption step

De Morgan's law when n=k and k>2

$(A_1 \cup A_2 \cup A_3 \ldots \cup A_k)^c = A_1^c \cap A_2^c \cap A_3^c \ldots \cap A_k^c \longrightarrow 1$

Induction step:

We must prove when n=k+1, De Morgan's law is true using above

$(A_1 \cup A_2 \cup A_3 \ldots \cup A_k \cup A_{k+1})^c = A_1^c \cap A_2^c \cap A_3^c \ldots \cap A_k^c \cap A_{k+1}^c \longrightarrow 2$

$B = (A_1 \cup A_2 \cup A_3 \ldots \cup A_k)$

Substituting B in the equation 2 to the left side the we have

$(A_1 \cup A_2 \cup A_3 \ldots \cup A_k \cup A_{k+1})^c = (B \cup A_{k+1})^c = B^c \cap A_{k+1}^c$ when n=2 De Morgan is true

$$(A_1 \cup A_2 \cup A_3 \ldots \cup A_k)^c \cap A_{k+1}^c$$

$$A_1^c \cap A_2^c \cap A_3^c \ldots \cap A_k^c \cap A_{k+1}^c$$

By the mathematical induction when n=k+1 De Morgan's law is true

Then we can prove that De Morgan law is true for all n Є N

iii. Distributive Laws for three non-empty finite sets A, B, and C.

Let x be an arbitrary element of A∪ (B ∩ C). Then,

$x \in A \cup (B \cap C) \Rightarrow x \in A$ or $x \in (B \cap C)$

$\Rightarrow x \in A$ or $(x \in B)$ and $x \in C$

$\Rightarrow (x \in A$ or $x \in B)$ and $(x \in A$ or $x \in C)$

[∵ 'or' distributes 'and']

$\Rightarrow x(A \cup B)$ and $x \in (A \cup C)$

$\Rightarrow x \in (A \cup B) \cap (A \cup C)$

$\therefore A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C) \longrightarrow$ **(a)**

Again. let y be an arbitrary element of $(A \cup B) \cap (A \cup C)$. Then,

$y \in (A \cup B) \cap (A \cup C) \Rightarrow y \in (A \cup B)$ and $y \in (A \cup C)$

$\Rightarrow (y \in A$ or $y \in B)$ and $(y \in A$ or $y \in C)$

$\Rightarrow y \in A$ or $(y \in B$ and $y \in C)$

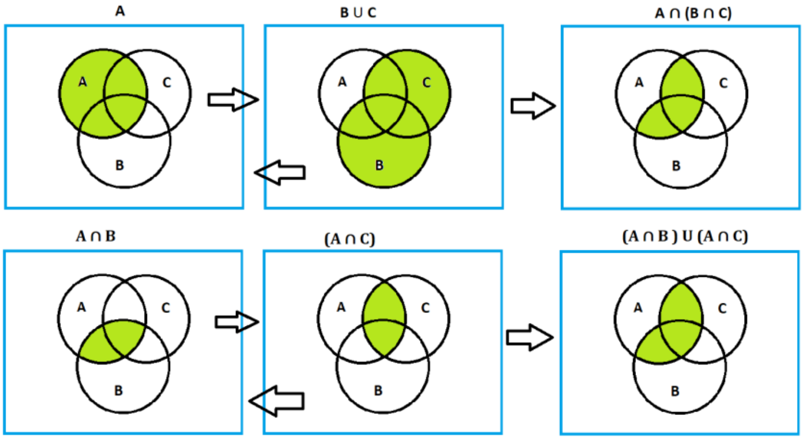
[∵ 'or distributes 'and']

$\Rightarrow y \in A$ or $y \in (B \cap C)$

$\Rightarrow y \in A \cup (B \cap C)$

$\therefore (A \cup B) \cap (A \cup C) \subseteq A \cup (B \cap C) \longrightarrow$ **(b)**

From (a) and (b), we get $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$



Let x be an arbitrary element of $A \cap (B \cup C)$. Then,

$x \in A \cap (B \cup C) \Rightarrow x \in A$ and $x \in (B \cup C)$

$\Rightarrow x \in A$ and ($x \in B$ or $x \in C$)

$\Rightarrow$ ($x \in A$ and $x \in B$) or ($x \in A$ and $x \in C$)

[$\because$ 'and' distributes 'or']

$\Rightarrow x \in (A \cap B)$ or $x \in (A \cap C)$

$\Rightarrow x (A \cap B) \cup (A \cap C)$

$\therefore A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C) \longrightarrow$ **(p)**

Again, let y be an arbitrary element of $(A \cap B) \cup (A \cap C)$. Then,

$y \in (A \cap B) \cup (A \cap C) \Rightarrow y \in \quad (A \cap B)$ or $y \in (A \cap C)$

$\Rightarrow$ ($y \in A$ and $y \in B$) or ($y \in A$ and $y \in C$)

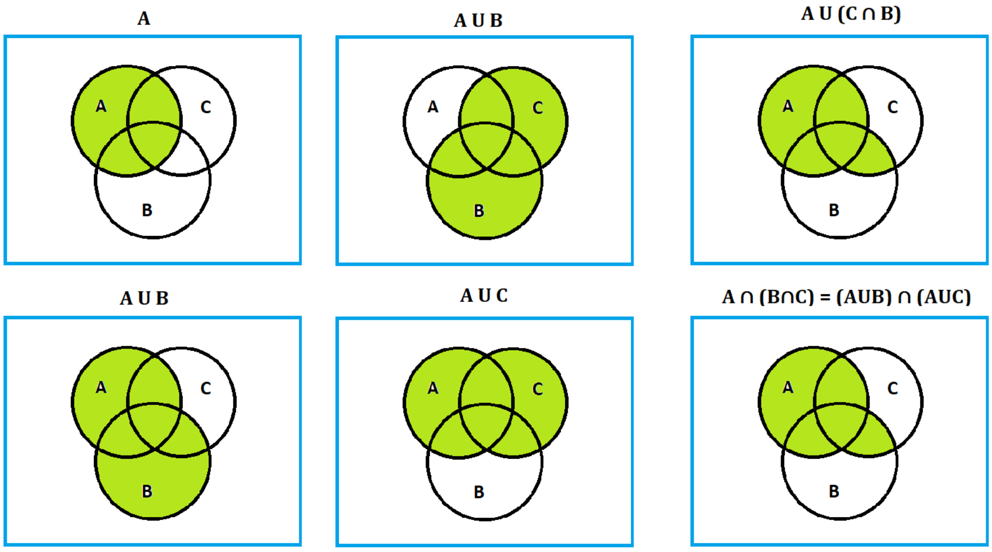
$\Rightarrow y \in A$ and ($y \in B$ or $y \in C$) [$\because$ and' distributes 'or']

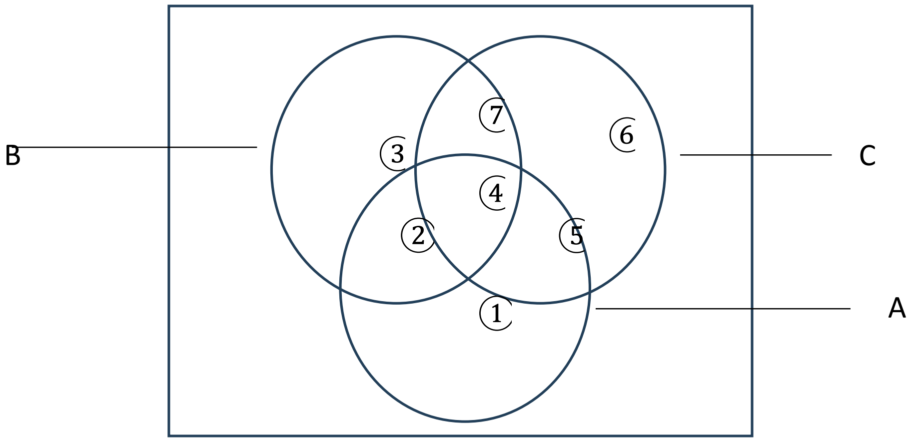
$\Rightarrow y \in A$ and $y \in (B \cup C)$

$\Rightarrow y \in A \cap (B \cup C)$.

$\therefore (A \cap B) \cup (A \cap C) \subseteq A \cap (B \cup C) \longrightarrow$ **(Q)**

From (p) and (q), we get $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.





A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)

LHS: A ∪ (B ∩ C)

((1) +(2) +(4) +(5)) ∪ ((2) +(3) +$\cancel{(4)}$+ $\cancel{(7)}$) ∩ ((4)+ (7) +(5)+ (6))

((1) +(2) +(4) +(5)) ∪ ($\cancel{(4)}$+ (7))

= ((1) +(2) +(4) +(5)+(7))

RHS: (A ∪ B) ∩ (A ∪ C)

((①+②+④+⑤) ∪ (~~②~~+③+~~④~~+ ⑦)) ∩ ((①+②+④+⑤) ∪ (~~④~~+ ⑦ +~~⑤~~+ ⑥))

((~~①+②+④+⑤+③+⑦~~)) ∩ (((①+②+④+⑤+⑦+~~⑥~~))

= (①+②+④+⑤+⑦)

∴ RHS= LHS

A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)

LHS: **A ∩ (B ∪ C)**

((①+②+④+⑤) ∩ (②+③+~~④~~+ ~~⑦~~) ∪ (④+ ⑦ +⑤+ ⑥))

(① +~~②+④+⑤~~) ∩ (~~②~~+③+~~④~~+ ⑦ +~~⑤~~+ ⑥))

= (②+④+⑤)

RHS: **(A ∩ B) ∪ (A ∩ C)**

(((①+~~②+④~~+⑤) ∩ (~~②~~+③+~~④~~+ ⑦)) ∪((①+②+~~④~~+~~⑤~~) ∩ (~~④~~+ ⑦ +~~⑤~~+ ⑥))

(((②+④)) ∪ ((~~④~~+⑤))

= (②+④+⑤)

∴RHS= LHS

Here we can conclude that the distributive laws for 3 non-empty sets is true as LHS=RHS

(A ∩ (B ∪C) = (A ∪ B) ∪ (A ∩ C))

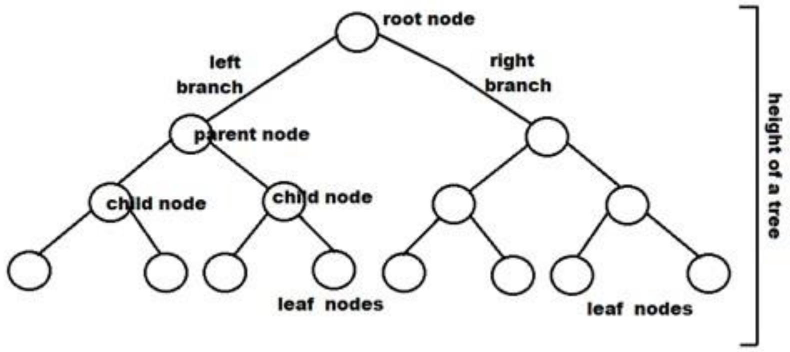
Activity 02

Part 1

1.

A Model two contextualized problems using binary trees both quantitatively and qualitatively

A binary tree is a non-linear data structure that can have up to two children per parent. Every node in a binary tree has a left and right reference, in addition to the data element. The root node is the highest node in a tree's structure. Parent nodes are those that contain other sub-nodes. (Sharma, 2025)



A parent node has two child nodes, the left and right child. A binary tree is used in a variety of applications, including hashing, network traffic routing, data compression, binary heap preparation, and binary search trees.

Linear Data Structures

A data structure is been regarded as linear when it is created with an element that is put in a sequential or one after another format. This establishes a system in a sense in which every component is related to the other, one after the other, in a series line or chain.

Quantitative Binary Trees

A quantitative binary tree is characterized by assigning numerical values to its nodes, allowing for a quantitative representation of the structure. This quantitative measure could be associated with various attributes such as quantities, weights, or any numeric parameter relevant to the given scenario. The numerical values assigned to nodes play a crucial role in quantitative analysis, enabling comparisons and computations within the tree structure. (javatpoint, 2025)

The typical structures that are linear are:

• Arrays – fixed-size collections where each item is stored in a specific index

• Stacks - they take place on the last come, first out (LIFO) principle, such as with plates.

• Queues- operated under the principle of First in First out (FIFO)- as in a queue of people.

• Linked Lists -collection of nodes that have pointers, which enable them to be used dynamically.

They are appropriate when you desire to work on items sequentially e.g. reading a file line by line or respond to customer care-related petitions.

Non-Linear Data Structures

A data structure on the other hand is non-linear when its elements are arranged hierarchically or branching in nature. This implies that one element can be related to more than one element forming levels or layers of data.

The most familiar of the non-characteristics data structures include trees and graphs. They enable the relationship between data points to be more intricate. To give an illustration, we have the trees in file systems and organization charts, the graphs in social networks and routing systems.

The structure of a tree is specific: it begins with a single node (it is referred to as the root) and extends into the sub-nodes. A graph on the contrary is more free form structure in which the connection (edges) may loop and does not necessarily have strict form of hierarchy.

What is Binary Tree?

A binary tree is a particular category of tree in which every kid has two at most children a left child and a right child.

A binary tree may be:

- Empty (null tree): it has no nodes.
- Single node: only the root without any harm.
- Ordered: having nodes at left focused and right subtrees.

Binary trees are highly utilized, as they are simple, and the basis of more complex data structures include Binary search trees, heaps, and decision trees.

Binary trees Types

- Strictly Binary Tree

In such a binary tree each inner (not-leaf) node has 2 children (one appearing on the right and one on the left).

When a tree contains only one child in a node, then such tree is not considered strictly binary.

- Full Binary Tree:

Each node in the internal nodes has two children and the leaf nodes all belong to the same depth or level in the tree.

This kind of tree is well balanced as far as its structure is concerned.

creating a study chart is simple and an easy task. It can be done by merely drawing a list of things to study on a piece of paper and leave it at that.

Example of Quantitative Binary Tree

The binary tree presented here shows a quantitative decision-making method for vacation planning. The root node represents the first decision to choose a holiday destination, and it divides into two categories: "Domestic Destinations" and "International Destinations." The decision-making process is further refined depending on the projected daily cost, with sub-branches for "Budget" and "Luxury" within each destination category. The quantitative part is calculating numerical cost figures to improve holiday planning.

