Higher Nationals

Internal verification of assessment decisions – BTEC (RQF)

INTERNAL VERIFICATION – ASSESSMENT DECISIONS			
Programme title	BTEC Higher National Diploma in Computing		
Assessor		**Internal Verifier**	
Unit(s)	Unit 18 : Discrete Mathematics		
Assignment title	Discrete mathematics in software engineering concepts		
Student'sname	Ryan Wickramaratne (COL 00081762)		
List which assessment criteria the Assessor has awarded.	**Pass**	**Merit**	**Distinction**

INTERNAL VERIFIER CHECKLIST		
Do the assessment criteria awarded match those shown in the assignment brief?	Y/N	
Is the Pass/Merit/Distinction grade awarded justified by the assessor's comments on the student work?	Y/N	
Has the work been assessed accurately?	Y/N	
Is the feedback to the student: Give details: • Constructive? • Linked to relevant assessment criteria? • Identifying opportunities for improved performance? • Agreeing actions?	 Y/N Y/N Y/N Y/N	
Does the assessment decision need amending?	Y/N	

Assessor signature		**Date**	
Internal Verifier signature		**Date**	
Programme Leader signature (if required)		**Date**	

Confirm action completed			
Remedial action taken Give details:			
Assessor signature		**Date**	
Internal Verifier signature		**Date**	
Programme Leader signature (if required)		**Date**	

Higher Nationals - Summative Assignment Feedback Form

Student Name/ID	Ryan Wickramaratne (COL 00081762)		
Unit Title	Unit 18 : Discrete Mathematics		
Assignment Number	1	**Assessor**	
Submission Date	07/07/2023	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	

Assessor Feedback:

LO1 Examine set theory and functions applicable to software engineering.

Pass, Merit & Distinction Descripts	**P1**	☐	**P2**	☐	**M1**	☐	D1 ☐

LO2 Analyse mathematical structures of objects using graph theory.

Pass, Merit & Distinction Descripts	**P3**	☐	**P4**	☐	**M2**	☐	D2 ☐

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

Pass, Merit & Distinction Descripts	**P5**	☐	**P6**	☐	**M3**	☐	D3 ☐

LO4 Explore applicable concepts within abstract algebra.

Pass, Merit & Distinction Descripts	**P7**	☐	**P8**	☐	**M4**	☐	D4 ☐

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.

ESOFT Shaping Lives, Creating Futures.

BTEC

Pearson
Higher Nationals in
Computing

Unit 18 : Discrete Mathematics

General Guidelines

1. A Cover page or title page – You should always attach a title page to your assignment. Use previous page as your cover sheet and make sure all the details are accurately filled.
2. Attach this brief as the first section of your assignment.
3. All the assignments should be prepared using a word processing software.
4. All the assignments should be printed on A4 sized papers. Use single side printing.
5. Allow 1" for top, bottom , right margins and 1.25" for the left margin of each page.

Word Processing Rules

1. The font size should be **12** point, and should be in the style of **Time New Roman**.
2. **Use 1.5 line spacing**. Left justify all paragraphs.
3. Ensure that all the headings are consistent in terms of the font size and font style.
4. 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.
5. Use word processing application spell check and grammar check function to help editing your assignment.

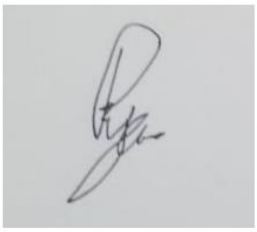
Important Points:

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. Carefully check the hand in date and the instructions given in the assignment. Late submissions will not be accepted.
3. Ensure that you give yourself enough time to complete the assignment by the due date.
4. Excuses of any nature will not be accepted for failure to hand in the work on time.
5. You must take responsibility for managing your own time effectively.
6. 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.
7. Failure to achieve at least PASS criteria will result in a REFERRAL grade .
8. Non-submission of work without valid reasons will lead to an automatic RE FERRAL. You will then be asked to complete an alternative assignment.
9. 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.
10. 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

Student Declaration

I hereby, declare that I know what plagiarism entails, namely to use another's work and to present it as my own without attributing the sources in the correct way. I further understand what it means to copy another's work.

1. I know that plagiarism is a punishable offence because it constitutes theft.
2. I understand the plagiarism and copying policy of the Edexcel UK.
3. I know what the consequences will be if I plagiaries or copy another's work in any of the assignments for this program.
4. I declare therefore that all work presented by me for every aspects of my program, will be my own, and where I have made use of another's work, I will attribute the source in the correct way.
5. I acknowledge that the attachment of this document signed or not, constitutes a binding agreement between myself and Pearson, UK.
6. I understand that my assignment will not be considered as submitted if this document is not attached to the attached.



ryandilthusha@gmail.com

7/07/2023

Student's Signature:
(*Provide E-mail ID*)

Date:
(*Provide Submission Date*)

Feedback Form

Formative feedback: Assessor to Student			
Action Plan			
Summative feedback			
Feedback: Student to Assessor			
Assessor's Signature		Date	
Student's Signature		Date	

Assignment Brief

Student Name /ID Number	Ryan Wickramaratne (COL 00081762)
Unit Number and Title	Unit 18 :Discrete Mathematics
Academic Year	2021/22
Unit Tutor	Mr. Thimira
Assignment Title	Discrete mathematics in Computing
Issue Date	
Submission Date	7/7/2023
IV Name & 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.

Assignment Brief and Guidance:

Activity 01

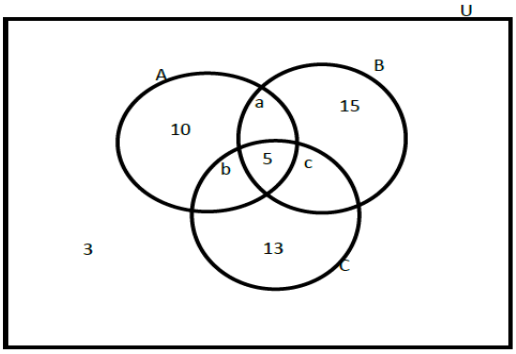
Part 1

1. Perform algebraic set operations in the following formulated mathematical problems.

i. Let A and B be two non-empty finite sets. If cardinalities of the sets A, B, and $A \cap B$ are 72, 28 and 13 respectively, find the cardinality of the set $A \cup B$.

ii. If n($A - B$)=45, n($A \cup B$)=110 and n($A \cap B$)=15, then find n(B).

iii. If n(A)=33, n(B)=36 and n(C)=28, find n($A \cup B \cup C$).



Part 2

1. Write the multisets (bags) of prime factors of given numbers.
   i. 160
   ii. 120
   iii. 250
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 technique.

i. $f : \Re \to \Re^+$, $f(x) = x^2$

ii. $f : \Re^+ \to \Re^+$, $f(x) = 1/x$

iii. $f : \Re^+ \to \Re^+$, $f(x) = x^2$

iv. $f : \left[-\pi/2, \pi/2\right] \to [-1,1]$, $f(x) = \sin x$

v. $f : [0, \pi] \to [-2, 2]$, $f(x) = 2\cos x$

Part 4

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

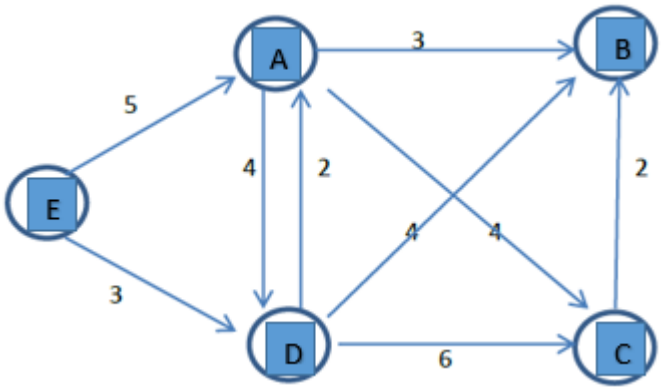
Activity 02

Part 1

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

Part 2

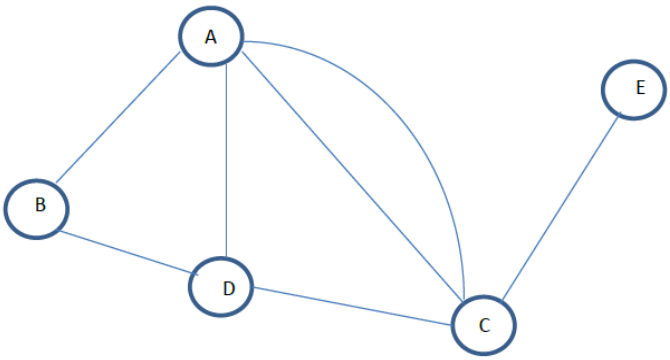
1. State the Dijkstra's algorithm for a directed weighted graph with all non-negative edge weights.
2. Use Dijkstra's algorithm to find the shortest path spanning tree for the following weighted directed graph with vertices A, B, C, D, and E given. Consider the starting vertex as E.



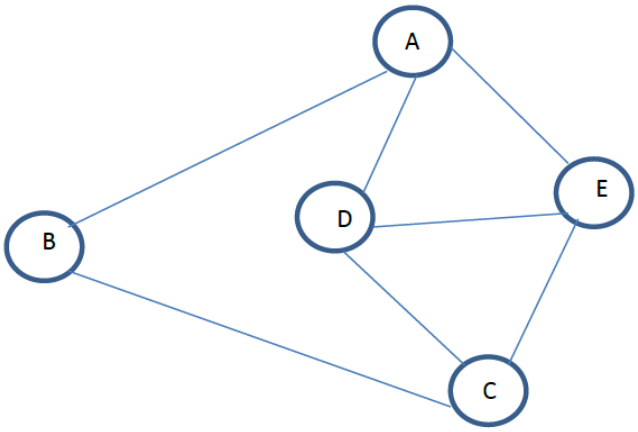
Part 3

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

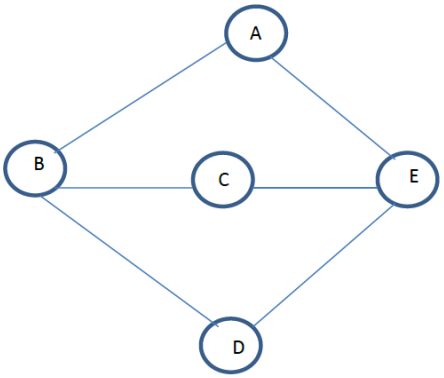
i.



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 driver is present **and** the driver has not buckled up **and** the ignition switch is on, **then** the warning light should turn on.
   ii. **If** it rains **and** you don't open your umbrella, **then** you will get wet.
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. $(A + \bar{B} + C)(A + B + C)(\bar{A} + B + \bar{C})$

Part 3

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

Part 4

1. Consider the K-Maps given below. For each K- Map
   i. Write the appropriate standard form (SOP/POS) of Boolean expression.
   ii. Design the circuit using AND, NOT and OR gates.
   iii. Design the circuit only by using
   - NAND gates if the standard form obtained in part (i) is SOP.
   - NOR gates if the standard form obtained in pat (i) is POS.

(a)

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

(b)

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

(c)

AB/C	0	1
00	1	0
01	1	1
11	1	0
10	0	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.
   i. State the relation between the order of a group and the number of binary operations that can be defined on that set.
   ii. How many binary operations can be defined on a set with 4 elements?
3.
   i. State the Lagrange's theorem of group theory.
   ii. For a subgroup H of a group G, prove the Lagrange's theorem.
   iii. Discuss whether a group H with order 6 can be a subgroup of a group with order 13 or not. Clearly state the reasons.

Part 3

1. Validate whether the set $S = \Re - \{-1\}$ is a group under the binary operation '*'defined as $a * b = a + b + ab$ for any two elements $a, b \in S$.

Part 4

1. Prepare a presentation for ten minutes to explore an application of group theory relevant to your course of study. (i.e. in Computer Sciences)

Grading Criteria	Achieved	Feedback
LO1 : Examine set theory and functions applicable to software engineering.		
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 technique.		
D1 Formulate corresponding proof principles to prove properties about defined sets.		
LO2 : Analyse mathematical structures of objects using graph theory.		
P3 Model contextualized 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.		
D2 Construct a proof of the Five colour theorem.		
LO3 : **Investigate solutions to problem situations using the application of Boolean algebra.**		
P5 Diagram a binary problem in the application of Boolean Algebra.		
P6 Produce a truth table and its corresponding Boolean equation from an applicable scenario.		
M3 Simplify a Boolean equation using algebraic methods.		

D3 Design a complex system using logic gates.		
LO4 : Explore applicable concepts within abstract algebra.		
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 given examples.		
M4 Validate whether a given set with a binary operation is indeed a group.		
D4 Explore with the aide of a prepared presentation the application of group theory relevant to your course of study		

Acknowledgement

I would like to express my special thanks of gratitude to my math lecturer Mr. Thimira for providing invaluable guidance and giving immense amount of knowledge to work on this assignment perfectly. I specially thanks him because he helped us in doing a lot of research and I came to know about so many new things about the math solving techniques.

Secondly, I would like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Executive Summery

This assignment extensively explores mathematical theory, focusing on set theory, function analysis, and number theory. It includes an examination of prime factorization, multiplicities, and cardinalities of multisets. Functions, their characteristics, injectivity, surjectivity, and invertibility were studied across quadratic, reciprocal, and trigonometric types. Additionally, we delved into set theory, offering a formal proof of De Morgan's laws and the distributive laws for finite sets. Lastly, graph theory was discussed, particularly focusing on Hamiltonian cycles. The assignment underscored thorough explanations, rigorous proofs, and precise formulations to ensure a comprehensive understanding of the principles involved.

List of figures

List of Tables

TABLE OF CONTENTS

Activity 1

Part 1

Question 1 – (i)

The cardinality of the set A union B is given by the formula |A U B| = |A| + |B| - |A ∩ B|, where |A|, |B|, and |A ∩ B| represent the cardinalities of the sets A, B, and their intersection respectively.

So, substituting the given values into the formula, we get:

|A U B| = |A| + |B| - |A ∩ B|

= 72 + 28 - 13

|A U B| =87

Therefore, the cardinality of the set A union B is 87.

Question 1 – (ii)

The formula for the cardinality of the union of two sets is n(A U B) = n(A) + n(B) - n(A ∩ B). The formula for the cardinality of the difference of two sets is n(A - B) = n(A) - n(A ∩ B).

From the problem, we know that n(A U B) = 110, n(A ∩ B) = 15, and n(A - B) = 45.

First, let's calculate n(A):

n(A) = n(A - B) + n(A ∩ B)

= 45 + 15

= 60

Then, we can use n(A) to calculate n(B):

n(B) = n(A U B) - n(A) + n(A ∩ B)

= 110 - 60 + 15

n(B) = 65

Therefore, the cardinality of the set B is 65.

Question 1 – (iii)

The cardinalities of sets A, B, and C, and their intersections can be represented as:

n(A) = 10 + a + b + 5 = 33,

n(B) = a + 5 + c + 15 = 36,

n(C) = b + 5 + c + 13 = 28,

n(A ∩ B ∩ C) = 5

We can find the values of a, b, and c by solving these equations.

From n(A) = 33, we can calculate:

15 + a + b = 33,

a + b = 33 - 15,

a + b = 18 → (Equation 1)

From n(B) = 36, we can calculate:

20 + a + c = 36,

a + c = 36 - 20,

a + c = 16 → (Equation 2)

From n(C) = 28, we can calculate:

18 + b + c = 28,

b + c = 28 - 18,

b + c = 10 → (Equation 3)

Subtracting Equation 1 from Equation 2 gives us:

$a + b = 18 \rightarrow$ (Equation 1)

$a + c = 16 \rightarrow$ (Equation 2)

$a + b - (a + c) = 18 - 6$

$a + b - a - c = 2$

$\cancel{a} + b - \cancel{a} - c = 2$

$b - c = 2 \rightarrow$ (Equation 4)

Addition Equation 1 from Equation 2 gives us:

$b + c = 10 \rightarrow$ (Equation 3)

$b - c = 2 \rightarrow$ (Equation 4)

$b + c + (b - c) = 10 + 2$

$b + c + b - c = 12$

$b + \cancel{c} + b - \cancel{c} = 12$

$2b = 12$

$\underline{\underline{b = 6}}$

Given that b = 6, we can substitute b into Equation 1 and Equation 3 to find the values of a and c.

Substitute b = 6 into Equation 1:

$a + b = 18$

$a + 6 = 18$

$a = 18 - 6$

$\underline{\underline{a = 12}}$

Substitute b = 6 into Equation 3:

b + c = 10

6 + c = 10

c = 10 - 6

c = 4

Now, we can find the cardinality of the union of sets A, B, and C by applying the Principle of Inclusion and Exclusion:

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

$= n(A) + n(B) + n(C) – (a+5) – (b+5) – (c+5) + n(A \cap B \cap C)$

Substituting the calculated values:

$= 33 + 36 + 28 - (12 + 5) - (6 + 5) - (4 + 5) + 5$

$= 97 - 17 - 11 - 9 + 5$

$= 65.$

So, $n(A \cup B \cup C) = 65$.

Part 2

Question 1 – (i)

To represent 160 as a multiset of its prime factors, first we need to find the prime factors of 160.

160 = 2 x 2 x 2 x 2 x 2 x 5

The multiset of prime factors of 160 is [2, 2, 2, 2, 2, 5].

Question 1 – (ii)

To represent 120 as a multiset of its prime factors, first we need to find the prime factors of 160.

120 = 2 x 2 x 2 x 3 x 5

The multiset of prime factors of 120 is [2, 2, 2, 3, 5].

Question 1 – (iii)

To represent 250 as a multiset of its prime factors, first we need to find the prime factors of 160.

250 = 2 x 5 x 5 x 5

The multiset of prime factors of 250 is [2, 5, 5, 5].

Question 2 – (i)

The multiset of prime factors of 160 is {2, 2, 2, 2, 2, 5}.

Multiplicity of 2 is 5 (because 2 appears five times). → $\mu_A(2) = 5$

Multiplicity of 5 is 1 (because 5 appears one time). → $\mu_A(5) = 1$

Question 2 – (ii)

The multiset of prime factors of 120 is {2, 2, 2, 3, 5}.

Multiplicity of 2 is 3 (because 2 appears three times). → $\mu_B(2) = 3$

Multiplicity of 3 is 1 (because 3 appears one time). → $\mu_B(3) = 1$

Multiplicity of 5 is 1 (because 5 appears one time). →$\mu_B(5) = 1$

Question 2 – (iii)

The multiset of prime factors of 250 is {2, 5, 5, 5}.

Multiplicity of 2 is 1 (because 2 appears one time). → $\mu_C(2) = 1$

Multiplicity of 5 is 3 (because 5 appears three times). → $\mu_C(5) = 3$

Question 3 – (i)

The multiset of prime factors of 160 is {2, 2, 2, 2, 2, 5}.

The cardinality of the multiset of prime factors of 160 is 6 (because there are five 2's and one 5).

5 (the multiplicity of 2) + 1 (the multiplicity of 5) = 6.

Question 3 – (ii)

The multiset of prime factors of 120 is {2, 2, 2, 3, 5}.

The cardinality of the multiset of prime factors of 120 is 5 (because there are three 2's, one 3, and one 5).

3 (the multiplicity of 2) + 1 (the multiplicity of 3) + 1 (the multiplicity of 5) = 5.

Question 3 – (iii)

The multiset of prime factors of 250 is {2, 5, 5, 5}.

The cardinality of the multiset of prime factors of 250 is 4 (because there are one 2 and three 5's).

1 (the multiplicity of 2) + 3 (the multiplicity of 5) = 4.

Part 3

Question 1 – (i)

$$f = \mathbb{R} \rightarrow \mathbb{R}^{+}$$

$$f(x) = x^{2}$$

Table 1. 1 Table for the function (Question 1-i)

x	f(x)
-2	4
-1	1
0	0
1	1
2	4

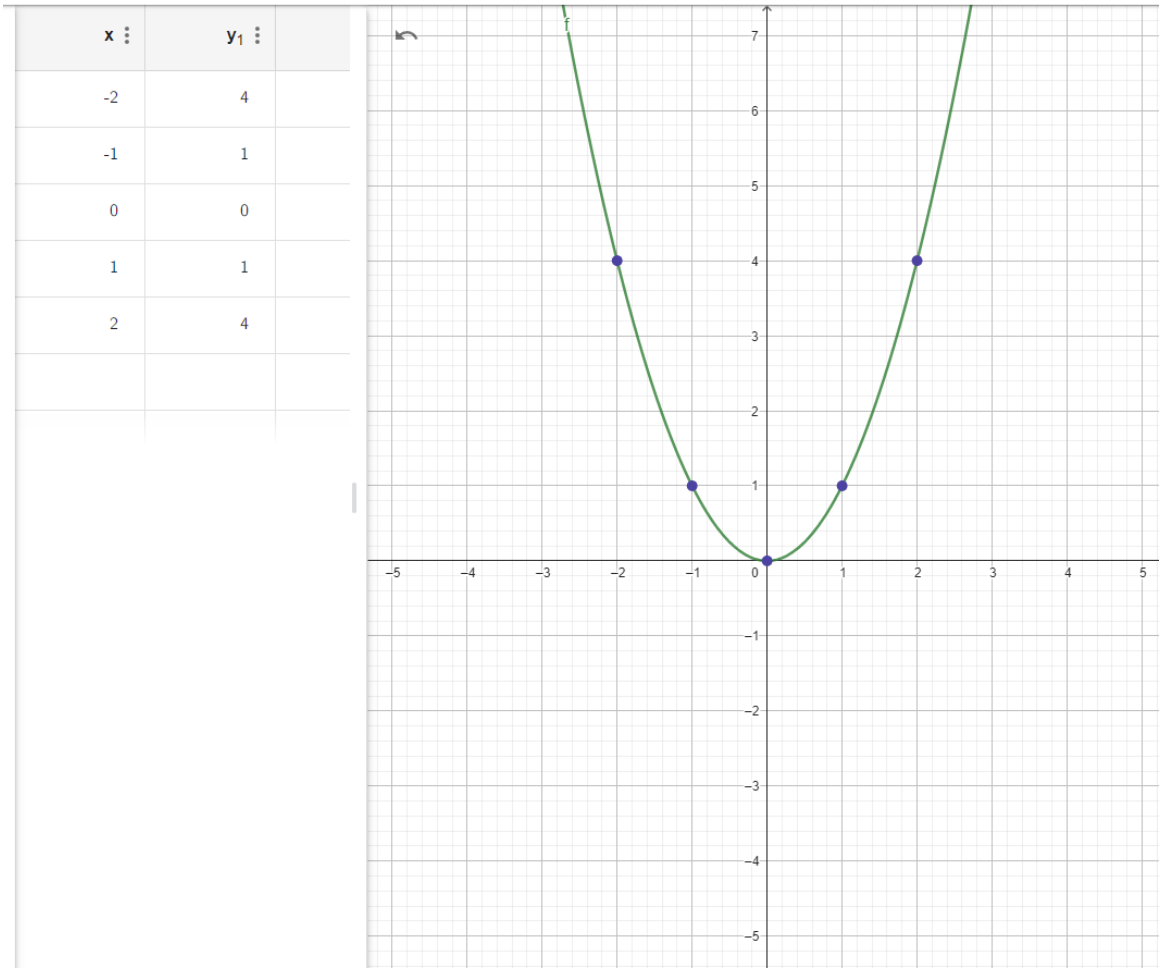


Figure 1. 1 Graph for the function (Question 1-i)

Horizontal Line Test of Injective (Finding one to one function):

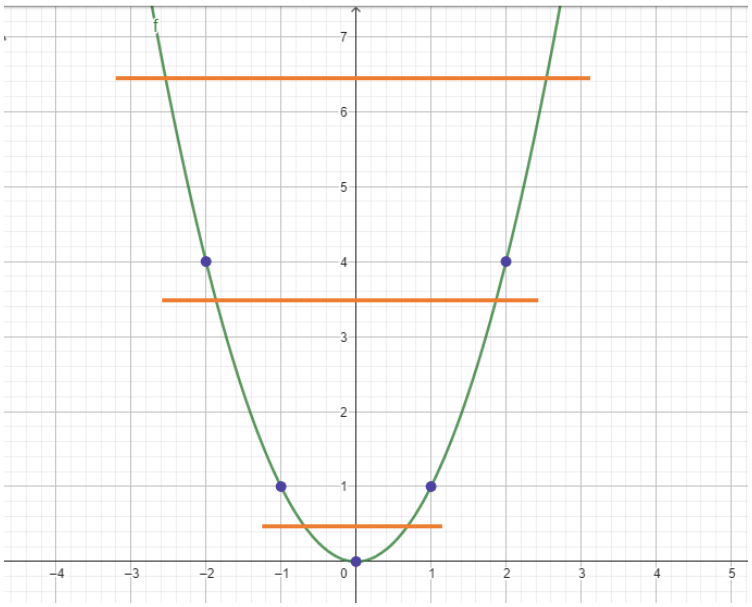


*Figure 1. 2 Graph (**Injective** Test) for the function (Question 1-i)*

The above function is not one-to-one function since if we draw a horizontal line anywhere in the graph of the function that intersects the graph in more than one place.

The function is not one-to-one (Injective).

Horizontal Line Test of Surjective (Finding on to function):

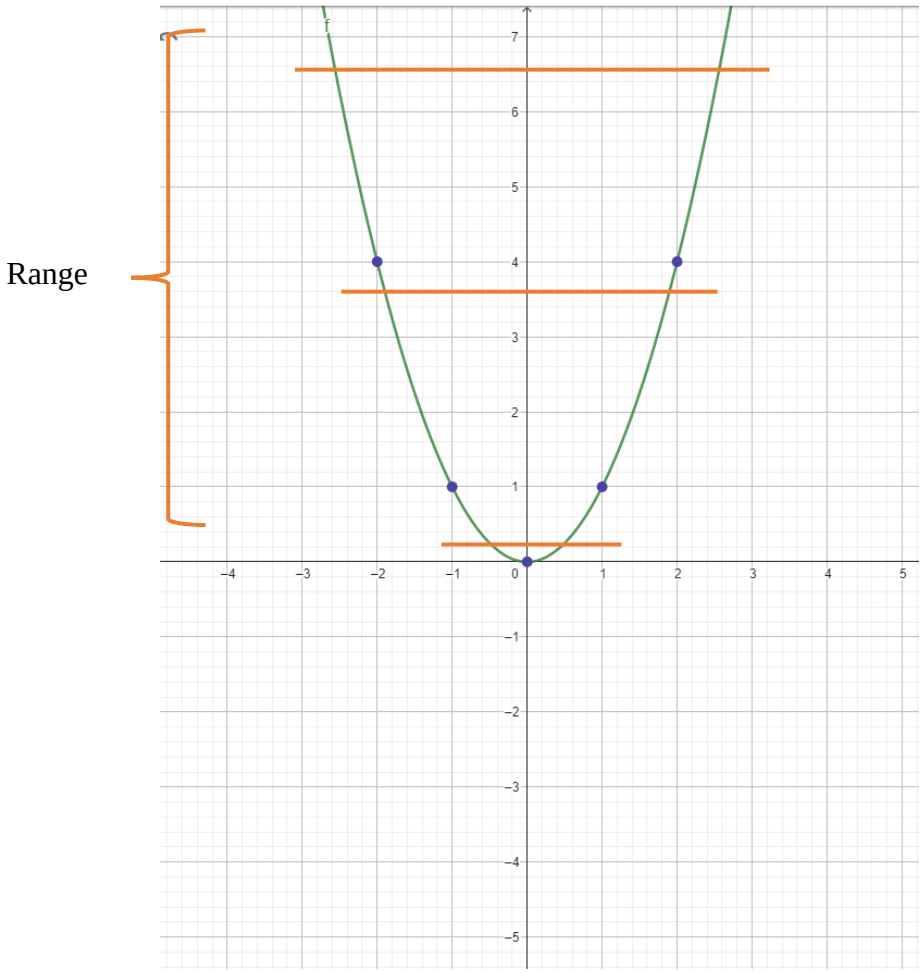


*Figure 1. 3 Graph **(Surjective** Test) for the function (Question 1-i)*

So, for every y in the codomain R+, there exists at least one x in the domain R such that f(x) = y. Therefore, the function is surjective (onto).

The function is on to (Surjective).

Invertibility requires a function to be both one-to-one (injective) and onto (surjective). As I mentioned earlier, this function is surjective, but it is not injective when the domain is all real numbers.

Therefore, since the function is not injective, it is not invertible.

Question 1 – (ii)

$$f = \mathbb{R}^+ \rightarrow \mathbb{R}^+$$

$$f(x) = \frac{1}{x}$$

Table 1. 2 Table for the function (Question 1-ii)

x	f(x)
0.5	2
1	1
2	0.5
3	1/3
4	0.25

Please Note: For x = 0, the function is undefined, so I've started the table from x = 0.5.

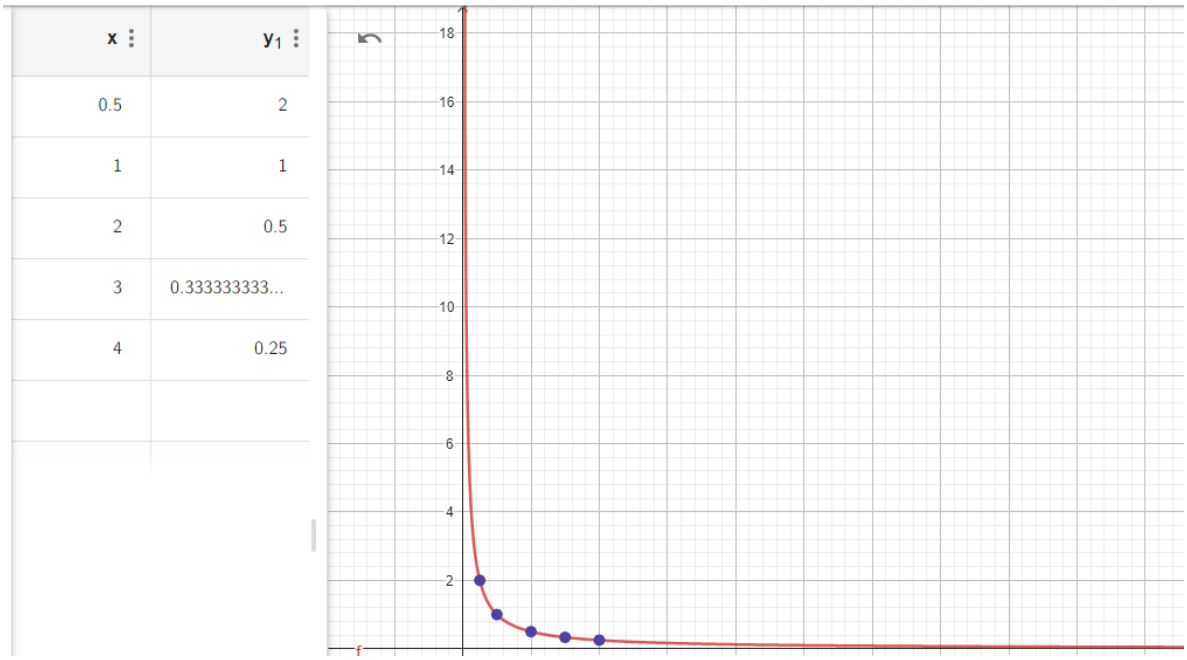


Figure 1. 4 Graph for the function (Question 1-ii)

Horizontal Line Test of Injective (Finding one to one function):

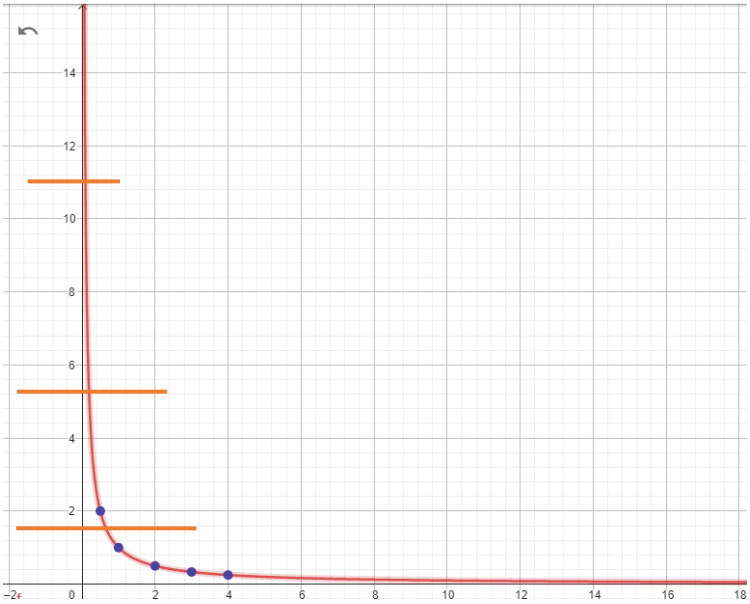


Figure 1. 5 Graph (Injective Test) for the function (Question 1-ii)

The above function is one-to-one function since if we draw a horizontal line in the graph of the function that intersects the graph at most once.

The function is one-to-one (Injective).

Horizontal Line Test of Surjective (Finding on to function):

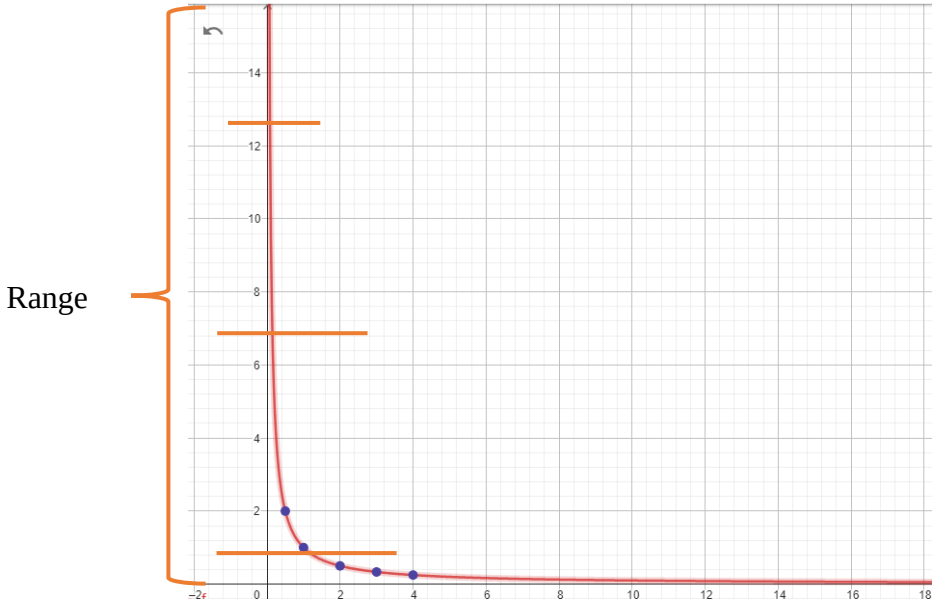


Figure 1. 6 Graph (Surjective Test) for the function (Question 1-ii)

For the function f(x) = $\frac{1}{x}$, every positive real number y has a corresponding positive real number. Therefore, the function covers all elements of the positive real numbers, making it surjective (onto) for this domain and codomain.

Hence, all horizontal lines go through the y axis intercept the function at least one time. Therefore:

The function is on to (Surjective).

Invertibility requires a function to be both one-to-one (injective) and onto (surjective). As I mentioned earlier, this function is surjective and injective when the domain is all positive real numbers.

Therefore, the function is invertible.

Finding the inverse of the function:

$$f(x) = \frac{1}{x}$$

$$f(x) \; is \; equal \; to \; y$$

$$y = \frac{1}{x}$$

$$x = \frac{1}{y}$$

$$The \; x \; is \; equal \; to \; inverse \; of the \; function \;\; and \; y \; is \; equal \; to \; x$$

$$f'(x) = \frac{1}{x}$$

Question 1 – (iii)

$$f = \mathbb{R}^+ \rightarrow \mathbb{R}^+$$

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

Table 1. 3 Table for the function (Question 1-iii)

x	f(x)
1	1
2	4
3	9
4	16
5	25

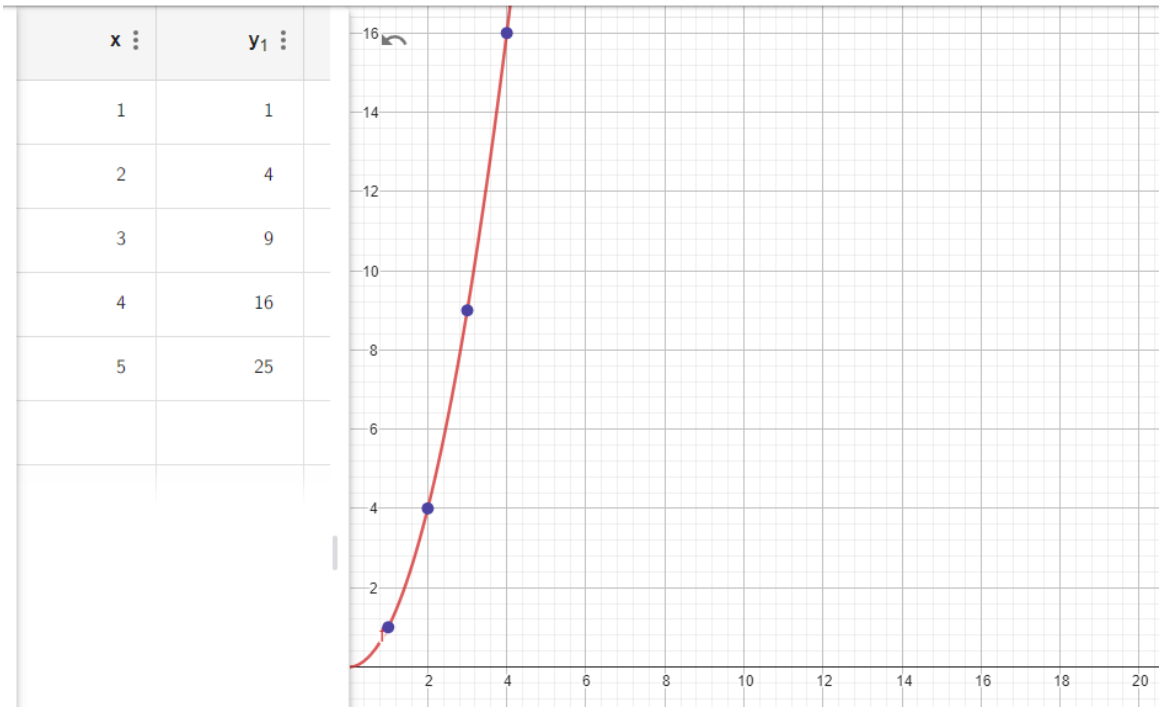


Figure 1. 7 Graph for the function (Question 1-iii)

Horizontal Line Test of Injective (Finding one to one function):

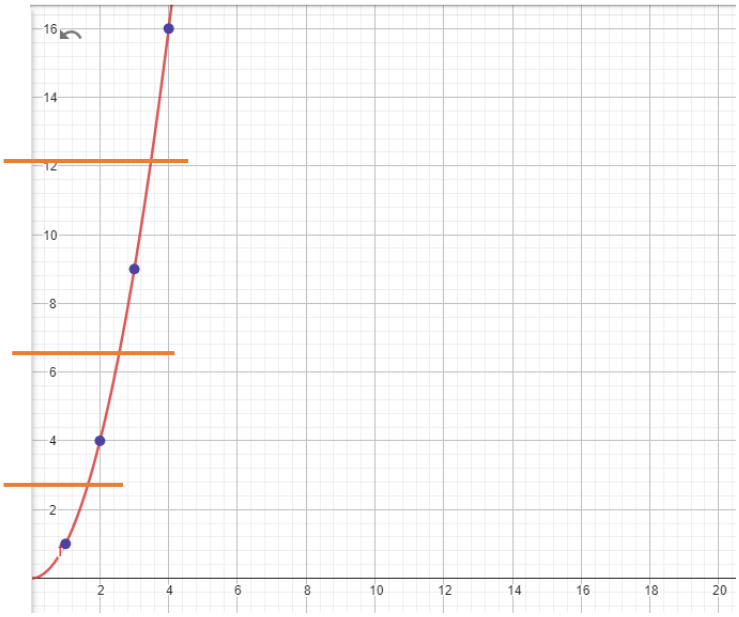


Figure 1. 8 Graph (Injective Test) for the function (Question 1-iii)

The above function is one-to-one function since if we draw a horizontal line in the graph of the function that intersects the graph at most once.

The function is one-to-one (Injective).

Horizontal Line Test of Surjective (Finding on to function):

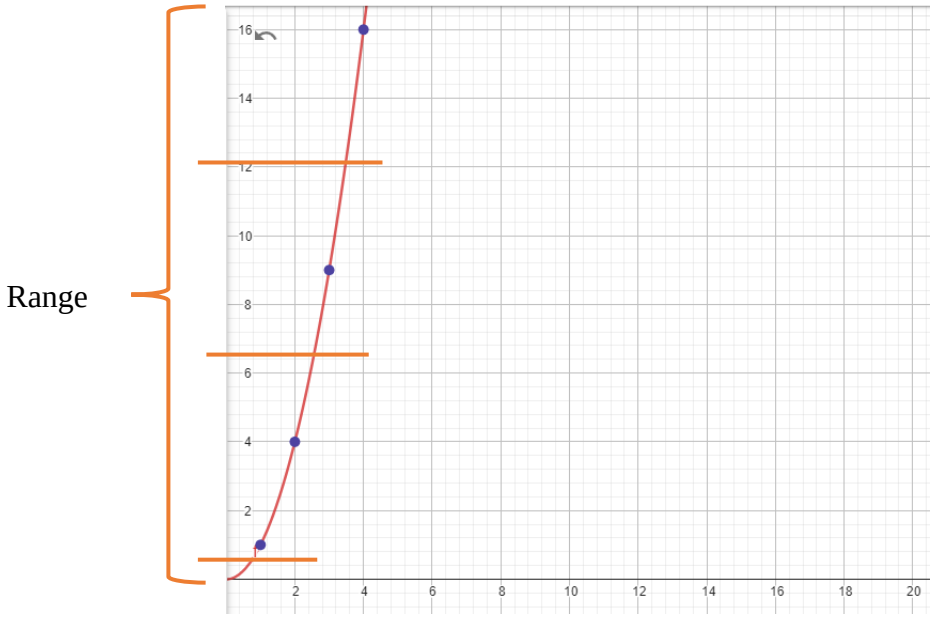


Figure 1. 9 Graph (Surjective Test) for the function (Question 1-iii)

For the function $f(x) = x^2$ every positive real number y has a corresponding positive real number. Therefore, the function covers all elements of the positive real numbers, making it surjective (onto) for this domain and codomain.

Hence, all horizontal lines go through the y axis intercept the function at least one time. Therefore:

The function is on to (Surjective).

Invertibility requires a function to be both one-to-one (injective) and onto (surjective). As I mentioned earlier, this function is surjective and injective when the domain is all positive real numbers.

Therefore, the function is invertible.

Finding the inverse of the function:

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

$$f(x) \; is \; equal \; to \; y$$

$$y = x^2$$

$$x = \sqrt{y}$$

$$The \; x \; is \; equal \; to \; inverse \; of the \; function \;\; and \; y \; is \; equal \; to \; x$$

$$f'(x) = \sqrt{x}$$

Question 1 – (iv)

$$f = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \rightarrow [-1, 1]$$

$$f(x) = \sin x$$

Table 1. 4 Table for the function (Question 1-iv)

x	f(x)
-π/2	-1
-π/4	-0.707
0	0
π/4	0.707
π/2	1

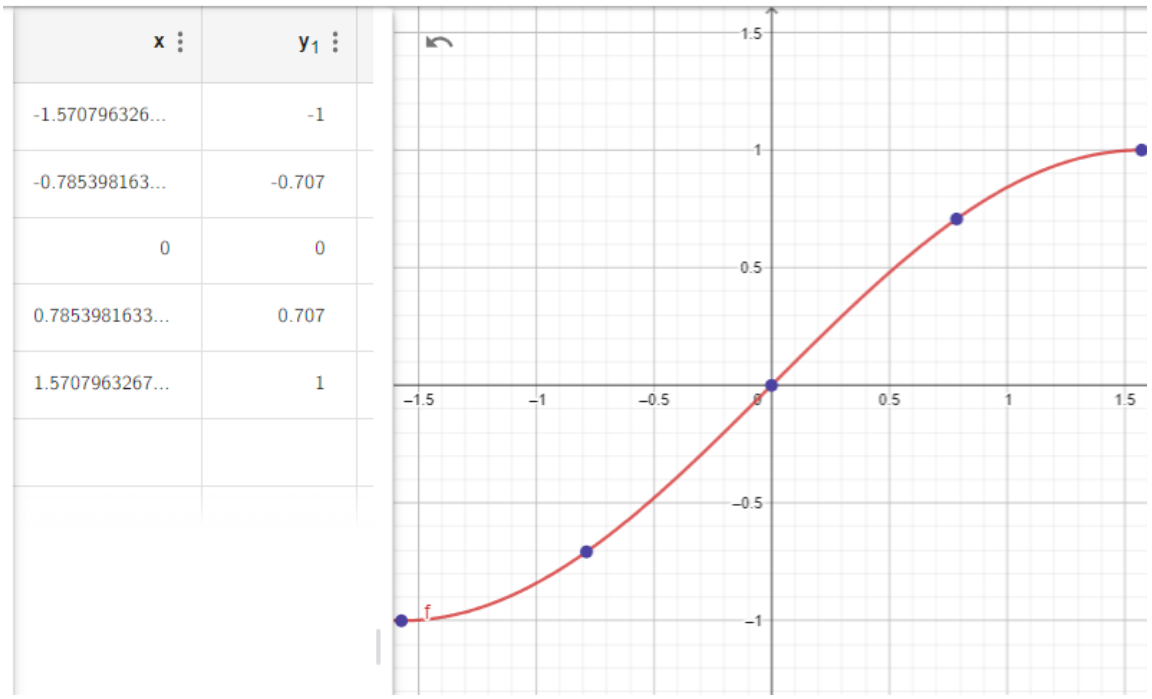


Figure 1. 10 Graph for the function (Question 1-iv)

Horizontal Line Test of Injective (Finding one to one function):

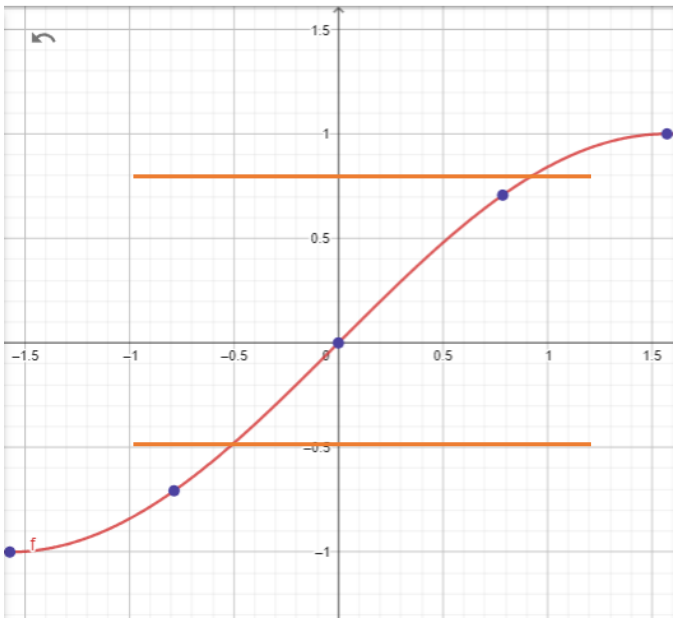


Figure 1. 11 Graph (Injective Test) for the function (Question 1-iv)

The above function is one-to-one function since if we draw a horizontal line in the graph of the function that intersects the graph at most once.

The function is one-to-one (Injective).

Horizontal Line Test of Surjective (Finding on to function):

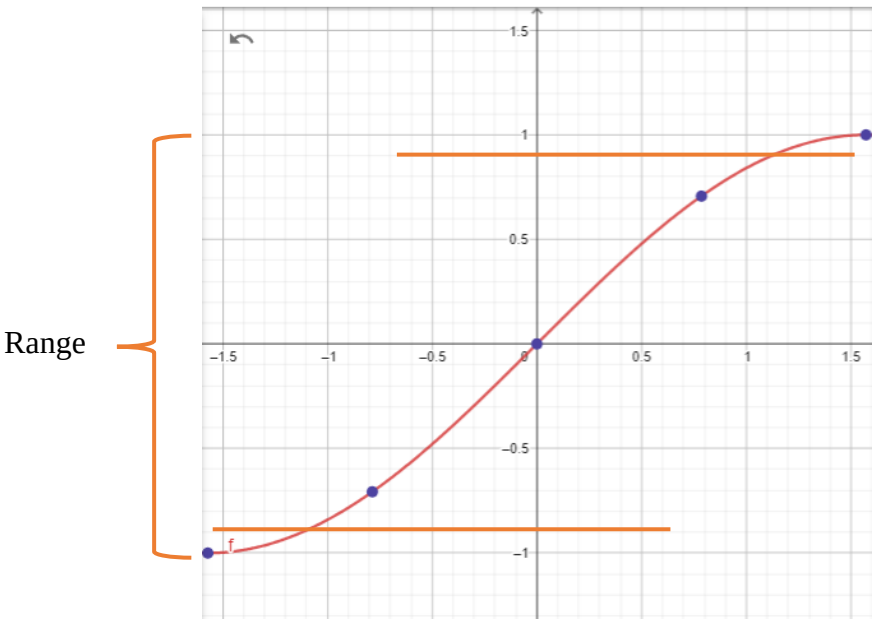


Figure 1. 12 Graph (Surjective Test) for the function (Question 1-iv)

For the function $f(x) = \sin x$ with a domain of -π/2 to π/2 and a codomain of -1 to +1, is indeed onto (surjective). This is because every y-value in the codomain (which is between -1 and 1, inclusive) can be obtained by some x-value in the domain.

Hence, all horizontal lines go through the y axis intercept the function at least one time. Therefore:

The function is on to (Surjective).

Invertibility requires a function to be both one-to-one (injective) and onto (surjective). As I mentioned earlier, this function is surjective and injective with a domain of -π/2 to π/2.

Therefore, the function is invertible.