Root Node

Vacation Decision

Left Child

Domestic Destinations

- Left Sub-Child

  Budget (Estimated cost per day **60$**)

- Right Sub-Child

  Luxury (Estimated cost per day **110$)**

Right Child

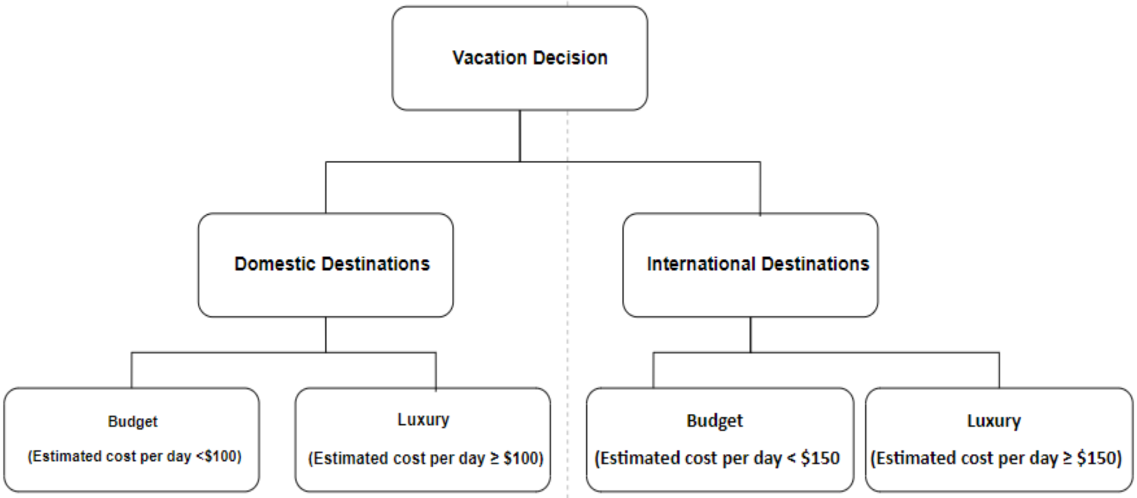
International Destinations

- Left Sub-Child

  Budget (Estimated cost per day **$120**)

- Right Sub-Child

  Luxury (Estimated cost per day **$200**)



Qualitative Binary Trees

Qualitative binary trees involve the assignment of non-numeric attributes to nodes, capturing characteristics that are not expressed through numerical values. These attributes could include labels, categories, or any qualitative descriptors that are relevant to the specific context. Qualitative binary trees provide a means to represent relationships and decision-making processes where non-numeric distinctions are more appropriate. (javatpoint, 2025)

Example of Qualitative Binary Trees

At the root node, which represents the initial application, the first decision branches into "Technical Positions" and "Non-Technical Positions." The following level introduces sub-conditions such as "Problem-Solving Skills" and "Communication Skills." Further refinement includes considerations like "Relevant Experience" and "Cultural Fit," providing a qualitative approach to

Root Node

Job Recruitment Decision

Left Child

Technical Positions

- Left Sub-Child

  Strong Problem-Solving Skills (Qualitative assessment)

- Left Sub-Sub-Child

  Relevant Experience (Decision: Hire)

- Right Sub-Child:

  Weak Problem-Solving Skills (Decision: Reject)

- Right Sub-Sub-Child

  No Relevant Experience (Decision: Reject)

Right Child

Non-Technical Positions

- Left Sub-Child

  Strong Communication Skills (Qualitative assessment)

- Left Sub-Sub-Child

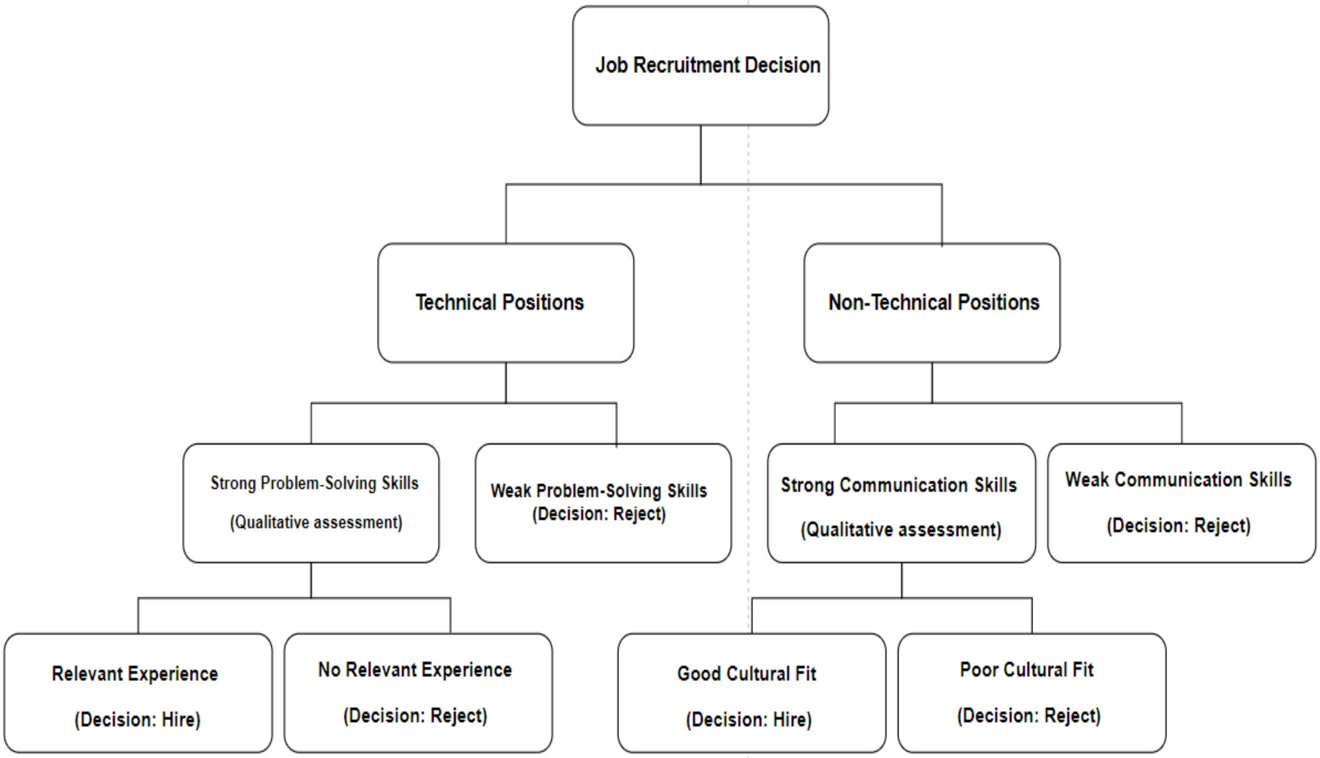
  Good Cultural Fit (Decision: Hire)

- Right Sub-Child

  Weak Communication Skills (Decision: Reject)

- Right Sub-Sub-Child

  Poor Cultural Fit (Decision: Reject)



These two examples illustrate the flexibility of binary trees, showcasing their adaptability to both quantitative and qualitative scenarios. The visual representations provide a tangible view of the structures, enhancing understanding and applicability in various contexts.

<u>Part 2</u>

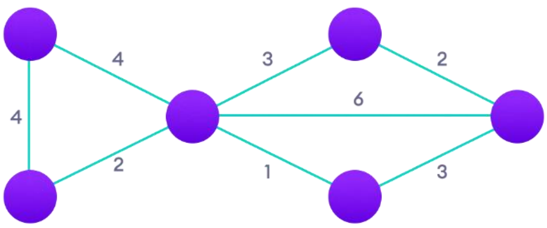
Question 01

Dijkstra's algorithm for a directed weighted graph with all non-negative edge weights is as follows.

Dijkstra's algorithm is a popular graph traversal algorithm used to find the shortest path between a starting node and all other nodes in a weighted graph. It was developed by Dutch computer scientist Edsger W. Dijkstra in 1956.

The algorithm works on graphs that have non-negative weights assigned to their edges. It maintains a set of unvisited nodes and a distance value for each node. Initially, the distance value for the starting node is set to 0, and the distance values for all other nodes are set to infinity. The algorithm then iteratively selects the node with the smallest distance value from the set of unvisited nodes and explores its neighboring nodes. During each iteration, Dijkstra's algorithm updates the distance values of the neighboring nodes if a shorter path is found. It calculates the tentative distance by adding the weight of the current edge to the distance value of the selected node. If this tentative distance is smaller than the current distance value of the neighboring node, the distance value is updated.

The algorithm continues to visit unvisited nodes and update their distances until all nodes have been visited or the target node (if specified) has been reached. Once the algorithm finishes, the resulting distance values represent the shortest path from the starting node to all other nodes in the graph. Dijkstra's algorithm guarantees finding the shortest path under the condition that the graph does not contain any negativeweight cycles. It is commonly used in various applications, such as network routing, transportation planning, and GPS navigation systems, where finding the optimal path is crucial. (Cheng, 2009)



Step: 1

Basic Characteristics of Dijkstra’s Algorithm

Below are the basic steps of how Dijkstra’s algorithm works.

- Dijkstra’s algorithm starts at the node source node we choose and then it analyzes the graph condition and its paths to find the optimal shortest distance between the given node and all other nodes in the graph.

- Dijkstra's algorithm keeps track of the currently known shortest distance from each node to the source node and updates the value after it finds the optimal path once the algorithm finds the shortest path between the source node and destination node then the specific node is marked as visited.

Basics requirements for Implementation of Dijkstra's Algorithm

- **Graph**

  Dijkstra's Algorithm can be implemented on any graph but it works best with a weighted Directed Graph with non-negative edge weights and the graph should be represented as a set of vertices and edges.

- **Source Vertex**

  Dijkstra's Algorithm requires a source node which is starting point for the search.

- **Destination vertex**

  Dijkstra's algorithm may be modified to terminate the search once a specific destination vertex is reached.

- **Non-Negative Edges**

  Dijkstra's algorithm works only on graphs that have positive weights this is because during the process the weights of the edge have to be added to find the shortest path. If there is a negative weight in the graph then the algorithm will not work correctly. Once a node has been marked as visited the current path to that node is marked as the shortest path to reach that node.

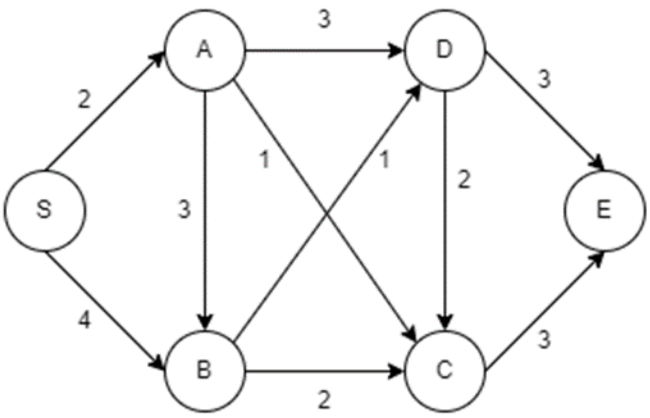
Algorithm for Dijkstra's Algorithm

- 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

Question 02

Use Dijkstra's algorithm to find the shortest distance from the source vertex (S) to all other vertices, and draw the minimum spanning tree for the following weighted directed graph with vertices S, A, B, C, D, and E.



shortest distances of each and every vertex

Initial stage/Vertex		**S**	**A**	**C**	**B**	**D**	**E**
S	0	-	-	-	-	-	-
A	∞	2	-	-	-	-	-
B	∞	4	4	4	-	-	-
C	∞	∞	3	-	-	-	-
D	∞	∞	5	5	5	-	-
E	∞	∞	∞	6	6	6	-

There the Shortest path for each and every from the source vertex S

Vertex B

Incoming edges S, A

From S

4 - 4

0 = 0 (shortest distance to s) TRUE

S TO B

From A

4 – 3

1 ≠ 2 (SHORTEST DISTANCE TO A) not true

Vertex C

Incoming edges A , B and D

From A

3 - 1

2 = 2 (shortest distance to A) true

S TO A TO C

FROM B

3 – 2

$1 \neq 4$ (shortest distance to B) NOT true

FROM D

$(3 - 2 = 1) \neq 5$

(Shortest distance to **D**) NOT TRUE

VERTEX E

INCOMING EDGES D , C

FROM D

6 – 3

3 ≠ 5 (shortest distance to D) NOT true

FROM C

6 – 3

3 = 3 (shortest distance to C) true

S TO A TO C TO

FROM 0

3 – 2

1 ≠ 5 (shortest distance to 0) NOT CORRECT

INCOMING EDGES A , B

FROM

5 - 3

2 = 2 (shortest distance to A)

S TO A TO D

FROM B

5 – 1

4 = 4 (shortest distance to B)

S TO B TO D

Vertex D

Incoming edges: (B) and (A)

From (A) : (5 - 3 = 2) = 2

(Shortest distance to (**A**)) TRUE

(S) TO (A) TO (D)

From (B): (5 - 1= 4) = 4

(Shortest distance to (**B**)) TRUE

(S) → (B)→(D)

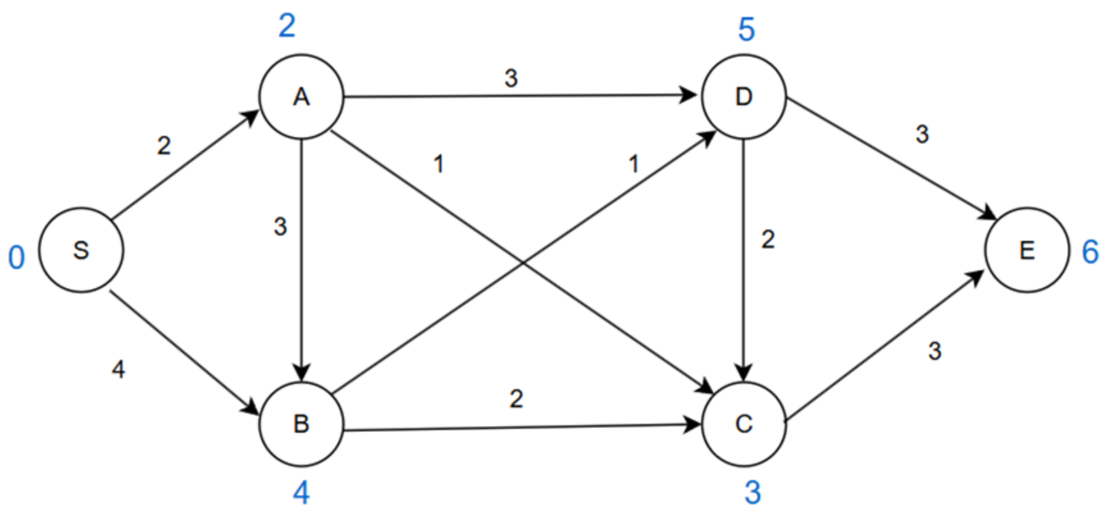
Vertex A

Incoming edges: (S)

From (S):   (2 - 2 = 0) = 0

(Shortest distance to (**S**)) TRUE

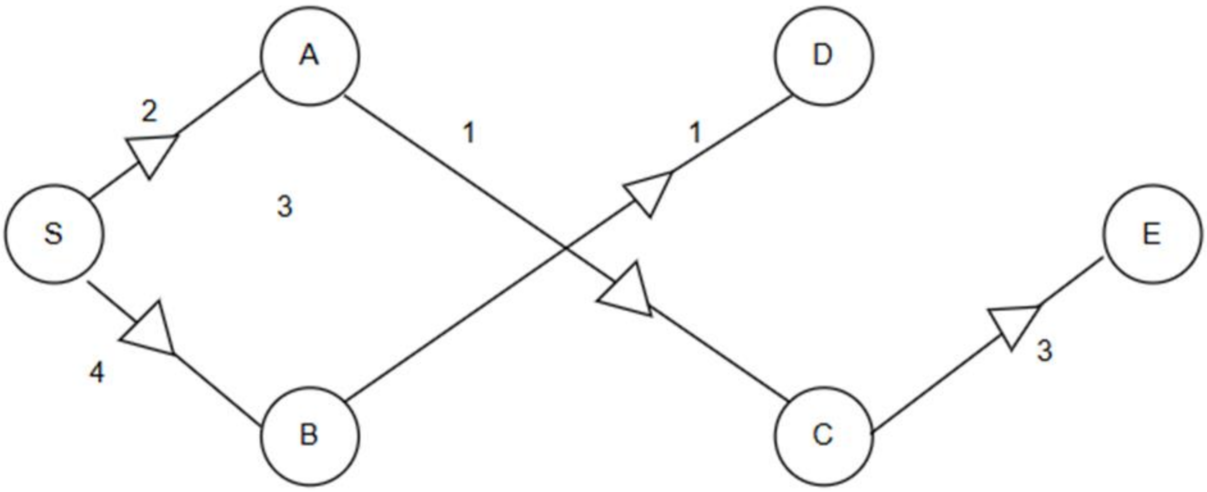
(S) → (A)



Vertex shortest path and shortest distance

Vertex	Shortest Distance	Shortest Path
S	0	-
A	2	S→A
B	4	S→B
C	3	S→A→C
D	5	S→A→D, S→B→D

E	6	S→A→C→E



Vertices = 6

Edges = 5

Weight = 4+2+1+1+3

= 11

Part 3

Question 01

How to find whether a given graph is Eulerian or not?

The difficulty is the same as the next question. "Is it possible to draw a given graph without lifting pencil from the paper and without tracing any of the edges more than once" .

A graph is considered Eulerian if it has an Eulerian Cycle and Semi-Eulerian if it has an Eulerian Path. The issue appears to be comparable to Hamiltonian Path, an NP-complete problem for generic graphs. Fortunately, we can determine if a given graph has an Eulerian

Path or not in polynomial time. In reality, we can locate it in O (V+E) time.

The following are some noteworthy characteristics of undirected graphs with an Eulerian route and cycle. We may use these qualities to determine if a graph is Eulerian.

Eulerian Cycle

An undirected graph has Eulerian cycle if following two conditions are true.

- All vertices with non-zero degree are connected. We don't care about vertices with zero degree because they don't belong to Eulerian Cycle or Path (we only consider all edges).
- All vertices have even degree.

Eulerian Path

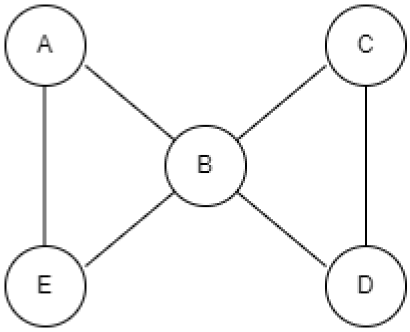
An undirected graph has Eulerian Path if following two conditions are true.

- Same as condition (a) for Eulerian Cycle.
- If zero or two vertices have odd degree and all other vertices have even degree. Note that only one vertex with odd degree is not possible in an undirected graph (sum of all degrees is always even in an undirected graph)

Note that a graph with no edges is considered Eulerian because there are no edges to traverse.

Assess whether the following undirected graphs have an Eulerian and/or a Hamiltonian cycle.

i.



Deg (A) =2

Deg (B) =4

Deg (C) =2

Deg (D) =2

Deg (E) =2

All the vertexes have even degrees; therefore, the graph has an Eularian tour

A→B→C→D→B→E→A= Eualr tour

Checking Hamiltonian cycle:

Dirac's theorem

Deg(v) >= n / 2

N = 5

Deg (A) = 2

2 >= 5 / 2

2 >= 2.5

THEOREM FAILS

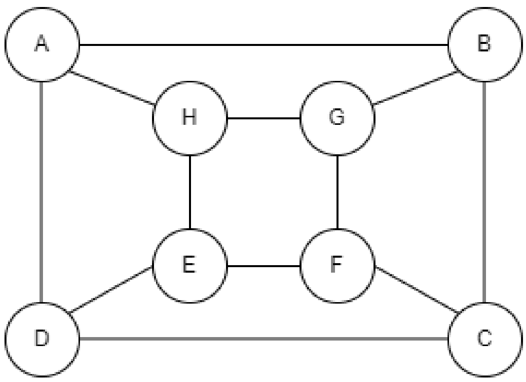
Deg (B) = 4

4 >= 5 / 2

4 >= 2.5 **Satisfies**

Dirac's theorem fails there is no Hamiltanian cycle Graph is not Hamiltanian.

ii.



Deg (A) =3

Deg (B) =3

Deg(C) =3

Deg (D) =3

Deg (E) =3

Deg (F) =3

Deg (G) =3

Deg (H) =3

Degree of all vertices are odd then theorem fails. There is no eular tour graph is not Eulerian.

Hamiltonian cycle

In this graph Dirac's theorem and ore's theorem failed still we can find the Hamiltonian cycle

Ores theorem

A, C

Deg (A) + Deg (C) >= n

3 + 3 >= 8

6 >= 8 fails

Dirac’s theorem a

N=8 degree (v)>=8/2

Deg(A) = 3 >= 8/2

3> = 4---→failed

Deg(B) = 3 >= 4---→failed

Deg(C) = 3 >= 4---→failed

Deg(D) = 3 >= 4---→failed

Deg(E) = 3 >= 4---→failed

Deg(F) = 3 >= 4---→failed

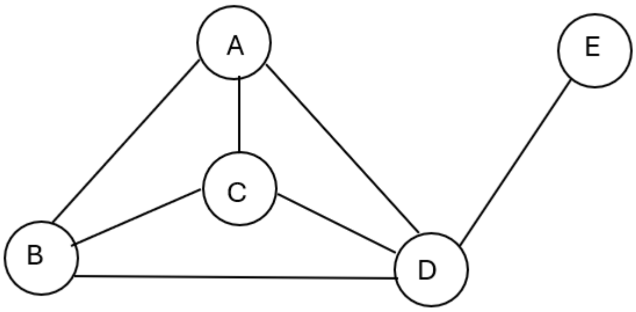
Deg(G) = 3 >= 4---→failed

Deg(H) = 3 >= 4---→failed

And then graph is Hamiltonian graph

A-D-C-B-G-F-E-H-A

III.



Deg (A) =3

Deg (B) =3

Deg (C) =3

Deg (D) =4

Deg (E) =1

Degree of A, B, C, E vertices are odd then theorem fails. There is no eular tour graph is not Eulerian.

Hamiltonian cycle

Dirac’s theorem

N = 5

Deg (A) >= N / 2

3 >= 5 / 2

3 >= 2.5 (pass)

Deg (E) >= n / 2

1 >= 5 / 2

1 >= 2.5 (fail)

Ore’s theorem

Deg (A) + Deg (C) >= n

3+1 >= 5

4 >= 5//--→failed

Dirac’s theorem fail, if we consider the vertex D when we try to find the Hamiltonian cycle the vertex D must be repeated then there is no Hamiltonian cycle. Graph is not Hamiltonian graph

The Dirac theorem does not hold, there exists no Hamiltonian cycle… consequently, and the graph is not Hamiltonian.

Part 4

Question 01

Construct a proof of the five-color theorem for every planar graph.

5-color theorem – Every planar graph is 5-colorable.

Proof:

Proof by contradiction.

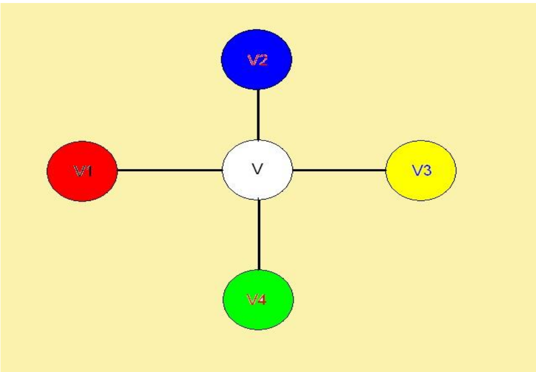
Let G be the smallest planar graph (in terms of number of vertices) that cannot be coloured with five colours.

Let v be a vertex in G that has the maximum degree. We know that deg(v) < 6 (from the corollary to Euler's formula).

Case #1: deg(v) ≤ 4. G-v can be coloured with five colours.

There are at most 4 colours that have been used on the neighbours of v. There is at least one colour then available for v.

So G can be coloured with five colours, a contradiction.

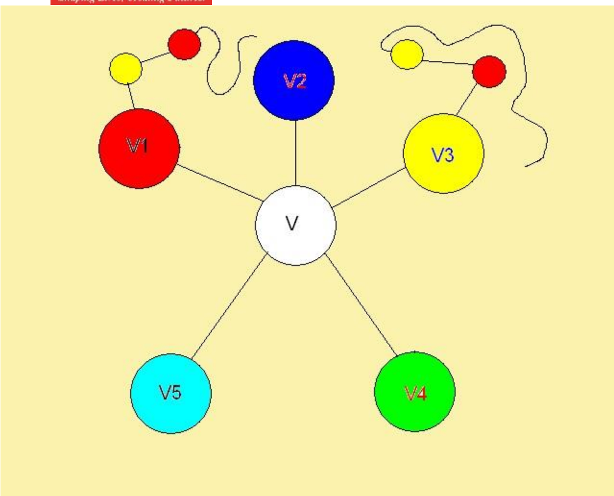


Case #2: deg(v) = 5. G-v can be coloured with 5 colours.

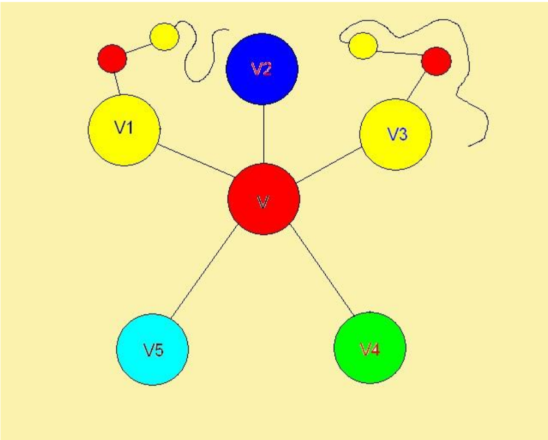
If two of the neighbours of v are coloured with the same colour, then there is a colour available for v.

So we may assume that all the vertices that are adjacent to v are coloured with colours 1, 2,3,4,5 in the clockwise order.

Consider all the vertices being coloured with colours 1 and 3 (and all the edges among them).

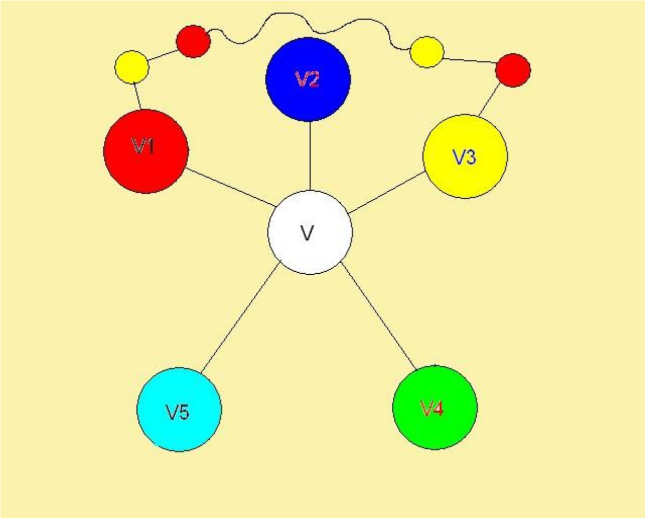


If this subgraph G is disconnected and v_1 and v_3 are in different components, then we can switch the colours 1 and 3 in the component with v_1.

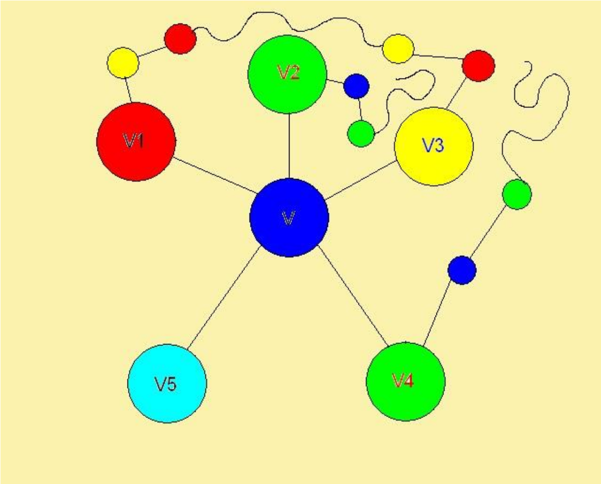


This will still be a 5-coloring of G-v. Furthermore, v_1 is coloured with colour 3 in this new 5-coloring and v_3 is still coloured with colour 3. Colour 1 would be available for v, a contradiction.

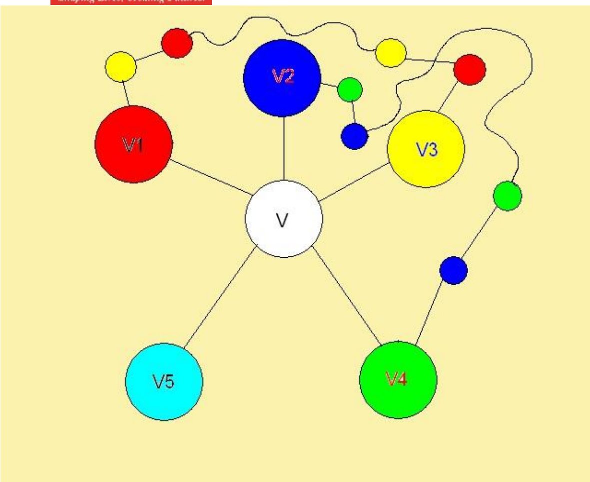
Therefore v_1 and v_3 must be in the same component in that subgraph, i.e. there is a path from v_1 to v_3 such that every vertex on this path is coloured with either colour 1 or color 3.



Now, consider all the vertices being coloured with colours 2 and 4 (and all the edges among them). If v_2 and v_4 don't lie of the same connected component then we can interchange the colors in the chain starting at v_2 and use left over colour for v.



If they do lie on the same connected component then there is a path from v_2 to v_4 such that every vertex on that path has either colour 2 or colour 4.



This means that there must be two edges that cross each other. This contradicts the planarity of the graph and hence concludes the proof. (5-color Theorem proof, 2025) (cgm, n.d.)

Activity 03

Part 1

Question 01

1. Two real-world binary problems in different fields with applications of Boolean Algebra is given below.

 i. **Office Building Security System**

 In this example, the security system for an office building controls access based on two factors:

 - Whether it's after hours (**A**)
 - Whether the employee has a valid access card (**B**)

 The security system should trigger an alarm (**Z**) if someone tries to access the building after hours without a valid access card

Boolean Expression

- A = 1 if it's after hours, 0 if not.
- B = 1 if the employee has a valid access card, 0 if not.
- Z = 1 if the alarm should be triggered, 0 if not.

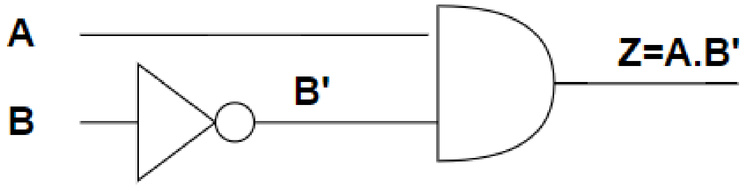
The alarm should trigger only if it's after hours and the employee does not have a valid access card

$Z=A.\bar{B}$

Truth table

A	**B**	**$\bar{B}$**	**$Z= A.\bar{B}$**
0	0	1	**0**
0	1	0	**0**
1	0	1	**1**
1	1	0	**0**

Logic Diagram



ii. **Voting System**

A voting system decides whether to accept a proposal based on votes from three committee members.

- Whether member M votes in favor (**M**)
- Whether member W votes in favor (**W**)
- Whether member D votes in favor (**D**)

The proposal is accepted (A) if any two or more members vote in favor.

Boolean Expression

- **M** = 1 if member M votes in favor, 0 if not.
- **W** = 1 if member W votes in favor, 0 if not.
- **D** = 1 if member D votes in favor, 0 if not.
- **A** = 1 if the proposal is accepted, 0 if not.