Finding the inverse of the function:

$$f(x) = \sin x$$

$$f(x) \; is \; equal \; to \; y$$

$$y = \sin x$$

$$x = \sin^{-1}(y)$$

$$The \; x \; is \; equal \; to \; inverse \; of the \; function \;\; and \; y \; is \; equal \; to \; x$$

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

Question 1 – (v)

$$f = [0\,, \pi] \rightarrow [-2\,, 2]$$

$$f(x) = 2\cos x$$

Table 1. 5 Table for the function (Question 1-v)

x	f(x)
0	2
π/4	sqrt(2)
π/2	0
3π/4	-sqrt(2)
π	-2

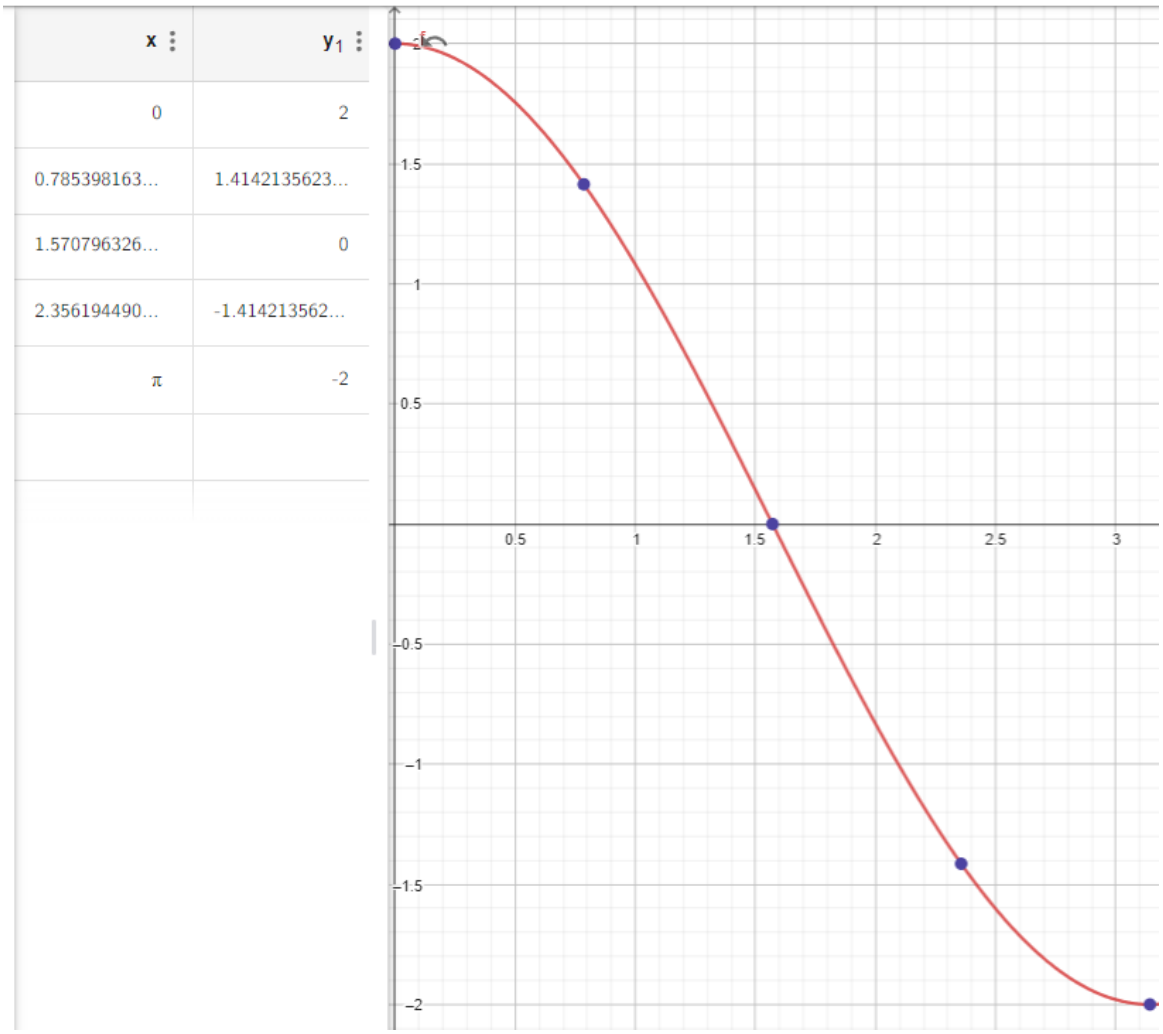


Figure 1. 13 Graph for the function (Question 1-v)

Horizontal Line Test of Injective (Finding one to one function):

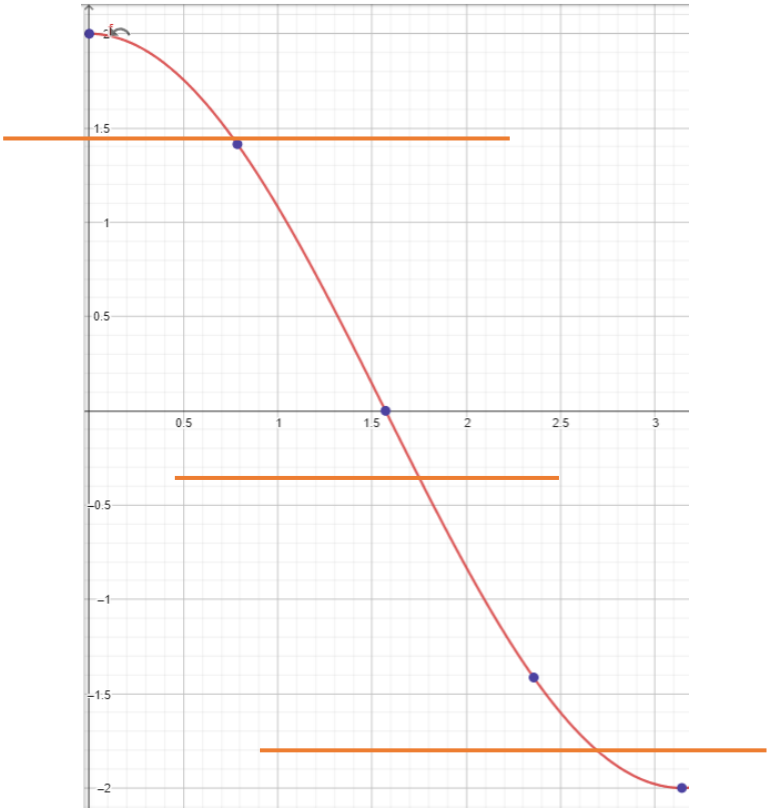


*Figure 1. 14 Graph (**Injective** Test) for the function (Question 1-v)*

The above function is one-to-one function since if we draw a horizontal line in the graph of the function that intersects the graph at most once.

The function is one-to-one (Injective).

Horizontal Line Test of Surjective (Finding on to function):

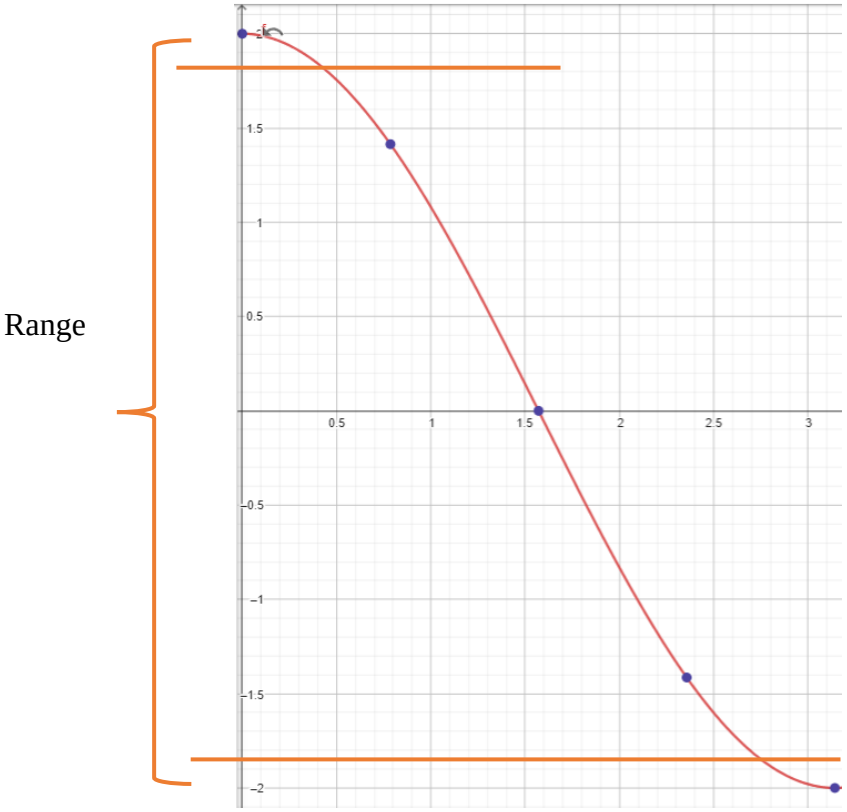


Figure 1. 15 Graph (Surjective Test) for the function (Question 1-v)

For the function $f(x) = 2\cos x$ with the domain [0, π] and co-domain [-2, 2] is surjective (onto).This is because every y-value in the codomain (which is between -2 and 2, inclusive) can be obtained by some x-value in the domain.

Hence, all horizontal lines go through the y axis intercept the function at least one time. Therefore:

The function is on to (Surjective).

Invertibility requires a function to be both one-to-one (injective) and onto (surjective). As I mentioned earlier, this function is surjective and injective with a domain of domain [0, π].

Therefore, the function is invertible.

Finding the inverse of the function:

$$f(x) = 2\cos x$$

$$f(x)\ is\ equal\ to\ y$$

$$y = 2\cos x$$

$$x = cos^{-1}\left(\frac{y}{2}\right)$$

$$The\ x\ is\ equal\ to\ inverse\ of the\ function\ \ and\ y\ is\ equal\ to\ x$$

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

Part 4

Question 1 – (i)

Assume A = B.

This means that every element in A is also in B, and every element in B is also in A. In set notation, we can say:

a) A ⊆ B: For every x, if x is in A, then x is also in B.
b) B ⊆ A: For every x, if x is in B, then x is also in A.

These two statements together establish that if A = B, then A is a subset of B and B is a subset of A.

Now, let's prove the converse: if A is a subset of B and B is a subset of A, then A = B.

a) A ⊆ B: For every x, if x is in A, then x is also in B.
b) B ⊆ A: For every x, if x is in B, then x is also in A.

If both these conditions are true, it means that every element of A is in B and every element of B is in A. Therefore, A = B.

So, we've shown that A = B if and only if A is a subset of B and B is a subset of A. This completes the proof.

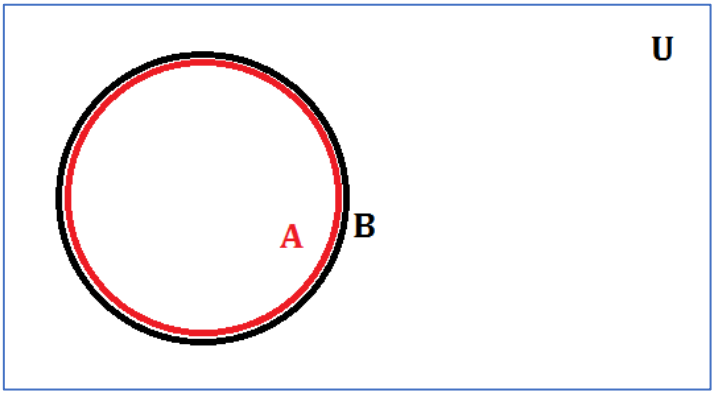


Figure 1. 16 Figure to show that A is in B and every element of B is in A

Question 1 – (ii)

De Morgan's laws state that for two sets A and B:

a) The complement of the union of two sets is equal to the intersection of their complements. Mathematically, it is represented as:

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

b) The complement of the intersection of two sets is equal to the union of their complements. Mathematically, it is represented as:

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

Proof of De Morgan's law: (A U B)' = A' ∩ B'

Let "H" = (A U B)' and "K" = A' ∩ B'

Let x be an arbitrary element of "H" then $x \in H \Rightarrow x \in (A \cup B)'$

$\Rightarrow x \notin (A \cup B)$

$\Rightarrow x \notin A$ and $x \notin B$

$\Rightarrow x \in A'$ and $x \in B'$

$\Rightarrow x \in A' \cap B'$

$\Rightarrow x \in K$

Therefore, $H \subset K \rightarrow$ (i)

Again, let y be an arbitrary element of "K" then $y \in K \Rightarrow y \in A' \cap B'$

$\Rightarrow y \in A'$ and $y \in B'$

$\Rightarrow y \notin A$ and $y \notin B$

⇒ y ∉ (A U B)

⇒ y ∈ (A U B)'

⇒ y ∈ H

Therefore, K ⊂ H → (ii)

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

Proof of De Morgan's law: (A ∩ B)' = A' U B'

Let "R" = (A ∩ B)' and "S" = A' U B'

Let x be an arbitrary element of "R" then x ∈ R ⇒ x ∈ (A ∩ B)'

⇒ x ∉ (A ∩ B)

⇒ x ∉ A or x ∉ B

⇒ x ∈ A' or x ∈ B'

⇒ x ∈ A' U B'

⇒ x ∈ S

Therefore, R ⊂ S → (iii)

Again, let y be an arbitrary element of "S" then y ∈ S ⇒ y ∈ A' U B'

⇒ y ∈ A' or y ∈ B'

⇒ y ∉ A or y ∉ B

⇒ y ∉ (A ∩ B)

⇒ y ∈ (A ∩ B)'

⇒ y ∈ R

Therefore, S ⊂ R → (iv)

When we combine (iii) and (iv) we get; R = S i.e. (A ∩ B)' = A' U B'

Question 1 – (iii)

The distributive laws for three non-empty finite sets A, B, and C are as follows:

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

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

First Law : A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)

Proof :

A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)

Let x ∈ A ∪ (B ∩ C). If x ∈ A ∪ (B ∩ C) then x is either in A or in (B and C).

x ∈ A or x ∈ (B and C)

x ∈ A or {x ∈ B and x ∈ C}

{x ∈ A or x ∈ B} and {x ∈ A or x ∈ C}

x ∈ (A or B) and x ∈ (A or C)

x ∈ (A ∪ B) ∩ x ∈ (A ∩ C)

x ∈ (A ∪ B) ∩ (A ∪ C)

x ∈ A ∪ (B ∩ C) => x ∈ (A ∪ B) ∩ (A ∪ C)

Therefore,

A ∪ (B ∩ C) ⊂ (A ∪ B) ∩ (A ∪ C) → (i)

Let x ∈ (A ∪ B) ∩ (A ∪ C). If x ∈ (A ∪ B) ∩ (A ∪ C) then x is in (A or B) and x is in (A or C).

x ∈ (A or B) and x ∈ (A or C)

{x ∈ A or x ∈ B} and {x ∈ A or x ∈ C}

x ∈ A or {x ∈ B and x ∈ C}

x ∈ A or {x ∈ (B and C)}

x ∈ A ∪ {x ∈ (B ∩ C)}

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

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

Therefore,

$(A \cup B) \cap (A \cup C) \subset A \cup (B \cap C)$ → (ii)

From equation (i) and (ii) we can prove $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

Second Law : $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

Proof :

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

Let $x \in A \cap (B \cup C)$. If $x \in A \cap (B \cup C)$ then $x \in A$ and $x \in$ (B or C).

$x \in A$ and $\{x \in B$ or $x \in C\}$

$\{x \in A$ and $x \in B\}$ or $\{x \in A$ and $x \in C\}$

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

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

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

Therefore,

$A \cap (B \cup C) \subset (A \cap B) \cup (A \cap C)$ → (iii)

Let $x \in (A \cap B) \cup (A \cap C)$. If $x \in (A \cap B) \cup (A \cap C)$ then $x \in (A \cap B)$ or $x \in (A \cap C)$.

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

$\{x \in A$ and $x \in B\}$ or $\{x \in A$ and $x \in C\}$

$x \in A$ and $\{x \in B$ or $x \in C\}$

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

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

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

Therefore,

$(A \cap B) \cup (A \cap C) \subset A \cap (B \cup C \rightarrow$ (iv)

From equation (iii) and (iv) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

Activity 2

Part 1

Question 1

Problem 1 :

Consider a scenario of an online shopping system. There are numerous product categories and sub-categories available. A binary tree can be employed to show these categories and their hierarchical relationship.

Qualitative model:

We can represent the major categories as root and sub-root nodes in the tree. The sub-categories or individual products could be represented as child nodes of the tree. The binary tree structure helps us navigate quickly to a particular category, thus improving the efficiency of the online shopping system.

The problem involves the organization of product categories and sub-categories in an online shopping system. Here, we're using a binary tree structure to represent these categories. The binary tree structure is ideal for this task because it mirrors the hierarchical relationships that exist between different product categories and sub-categories. At the very top, or root of the tree, we have the most general categories.

For example, the root node could represent the "All Products" category. This root node would then have two child nodes, representing two major product categories. As an example, these could be "Electronics" and "Home & Kitchen".

Each of these nodes would further branch into two child nodes each, representing sub-categories. For instance, the "Electronics" node could have "TVs" and "Laptops" as its child nodes. Similarly, the "Home & Kitchen" node could have "Furniture" and "Appliances" as its child nodes.

This structure enables easy navigation through the categories. If a customer wants to browse through all the available TVs, they would start at the root node, navigate to the "Electronics" node, and finally reach the "TVs" node.

This tree-like structure can keep branching out depending on how detailed the categorization is. The level of detail can be adjusted according to the need of the application, allowing for a flexible and scalable system. In addition, such a structure also helps in maintaining and updating the categories since changes to a specific category can be made without disrupting the others.

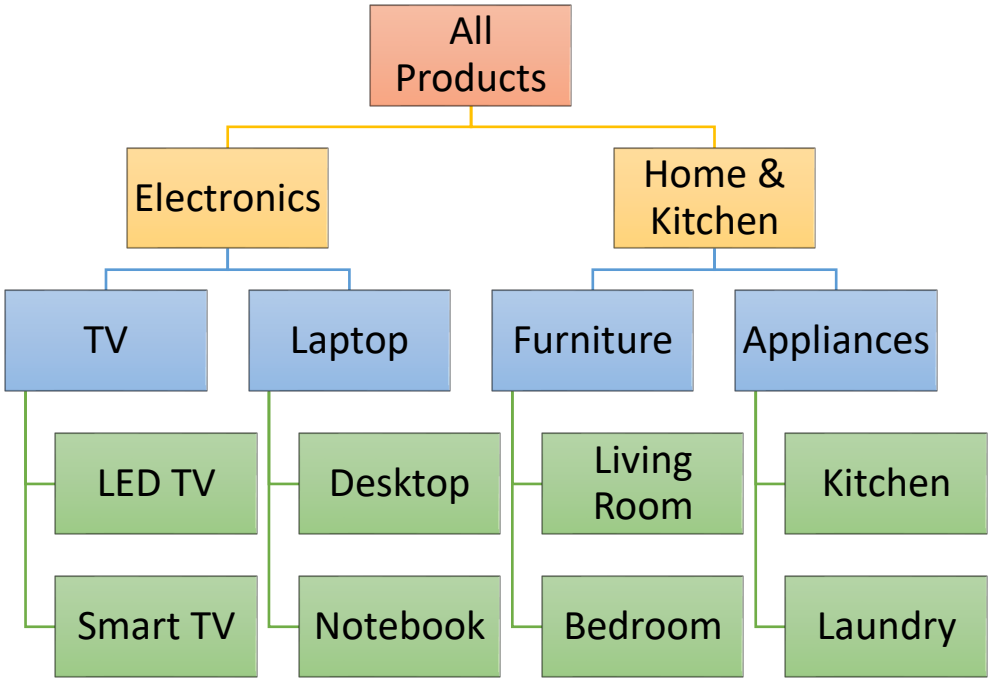


Figure 2. 1 Binary Tree 1 for problem 1

Quantitative model:

The binary tree can also be used quantitatively to represent sales data. Each node can store the number of items sold for the particular category.

In an E-commerce platform, a binary tree can be utilized not just to represent the categories (qualitative data), but also to represent various numerical data such as inventory levels or average ratings of products under each category.

To illustrate this, I've used a binary tree for the 'Electronics' category, but this time with inventory levels:

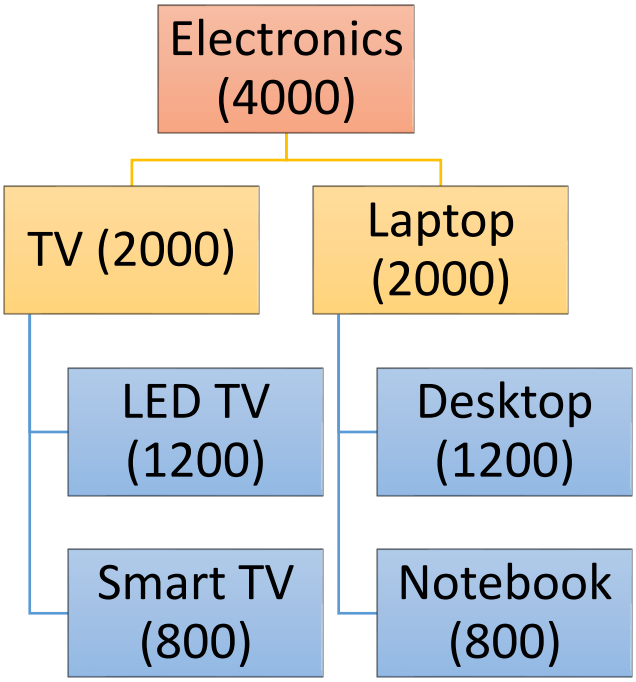


Figure 2. 2 Binary Tree 2 for problem 1

In this case, the numbers in parentheses represent the number of items in stock for each category. For example, there are 2000 TVs in stock in total, of which 1200 are LED TVs and 800 are Smart TVs. Similarly, there are 2000 Laptops in stock in total, of which 1200 are Desktops and 800 are Notebooks.

This binary tree can be very helpful for inventory management. With this structure, a company can quickly check the inventory status and determine if more items need to be ordered for a particular category. In this way, the binary tree not only represents the hierarchical structure of the categories but also provides quantitative data about the number of items available in each category.

Problem 2 :

Let's consider a decision-making process in sports that has multiple levels of decision-making criteria.

Qualitative model:

We can represent this hierarchy of decisions using a binary tree, where each node is a decision criterion. The root is the main decision to be made, and each child node represents a sub-decision or factor contributing to the parent decision.

I'll explain a Qualitative Binary Tree Model based on the Football Match Statistics example. In a football match, a binary tree can be used to represent various qualitative data about the match events. I'll illustrate this with a binary tree representing the significant events in a hypothetical football match.

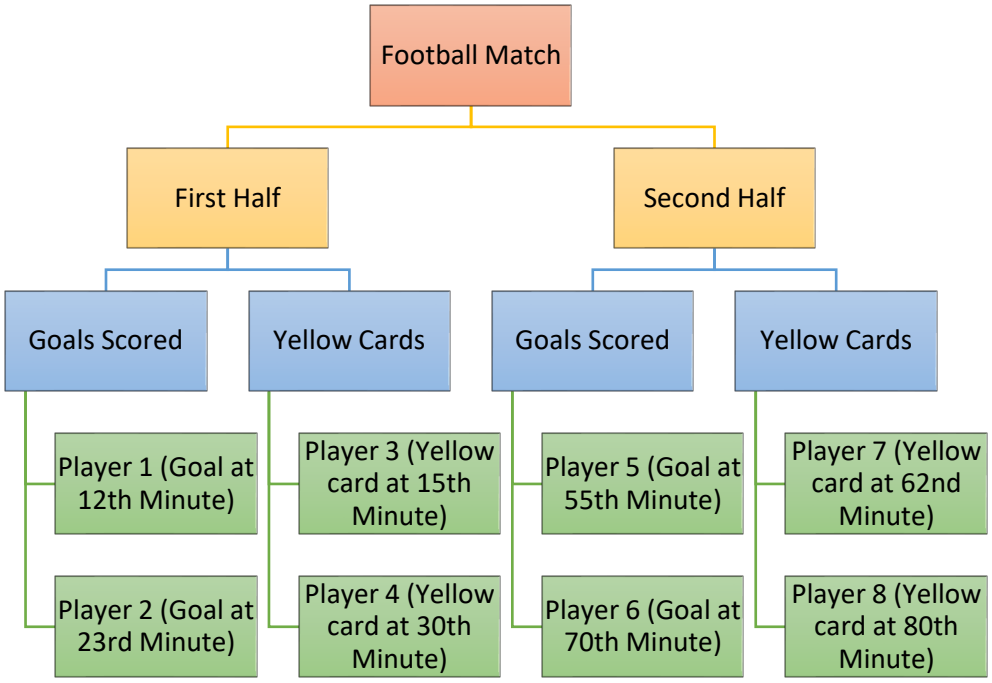


Figure 2. 3 Binary Tree 1 for problem 2

This binary tree helps in quickly accessing the events of the match. The left subtree represents the events of the first half, and the right subtree represents the events of the second half. Under each half, the left child node represents the goals scored and the right child node represents the yellow cards given. Each of these nodes further has left and right child nodes representing the details of each event.