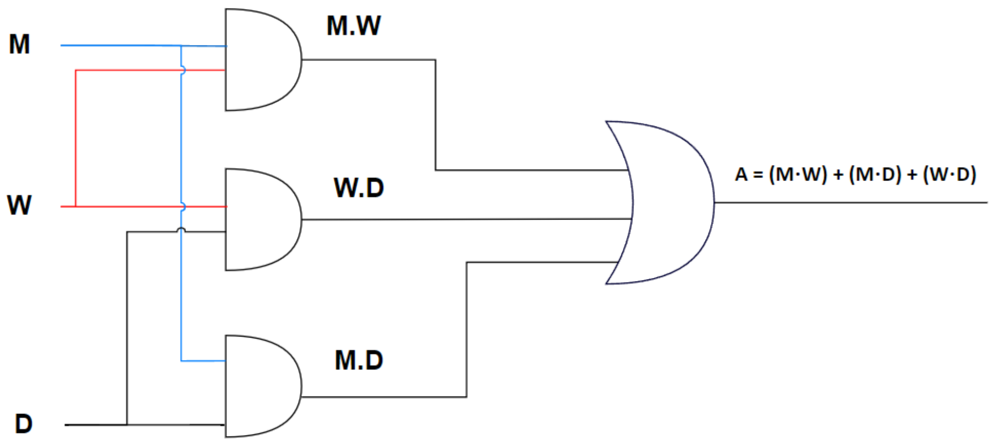
The proposal is accepted if any two or more members vote in favor

A= (M·W) + (M·D) + (W·D)

Truth table

M	W	D	M·W	M·D	W·D	A = (M·W) + (M·D) + (W·D)
0	0	0	0	0	0	**0**
0	0	1	0	0	0	**0**
0	1	0	0	0	0	**0**
0	1	1	0	0	1	**1**
1	0	0	0	0	0	**0**
1	0	1	0	1	0	**1**
1	1	0	1	0	0	**1**
1	1	1	1	1	1	**1**

Logic Diagram



Part 02

Question 01

i. If the room temperature is above a certain threshold and the fan is not turned on, and someone is present in the room, then an alert should be activated.

- The room temperature is high (above a defined threshold),
- The fan is off, and
- There is someone present in the room.

This ensures that discomfort or overheating does not go unnoticed when ventilation is not active and a person is present.

Inputs

- The room temperature is above **(A)**
- The fan is turned on **(B)**
- present in the room **(C)**
- The fan is not turned off **(($\bar{B}$)**

Outputs

Y = then an alert should be activated.

$Y = A.\bar{B}.C$

Boolean Logic:

To activate the alert, all three conditions must be true:

- A is true (temperature is high),

- B is false (fan is off, i.e., $\bar{B}$ is 1),
- C is true (someone is present).

A	B	C	$\bar{B}$	Y
1	1	1	0	0
1	1	0	0	0
1	**0**	**1**	**1**	**1**
1	0	0	1	0
0	1	1	0	0
0	1	0	0	0
0	0	1	1	0
0	0	0	1	0

ii. If there is a power outage and you don't have a flashlight, then you will be in the dark.

You will only be in the dark if both of the following are true:

- There is a power outage, and
- You do not have a flashlight.

- power outage **(A)**
- have a flashlight **(B)**
- don't have a flashlight **($\bar{B}$)**

Outputs

will be in the dark **(Y)**

$Y = A.\bar{B}$

A	B	$\bar{B}$	Y
1	1	0	0
1	0	1	1
0	1	0	0
0	0	1	0

Question 02

i. 1. $\bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC + \bar{A}B\bar{C}$

Y = $\bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC + \bar{A}B\bar{C}$

A	**B**	**C**	$\bar{A}$	$\bar{B}$	$\bar{C}$	$\bar{A}\bar{B}C$	$A\bar{B}\bar{C}$	**ABC**	$\bar{A}B\bar{C}$	Y
0	0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	1	0	0	0	1
0	1	0	1	0	1	0	0	0	1	1
0	1	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	0	1	0	0	1
1	0	1	0	1	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	1	0	1

ii. $(A + \bar{B} + C)(A + B + C)(\bar{A} + B + \bar{C})$

$Y = (A + \bar{B} + C)(A + B + C)(\bar{A} + B + \bar{C})$

A	**B**	**C**	$\bar{A}$	$\bar{B}$	$\bar{C}$	$A + \bar{B} + C$	**A+ B+C**	$\bar{A}$+B+$\bar{C}$	Y

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

iii. $\bar{A}\bar{B}C + \bar{A}BC + A\bar{B}$CD

A	**B**	**C**	**D**	**$\bar{A}$**	**$\bar{B}$**	**$\bar{A}\bar{B}C$**	**$\bar{A}BC$**	**$A\bar{B}CD$**	**X**
0	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**1**
0	**0**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**1**
0	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**1**
0	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**
1	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
1	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**

1	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
1	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**
1	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
1	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
1	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**

Part 03

Question 01

Rules of Boolean algebra

- **Commutative laws**

  The order of the variables does not affect the outcome of the operation.

  For example**, A + B = B + A and A * B = B * A.**

- **Associative laws**

  The grouping of the variables does not affect the outcome of the operation.

  For example, **(A + B) + C = A + (B + C) and (A * B) * C = A * (B * C).**

- **Distributive laws**

  The product of a sum is equal to the sum of the products.

  For example, **A * (B + C) = AB + AC and (A + B) * (C + D) = AC + AD + BC + BD.**

- **Identity laws**

The binary values 0 and 1 have special properties in Boolean algebra.

For example, **A + 0 = A and A * 1 = A.**

- **Complement laws.**

  Each variable has a complement which is denoted by a bar over the variable.

  For example, **A + A' = 1 and A * A' = 0.**

- **Idempotent laws**

  A variable added or multiplied by itself is equal to itself.

  For example**, A + A = A and A * A = A.**

- **Annihilation laws**

  A variable multiplied by 0 is equal to 0 and a variable added to 1 is equal to 1.

  For example**, A * 0 = 0 and A + 1 = 1.**

- **De Morgan's laws.**

  The complement of a sum is equal to the product of the complements and the complement of a product is equal to the sum of the complements.

  For example, **(A + B)' = A' * B' and (A * B)' = A' + B'**.

Simplify the following Boolean expressions using algebraic methods.

i. $\boldsymbol{A}(\boldsymbol{A}+\boldsymbol{B})+\boldsymbol{B}(\boldsymbol{B}+\boldsymbol{C})+\boldsymbol{C}(\boldsymbol{C}+\boldsymbol{A})$

= AA+AB+BB+BC+CC+CA: Distributive law

= A+AB+B+BC+C+CA: Impotency law

= A (1+B) +B(1+C)+C(1+A): Distributive law

= **A+B+C**

ii. $\boldsymbol{\overline{A}BC+A\overline{B}\overline{C}+\overline{A}\overline{B}\overline{C}+A\overline{B}C+ABC}$

$\overline{A}BC+A\overline{B}\overline{C}+\overline{A}\overline{B}\overline{C}+A\overline{B}C+ABC$

$BC(\overline{A} + A) + A\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C} + A\overline{B}C$ (Distributive Law)

$BC1 + A\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C} + A\overline{B}C$ (Complement Law)

$BC + A\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C} + A\overline{B}C$(Identity Law)

$BC + \overline{B}\overline{C}(A + \overline{A}) + A\overline{B}C$ (Distributive Law)

$BC + \overline{B}\overline{C}1 + A\overline{B}C$ (Complement Law)

$BC + \overline{B}\overline{C} + A\overline{B}C$ (Identity Law)

$\overline{B}\overline{C} + C(A\overline{B} + B)$ (Distributive Law)

$\overline{B}\overline{C} + C(A + B)$ (Absorption Law)

$\overline{B}\overline{C} + CA + CB$ (Distribution)

iii. $(A + \overline{B} + \overline{C})(A + \overline{B} + C)(A + B + \overline{C})$

$(A + \overline{B} + C)\ (A + B + \overline{C})\ A + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{B} + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{C}$

(Distribution law)

$(A + B + \overline{C})\ AA + (A + B + \overline{C})\ A\overline{B} + (A + B + \overline{C})\ AC + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{B} +$

$(A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{C}$ (Distribution law)

$(A + B + \overline{C})\ A + (A + B + \overline{C})\ A\overline{B} + (A + B + \overline{C})\ AC + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{B} +$

$(A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{C}$ (Idempotent Law)

$(A + B + \overline{C})\ A + (A + B + \overline{C})\ AC + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{B} + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{C}$

(Absorption Law)

$(A + B + \overline{C})\ A + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{B} + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{C}$

(Absorption Law)

$AA + AB + A\overline{C}\ (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{B} + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{C}$

(Distribution Law)

$A + AB + A\overline{C}\ (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{B} + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{C}$

(Idempotent Law)

$A + A\overline{C}\ (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{B} + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{C}$

(Absorption Law)

$A + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{B} + (A + \overline{B} + C)\ (A + B + \overline{C})\ \overline{C}$

(Absorption Law)

$A + (A + B + \overline{C})\,\overline{B}A + (A + B + \overline{C})\,\overline{B}\overline{B} + (A + B + \overline{C})\,\overline{B}C + (A + \overline{B} + C)\,(A + B + \overline{C})\,\overline{C}$

(Distribution Law)

$A + (A + B + \overline{C})\,\overline{B}A + (A + B + \overline{C})\,\overline{B}\,(A + B + \overline{C})\,\overline{B}C + (A + \overline{B} + C)\,(A + B + \overline{C})\,\overline{C}$

(Idempotent Law)

$A + (A + B + \overline{C})\,\overline{B} + (A + B + \overline{C})\,\overline{B}C + (A + \overline{B} + C)\,(A + B + \overline{C})\,\overline{C}$

(Absorption Law)

$A + (A + B + \overline{C})\,\overline{B} + (A + \overline{B} + C)\,(A + B + \overline{C})\,\overline{C}$

(Absorption Law)

$A + \overline{B}A + \overline{B}B + \overline{B}\overline{C} + (A + \overline{B} + C)\,(A + B + \overline{C})\,\overline{C}$

(Distribution Law)

$A + \overline{B}A + 0 + \overline{B}\overline{C} + (A + \overline{B} + C)\,(A + B + \overline{C})\,\overline{C}$

(Complement Law)

$A + \overline{B}A + \overline{B}\overline{C} + (A + \overline{B} + C)\,(A + B + \overline{C})\,\overline{C}$

(Identity Law)

$A + \overline{B}\overline{C} + (A + \overline{B} + C)\,(A + B + \overline{C})\,\overline{C}$

(Absorption Law)

$A + \overline{B}\overline{C} + (A + B + \overline{C})\overline{C}A + (A + B + \overline{C})\overline{C}\overline{B} + (A + B + \overline{C})\,\overline{C}C$

(Distribution Law)

$A + \overline{B}\overline{C} + (A + B + \overline{C})\overline{C}A + (A + B + \overline{C})\overline{C}\overline{B} + 0$

(Complement Law)

$A + \overline{B}\overline{C} + (A + B + \overline{C})\overline{C}A + (A + B + \overline{C})\overline{C}\overline{B}$

(Identity Law)

$A + \overline{B}\overline{C} + (A + B + \overline{C})\overline{C}\overline{B}$

(Absorption Law)

$A + \overline{B}\overline{C}$

iv. $\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

$BC(A + \bar{A}) + A\bar{B}C + AB\bar{C}$ (Distributive Law)

$BC1 + A\bar{B}C + AB\bar{C}$ (Complement Law)

$BC + A\bar{B}C + AB\bar{C}$ (Identity Law)

$C(A\bar{B} + B) + AB\bar{C}$ (Distributive Law)

$C(A + B) + AB\bar{C}$ (Absorption Law)

$CA + CB + AB\bar{C}$ (Distribution Law)

$CB + A(B\bar{C} + C)$ (Distributive Law)

$CB + A(B + C)$ (Absorption Law)

<u>CB+AB+AC</u>

.

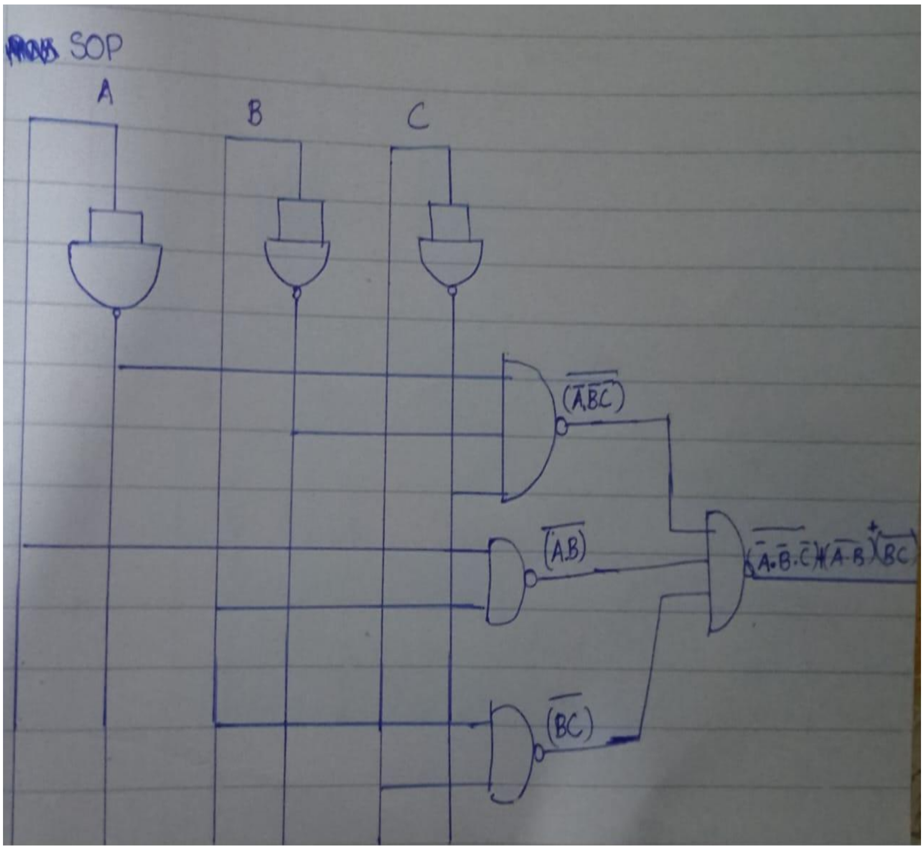
Part 04

Question 01

(a)

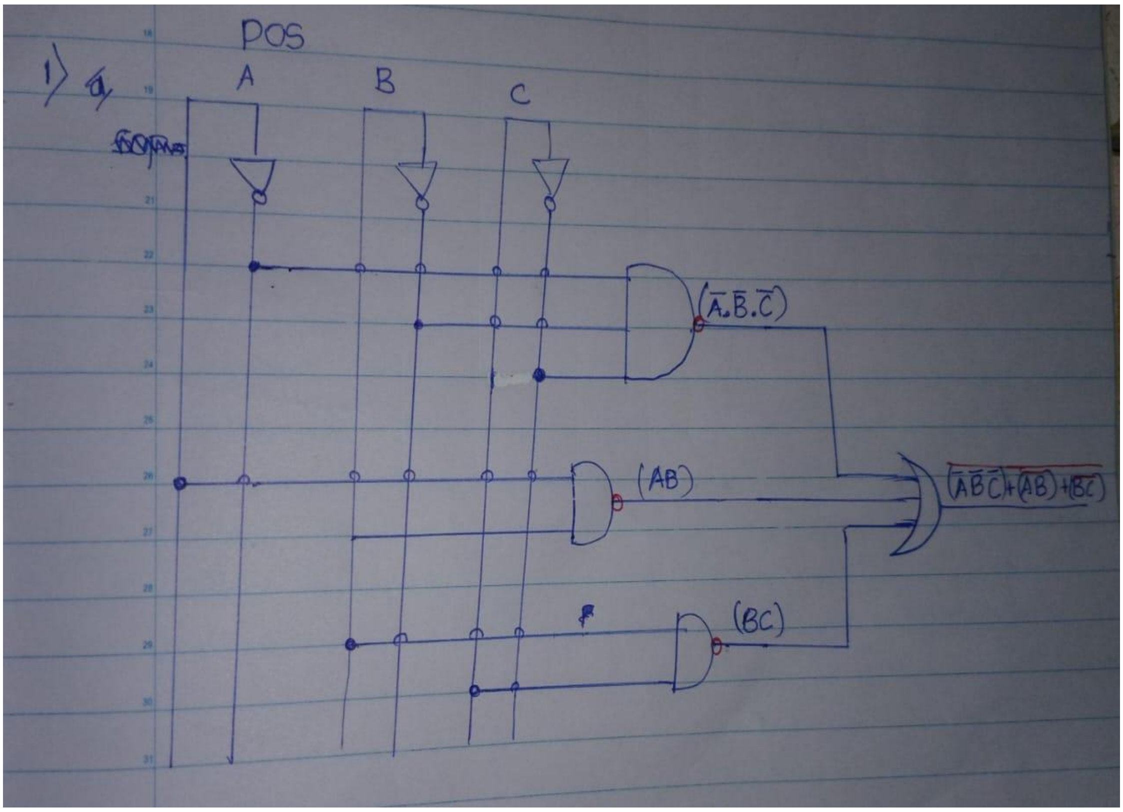
SOP= $\overline{A}\overline{B}\overline{C} + AB + BC$

AB/C	0	1
00	1	0
01	0	1
11	1	1
10	0	0



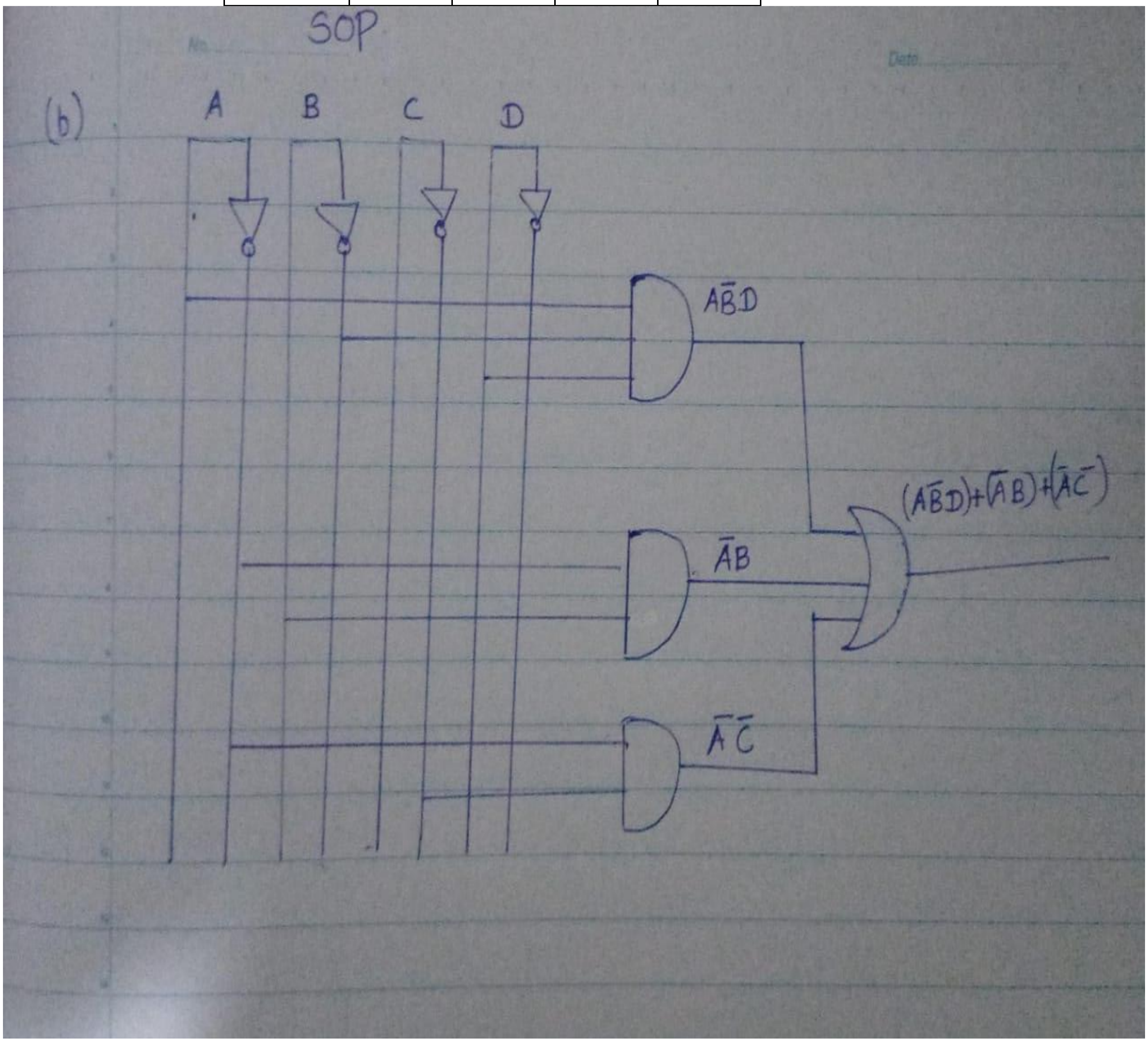
POS= $(\bar{\mathbf{A}}+\boldsymbol{B})(\mathbf{A}+\bar{\mathbf{B}}+\boldsymbol{C})(\mathbf{B}+\bar{\mathbf{C}})$

AB/C	0	1
00	1	0
01	0	1
11	1	1
10	0	0



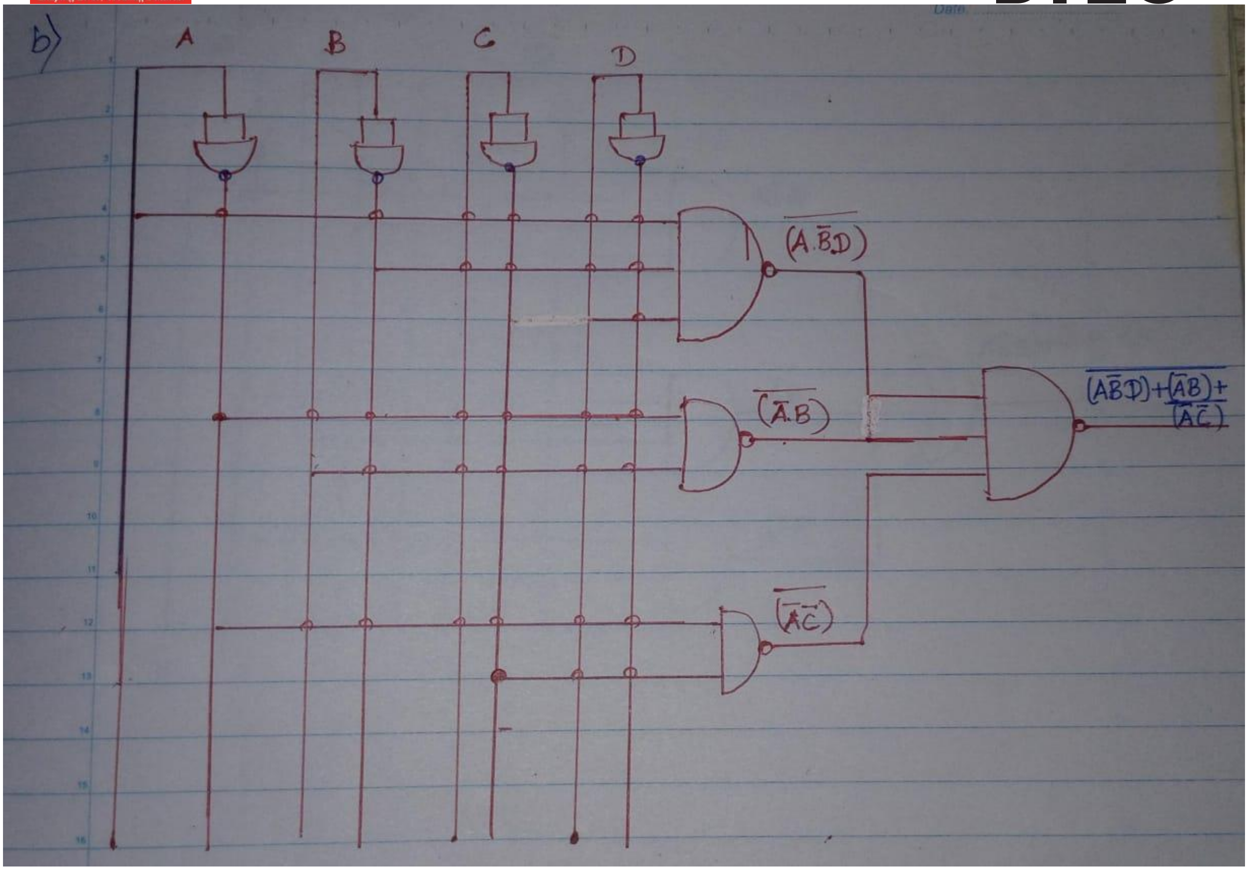
(b) SOP= $\mathbf{\bar{A}B + \bar{A}\bar{C} + A\bar{B}D}$

AB/CD	00	01	11	10
00	1	1	0	0
01	1	1	1	1
11	0	0	0	0
10	0	1	1	0



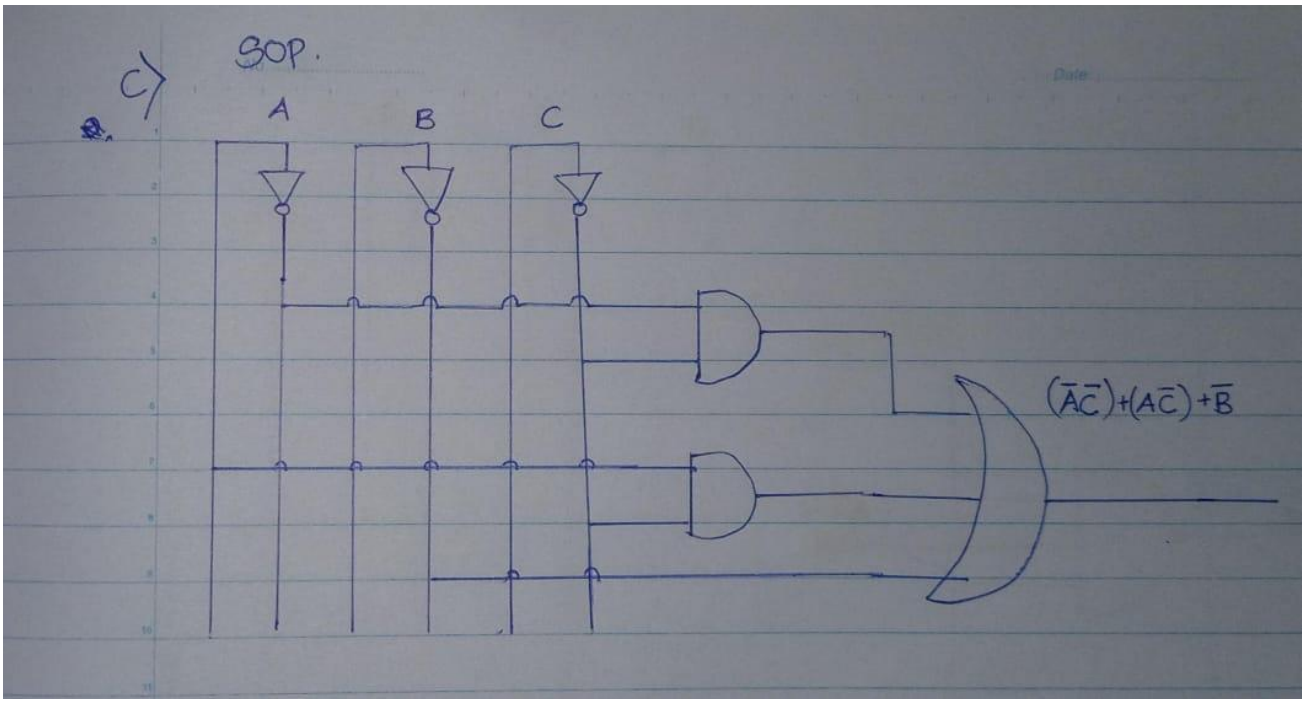
POS= $(\bar{\mathbf{A}} + \bar{\mathbf{B}})(\bar{\mathbf{A}} + \mathbf{D})(\mathbf{A} + \boldsymbol{B} + \bar{\mathbf{C}})$

AB/CD	00	01	11	10
00	1	1	0	0
01	1	1	1	1
11	0	0	0	0
10	0	1	1	0


b)
A
B
C
D
$\overline{(A.\bar{B}D)}$
$\overline{(\bar{A}.B)}$
$\overline{(\bar{A}\bar{C})}$
$\overline{(A\bar{B}D)+(\bar{A}B)+(\bar{A}\bar{C})}$

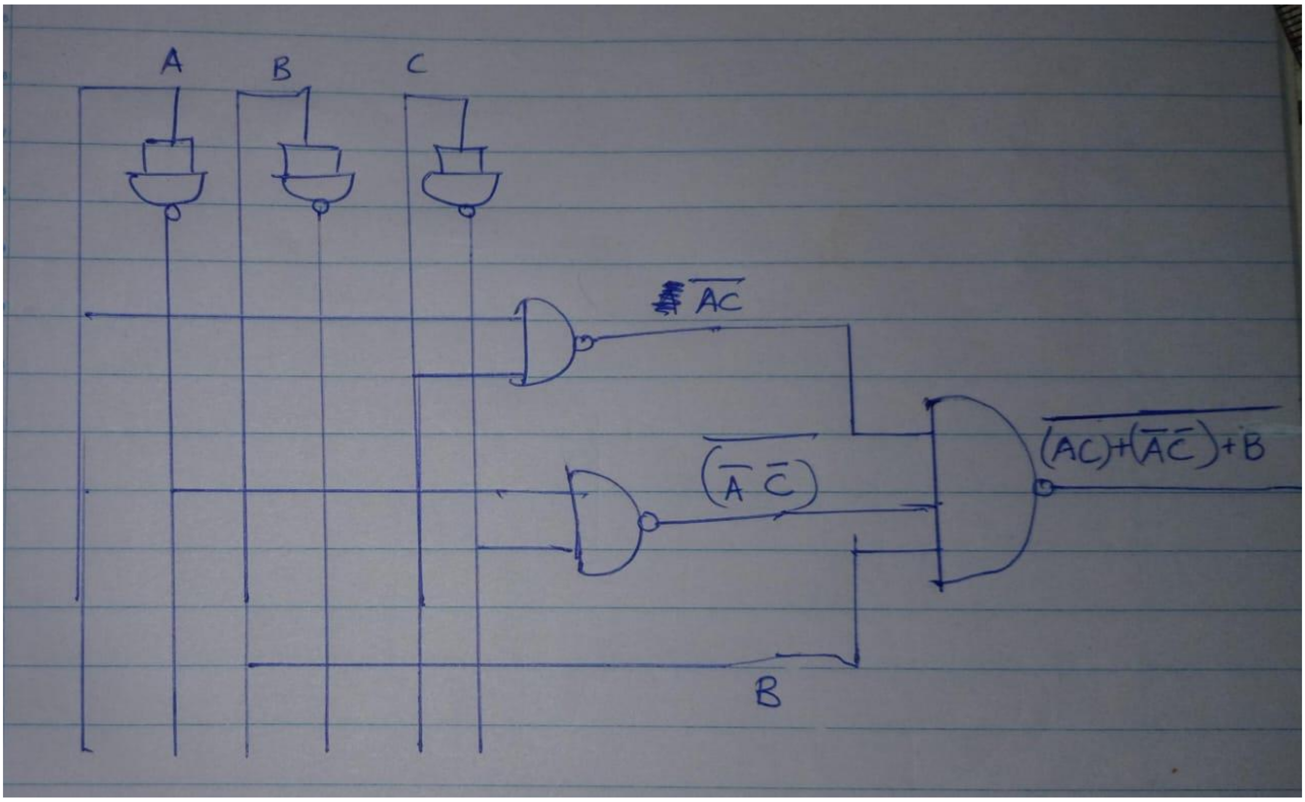
c) **SOP = $\bar{B} + \bar{A}\bar{C} + AC$**

AB/C	0	1
00	1	1
01	1	0
11	0	1
10	1	1



POS = $(A + \bar{B} + \bar{C})(\bar{A} + \bar{B} + C)$

AB/C	0	1
00	1	1
01	1	0
11	0	1
10	1	1



Activity 04

Part 1

1.