I'll use approach to split the game into different components rather than periods of play:

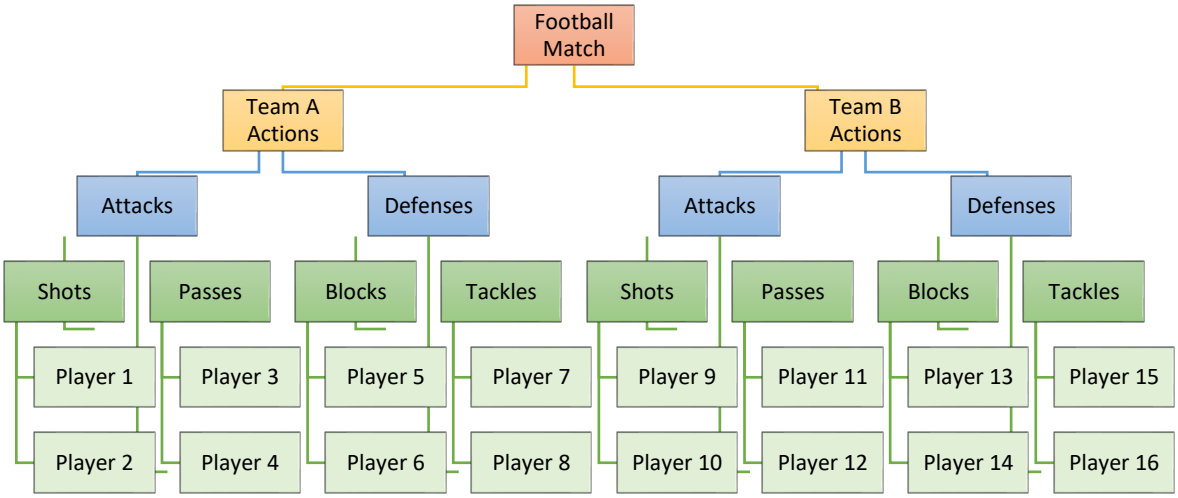


Figure 2. 4 Binary Tree 2 for problem 2

This tree splits the match into "Team A Actions" and "Team B Actions". These nodes further split into "Attacks" and "Defenses" nodes. Each of these nodes splits into more detailed actions such as "Shots", "Passes", "Blocks", and "Tackles". Each of these nodes further splits into player nodes (P1 to P16), representing the players who executed each action.

Quantitative model:

Each node could also hold a quantitative score indicating the outcome of that decision. The binary tree helps in organizing and structuring the decision-making process, and provides a clear visualization of the decision hierarchy. I'll expand on the application of a binary tree in the context of a game like chess.

A binary tree can be used to quantify the game of chess by representing the various potential moves and counter moves that can be made from a given position. Each node in the binary tree would represent a board state, and each edge would represent a move.

Let's consider a simplified scenario in a game of chess. Assume that there are only two pieces left on each side: the king and a pawn. Our objective is to model the next possible moves and evaluate them.

In this quantitative model, we can label the nodes with specific values to represent the evaluation of the board state from the perspective of one player. Positive values might indicate an advantageous state, while negative values might indicate a disadvantageous state.

To make it even more concrete, let's imagine a partial tree with 3 levels:

Level 0 → The root node could represent the current board state and has a value of 0 (since we don't know yet who has the advantage).

Level 1 → In the first level, the two children of the root represent the two possible moves for Player 1.

- The left child might represent moving the pawn forward, which could put pressure on the opponent's king and thus be advantageous. This could be represented with a value of 1.
- The right child might represent moving the king, which could potentially expose the king to danger. This could be represented with a value of -1.

Level 2 → In the second level, each node represents a potential countermove by Player 2 in response to each of Player 1's possible moves.

- For example, the children of the left node in the first level (the scenario where Player 1 moved the pawn) could represent either attacking the pawn with the king (potentially dangerous, thus -2) or moving the king away (neutral, thus 0).
- The children of the right node in the first level (the scenario where Player 1 moved the king) could represent either attacking the king (very advantageous, thus 2) or moving the pawn (neutral, thus 0).

By constructing such a binary tree, a player can explore the potential outcomes of different moves, predict the opponent's responses, and thus make a more informed decision about the best move.

Here is a textual representation of the binary tree for the above chess game scenario:

Level 0:

Root - Current state of the board (Value = 0)

Level 1:

Left Child of Root - Player 1 moves the pawn forward (Value = 1)

Right Child of Root - Player 1 moves the king (Value = -1)

Level 2:

Left Child of Left Child of Root - Player 2 attacks the pawn with the king (Value = -2)

Right Child of Left Child of Root - Player 2 moves the king away (Value = 0)

Left Child of Right Child of Root - Player 2 attacks the king (Value = -2)

Right Child of Right Child of Root - Player 2 does nothing (Value = 0)

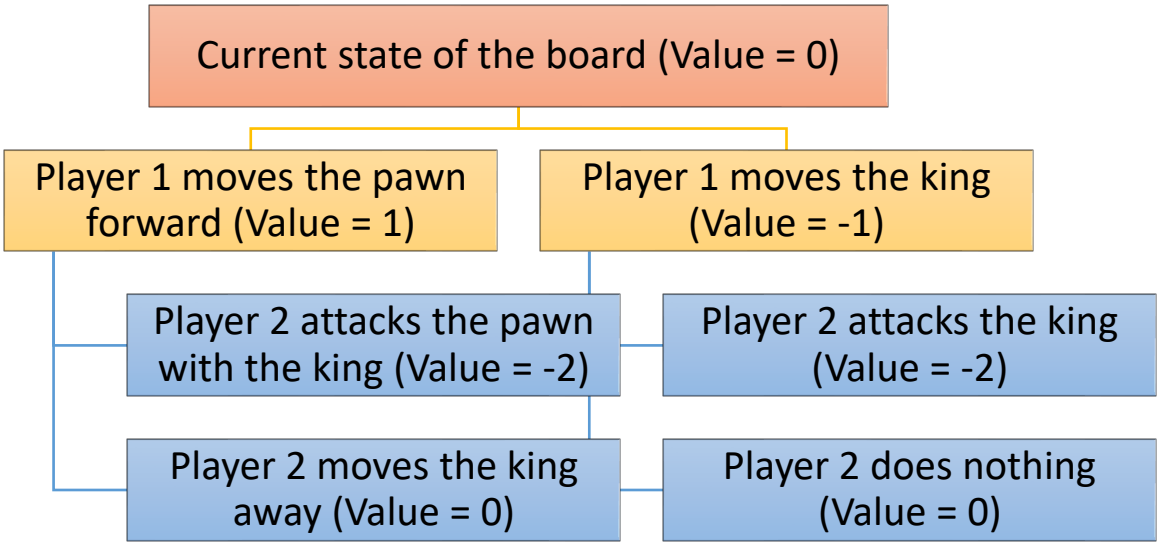


Figure 2. 5 Binary Tree 3 for problem 2

Let's consider a stock trading problem. A binary tree can be used to model different possible trading decisions based on two choices at each time - buying a stock or selling a stock. Each node in the tree could represent the net profit (or loss) at each step.

For example, consider a scenario where a trader has the option to buy or sell a single share of a particular stock each day. The price of the stock for the next three days is known to be $5, $7, and $9.

Here's a possible textual representation of a binary tree for this scenario:

Level 0:

Root - Initial state, no shares owned (Value = $0)

Level 1:

Left Child of Root - Buy 1 share at $5 (Value = -$5)

Right Child of Root - Do not buy (Value = $0)

Level 2:

Left Child of Left Child of Root - Sell share at $7 (Value = $2)

Right Child of Left Child of Root - Do not sell, buy another share at $7 (Value = -$12)

Left Child of Right Child of Root - Buy 1 share at $7 (Value = -$7)

Right Child of Right Child of Root - Do not buy (Value = $0)

Level 3:

Left Child of Left Child of Left Child of Root - Buy 1 share at $9 (Value = -$7)

Right Child of Left Child of Left Child of Root - Do not buy (Value = $2)

Left Child of Right Child of Left Child of Root - Sell 1 share at $9 (Value = -$3)

Right Child of Right Child of Left Child of Root - Do not sell, buy another share at $9 (Value = -$21)

Left Child of Left Child of Right Child of Root - Sell share at $9 (Value = $2)

Right Child of Left Child of Right Child of Root - Do not sell, buy another share at $9 (Value = -$16)

Left Child of Right Child of Right Child of Root - Buy 1 share at $9 (Value = -$9)

Right Child of Right Child of Right Child of Root - Do not buy (Value = $0)

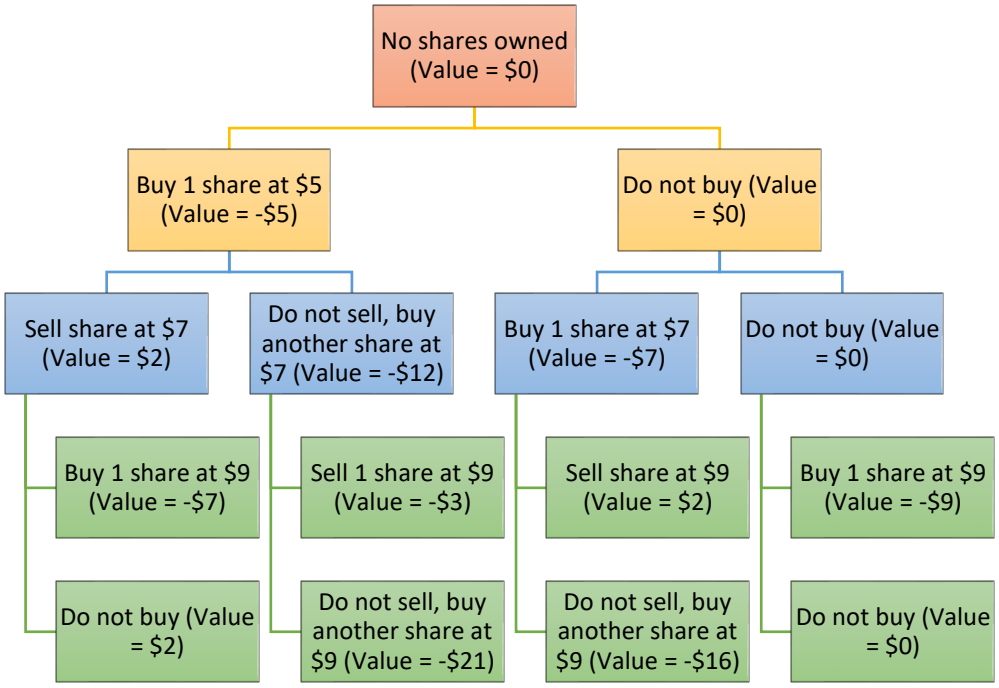


Figure 2. 6 Binary Tree 4 for problem 1

This tree continues expanding with each subsequent level representing further buying or selling decisions, each node holding a value representative of the net profit or loss at that point. The trader could use this decision tree to help decide their best strategy.

Part 2

Question 1

Dijkstra's Algorithm is a famous greedy algorithm used to find the shortest path from a single source vertex to all other vertices in the weighted graph (graph with non-negative edges). This is how Dijkstra's algorithm works:

1. Create a set that keeps track of vertices included in the shortest path tree (SPT), which at first includes only the source vertex.
2. Assign a tentative distance value to every vertex from the source vertex. Initialize the distance of the source vertex as 0, and set all other vertices to infinity as they are initially unknown.
3. For the current vertex, consider all of its unvisited neighbors and calculate their tentative distances through the current vertex. If the new distance is less than the current assigned value (this is the 'relaxation' procedure), update the shortest distance.
4. After considering all of the unvisited neighbors of the current vertex, mark the current node as visited. A visited node will not be checked again.
5. If the destination vertex has been marked visited or if the smallest tentative distance among the vertices left is infinity (when the queue is empty), the algorithm has finished.
6. Otherwise, select the unvisited vertex that is marked with the smallest tentative distance, set it as the new "current vertex", and go back to step 3.

At the end of the algorithm, the shortest path from the source vertex to all other vertices in the graph can be determined.

In this assignment question we've to use the Dijkstra's algorithm for a directed weighted graph with all non-negative edge weights.

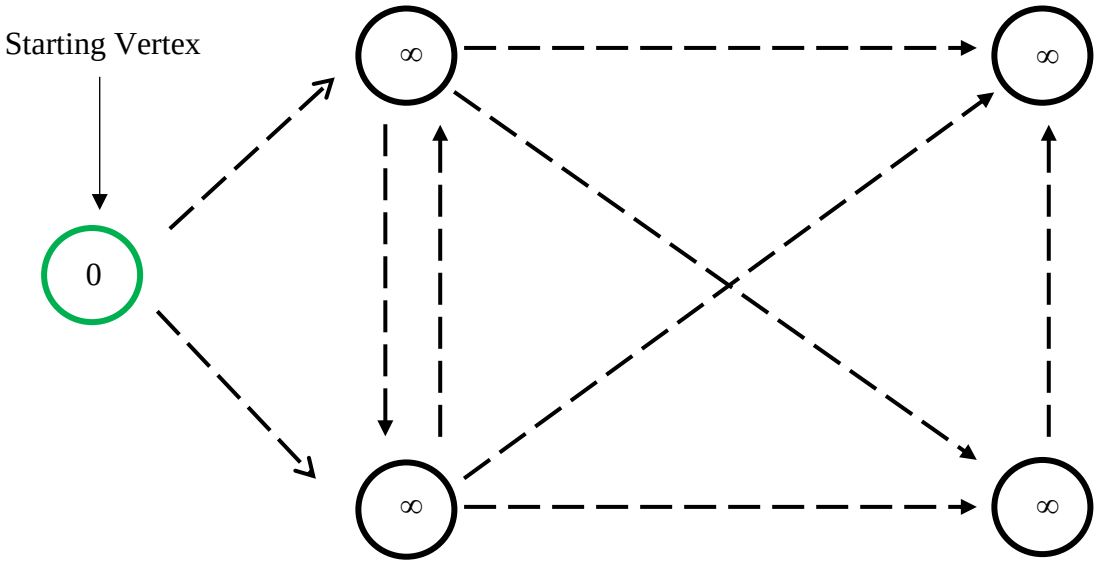
So this is the a step-by-step guide on how I construct this graph and run Dijkstra's algorithm on it.

1. Vertices (Nodes): We have five vertices A, B, C, D, E.
2. Edges: Draw directed edges as mentioned with their weights.
3. Starting Vertex: Vertex E is our starting point.

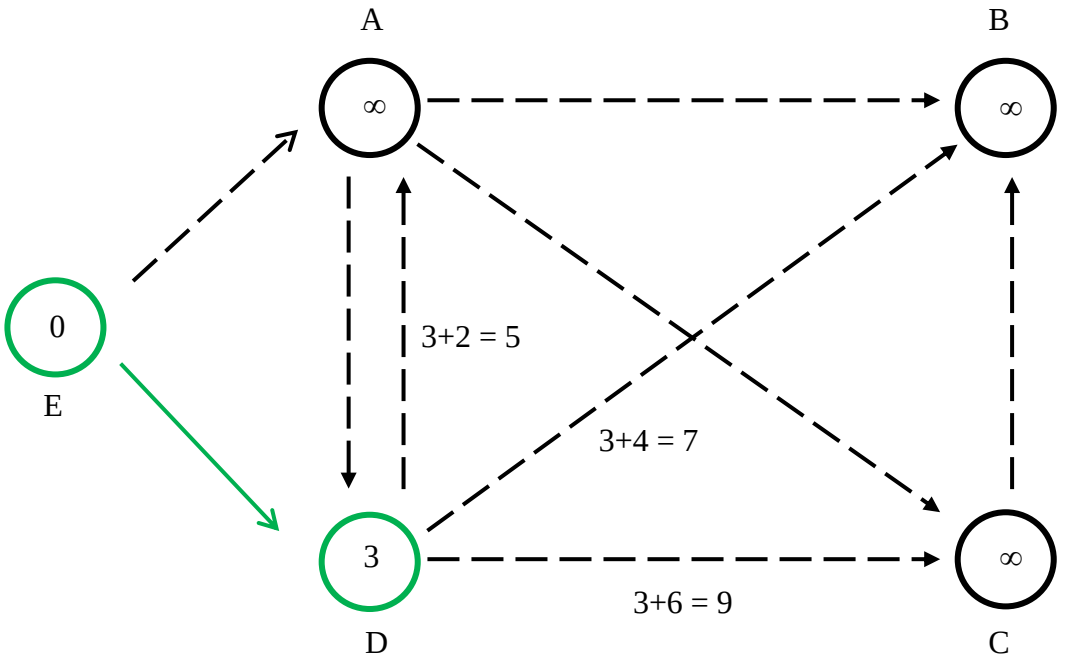
After drawing the graph, I can start Dijkstra's algorithm from vertex E:

1. I can initialize the distance to start node to 0, and for all other nodes to infinity. The set of visited nodes is empty initially.
2. I can repeat the following steps until all nodes have been visited:
    - Select the unvisited node with the smallest distance (at the start, this would be E), mark it as visited, and update the distances to all its adjacent nodes.
    - For each adjacent node, if the sum of the distance to the current node and the edge weight to the adjacent node is less than the current distance to the adjacent node, update the distance.
3. After running the algorithm, the shortest path from E to each vertex is the value stored for each vertex.

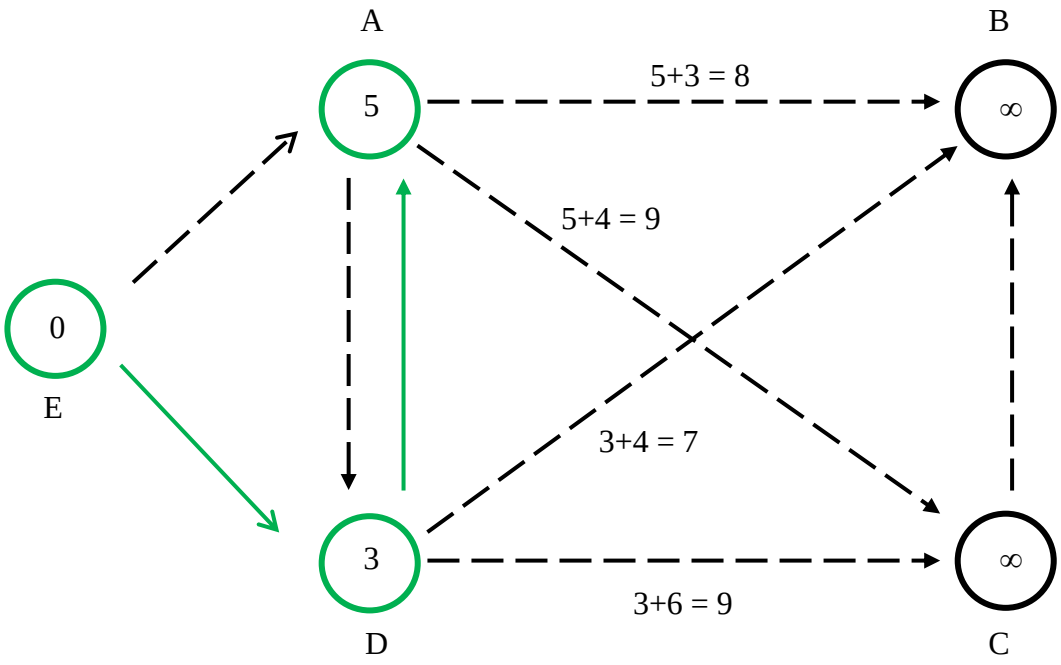
Question 2



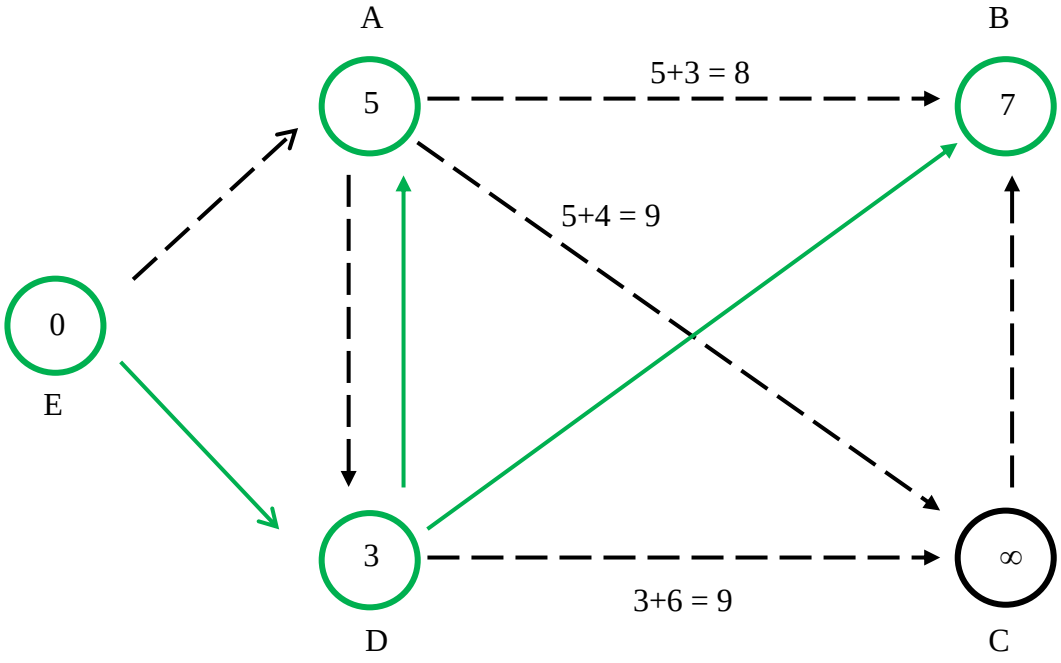
I initialized the distance to start node to 0, and for all other nodes to infinity. The set of visited nodes is empty initially.



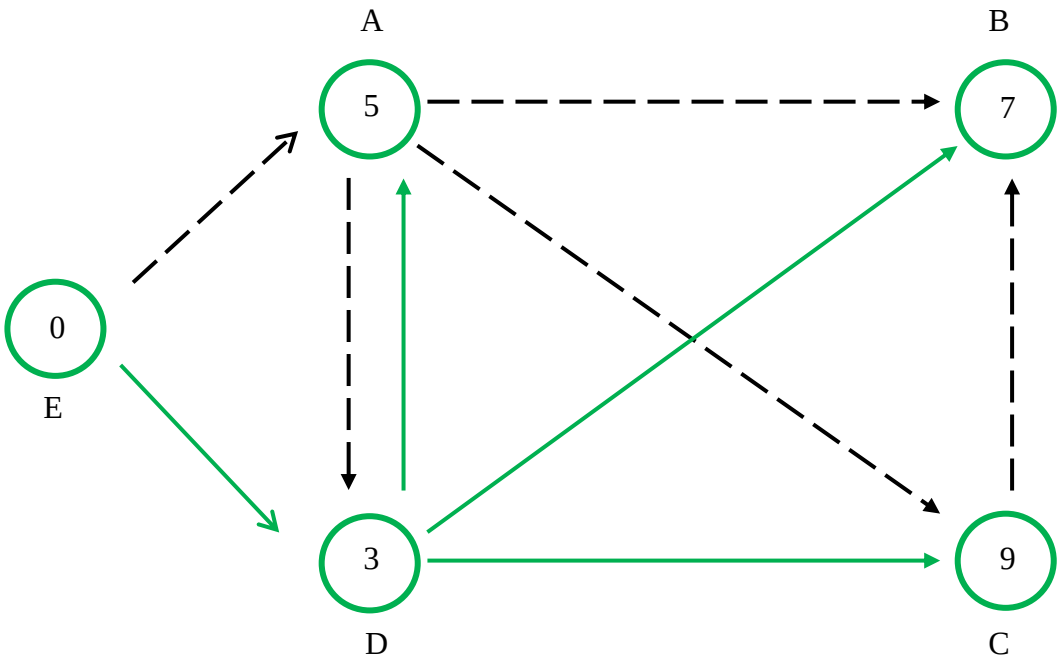
D is the unvisited node with the shortest distance from E. So I marked it as visited, and update the distances to all its unvisited adjacent nodes.



A is the unvisited node with the shortest distance from D. So I marked it as visited, and update the distances to all its unvisited adjacent nodes.



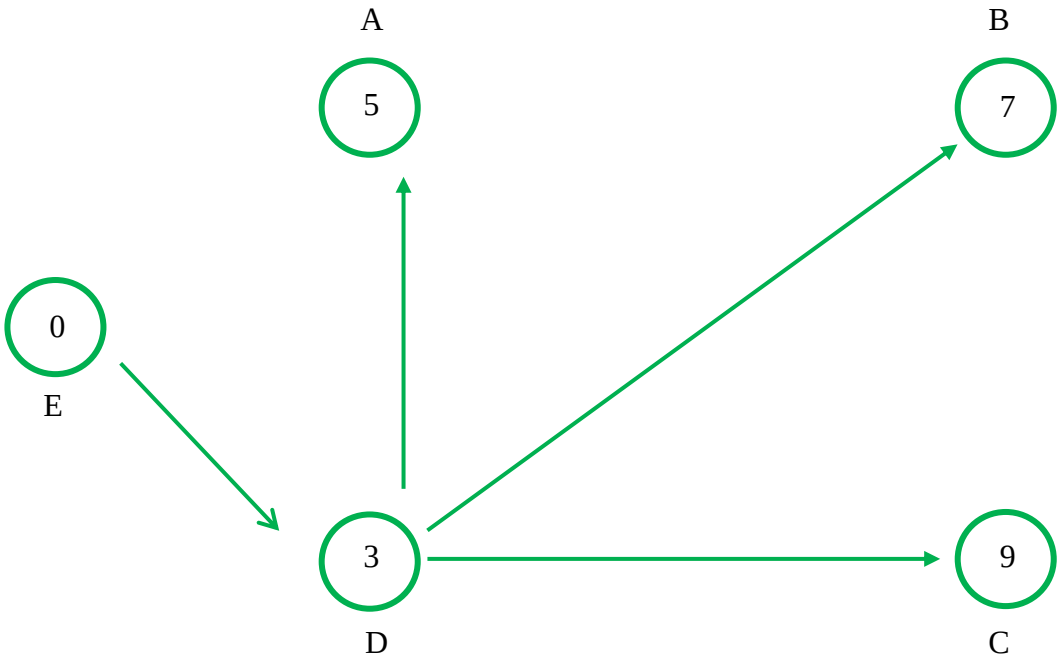
B is the unvisited node with the shortest distance from D. So I marked it as visited, and update the distances to all its unvisited adjacent nodes.