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

Binary operation.

Just as we get a number when two numbers are either added or subtracted or multiplied or are divided. The binary operations associate any two elements of a set. The resultant of the two are in the same set. Binary operations on a set are calculations that combine two elements of the set (called operands) to produce another element of the same set.

The binary operations * on a non-empty set A are functions from A × A to A. The binary operation, *: A × A → A. It is an operation of two elements of the set whose domains and co-domain are in the same set. (toppr, 2025)

Basic Binary Operations

Operation	**Symbol**	**Representation**
Addition	+	**A+B**
Subtraction	-	**A-B**

Multiplication	×	**A×B**
Division	÷	**A ÷B**

- **Commutative property**

The commutative property refers to the property of a binary operation where the order of the elements being operated does not affect the result. In other words, if a binary operation is commutative, swapping the order of the elements being operated will not change the outcome of the operation.

To explain the commutative property using the variables 'a' and 'b', let's consider an example **with addition as the binary operation.**

Commutative property of addition.

For any elements 'a' and 'b', the commutative property states that a + b = b + a. This means that if you add 'a' and 'b' together, the result will be the same regardless of the order in which you add them. For example: If a = 3 and b = 5, a + b = 3 + 5 = 8.

By applying the commutative property, we can swap the order of 'a' and 'b' in the addition operation b + a = 5 + 3 = 8.

As you can see, the result remains the same (8) regardless of whether we add 'a' to 'b' or 'b' to 'a'. This is because addition is a commutative operation.

However, it's important to note that not all binary operations are commutative. For example, subtraction is not commutative.

If we have a = 3 and b = 5

a - b = 3 - 5 = -2,

but swapping the order.

b - a = 5 - 3 = 2.

In this case, the result is different depending on the order of the subtraction. Therefore, subtraction does not possess the commutative property.

- **Associative Property**

The associative property is a property of binary operations that states that the grouping of elements being operated does not affect the final result. In other words, when performing an operation on three elements, it doesn't matter which two elements are combined first.

Let's consider the binary operation denoted by "*". Using the variables x, y, and z, the associative property can be expressed as.

$(x * y) * z = x * (y * z)$

$(1 \times 2) \times 3 = 1\times(2\times3)$

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

$(1 – 2) – 3! = 1-(2-3)$

$(1/2)/3! = 1/(2/3)$

$(2^3)^2! = 2(3^2)$

This property implies that no matter how we group the elements, the result will be the same. The operation "*", which can represent addition, multiplication, or any other binary operation, exhibits the associative property if the equation holds true for all possible values of x, y, and z.

To illustrate this further, let's consider an example using addition.

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

Regardless of the specific values of x, y, and z, as long as addition is the binary operation, the sum of the left-hand side and the sum of the right-hand side will always be equal.

For instance, let's say x = 2, y = 3, and z = 4. Applying the associative property.

(2 + 3) + 4 = 5 + 4 = 9

2 + (3 + 4) = 2 + 7 = 9

In this example, the associative property of addition holds true.

The associative property is particularly useful when dealing with multiple operations or when rearranging expressions. It allows us to simplify calculations by changing the grouping of elements without changing the final result.

- **Identity Property**

The identity property, also known as the identity element property, refers to the property of a binary operation where there exists a specific element that, when combined with any other element under the operation, leaves the other element unchanged.

Let's use the variables x and y to illustrate the identity property. Suppose we have a binary operation denoted by "*".

The identity property states that there exists an identity element, denoted as "e," such that for any element x in the set, the following holds:

x * e = e * x = x

In other words, when we perform the operation "*" between any element x and the identity element e, the result is always x. Similarly, if we interchange the order and perform the operation between e and x, the result is still x.

Let's consider an example to understand this property better. Suppose we have a set of integers and the binary operation is addition (+). In this case, the identity element is 0.

Using the identity property, we have:

+ 0 = 0 + x = x

For any integer value of x, adding 0 to it will always give us the original value of x.

Similarly, if we swap the order and add x to 0, we again get x.

For instance:

If x = 5, then 5 + 0 = 0 + 5 = 5.

If x = -2, then -2 + 0 = 0 + (-2) = -2.

The identity property guarantees that there is a specific element within the set that does not affect other elements when combined under the binary operation. This property is essential in various mathematical structures and computations, providing a foundation for algebraic manipulations and calculations.

- **Inverse binary operation**

Inverse binary operation refers to an operation that "undoes" or reverses the effects of a given binary operation. In the context of two elements x and y, an inverse binary operation is an operation that, when applied to y and the result obtained by applying the original binary operation to x and y, yields x again.

More formally, let's assume we have a binary operation denoted by "". If we have elements x and y, an inverse binary operation denoted by "^-1" would satisfy the following condition:

$* (*^{-1}(x, y)) = x$

In other words, applying the inverse operation to y and the result obtained by applying the original operation to x and y should give us back x.

For example, if we consider the addition operation as the binary operation, the inverse binary operation would be subtraction. If we have two numbers x and y, and we add them together (x + y), the inverse operation of addition is subtraction. So, if we subtract y from the result (x + y) using the inverse operation, we should get back the original number x:

$(x + y) - y = x$

This demonstrates the inverse relationship between addition and subtraction as binary operations.

Similarly, in multiplication, the inverse operation is division. If we multiply x and y (x * y), dividing the result by y using the inverse operation should give us back x:

$(x * y) / y = x$

It's important to note that not all binary operations have inverse operations. In order for an operation to have an inverse, it needs to satisfy certain properties, such as being associative, having an identity element, and every element having an inverse.

<u>Part 2</u>

1.

Order 01

*	e
e	e

Ex :-

*	0
0	0

*	1
1	1

Order 02

*	e	a
e	a	e
a	e	a

Order 03

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

Order 04

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

Question 02

State the relation between the order of a group and the number of binary operations that can be defined on that set.

Let the order of a group is n then. The number of possible binary is n^{n^2}.

Ex:

Let n=2

Then the number of possible binary

Operations = n^{n^2}. = 2^{22}= 2^4= 16

Let n = 3

Then, the number of possible binary operation

$3^{3\,2} = 3^9 = 19683$

i. **How many binary operations can be defined on a set with 3 elements**

A **binary operation** on a set is a rule that combines **any two elements** of the set to produce **another element from the same set**.

n = {a, b, c}

This set has **3 elements**.

Let n = 3

$3 \times 3 = 9$ input pairs

Then, the number of possible binary operation

n^(n^2) = 3^ (3^2)

$3^{9=}$ 19,683

Therefore, there are 19,683 possible binary operations in set with 3 elements.

Question 03

State the Lagrange's theorem of group theory.

Lagrange theorem is one of the central theorems of abstract algebra. It states that in group theory, for any finite group say G, the order of subgroup H of group G divides the order of G. The order of the group represents the number of elements. This theorem was given by Joseph-Louis Lagrange. (BYJUS, 2025)

i. **For a subgroup H of a group G, prove the Lagrange's theorem.**

Before proving Lagrange's Theorem, proving three lemmas.

Lemma If H is a finite subgroup of a group G and H contains n elements then any right coset of H contains n elements.

Proof For any element x of G, Hx = {h • x | h is in H} defines a right coset of H. By the cancellation law each h in H will give a different product when multiplied on the left onto x. Thus each element of H will create a corresponding unique element of Hx.

Thus Hx will have the same number of elements as H.

Lemma Two right cosets of a subgroup H of a group G are either identical or disjoint.

Proof Suppose Hx and Hy have an element in common. Then for some elements h1 and h2 of H

h1 • x = h2 • y

This implies that x = h1-1 • h2 • y. Since H is closed this means there is some element h3 (which equals h1-1 • h2) of H such that x = h3 • y. This means that every element of Hx can be written as an element of Hy by the correspondence

h • x = (h • h3) • y

for every h in H. We have shown that if Hx and Hy have a single element in common then every element of Hx is in Hy. By a symmetrical argument it follows that every element of Hy is in Hx and therefore the "two" cosets must be the same coset. Since every element g of G is in some coset (namely it's in Hg since e, the identity element is in H) the elements of G can be distributed among H and its right cosets without duplication. If k is the number of right cosets and n is the number of elements in each coset then |G| = kn.

Alternate Proof In the last chapter we showed that a • b-1 being an element of H was equivalent to a and b being in the same right coset of H. We can use this Idea establish Lagrange's Theorem.

Define a relation on G with a ~ b if and only if a • b-1 is in H. Lemma: The relation a ~ b is an equivalence relation.

Proof We need to establish the three properties of an equivalence relation -- reflexive, symmetrical and transitive.

(1) Reflexive: Since a • a-1 = e and e is in H it follows that for any a in G

a ~ a

(2) Symmetrical: If a ~ b then a • b-1 is in H. Then the inverse of a • b-1 is in H. But the inverse of a • b-1 is b • a-1 so

b ~ a

(3) Transitive: If a ~ b and b ~ c then both a • b-1 and b • c-1 are in H. Therefore their product (a • b-1) • (b • c-1) is in H. But the product is simply a • c-1. Thus

a ~ c

And we have shown that the relation is an equivalence relation.

It remains to show that the (disjoint) equivalence classes each have as many elements as H.

Lemma The number of elements in each equivalence class is the same as the number of elements in H.

Proof For any a in G the elements of the equivalence class containing a are exactly the solutions of the equation

a • x-1 = h

Where h is any element of H. By the cancellation law each member h of H will give a different solution. Thus the equivalence classes have the same number of elements as H.

One of the imediate results of Lagrange's Theorem is that a group with a prime number of members has no nontrivial subgroups.

Definition if H is a subgroup of G then the number of left cosets of H is called the index of H in G and is symbolized by (G:H). From our development of Lagrange's theorem we know that

|G| = |H| (G:H)

Converse of Lagrange's Theorem One of the most interesting questions in group theory deals with considering the converse of Lagrange's theorem. That is if a number n divides the order of group G does that mean that G must have a subgroup of order n? The answer is no in general but the special cases where it does work out are many and interesting. They are dealt with in detail in the Sylow Theorems which we will treat later. As a tidbit we look at the following

Theorem If the order of a group G is divisible by 2 then G has a subgroup of two elements.

Proof The proof is left as an exercise for the student. Proof. By the Theorem that, "If H is a subgroup of a finite group G, then the right cossets Ha form a partition of G, and by the statement that "Any cosset Ha have the same number of elements" the each cost that partition G has r elements. Therefore, G has "r ×s " elements, and so the order of H orders of H divides the order of G. (Dogfrey, 1999)

iii. Discuss whether a group H with order 5 can be a subgroup of a group with order 20 or not. Clearly state the reasons.

Let H be a subgroup of a group G. According to **Lagrange's Theorem** in group theory, the order (which means the number of elements) of a subgroup H must divide exactly into the order of the whole group G. This means that if the number of elements in H does not divide the number of elements in G, then H cannot be a subgroup of G. In this case, the order of the group H is given as 5, and the order of the main group G is 20.

Now we need to check whether 5 divides 20. If we divide 20 by 5, we get 4, which is a whole number. So, **5 divides 20 exactly** with no remainder. That means a subgroup with 5 elements can exist within a group that has 20 elements. This situation satisfies the condition stated by Lagrange's Theorem.

To understand this better, we can look at all the possible divisors of 20. The positive integers that divide 20 evenly are: 1, 2, 4, 5, 10, and 20. According to group theory, any subgroup of G must have an order equal to one of these divisors. So a subgroup of order 5 is definitely possible.

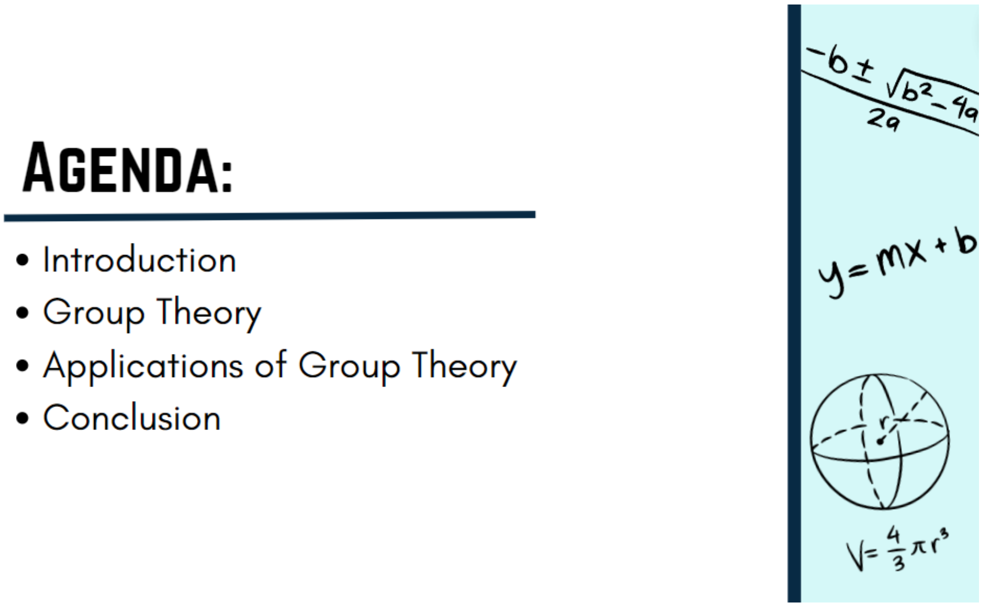
An example of this is the cyclic group of order 20. A cyclic group of order n always has exactly one subgroup for each positive integer that divides n. Since 5 divides 20, a cyclic group of order 20 will certainly have a subgroup of order 5. This subgroup will itself be cyclic and will follow all the rules and properties of a group.

Therefore, it is completely valid and mathematically correct to say that a group of order 20 can have a subgroup of order 5. This conclusion is fully supported by **Lagrange's Theorem**, and it shows that group theory provides clear and reliable rules for how group structures can be formed.