C is the unvisited node with the shortest distance from both A and D. So I marked it as visited from D as my preference.

Since all nodes have been visited I stopped the algorithm.

The following spanning tree is the shortest path spanning tree for the given directed graph.



Here's what I got after running Dijkstra's:

- From E to A, the shortest path is E → D → A with a total weight of 5.
- From E to B, the shortest path is E → D → B with a total weight of 7.
- From E to C, the shortest path is E → D → C with a total weight of 9.
- From E to D, the shortest path is E → D with a total weight of 3.
- From E to E, the shortest path is just E with a total weight of 0.

Part 3

Eulerian Cycle:

An Eulerian cycle, also known as an Eulerian circuit, in an undirected graph is a cycle that visits every edge exactly once and starts and ends at the same vertex.

To check whether an undirected graph has an Eulerian cycle, we need to follow these steps:

1. We need to ensure that the graph is connected. This means there is a path between every pair of vertices. If the graph is not connected, it cannot have an Eulerian cycle.
2. We need to check the degree of every vertex in the graph. The degree of a vertex is the number of edges that touch it. For the graph to have an Eulerian cycle, every vertex must have an even degree. If one or more vertices have an odd degree, the graph cannot have an Eulerian cycle.

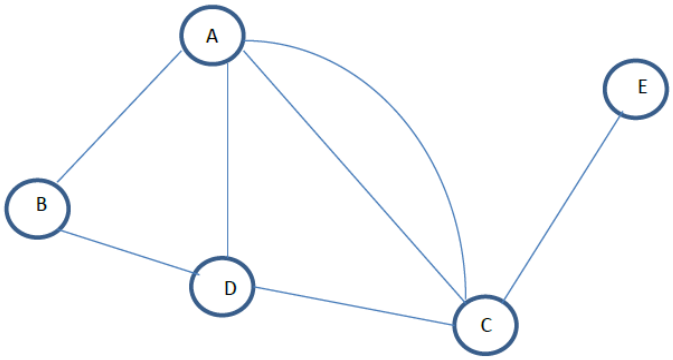
These two conditions together are also sufficient: if a graph is connected and all vertices have an even degree, it has an Eulerian cycle.

Hamiltonian Cycle:

A Hamiltonian cycle in an undirected graph is a cycle that visits every vertex exactly once and starts and ends at the same vertex.

Determining whether such a cycle exists in a given graph is known as the Hamiltonian cycle problem, and it's famously difficult. In fact, it's NP-complete, meaning that no efficient solution is known and it's believed that none exists.

Question 1 – (i)



d (A) = 4

d (B) = 2

d (C) = 4

d (D) = 3

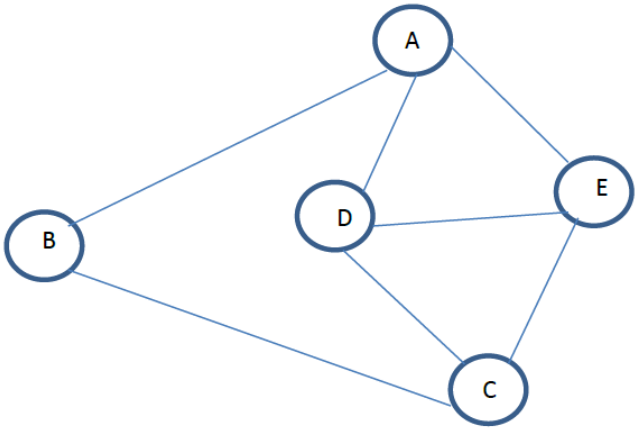
d (E) = 1

This graph is connected.

For the graph to have an Eulerian cycle, every vertex must have an even degree. Since D and E vertices have an odd degree, the graph does not have an Eulerian cycle.

The graph does not have a Hamiltonian cycle. Because there is no cycle that visits every vertex exactly once and starts and ends at the same vertex. Due to node E is connected with node C with 1 edge is it not possible to draw Hamiltonian cycle for the given graph.

Question 1 – (ii)



d (A) = 3

d (B) = 2

d (C) = 3

d (D) = 3

d (E) = 3

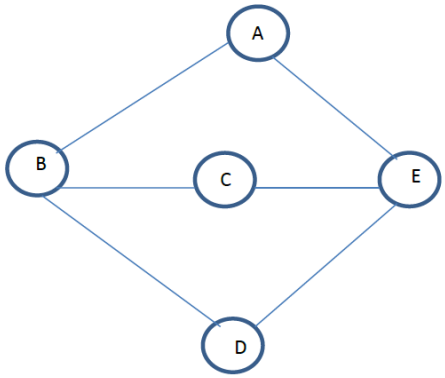
This graph is connected.

For the graph to have an Eulerian cycle, every vertex must have an even degree. Since A, C, D and E vertices have an odd degree, the graph does not have an Eulerian cycle.

The graph does have a Hamiltonian cycle.

The Hamiltonian Cycle for the above graph is = {A, D, E, C, B, A}

Question 1 – (iii)



d (A) = 2

d (B) = 3

d (C) = 2

d (D) = 2

d (E) = 3

This graph is connected.

For the graph to have an Eulerian cycle, every vertex must have an even degree. Since A, C, D and E vertices have an odd degree, the graph does not have an Eulerian cycle.

The graph does not have a Hamiltonian cycle.

Part 4

Question 1

Proof:

Proof by contradiction.

Let G be the smallest planar graph (in terms of vertex count) that cannot be colored with five different colors.

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

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

On the neighbors of v, at most four colors have been used. Then there is at least one color available for v.

So G can be colored with five different colors, which is a contradiction.

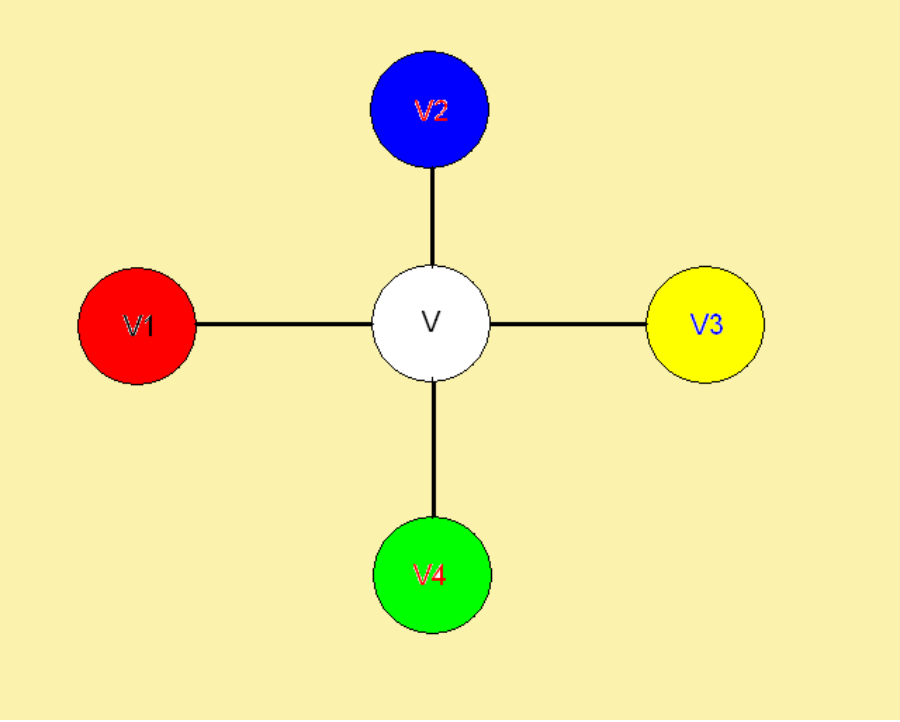


Figure 2. 7 5-Color Theorem figure 1

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

If two of v's neighbors have the same color, then there is a color available for v.

So we can assume that all the vertices adjacent to v are colored with the numbers 1,2,3,4,5 in clockwise sequence.

Consider all the vertices (and all the edges between them) to be colored with colors 1 and 3.

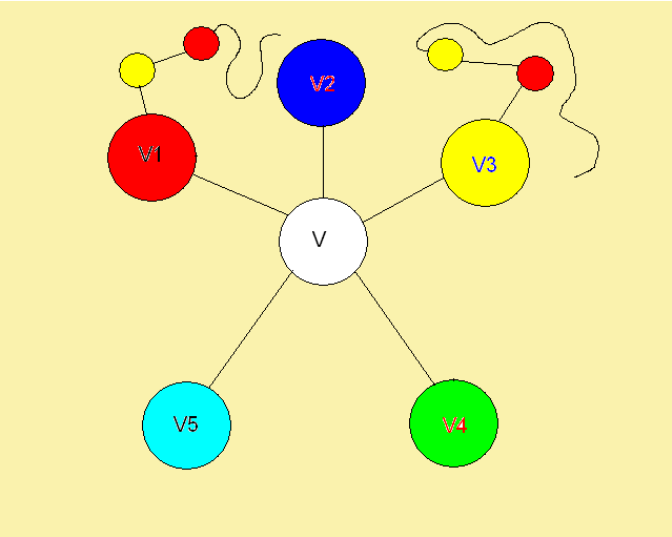


Figure 2. 8 5-Color Theorem figure 2

If we disconnect this subgraph G and v1 and v3 are in different components, we can switch the colors 1 and 3 in the component with v1.

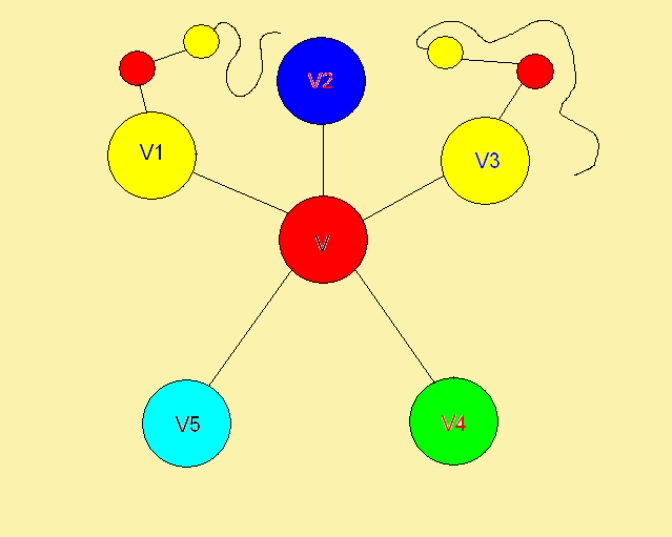


Figure 2. 9 5-Color Theorem figure 3

This will remain a 5-coloring of G-v. Furthermore, in this new 5-coloring, v1 is colored with color 3 and v3 is still colored with color 3. Color 1 would be available for v, which would be a contradiction.

Hence v1 and v3 must be in the same component in that subgraph, i.e. there is a path from v1 to v3 such that every vertex on this path is colored with either color 1 or color 3.

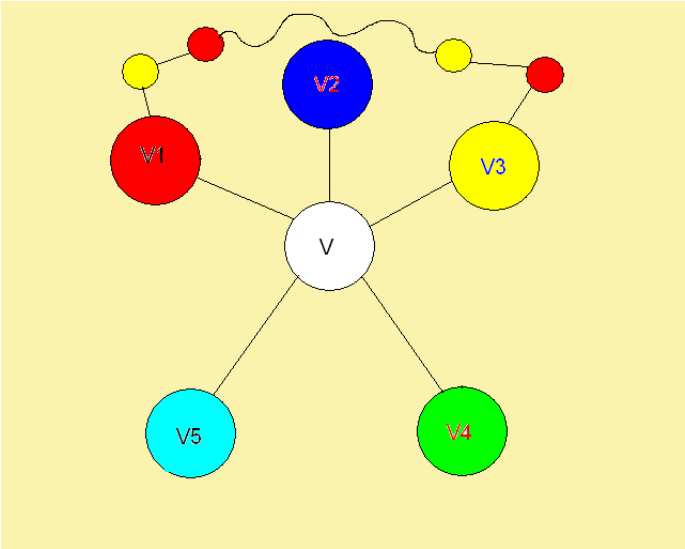


Figure 2. 10 5-Color Theorem figure 4

Consider all of the vertices (and edges) that are colored with colors 2 and 4. If v2 and v4 do not belong to the same connected component, we can swap the colors in the chain beginning with v2 and utilize the leftover color for v.

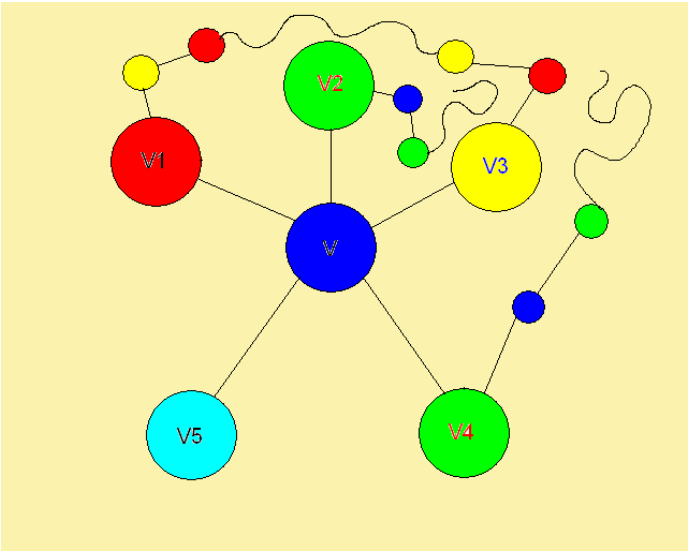


Figure 2. 11 5-Color Theorem figure 5

If they are on the same connected component, there is a path from v2 to v4 with every vertex having either color 2 or color 4.

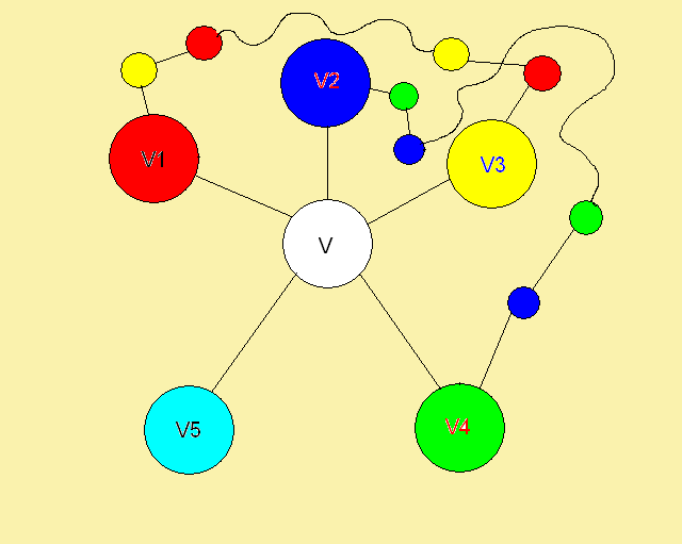


Figure 2. 12 5-Color Theorem figure 6

This implies that there must be two intersecting edges. This contradicts the planarity of the graph and hence concludes the proof.

The Five Color Theorem is a result from graph theory which states that every planar graph (a graph that can be drawn on a plane without edges crossing) can be colored with no more than five colors in such a way that no two adjacent vertices share the same color. This theorem is a weaker version of the Four Color Theorem, which is significantly more difficult to prove.

Here is a proof of the Five Color Theorem:

Proof:

1. **Base Case:** Any planar graph with fewer than or equal to five vertices can obviously be colored using five colors. This is our base case for induction.
2. **Inductive Step:** Now, let's assume that the theorem holds for all planar graphs with n or fewer vertices, where n >= 5. We need to show that if a planar graph has (n+1) vertices, then it can be colored with at most five colors.
3. By Euler's formula for planar graphs (v - e + f = 2, where v = vertices, e = edges, f = faces), any planar graph with v vertices and e edges satisfies e <= 3v - 6.
4. Since each vertex has at least one edge, the average degree of a vertex in a planar graph is less than 6 (2e/v < 6). This means that there must be at least one vertex (call it v) of degree five or less.
5. Remove v from the graph, leaving a planar graph with n vertices. By our inductive hypothesis, this smaller graph can be colored with five colors.
6. Now, put v back into the graph. Since v has at most five neighbors, and since the graph is five-colored, there are at most five colors already in use around v. If there are fewer than five colors in use, then v can be colored with one of the unused colors.
7. The problematic case is when v has five neighbors each of a different color. In this case, construct a graph whose vertices are color classes, with two vertices connected by an edge if they are both adjacent to v. Since v has degree 5, this graph has 5 vertices and 5 edges, and thus is a cycle of length 5.

8. If we can find two nonadjacent color classes in this graph, then we can swap the colors in one of those classes in the larger graph, making a color available for v. If we cannot, then this graph must be a complete graph, which would mean that v has neighbors of five different colors. But in this case, there is a "color path" from one color class to another, avoiding the vertex v.
9. We can switch the colors along this path, then color v with the now unused color.

This completes the proof: every planar graph can be colored with no more than five colors.

Activity 3

Part 1

Question 1

Problem 1: Computer Science - Search Algorithms

Search algorithms in computer science often employ Boolean logic to refine the search criteria and provide more accurate results. Consider a situation where we're developing a search algorithm for a library system. The user may want to search for a book that matches multiple categories, such as "Fiction" and "Published before 2000".

We can represent this problem in Boolean algebra where "Fiction" is represented by A and "Published before 2000" is represented by B. The user wants a book that is both A and B. This is a binary AND operation.

Boolean Chart:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

In the chart, '0' denotes 'False' (or 'No') and '1' denotes 'True' (or 'Yes').

So, a book would only be selected if it satisfies both conditions (is a fiction and published before 2000).

Let's dive a little deeper into the computer science problem involving the library system. Suppose our library system has three categories that books can fall under: "Fiction", "Published before 2000", and "Author is J.K. Rowling". In Boolean terms, these can be represented as A, B, and C respectively.

Let's consider a complex search query: Find a book that is either a Fiction and published before 2000 or is authored by J.K. Rowling. This can be represented in Boolean algebra as (A AND B) OR C.

Boolean Chart:

A	B	C	(A AND B) OR C
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

In this chart, '0' denotes 'False' (or 'No') and '1' denotes 'True' (or 'Yes').

So, a book would be selected if it is both a fiction and published before 2000 or is authored by J.K. Rowling.

For a visual representation, consider the following Venn diagram:

- Circle A represents "Fiction" books
- Circle B represents "Published before 2000" books
- Circle C represents "Authored by J.K. Rowling" books

The intersection of circle A and B (the area where both A and B overlap but C doesn't) represents books that are both "Fiction" and "Published before 2000".

Now, to represent the Boolean operation (A AND B) OR C, the intersection of circle A and B and also the entire area of Circle C. This area represents the books that would be selected by our search query, i.e., either a book falls in the intersection of A and B or it is in C.

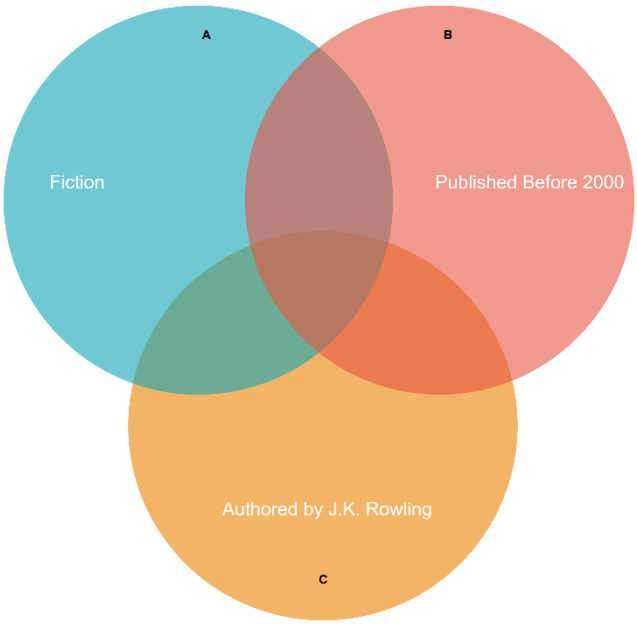


Figure 3. 1 Venn diagram for problem 1

Logic Gate:

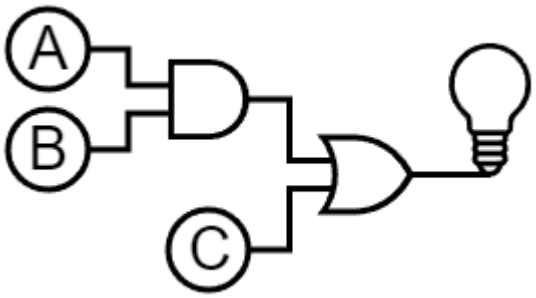


Figure 3. 2 Logic Gate for problem 1

Problem 2: Electrical Engineering - Logic Gates Design

In electrical engineering, especially in the design of digital circuits, Boolean algebra is used to manipulate the inputs and outputs of logic gates.

Consider a situation where we need to design an alarm system for a secure door. The alarm should sound if either the door is open OR the security system is activated. In this case, the state of the door can be represented by A and the state of the security system can be represented by B. The OR operation in Boolean algebra represents our condition perfectly.

Boolean Chart:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Here, the alarm will sound in any situation other than both the door being closed (A=0) and the security system being deactivated (B=0).

Let's consider an automated lighting system in a smart home that uses Boolean logic to decide when to turn on the lights. The system has three inputs: it's dark outside (A), there's someone at home (B), and it's not bedtime (C). The output (D) is whether the lights should be on or not. This situation can be modeled with the Boolean equation D = A AND B AND C.

We can explain this problem through a truth table, which represents all possible combinations of inputs (A, B, C) and the resulting output (D).

Boolean Chart:

A (Dark)	B (Home)	C (Not Bedtime)	D (Lights on)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Here 1 represents True and 0 represents False.

The lights will only turn on when it's dark, there's someone at home, and it's not bedtime (the last row in the table).

Let's take a closer look at the other scenarios as well.

Scenario: Nobody's Home during Daytime Here, A (Dark outside) = 0, B (Someone at home) = 0, and C (Not bedtime) = 1. Substituting these into our Boolean equation D = A AND B AND C, we get D = 0 AND 0 AND 1, which simplifies to D = 0. This means the lights will not turn on, which is exactly what we want when nobody's home and it's not dark outside.
Scenario: Somebody's Home but it's Bedtime Here, A (Dark outside) = 1, B (Someone at home) = 1, and C (Not bedtime) = 0. Substituting these into our Boolean equation, we get D = 1 AND 1 AND 0, which simplifies to D = 0. In this case, the lights should not be on since it's bedtime, which aligns with the output of our Boolean equation.

Scenario: Somebody's Home, it's Dark, and Not Bedtime Here, A (Dark outside) = 1, B (Someone at home) = 1, and C (Not bedtime) = 1. Substituting these into our Boolean equation, we get D = 1 AND 1 AND 1, which simplifies to D = 1. So, the lights will turn on, which is exactly the scenario we want for this situation.
Scenario: It's Dark outside, but Nobody's Home Here, A (Dark outside) = 1, B (Someone at home) = 0, and C (Not bedtime) = 1. Substituting these into our Boolean equation, we get D = 1 AND 0 AND 1, which simplifies to D = 0. So, the lights will not turn on, as expected, because even though it's dark and not bedtime, there's no one home.

Logic Gate:

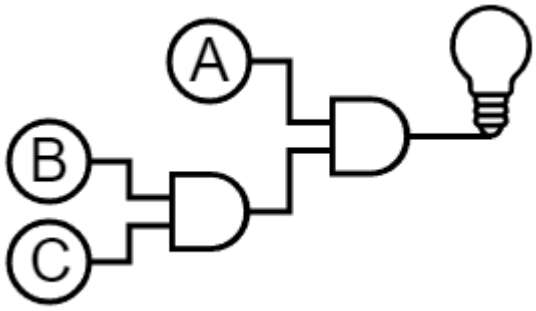


Figure 3. 3 Logic Gate for problem 2

Part 2

Question 1 – (i)

Let's denote:

- A: Driver is present = 1
- B: Driver has buckled up = 1
- C: Ignition switch is on = 1
- W: Warning light should turn on = 1

The statement "If the driver is present and the driver has not buckled up and the ignition switch is on, then the warning light should turn on" translates to the Boolean equation W = A AND (NOT B) AND C.

F (A, B, C) = $A \cdot \bar{B} \cdot C$

Here's the truth table for this situation:

Table 3. 1 Part 2 Question 1 – (i) truth table

A	B	C	W
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

In the table, 0 represents false and 1 represents true. As we can see, the warning light (W) turns on (is 1) only when the driver is present (A is 1), the driver has not buckled up (B is 0), and the ignition switch is on (C is 1).

Logic Gate:

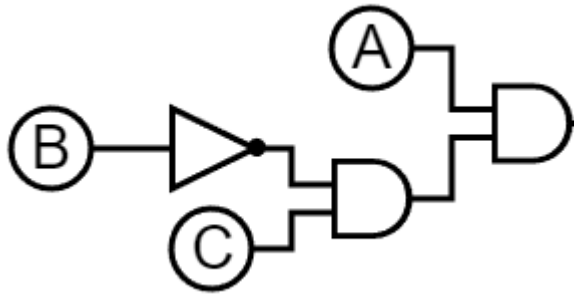


Figure 3. 4 Part 2 Question 1 – (i) logic gate

Question 1 – (ii)

Let's denote:

- R: It rains = 1
- U: You open your umbrella = 1
- W: You will get wet = 1

The statement "If it rains and you don't open your umbrella, then you will get wet" translates to the Boolean equation W = R AND (NOT U).

F (R, U) = $R \cdot \overline{U}$

Here's the truth table for this situation:

Table 3. 2 Part 2 Question 1 – (ii) truth table

R	U	W
0	0	0
0	1	0
1	0	1
1	1	0

In the table, 0 represents false and 1 represents true. As we can see, you will get wet (W is 1) only when it rains (R is 1) and you don't open your umbrella (U is 0).

Logic Gate:

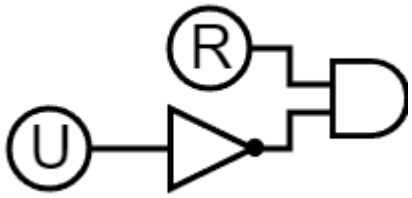


Figure 3. 5 Part 2 Question 1 – (ii) logic gate

Question 2 – (i)

This Boolean expression consists of four terms: A'B'C, AB'C', ABC, A'BC'.

Let's define:

- A, B, C are Boolean variables.
- ' denotes the NOT operation.
- + denotes the OR operation.

The expression can be read as: (NOT A AND NOT B AND C) OR (A AND NOT B AND NOT C) OR (A AND B AND C) OR (NOT A AND B AND NOT C).

Here's the truth table for this Boolean expression:

Table 3. 3 Part 2 Question 2 – (i) truth table

A	B	C	A'B'C	AB'C'	ABC	A'BC'	A'B'C + AB'C' + ABC + A'BC'
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	1
0	1	1	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	0	0	1	0	1

In the table, 0 represents false and 1 represents true. The last column represents the result of the entire Boolean expression for all possible combinations of A, B, and C.

Question 2 – (ii)

This Boolean expression consists of three terms: (A + B' + C), (A + B + C), and (A' + B + C').

Let's define:

- A, B, C are Boolean variables.
- ' denotes the NOT operation.
- + denotes the OR operation.
- ( ) denotes the AND operation.

The expression can be read as: (A OR NOT B OR C) AND (A OR B OR C) AND (NOT A OR B OR NOT C).

Here's the truth table for this Boolean expression:

Table 3. 4 Part 2 Question 2 – (ii) truth table

A	B	C	A+B'+C	A+B+C	A'+B+C'	(A+B'+C)(A+B+C)(A'+B+C')
0	0	0	1	0	1	0
0	0	1	1	1	1	1
0	1	0	0	1	1	0
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	0	0
1	1	0	1	1	1	1
1	1	1	1	1	1	1

In the table, 0 represents false and 1 represents true. The last column represents the result of the entire Boolean expression for all possible combinations of A, B, and C.

Part 3

Question 1 – (i)

Given expression:

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

Step 1: Apply the Distributive Law

= AA + AB + BB + BC + CC + CA

Step 2: Apply the Multiplicative Identities (AA = A, BB = B, CC = C)

= A + AB + B + BC + C + CA

Step 3: Regroup the terms (Commutative Law)

= A + AB + CA + B + BC + C

Step 4: Apply the Distributive Law

= A*(1 + B + C) + B*(1 + C) + C

Step 5: Apply the Additive Identity [A(1+B) = A*1]

= A(1) + B(1) + C

Answer

= A + B + C

Logic Gate as per question:

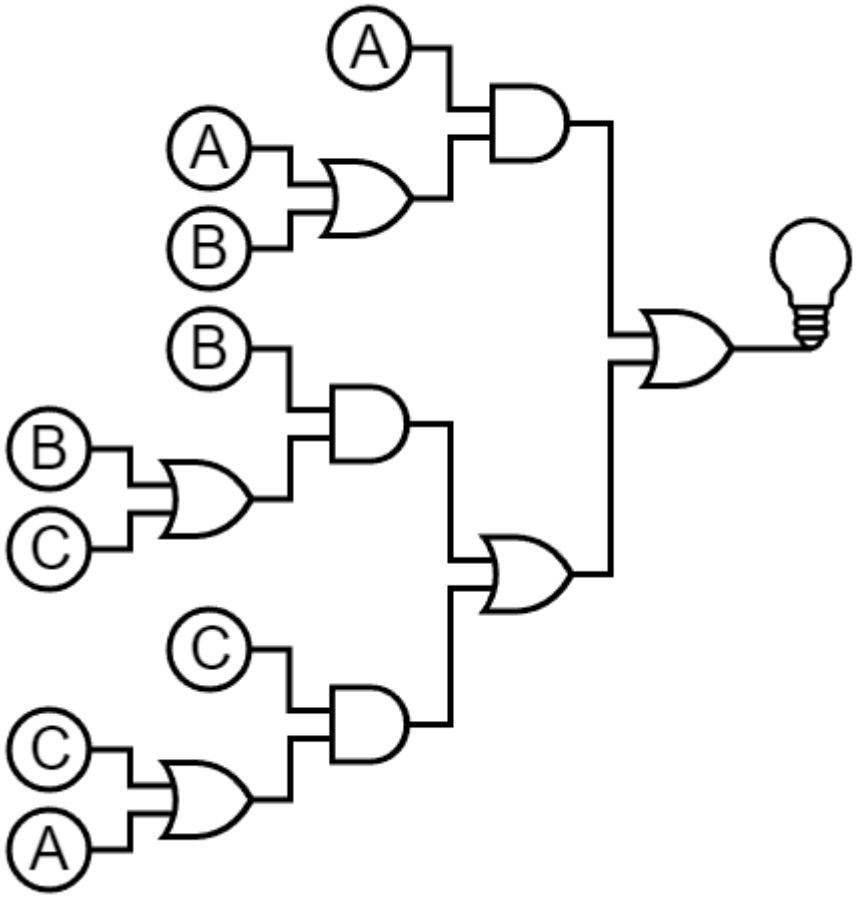


Figure 3. 6 Part 3 Question 1 – (i) logic Gate as per question

Logic Gate as per the answer:

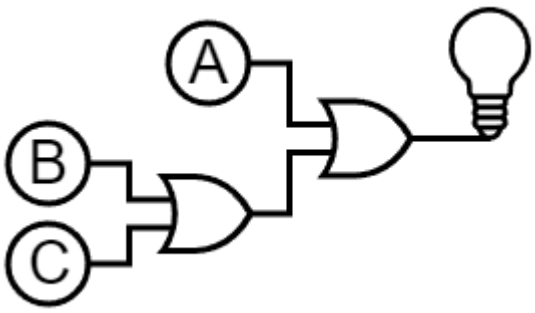


Figure 3. 7 Part 3 Question 1 – (i) logic Gate as per answer

Question 1 – (ii)

Given expression:

= (A+B') (B+C) + (A+B) (C+A')

Step 1: Apply the Distributive Law

= AB + AC + B'B + B'C + AC + AA' + BC + BA'

Step 2: Apply the Multiplicative Identity (B'B = 0)

= AB + AC + 0 + B'C + AC + 0 + BC + BA'

Step 3: Apply the Additive Identity (AB + 0 = AB)

= AB + AC + B'C + AC + BC + BA'

Step 4: Regroup the terms (Commutative Law)

= AB + AC + AC + B'C + BC + BA'

Step 5: Apply the Additive Identity (AB + AB = AB)

= AB + AC + B'C + BC + BA'

Step 6: Regroup the terms (Commutative Law)

= BA + BA' + BC + B'C + AC

Step 7: Apply the Distributive Law

= B(A + A') + BC + B'C + AC

Step 8: Apply the Additive Identity (A + A' = 1)

= B(1) + BC + B'C + AC

Step 9: Apply the Distributive Law

= B (1 + C) + B'C + AC

Step 10: Apply the Additive Identity [A(1+B) = A*1]

= B (1) + B'C + AC

Step 11: Apply the Redundant Literal Rule A + A'B = A + B

= B + C + AC

Step 12: Apply the Distributive Law

= B + C (1+ A)

Step 13: Apply the Additive Identity [A(1+B) = A*1]

= B + C(1)

Answer

= B + C

Logic Gate as per question:

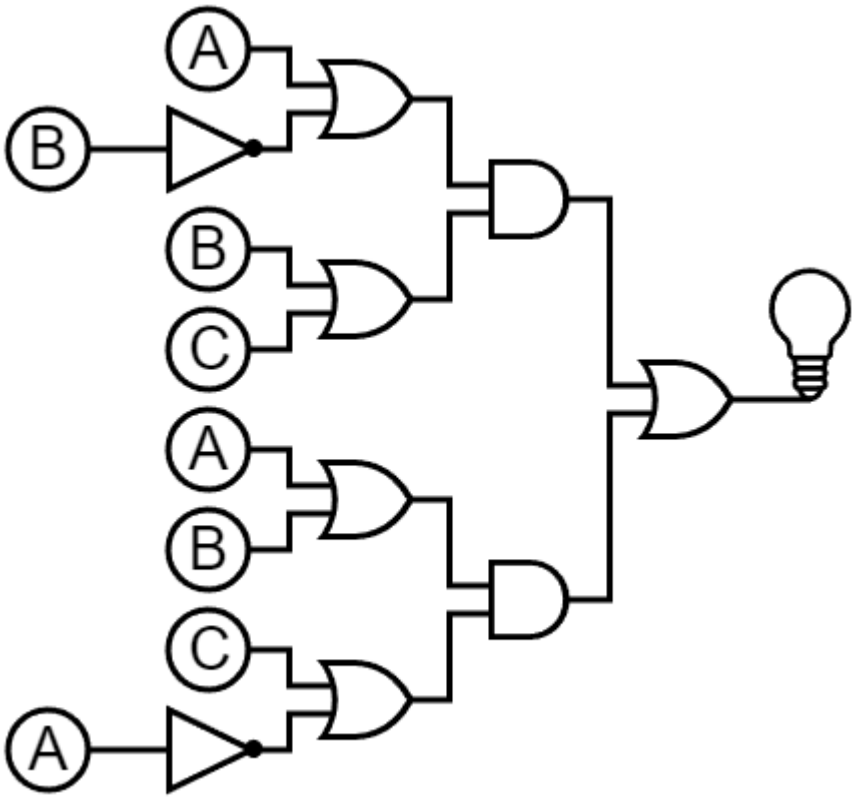


Figure 3. 8 Part 3 Question 1 – (i) logic Gate as per question

Logic Gate as per the answer:

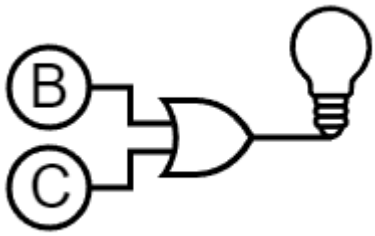


Figure 3. 9 Part 3 Question 1 – (i) logic Gate as per answer

Question 1 – (iii)

Given expression:

= (A + B) (AC + AC') + AB + B

Step 1: Apply the Distributive Law

= AAC + AAC' + BAC + BAC' + AB + B

Step 2: Apply the Multiplicative Identity (AA = A)

= AC + AC' + BAC + BAC' + AB + B

Step 3: Regroup the terms (Commutative Law)

= AC + BAC + AC' + BAC' + AB + B

Step 4: Apply the Distributive Law

= AC(1 + B) + AC'(1 + B) + B (A +1)

Step 5: Apply the Additive Identity [A(1+B) = A*1]

= AC + AC' + B

Step 6: Apply the Distributive Law

= A (C + C') + B

Step 7: Apply the Additive Identity (A + A' = 1)

= A (1) + B

Answer

= A + B

Logic Gate as per the question:

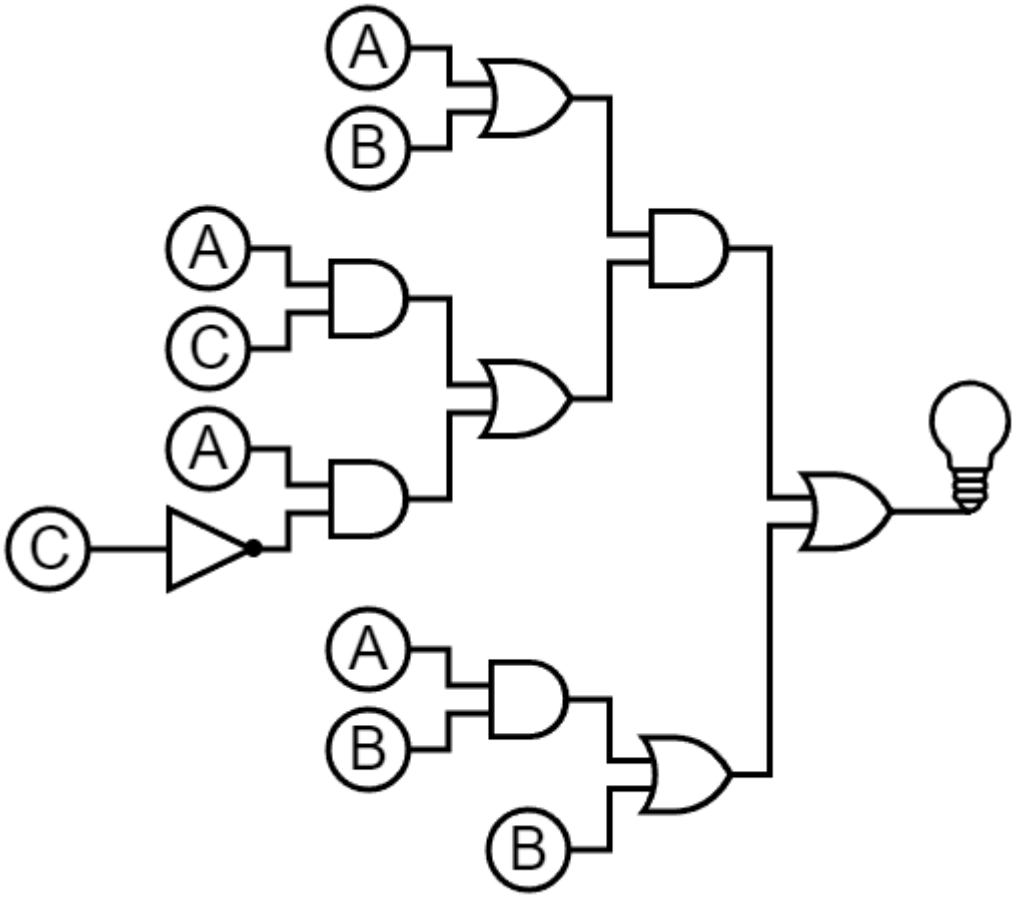


Figure 3. 10 Part 3 Question 1 – (i) logic Gate as per question

Logic Gate as per the answer:

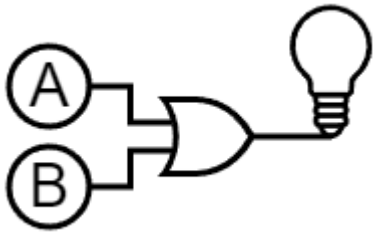


Figure 3. 11 Part 3 Question 1 – (i) logic Gate as per answer

Question 1 – (iv)

Given expression:

= A'(A+B) + (B+A) (A+B')

Step 1: Apply the Distributive Law

= AA' + A'B + BA + BB' + AA + AB'

Step 2: Apply the Multiplicative Identity (AA = A), AA' = 0)

= 0 + A'B + BA + 0 + A + AB'

Step 3: Apply the Additive Identity (AB + 0 = AB)

= A'B + BA + A + AB'

Step 4: Regroup the terms (Commutative Law)

= A'B + AB + A + AB'

Step 5: Apply the Distributive Law

= A'B + A (B + 1 + B')

Step 6: Apply the Additive Identity [A(1+B) = A*1]

= A'B + A (1)

Step 7: Regroup the terms (Commutative Law)

= A + A'B

Step 9: Apply the Redundant Literal Rule A + A'B = A + B

= A + B

Answer

= A + B

Logic Gate as per the question:

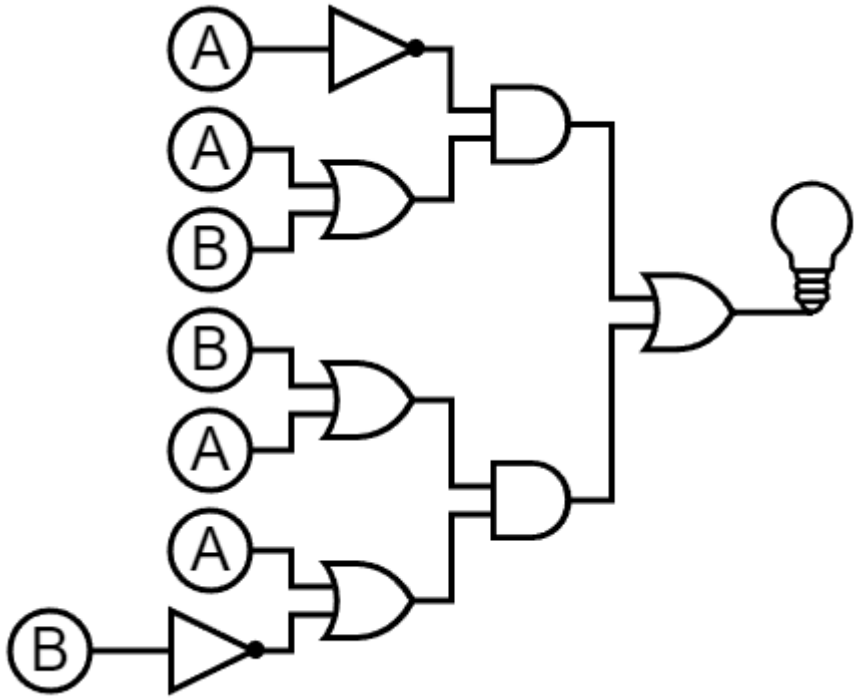


Figure 3. 12Part 3 Question 1 – (i) logic Gate as per question

Logic Gate as per the answer:

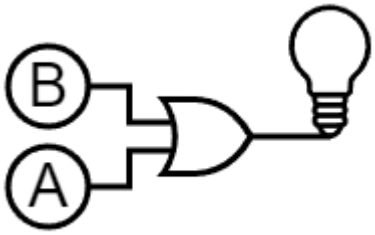


Figure 3. 13 Part 3 Question 1 – (i) logic Gate as per answer

Part 4

Question (a) – (i)

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

Before writing the appropriate standard form (SOP/POS) of Boolean expression let's put the K Map values into a chart according to the given values of the function for each combination of A, B, and C.

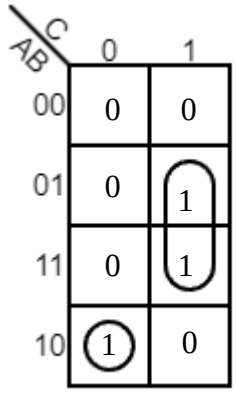
Table 3. 5 Part 4 Question (a) – (i) truth table with minterms and maxterms

A	B	C	F (A,B,C)	Minterm	Maxterm
0	0	0	0		A+B+C
0	0	1	0		A+B+C'
0	1	0	0		A+B'+C
0	1	1	1	A'BC	
1	0	0	1	AB'C'	
1	0	1	0		A'+B+C'
1	1	0	0		A'+B'+C
1	1	1	1	ABC	

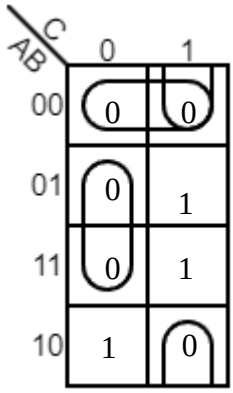
SOP according to chart = A'BC + AB'C' + ABC

POS according to chart = (A+B+C) (A+B+C') (A+B'+C) (A'+B+C') (A'+B'+C)

K-Map for above data :



Minimum SOP according to K-Map = AB'C' + BC



Minimum POS according to K-Map = (B+C') (B'+C) (A+B)

Question (a) – (ii)

This is the design of the circuit using AND, NOT and OR gates for SOP form.

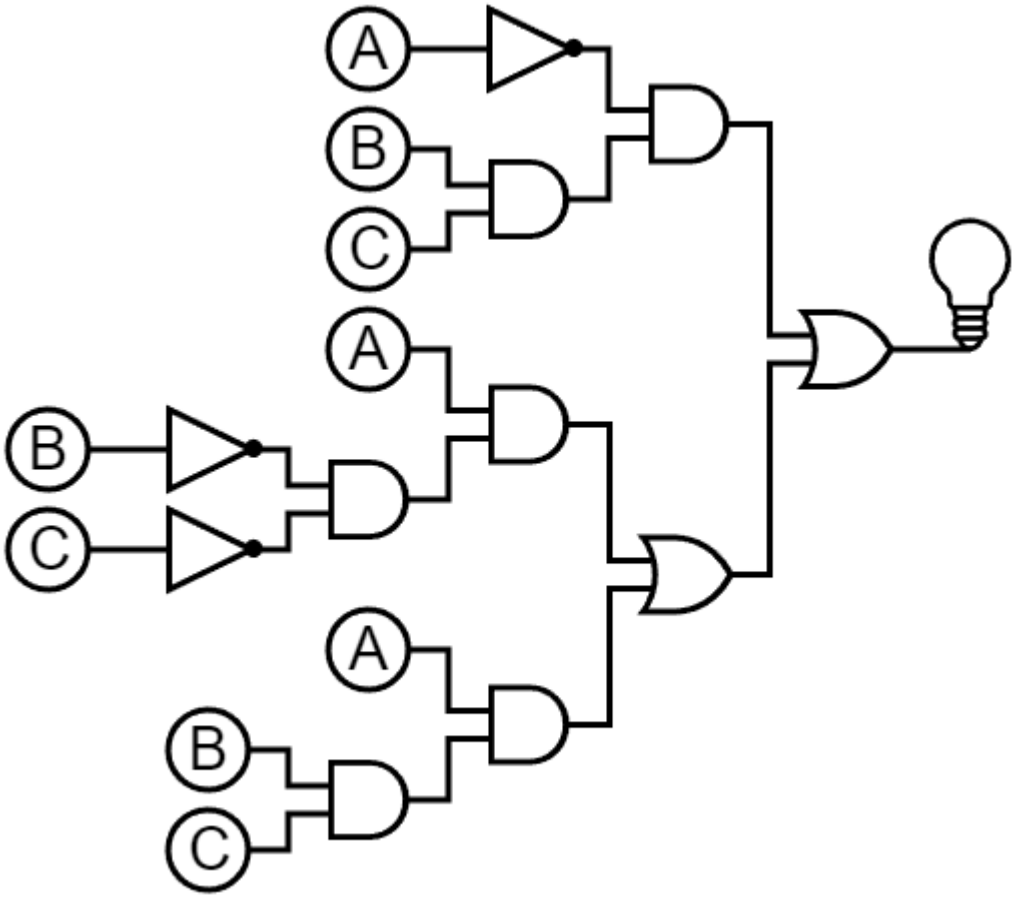


Figure 3. 14 Part 4 Question (a) – (ii) the circuit using AND, NOT and OR gates for SOP form

Question (a) – (iii)

The designed circuit only by using NAND gates since the standard form obtained in part (i) is SOP.

The given SOP (Sum of Products) expression is:

SOP according to chart = A'BC + AB'C' + ABC

First, let's convert this SOP according to chart expression to use only NAND gates, which are universal gates. This can be done by using De Morgan's law.