(Godfrey, 1999)

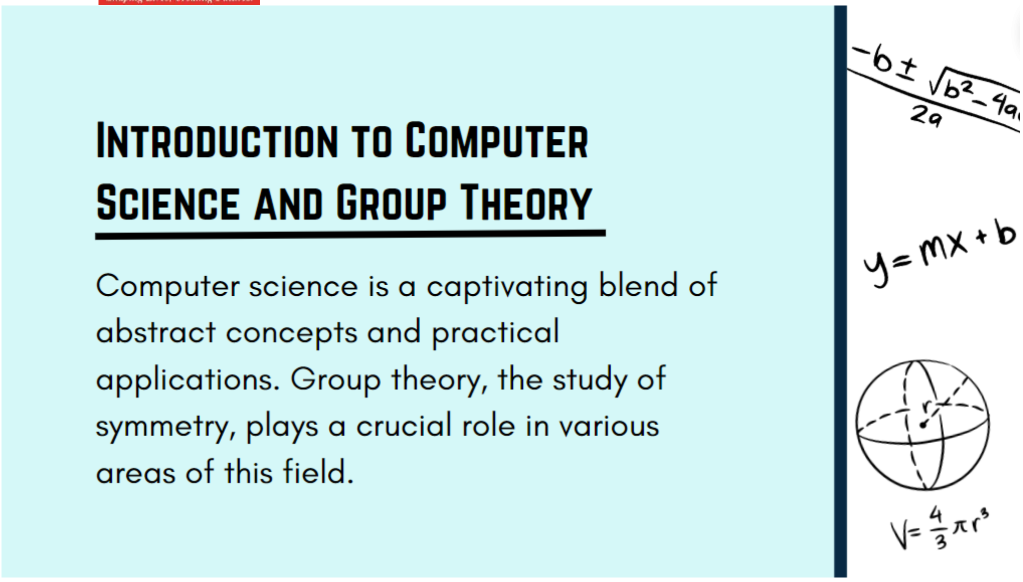
Slide 1

Good [morning/afternoon], everyone. My name is AASHIK from batch 107. Today, we are going to dive into the fascinating world of computer science by exploring group theory and its diverse applications. Group theory is an area of mathematics that deals with symmetry and abstract algebraic structures. I hope this session will be both informative and engaging for all of you



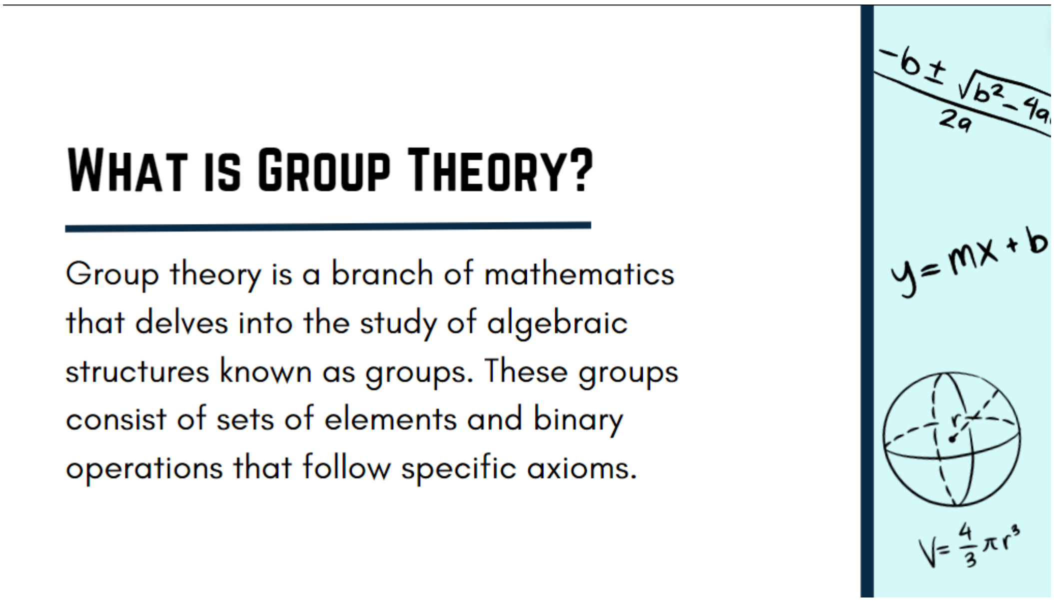
Slide 2

Here's our agenda for today's presentation. We will start with an introduction to computer science and group theory, providing a foundational understanding of the concepts. Next, we will explore what group theory entails. Following that, we will delve into the specific applications of group theory within computer science, focusing on cryptography, error correcting codes, computer graphics, and robotics. Finally, we will conclude with a summary of what we've learned and look forward to the future potential of group theory in emerging fields



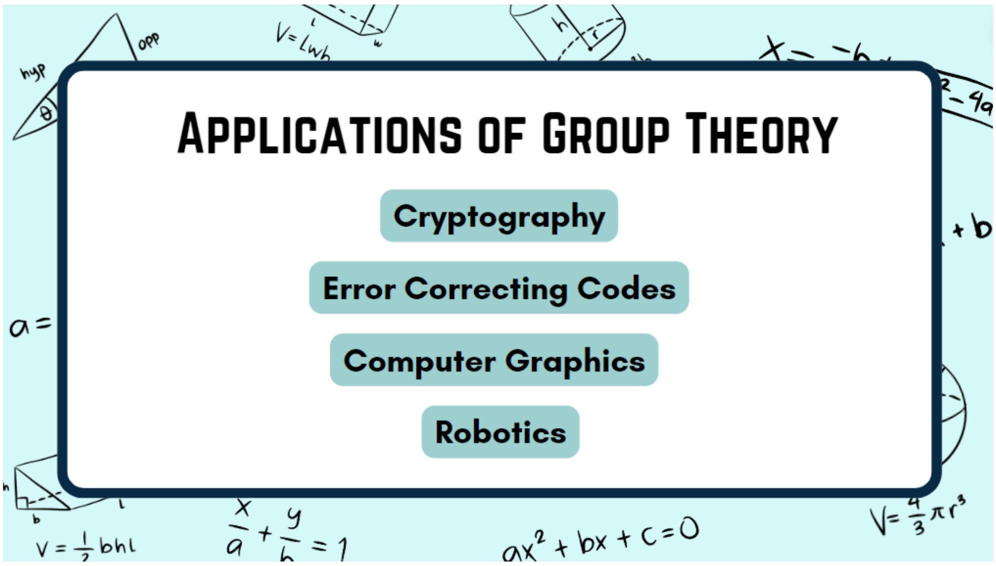
Slide 3

Computer science is a field that blends theoretical foundations with practical applications. One of the theoretical underpinnings of computer science is group theory, which studies symmetry and structure in mathematical terms. Group theory provides a way to analyze and understand the properties of symmetrical objects and systems. This theoretical framework has found numerous applications in computer science, making it an invaluable tool for both researchers and practitioners



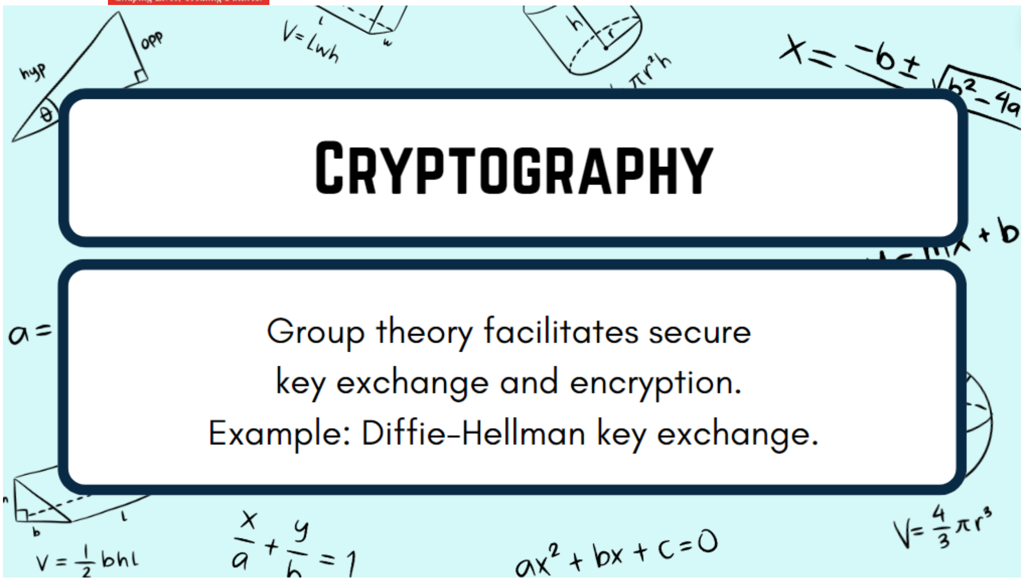
Slide 4

Group theory is a branch of mathematics that investigates algebraic structures known as groups. A group is composed of a set of elements combined with a binary operation that satisfies four main axioms: closure, associativity, the identity element, and the inverse element. These properties might seem abstract at first, but they provide a robust framework for understanding a wide variety of mathematical and real-world phenomena. Groups are used to model symmetry and are foundational in fields like physics, chemistry, and of course, computer science.



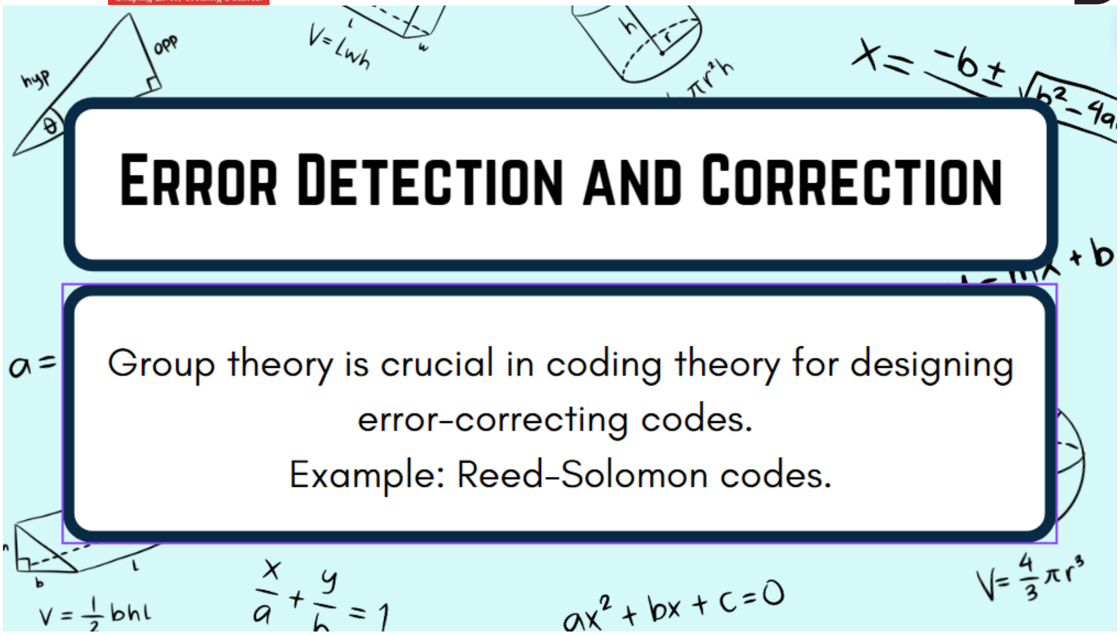
Slide 5

Group theory's abstract principles have practical applications in many areas of computer science. Today, we will focus on four major applications: cryptography, error correcting codes, computer graphics, and robotics. Each of these areas utilizes group theory to solve complex problems, enhance security, ensure data integrity, and improve functionality. By exploring these applications, we will see how the abstract concepts of group theory translate into real-world solutions.



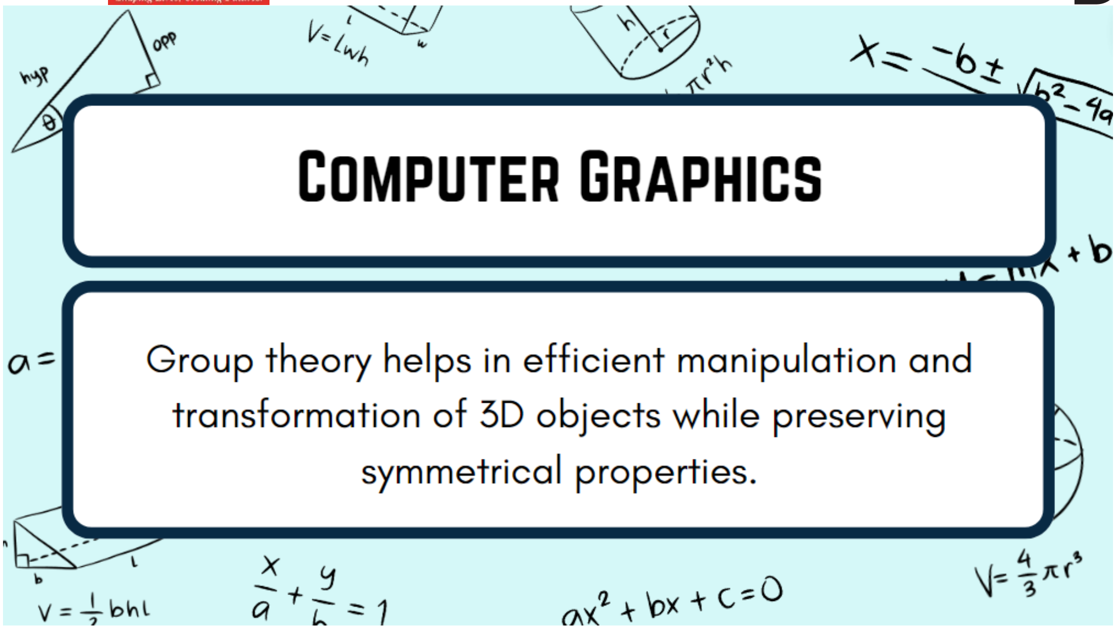
Slide 6

Cryptography is essential for securing communication in the digital age. Group theory plays a critical role in developing cryptographic algorithms that ensure data confidentiality and integrity. One of the most notable applications is the Diffie-Hellman key exchange protocol. This protocol allows two parties to securely exchange cryptographic keys over a public channel by using the properties of cyclic groups. By leveraging group theory, cryptography can provide robust security mechanisms to protect sensitive information from unauthorized access.



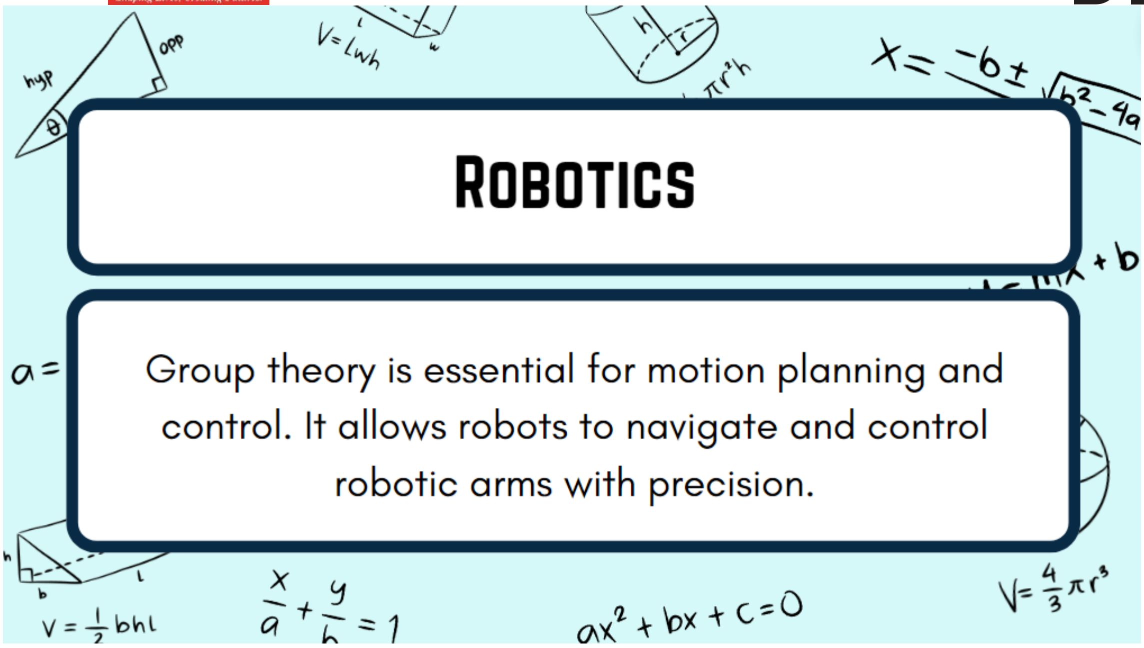
Slide 7

Error detection and correction are vital in maintaining data integrity during transmission and storage. Group theory is instrumental in coding theory, which deals with designing error-correcting codes. These codes can detect and correct errors that occur during data transmission. Reed-Solomon codes, for example, are widely used in digital communications and data storage systems. They use the principles of finite fields and polynomial algebra, rooted in group theory, to improve the reliability of data transfer and storage.