The NAND expression for SOP according to chart can be given by double negation:

[(A'BC)' . (AB'C')' . (ABC)'] '

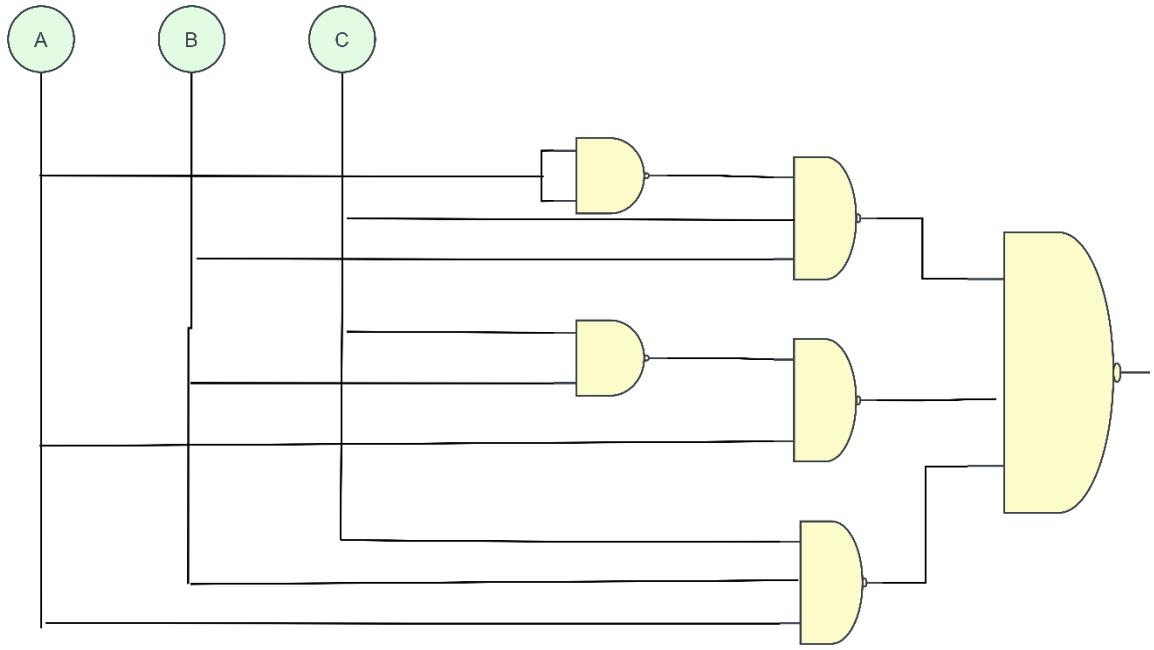


Figure 3. 15 The designed circuit only by using NAND for the standard SOP form obtained in part (i)

Minimum SOP according to K-Map = AB'C' + BC

First, let's convert this minimum SOP expression to use only NAND gates, which are universal gates. This can be done by using De Morgan's law.

The NAND expression for minimum SOP can be given by double negation:

[(AB'C')' . (BC)'] '

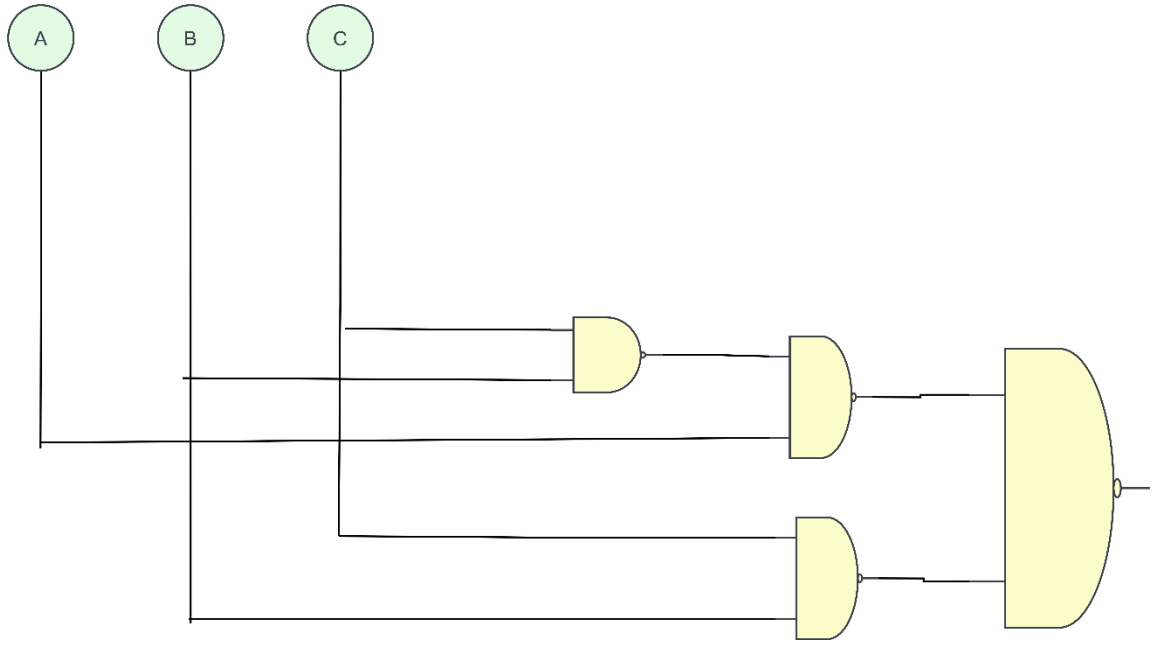


Figure 3. 16 The designed circuit only by using NAND for the SOP form obtained from K-Map

Question (b) – (i)

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

Before writing the appropriate standard form (SOP/POS) of Boolean expression let's put the K Map values into a chart according to the given values of the function for each combination of A, B, and C.

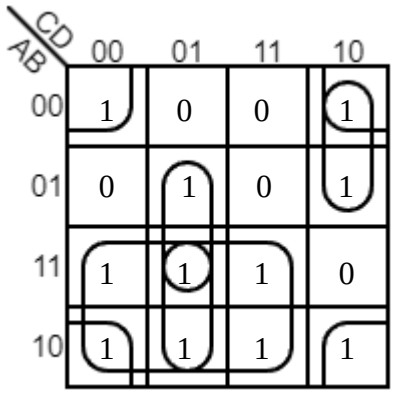
Table 3. 6 Part 4 Question (b) – (i) truth table with minterms and maxterms

A	B	C	D	F (A,B,C,D)	Minterm	Maxterm
0	0	0	0	1	A'B'C'D'	
0	0	0	1	0		A+B+C+D'
0	0	1	0	1	A'B'CD'	
0	0	1	1	0		A+B+C'+D'
0	1	0	0	0		A+B'+C+D
0	1	0	1	1	A'BC'D	
0	1	1	0	1	A'BCD'	
0	1	1	1	0		A+B'+C'+D'
1	0	0	0	1	AB'C'D'	
1	0	0	1	1	AB'C'D	
1	0	1	0	1	AB'CD'	
1	0	1	1	1	AB'CD	
1	1	0	0	1	ABC'D'	
1	1	0	1	1	ABC'D	
1	1	1	0	0		A'+B'+C'+D
1	1	1	1	1	ABCD	

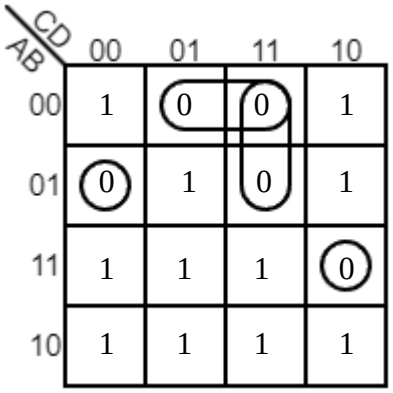
SOP according to chart = A'B'C'D' + A'B'CD' + A'BC'D + A'BCD' + AB'C'D' + AB'C'D + AB'CD' + AB'CD + ABC'D' + ABC'D + ABCD

POS according to chart = (A+B+C+D') (A+B+C'+D') (A+B'+C+D) (A+B'+C'+D') (A'+B'+C'+D)

K-Map for above data :



Minimum SOP according to K-Map = B'D' + AD + AC' + A'CD' + BC'D



Minimum POS according to K-Map = (A + B + D') (A + B' + C + D) (A + C' + D') (A' + B' + C' + D)

Question (b) – (ii)

This is the design of the circuit using AND, NOT and OR gates for SOP form.

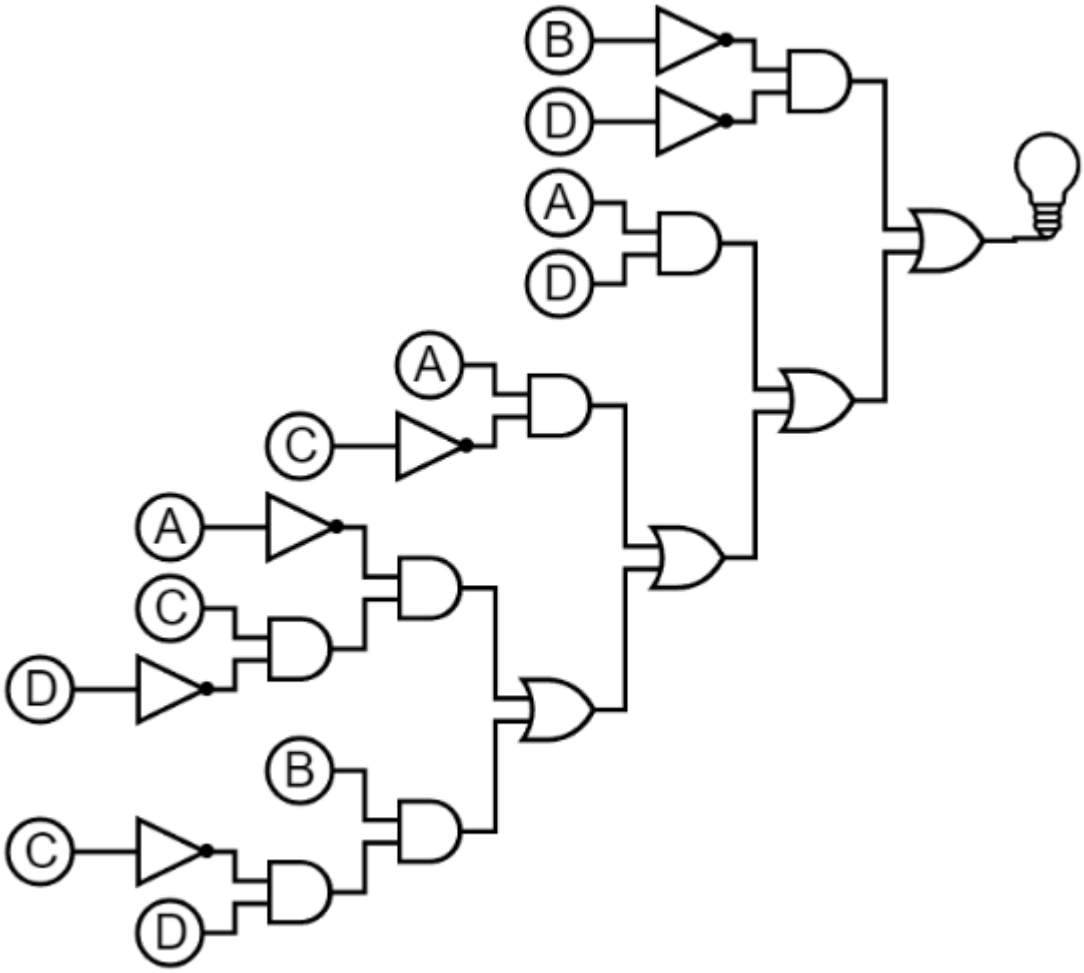


Figure 3. 17 Part 4 Question (b) – (ii) the circuit using AND, NOT and OR gates for SOP form

Question (b) – (iii)

The designed circuit only by using NAND gates since the standard form obtained in part (i) is SOP.

The given SOP (Sum of Products) expression is:

SOP according to chart = A'B'C'D' + A'B'CD' + A'BC'D + A'BCD' + AB'C'D' + AB'C'D + AB'CD' + AB'CD + ABC'D' + ABC'D + ABCD

First, let's convert this SOP according to chart expression to use only NAND gates, which are universal gates. This can be done by using De Morgan's law.

The NAND expression for SOP according to chart can be given by double negation:

[(A'B'C'D')'.(A'B'CD')'.(A'BC'D)'.(A'BCD')'.(AB'C'D')'.(AB'C'D)'.(AB'CD')'.(AB'CD)'.(ABC'D')'.(ABC'D)'.(ABCD)']'

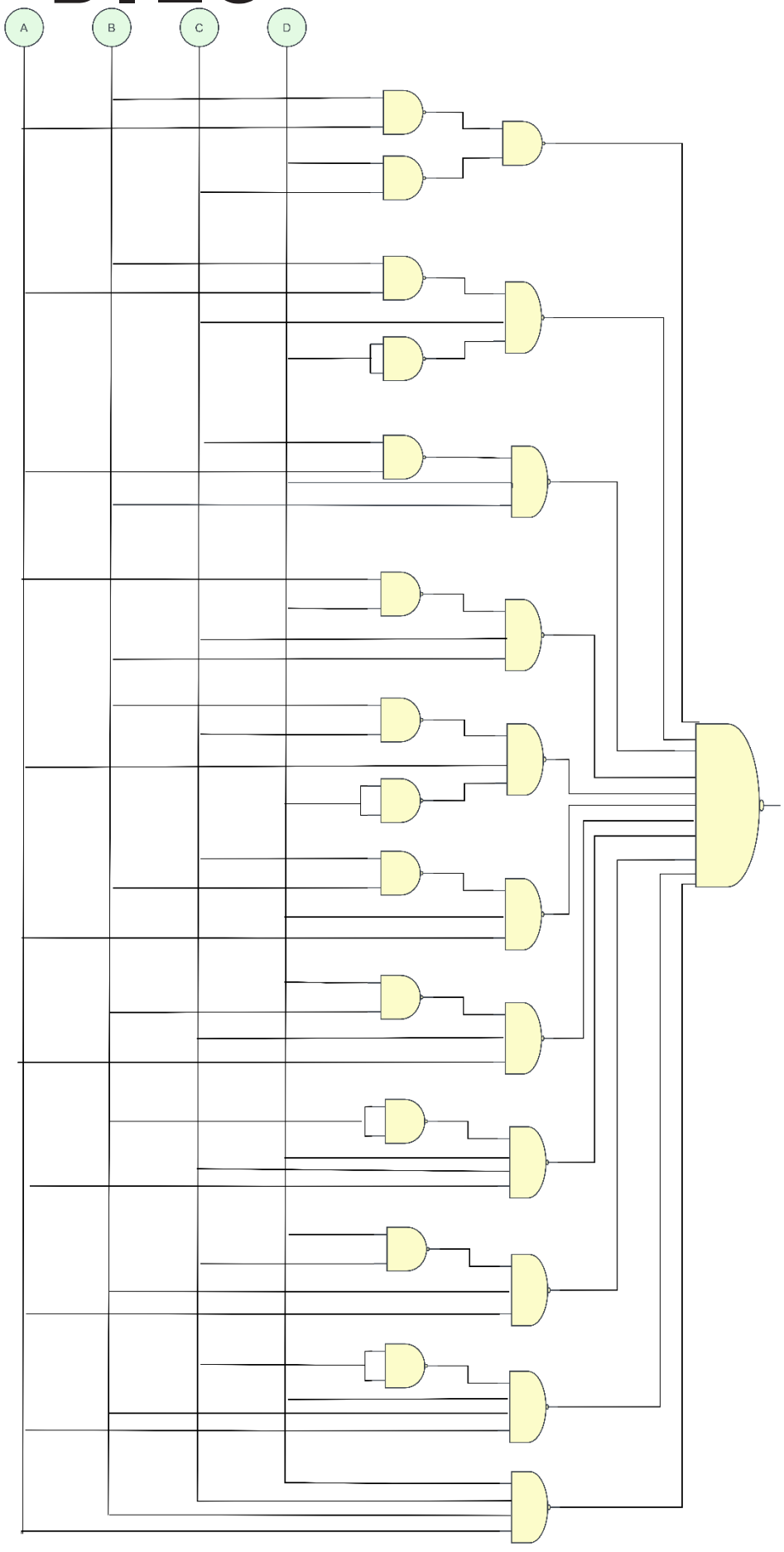


Figure 3. 18 The designed circuit only by using NAND for the standard SOP form obtained in part (i)

Minimum SOP according to K-Map = B'D' + AD + AC' + A'CD' + BC'D

First, let's convert this minimum SOP expression to use only NAND gates, which are universal gates. This can be done by using De Morgan's law.

The NAND expression for minimum SOP can be given by double negation:

[(B'D')' . (AD)' . (AC')' . (A'CD')' . (BC'D)'] '

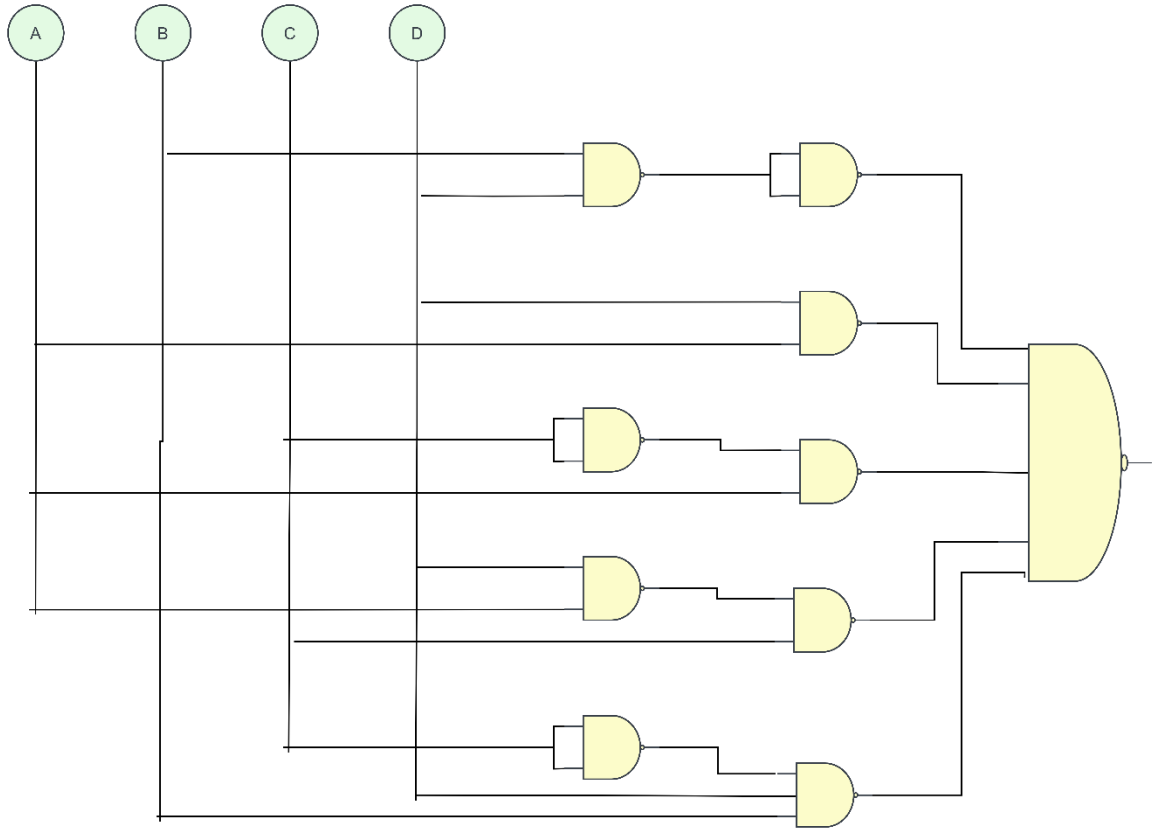


Figure 3. 19 The designed circuit only by using NAND for the SOP form obtained from K-Map

Question (c) – (i)

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

Before writing the appropriate standard form (SOP/POS) of Boolean expression let's put the K Map values into a chart according to the given values of the function for each combination of A, B, and C.

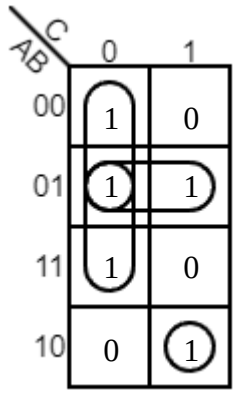
Table 3. 7 Part 4 Question (c) – (i) truth table with minterms and maxterms

A	B	C	F (A,B,C)	Minterm	Maxterm
0	0	0	1	A'B'C'	
0	0	1	0		A+B+C'
0	1	0	1	A'BC'	
0	1	1	1	A'BC	
1	0	0	0		A'+B+C
1	0	1	1	AB'C	
1	1	0	1	ABC'	
1	1	1	0		A'+B'+C'

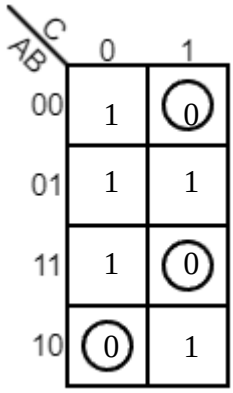
SOP according to chart = A'B'C' + A'BC' + A'BC + AB'C + ABC'

POS according to chart = (A+B+C') (A'+B+C) (A'+B'+C')

K-Map for above data :



Minimum SOP according to K-Map = A'C' + A'B + BC' + AB'C



Minimum POS according to K-Map = (A+B+C') (A'+B'+C') (A'+B+C)

Question (c) – (ii)

This is the design of the circuit using AND, NOT and OR gates for SOP form.

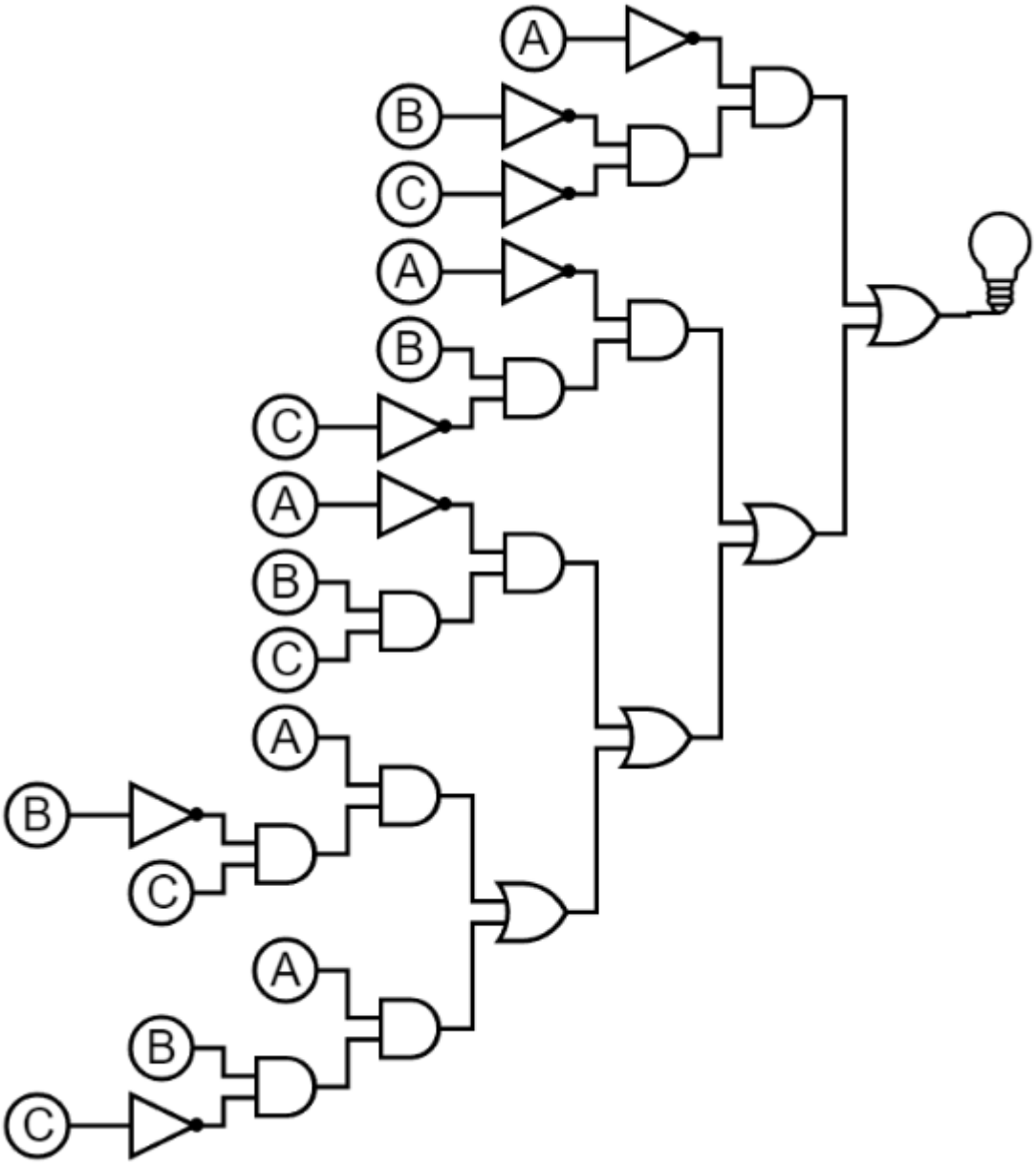


Figure 3. 20 Part 4 Question (c) – (ii) the circuit using AND, NOT and OR gates for SOP form

Question (c) – (iii)

The designed circuit only by using NAND gates since the standard form obtained in part (i) is SOP.

The given SOP (Sum of Products) expression is:

SOP according to chart = A'B'C' + A'BC' + A'BC + AB'C + ABC'

First, let's convert this SOP according to chart expression to use only NAND gates, which are universal gates. This can be done by using De Morgan's law.