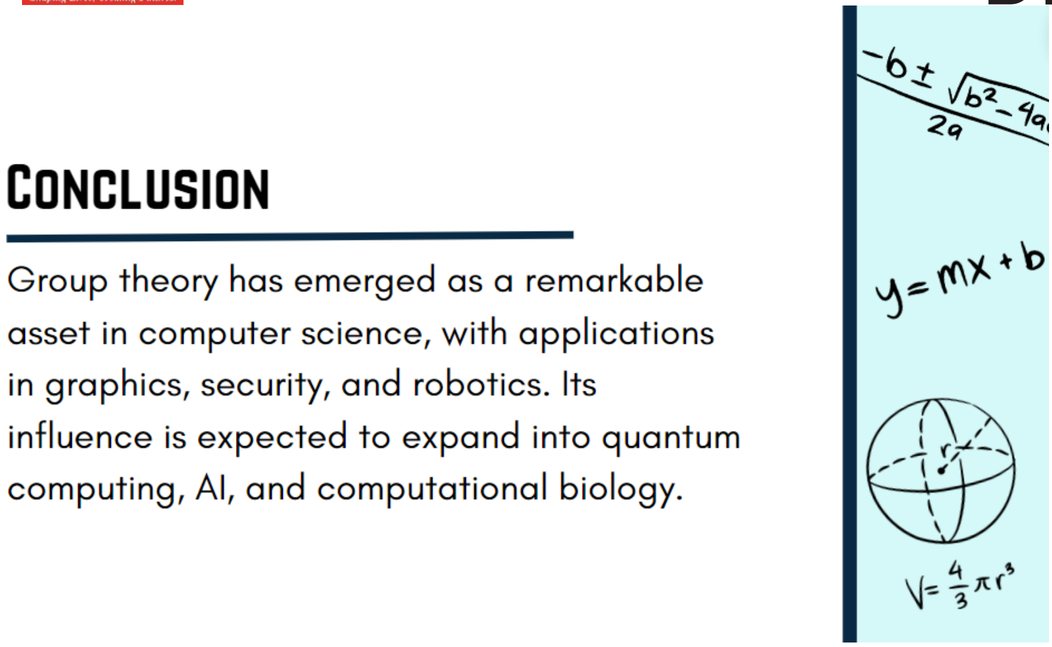
Slide 8

In computer graphics, group theory helps manage the manipulation and transformation of 3D objects. Group theory ensures that transformations such as rotations, translations, and scaling are applied consistently, preserving the object's symmetrical properties. This is crucial for rendering realistic and accurate graphical representations. The mathematical foundation provided by group theory allows for efficient algorithms that can handle complex transformations and animations in real-time graphics applications.



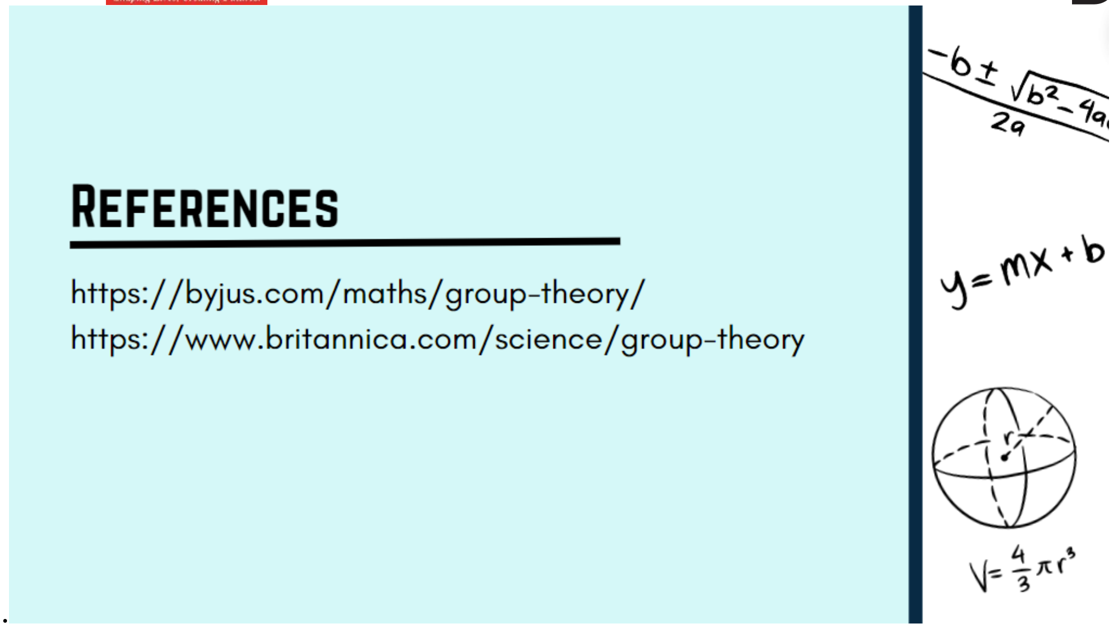
Slide 9

Robotics relies heavily on group theory for motion planning and control. Group theory helps robots navigate through complex environments and perform precise movements. For instance, the kinematics of robotic arms, which involve calculating joint angles to achieve a desired position, are modeled using group theory. This enables robots to perform tasks with high precision, whether in manufacturing, surgery, or exploration. By understanding the mathematical structure of movement, robots can operate more efficiently and accurately.



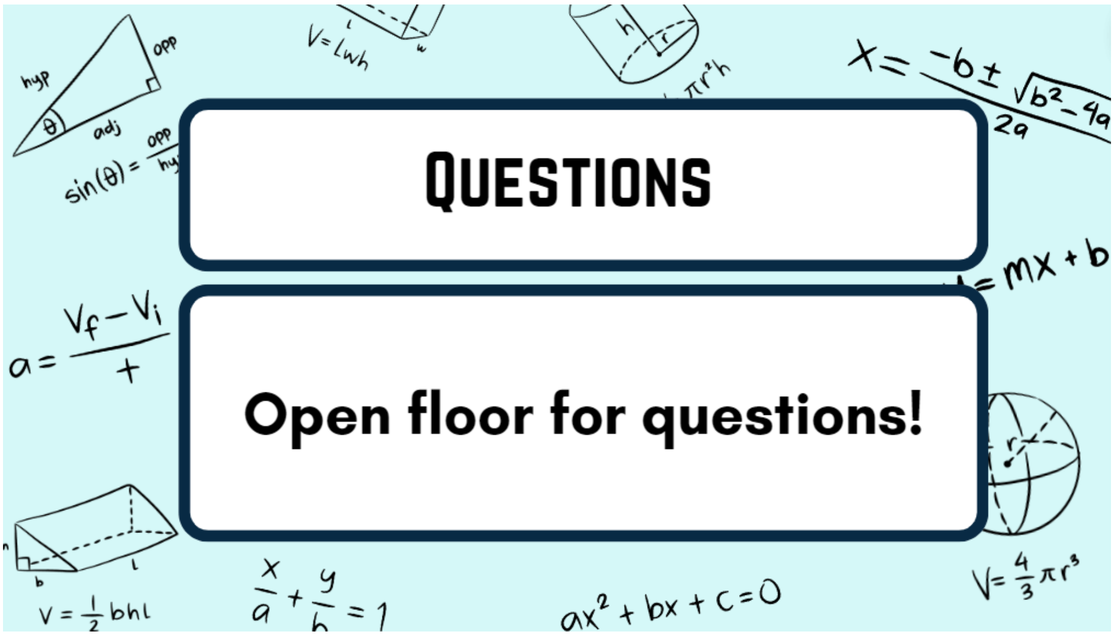
Slide 10

To sum up, group theory is a powerful tool in computer science with applications in cryptography, error correction, computer graphics, and robotics. Its ability to provide a structured way to analyze and manipulate symmetry and transformations makes it indispensable in these fields. Looking ahead, we can expect group theory to play a significant role in emerging technologies like quantum computing, artificial intelligence, and computational biology. Its potential to drive innovation and solve complex problems is immense



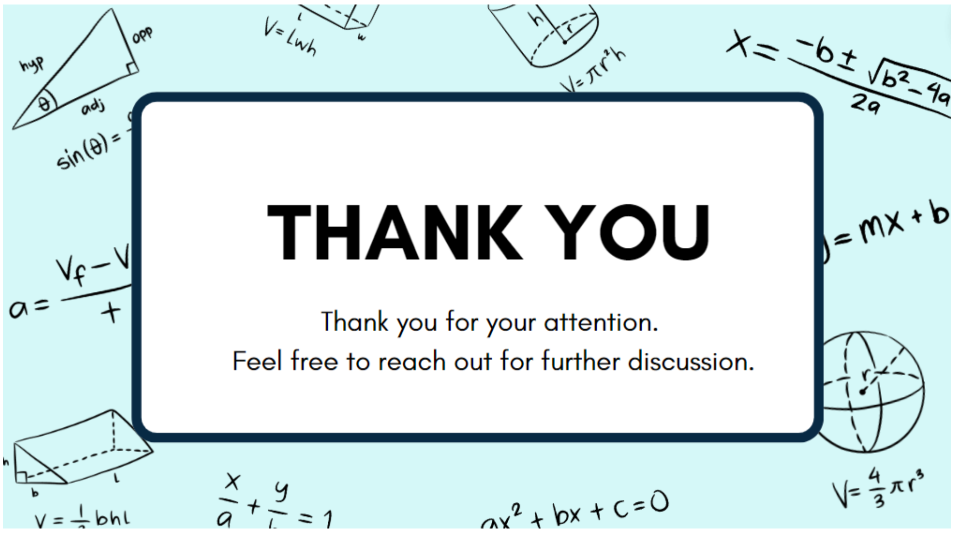
Slide 11

These are the references and sources I used to gather information for this presentation. They provide a wealth of knowledge on group theory and its applications. I encourage you to explore these resources further to deepen your understanding of the topic.



Slide 12

Now, I'd like to open the floor for questions. Please feel free to ask about any aspect of group theory or its applications that we've discussed today. I'm here to help clarify any points and discuss the material in more detail.



Slide 13

Thank you all for your attention and participation. I hope you found this presentation insightful and that it has sparked your interest in group theory and its applications in computer science. If you have any further questions or would like to explore these topics further, please don't hesitate to reach out. Thank you once again.

QUESTION 03

1. **Validate whether the set ℤ is a group under the binary operation '*'defined as a*b=a+b+2 for any two integers a, b.**

1. Closure Property

For closure, we need to show that for any two integers a, b ∈ Z, the result of a * b = a + b + 2 is also an integer.

Let a, b ∈ Z. Since a and b are integers and the sum of integers is also an integer, a + b + 2 ∈ Z.So, a * b ∈ Z, which implies that the binary operation '*' is closed over Z.Closure property is satisfied.

2. Associative Property

We need to check whether (a * b) * c = a * (b * c) for all a, b, c ∈ Z.

Let's compute both sides:

LHS:

$(a*b)*c=(a+b+2)*c=(a+b+2)+c+2=a+b+c+4$(a∗b)∗c=(a+b+2)∗c=(a+b+2)+c+2=a+b+c+4

RHS:

$a*(b*c)=a*(b+c+2)=a+(b+c+2)+2a+b+c+4$a∗(b∗c)=a∗(b+c+2)=a+(b+c+2)+2=a+b+c+4

Since LHS = RHS, the operation is associative.

Associative property is satisfied.

3. Commutative Property

Check whether a * b = b * a for all a, b ∈ Z.

Compute both:

LHS=a * b = a + b + 2

RHS=b * a = b + a + 2 = a + b + 2

Since a * b = b * a, the operation is commutative.

Since addition is commutative and multiplication is commutative, we can see that both sides of the equation are equal. Therefore, the commutative property is satisfied.

4. Identity Element

We need to find an element e ∈ Z such that for all a ∈ Z,

$a*e=a \text{and} e*a=a$a∗e=aande∗a=a

Start from:

$a*e=a+e+2=a \Rightarrow e+2=0 \Rightarrow e=-2$a∗e=a+e+2=a⇒e+2=0⇒e=−2

Verify:

a * (-2) = a + (-2) + 2 = a

(-2) * a = -2 + a + 2 = a

So, e = -2 is the identity element.

Identity element property is satisfied.

5. Inverse Element

For each a ∈ Z, there must exist an element b ∈ Z such that:

$a*b=e=-2$

So:

$a+b+2=-2 \Rightarrow a+b=-4 \Rightarrow b=-4-a$

Let’s verify:

a * (-4 - a) = a + (-4 - a) + 2 = -4 + 2 = -2

Since b = -4 - a ∈ Z, every element has an inverse in Z.

Inverse element property is satisfied.

Since all five group properties closure, associativity, commutativity, identity, and inverse are satisfied,

The set Z under the binary operation a * b = a + b + 2 forms an abelian group.

Gant chart

Months / Activities	April			May				June				July		
	W 2	W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3
Task 01														
Task 02														
Task 03														
Task 04														

References

5-color Theorem proof, 2025. *csustan.* [Online]

Available at: : https://csustan.csustan.edu/~tom/Lecture-Notes/5-colors/5-color.pdf

[Accessed 04 07 2025].

BYJUS, 2025. *BYJUS.* [Online]

Available at:

https://byjus.com/maths/lagrangetheorem/#:~:text=Lagrange%20theorem%20is%20one%20of,given%20by%20Josep

[Accessed 3 07 2025].

cgm, n.d. *cgm.* [Online]

Available at: https://cgm.cs.mcgill.ca/~athens/cs507/Projects/2003/MatthewWahab/5color.html

[Accessed 1 07 2025].

Cheng, K.-T., 2009. *sciencedirect.* [Online]

Available at: https://www.sciencedirect.com/topics/computer-science/dijkstraalgorithms#:~:text=Dijkstra's%20algorithm%20solves%20the%20shortest,all%20e

[Accessed 04 07 2025].

Dogfrey, 1999. *Dogfrey.* [Online]
Available at: https://dogschool.tripod.com/lagrange.html
[Accessed 4 07 2025].

javatpoint, 2023. *javatpoint.* [Online]
Available at: https://www.javatpoint.com/proof-of-de-morgans-law
[Accessed 15 07 2025].

javatpoint, 2025. *javatpoint.* [Online]
Available at: https://www.javatpoint.com/discrete-mathematics-binary-trees
[Accessed 04 07 2025].

Sharma, R., 2025. *upgrad.* [Online]
Available at: https://www.upgrad.com/blog/5-types-of-binary-tree/
[Accessed 02 07 2025].

toppr, 2025. *toppr.* [Online]
Available at: https://www.toppr.com/guides/maths/relations-and-functions/binaryoperations/
[Accessed 15 07 2025].