The NAND expression for SOP according to chart can be given by double negation:

[(A'B'C')'(A'BC')'(A'BC)'(AB'C)'(ABC')']'

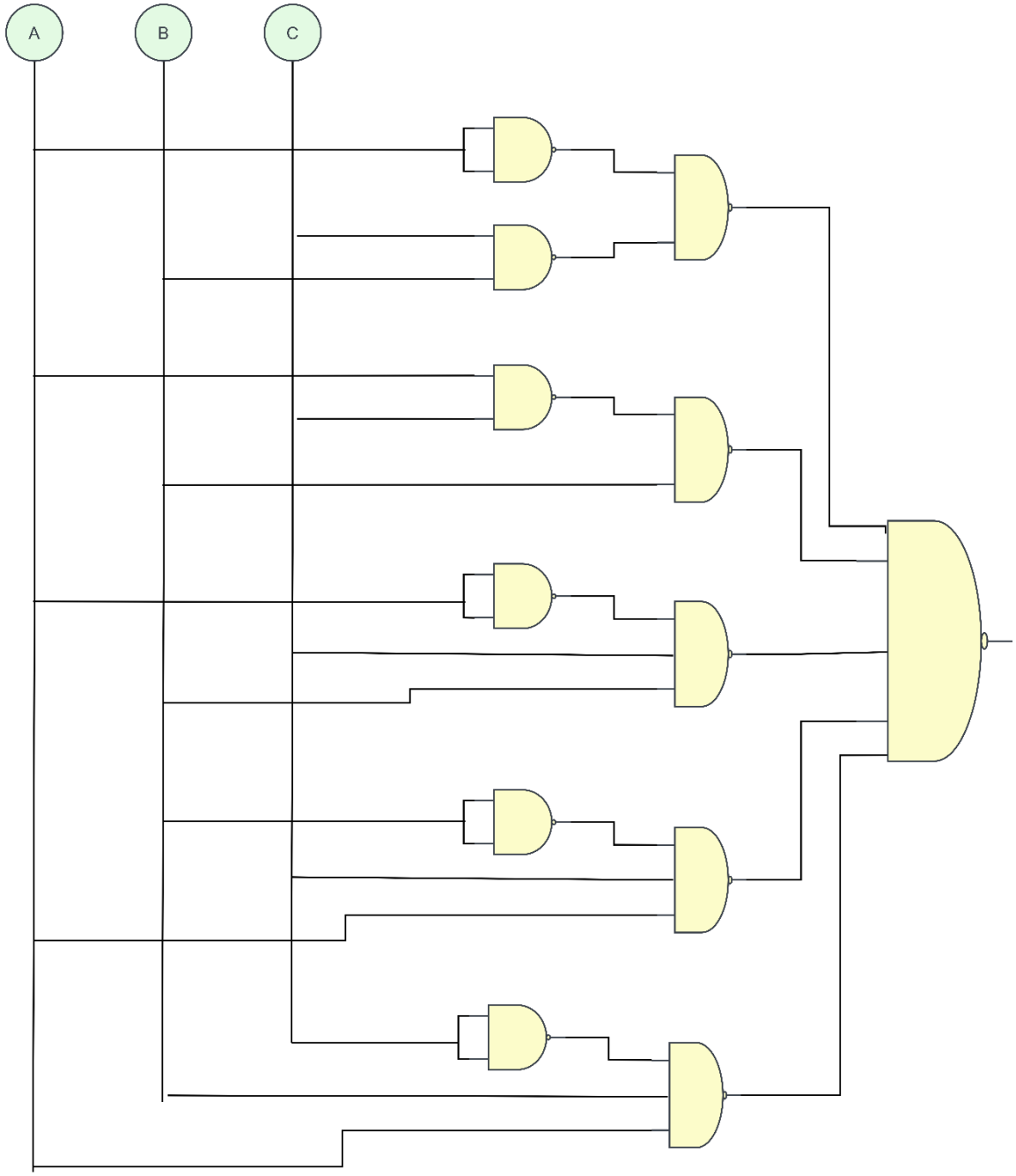


Figure 3. 21 The designed circuit only by using NAND for the standard SOP form obtained in part (i)

Minimum SOP according to K-Map = A'C' + A'B + BC' + AB'C

First, let's convert this minimum SOP expression to use only NAND gates, which are universal gates. This can be done by using De Morgan's law.

The NAND expression for minimum SOP can be given by double negation:

[(A'C')' . (A'B)' . (BC')' . (AB'C)'] '

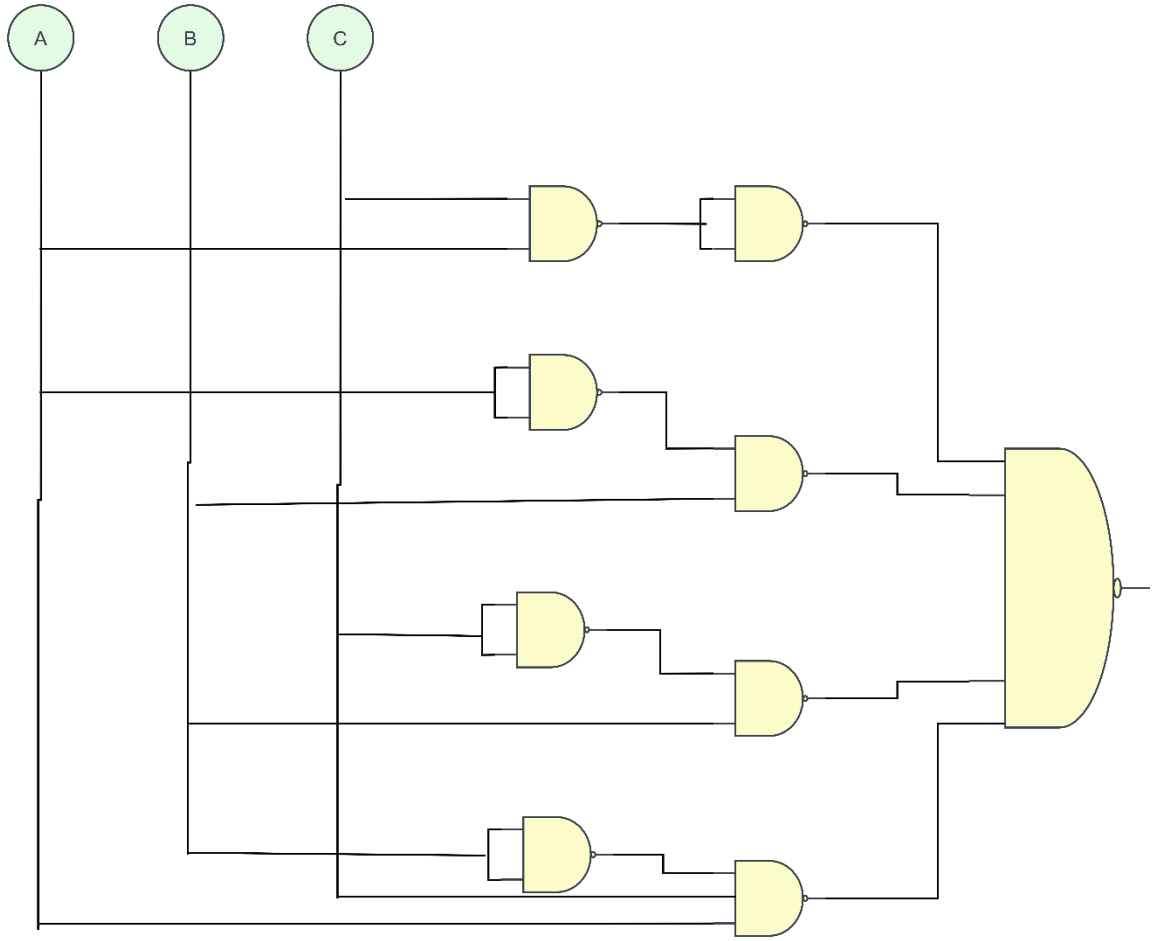


Figure 3. 22 The designed circuit only by using NAND for the SOP form obtained from K-Map

Activity 4

Part 1

Question 1

A binary operation is a mathematical operation that applies to two quantities or values from a specified set. When we run various binary operations on the same set, the results or characteristics associated with the operation can vary significantly.

Some distinguishing characteristics include as below:

1) **Commutativity:** Some binary operations are commutative, meaning the order in which the elements are operated on does not change the result. An example is an addition in the set of real numbers ($x + y = y + x$ for any real numbers x and y). However, not all operations are commutative - for example, subtraction is not ($x - y \neq y - x$).

Here are examples to illustrate the principle of commutativity:

Addition →

3 + 5 = 8

5 + 3 = 8

Here, changing the order of the numbers does not change the result. Therefore, addition is commutative.

Multiplication →

7 * 4 = 28

4 * 7 = 28

Just like addition, multiplication is also commutative, as changing the order of the factors does not change the product.

Not all operations are commutative, though. Here are counterexamples:

Subtraction →

9 - 4 = 5

4 - 9 = -5

In this case, changing the order of the numbers does change the result. Therefore, subtraction is not commutative.

Division →

12 / 3 = 4

3 / 12 = 0.25

Again, changing the order of the numbers does change the result. Therefore, the division is not commutative.

Special Note: The concept of commutativity is not restricted to numbers. It also applies (with some limitations) to some operations on other mathematical objects such as sets, vectors, and matrices. For instance, the union and intersection of sets are commutative operations.

2) **Associativity:** Certain binary operations are associative, i.e., when three or more values are involved, the grouping of these values doesn't affect the final result. Addition and multiplication of real numbers are associative operations. In contrast, operations like subtraction and division are not associative.

Here are examples to illustrate the principle of associativity:

Addition →

(2 + 3) + 4 = 2 + (3 + 4)

Here, whether you add 2 to 3 first, or 3 to 4 first, you still get 9. Therefore, addition is associative.

Multiplication →

(4 * 5) * 6 = 4 * (5 * 6)

Like addition, whether you multiply 4 and 5 first, or 5 and 6 first, you still get 120. Therefore, multiplication is associative.

However, not all operations are associative. Here are counterexamples:

Subtraction →

(7 - 3) - 2 = 2, whereas 7 - (3 - 2) = 6

Here, the order in which operations are performed does matter. Therefore, subtraction is not associative.

Division →

(8 / 4) / 2 = 1, whereas 8 / (4 / 2) = 4

Similarly, the results differ depending on how the operations are grouped. Therefore, division is not associative.

Special Note: It is crucial to emphasize that associativity is not limited to numbers or mathematical operations. It can be applied to operations on other mathematical objects, such as matrices, functions, and sets, as long as the operations are stated in an associative manner. For example, the operation of function composition is associative.

3) **Existence of Identity:** Some binary operations have an identity element, which is an element that when combined with other components leaves them unchanged. In the set of integers, for example, 0 is the identity element for addition and 1 is the identity element for multiplication.

This identity value is determined by both the set of numbers used and the operation executed. Here are a few examples:

Additive Identity →

The additive identity is the number 0. This is because adding zero to any number doesn't change the value of that number. For example:

5 + 0 = 5

0 + 123 = 123

For every real number x, we have x + 0 = 0 + x = x. Thus, 0 is the additive identity.

Multiplicative Identity →

The multiplicative identity is the number 1. This is because multiplying any number by one doesn't change the value of that number. For example:

7 * 1 = 7

1 * 0.5 = 0.5

For every real number x, we have x * 1 = 1 * x = x. Thus, 1 is the multiplicative identity.

Special Note: The identity for various mathematical structures may differ. For 2x2 matrices, for example, the identity matrix is the one in which the diagonal from top left to bottom right is made up of ones and all other entries are zero. We get the original matrix when we multiply any matrix by this identity matrix:

A * e = A

e * A = A

where A is any 2x2 matrix, and "e" is the identity matrix:

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

The concept of an identity is a fundamental building block in many areas of mathematics, including abstract algebra and linear algebra.

4) **Existence of Inverse:** Every element in certain binary operations has an inverse, which is an element that may be joined with the original element to provide the identity. For example, in the set of all integers, the additive inverse of an integer is its negation, and in the set of non-zero rational numbers, the multiplicative inverse of a number is its reciprocal.

In mathematics, an inverse of an element is another element that, when combined with the original element via a specific operation, provides the identity element for that operation.

Additive Inverse →

The additive inverse of a number is the negation of that number. If we add a number and its additive inverse, we get the additive identity, which is 0. For example:

The additive inverse of 5 is -5 because 5 + (-5) = 0.

The additive inverse of -3 is 3 because -3 + 3 = 0.

In general, for any real number x, the additive inverse is -x, because x + (-x) = 0.

Multiplicative Inverse →

The multiplicative inverse of a number is the reciprocal of that number. If we multiply a number by its multiplicative inverse, we get the multiplicative identity, which is 1. For example:

The multiplicative inverse of 4 is 1/4 because 4 * (1/4) = 1.

The multiplicative inverse of 1/2 is 2 because (1/2) * 2 = 1.

In general, for any non-zero real number x, the multiplicative inverse is 1/x, because x * (1/x) = 1.

Special Note: Please note that the concept of an inverse is not limited to real numbers or even to numbers in general. In many mathematical structures, such as matrices, functions, and abstract algebraic structures, elements can have inverses with respect to specific operations.

For example, the inverse of a 2x2 invertible matrix A is a matrix B such that AB = BA = I, where I is the identity matrix.

Similarly, the inverse of a function f is a function g such that f(g(x)) = g(f(x)) = x for all x in the domain of definition.

5) **Closure:** A binary operation is considered to be closed if applying it to two elements in the set always results in another element in the set. For example, addition is a closed operation on the set of integers because adding any two integers always results in another integer.

Here are some examples of closure:

Closure under Addition for Integers →

The integer set is closed by the addition operation ($\mathbb{Z}$, +). This means that if we add any two integers, we will always obtain another integer. For example:

3 + 4 = 7

(-5) + 2 = -3

Each of these sums is an integer, so the set of integers is closed under addition.

Closure under Multiplication for Real Numbers →

The operation of multiplication closes the set of real numbers ($\mathbb{R}$, ×). This indicates that if we multiply any two real numbers, we will always obtain a real number. For example:

2.3 * 4.5 = 10.35

$\sqrt{2} * \pi$ = 4.44288... (approximately)

Each product is a real number, so the set of real numbers is closed under multiplication.

In contrast, not all sets are closed by all operations. Here are some non-closure examples:

Non-closure under Division for Integers →

The operation of division does not close the set of integers ($\mathbb{Z}$, ÷). This is because dividing one integer by another does not always result in an integer. For example:

5 / 2 = 2.5

In this case, the result is not an integer, so the set of integers is not closed under division.

Non-closure under Subtraction for Natural Numbers →

The set of natural numbers (counting numbers starting from 1) is not closed under the operation of subtraction ($\mathbb{N}$, +). This is due to the fact that subtracting a greater number from a smaller number produces a negative number, which is not a natural number. For example:

3 - 5 = -2

The result is a negative number, which is not a natural number, so the set of natural numbers is not closed under subtraction.

The idea of closure is used in abstract algebra to create structures such as groups, rings, and fields, which are sets that are closed under one or more operations and meet additional properties.

Conclusion :

In conclusion, it is important to understand the distinctive characteristics of binary operations when working with sets since they specify the operation's behavior and the structure that results from its use. Commutativity, associativity, the existence of an identity, the existence of inverses, and closure are the key qualities of binary operations considered in the context of Group Theory.

Commutativity and associativity define the order and grouping in which elements are operated on, respectively, whereas the existence of an identity and an inverse focus on the properties of individual elements within the operation. Closure is an important characteristic that ensures that each operation within the set results in an element from the same set.

It is crucial to note, however, that not all binary operations have all of these qualities. The properties they do possess ultimately define the structure of the set and its behavior under the operation, contributing to the formation of mathematical structures such as groups, rings, and fields.

These principles form the foundation for more complex studies in abstract algebra and have applications across various fields of mathematics, computer science, and even physics. They are critical to software engineering in contexts such as data processing, cryptography, and error detection and correction, emphasizing their significance in both theory and practice.

Part 2

Question 1

Groups are mathematical structures that are made up of a set of elements and an operation that combines any two of them to generate the third element in the set. They must meet four criteria: closure, associativity, the existence of an identity element, and the existence of an inverse for each element.

Let's use the elements {a, b, c, e} to construct operation tables for groups of orders 1, 2, 3 and 4. In each group, 'e' serves as the identifying element, and '*' designates the group operation.

Group G of order 1:

The only group of order 1 consists of the identity element only.

*	e
E	e

Group G of order 2:

There are only two elements: e and a. By the identity and inverse properties, the other element must be its own inverse, so a*a = e.

*	e	a
e	e	a
a	a	e

Group G of order 3:

There are three elements: e, a and b. We may fill in the multiplication table using the identity and inverse properties.

Since there is no other identity element except e, a and b cannot square to e. As a result, they must square each other and, by elimination, multiply to e.

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

Group G of order 4:

There are four elements: e, a, b, and c. One possibility is that each element is its own inverse. This gives us the Klein four-group.

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

Conclusion :

The structure of the groups is represented by the operation tables. They can be used to find the result of any combination of elements in a group operation, and they capture the identity and inverse properties.

It is important to note that, depending on the properties of the operation and elements, there are numerous alternatives for the operation table of a group of order 4. The one shown above is just an example.

The structure of an order 4 group is not unique. In fact, there are two groups of order 4 up to isomorphism: the Klein four-group, which we have already looked at, and the cyclic group of order 4.

Let's define the multiplication table for the cyclic group of order 4, again using the elements {e, a, b, c} and e as the identity element:

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

This group is called a cyclic group because we can generate all other elements by repeatedly applying the operation to one element.

In this case, e is the identity, a is a generator, b = a * a, and c = a * a * a (or b * a).

If we apply the operation to a once more, we cycle back to the identity: a * a * a * a = e.

Also, the abstract symbols a, b, c, and e are used in the tables above. In practice, the elements can be numbers, matrices, permutations, or any other object that obeys the group laws. As long as the group laws are satisfied, the operation could be anything from addition and multiplication to function composition and matrix multiplication.

Question 2 – (i)

The number of elements in a group determines its order. A binary operation on a group is a rule that takes any two elements in the group and creates another element inside the group.

The number of binary operations that can be defined on a set depends on the number of elements in the set. If a set has n elements, then a binary operation can be seen as a function from n × n pairs of elements to n elements.

Therefore, there are $(n)^{n^2}$ possible binary operations that can be defined on the set.

However, not all of these operations would be compatible with the requirements to form a group. For the operation to give rise to a group structure, it must satisfy the group axioms: closure, associativity, identity, and inverses.

To illustrate this, consider the set with two elements {e, a}. There are $(2)^{2^2}$ = 16 potential binary operations. However, only four of these operations satisfy the group axioms and result in a group. These operations can be represented by the following tables:

*	e	a
e	e	a
a	a	e

*	e	a
e	e	a
a	e	e

*	e	a
e	e	a
a	e	a

*	e	a
e	e	a
a	a	a

As these examples show, even though there are many possible binary operations for a given set, only a few of them will satisfy the properties required to form a group. The complexity of finding these valid operations increases with the size of the set, making it a challenging task in abstract algebra.

Question 2 – (ii)

A binary operation on a set can be considered a function that maps pairs of set elements to set elements. If a set has n elements, then a binary operation is a function that maps n x n = n^2 pairs of elements to n possible outcomes. T

Therefore, for a set with n elements, there are $(n)^{n^2}$ possible binary operations.

In the case of a set with 4 elements, this means there are $(4)^{4^2}$ = $(4)^{16}$ = 4,294,967,296 possible binary operations.

Question 3 – (i)

Lagrange's Theorem is a fundamental theorem in group theory, a subfield of abstract algebra. The theorem was named after Joseph-Louis Lagrange.

Lagrange's Theorem states:

"In a finite group, the order (number of elements) of every subgroup is a divisor of the order of the group."

To put it another way, if G is a finite group with |G| elements (the order of the group), and H is a subgroup of G with |H| elements (the order of the subgroup), then |H| divides |G|.

In notation form, if |G| = n and |H| = m, then n mod m = 0.

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

For instance, if we have a group G with 12 elements, any subgroup of G must have 1, 2, 3, 4, 6, or 12 elements, as these are the divisors of 12.

No subgroup of G can, for example, have 5 or 7 elements, as 5 and 7 do not divide 12 evenly.

Lagrange's Theorem is a statement with regard to the structure of finite groups and their subgroups that has wide-ranging implications in mathematics and computer science.

Question 3 – (ii)

Cosets are an important concept in group theory. If we have a group G and a subgroup H of G, and if g is an element of G, then we can form what's called the left coset of H with respect to g, denoted by gH = {gh: h belongs to H}. There is also a similar concept of a right coset, denoted by Hg.

Before we can delve into the proof of Lagrange's Theorem, we need to understand three key assertions:

Assertion 1: For a finite group G and its subgroup H, each element g in G corresponds one-to-one with each coset of H.

Assertion 2: If we have two elements g1 and g2 in G, they are said to be related (g1 ~ g2) if and only if their left cosets with respect to H are the same (g1H = g2H). This forms an equivalence relation.

Assertion 3: For any set S and an equivalence relation ~ on S, if A and B are two equivalence classes with no overlap (A ∩ B = ∅), then A and B are the same equivalence class.

Now, let's revisit the proof of Lagrange's Theorem:

Proof:

Consider a finite group G with order m, and a subgroup H of order n. If we break down G into left cosets with respect to H, each coset, say gH, contains n unique elements.

Suppose H = {h1, h2, ..., hn}, then gh1, gh2, ..., g*hn are n distinct elements of the coset gH.

By the cancellation law in group theory, if ghi = ghj then hi = hj.

Since G is finite, the number of distinct left cosets is also finite, say p. Therefore, the total number of elements in all cosets is np, which equals the total number of elements in G. Hence, we have m = np, or p = m/n.

This shows that the order of H (n) divides the order of G (m). It also shows that the index p is also a divisor of the order of the group.

Therefore, we've proven Lagrange's Theorem: the order of any subgroup H divides the order of the group G, or in mathematical terms, |G| = k*|H| for some integer k. This is a fundamental result in group theory with many implications.

Question 3 – (iii)

Based on Lagrange's Theorem, a group H with order 6 cannot be a subgroup of a group G with order 13.

Lagrange's Theorem states that the order (the number of elements) of a subgroup divides the order of the group. In other words, if H is a subgroup of G, then the number of elements in G is a multiple of the number of elements in H.

In this case, the order of H is 6, and the order of G is 13. Since 13 is not a multiple of 6, by Lagrange's Theorem, it is impossible for a group of order 6 to be a subgroup of a group of order 13.

$$\frac{|G|}{|A|} = \frac{13}{6} = 2.1666 \ \ (Not\ Divisible)$$

Therefore, a group H with order 6 cannot be a subgroup of a group with order 13.

Part 3

Question 1

To determine whether a set forms a group under a certain operation, we must check whether the four fundamental properties of a group hold:

- Closure: For all a, b in the group, the result of the operation a * b is also in the group.
- Associativity: For all a, b, c in the group, (a * b) * c = a * (b * c).
- Identity: There is an element e in the group such that for every element a in the group, the equations e * a and a * e return a.
- Inverses: For each a in the group, there exists an element b in the group such that a * b = b * a = e, where e is the identity element of the group.

For the given set S = R - {-1} (the set of all real numbers excluding -1) under the operation *, we need to determine whether these four properties hold:

1) Closure:

To test closure, we need to verify whether the result of the operation * on any two elements of S is also an element in S. That is, for any two real numbers a and b (which are not -1), the result of the operation a * b = a + b + ab should be a real number that is not -1.

Since the operation * is essentially a combination of addition and multiplication (both of which are closed on the real numbers), it should be clear that for any “a” and “b” in “S”, a * b will yield another real number.

Ex:

$a, b \in S = a + b + ab \in S$

$0,1 \in S = 0 + 1 + 0 \times 1 = 1 \in S$

$5, -5 \in S = 5 + (-5) + 5 \times (-5) = -25 \in S$

However, to ensure closure, we must also confirm that the result is not -1. Here, we observe that the only possible way for a + b + ab to equal -1 is if a and/or b is -1, which is not possible as we explicitly excluded -1 from our set S. Hence, S is closed under the operation *.

Ex:

$a, b \in S = a + b + ab \in S$

$0, -1 \in S = 0 + (-1) + 0 \times (-1) = -1 \in S$

$5, -1 \in S = 5 + (-1) + 5 \times (-1) = -1 \in S$

$-1,0 \in S = (-1) + 0 + (-1) \times 0 = -1 \in S$

$-1,5 \in S = (-1) + 5 + (-1) \times 5 = -1 \in S$

$-1, -1 \in S = (-1) + (-1) + (-1) \times (-1) = -1 \in S$

And also, in the question it is mentioned that a * b = a + b + ab is a binary operation for "S". So it is already satisfy the closure property.

2) Associativity:

In group theory, associativity is a fundamental property. It states that when we have three elements and a binary operation, the order in which they are grouped has no effect on the final output.

Formally, for any elements a, b, c in a group G, the operation '*' must satisfy the property (a * b) * c = a * (b * c).

Let's discuss associativity in the context of our specific set and operation. Our set is S = R - {-1}, the set of all real numbers excluding -1, and our operation is '*', defined by a * b = a + b + ab.

We need to prove that for all a, b, c in S, (a * b) * c = a * (b * c).

Firstly, let's look at the left side of the equation:

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

= (a + b + ab) + C + (a + b + ab) × c

= a + b + ab + c + ac + bc + abc ← ①

Now, let's look at the right side of the equation:

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

= (a + b + ab) + C + (a + b + ab) × c

= a + b + c + bc + ab + ac + abc ← ②

L.H.S. = R.H.S.

As we can see, after expanding both sides, the results are equal. Hence, the operation '*' is associative on S. This means that no matter how we group the elements when applying the operation, we'll always get the same result.

Associativity is important as it provides consistency and allows us to manipulate elements within the group without having to worry about the order of operations.

3) Existence of Identity Element:

The identity property in group theory states that there exists an element within the group such that when the group's operation is applied to any element in the group and the identity, the result is the original element.

In mathematical terms, for a group G with an operation '*', there exists an element 'e' (the identity) in G such that for all elements 'a' in G, we have e * a = a * e = a.

Our set is S = R - {-1}, the set of all real numbers excluding -1, and our operation is '*', defined by a * b = a + b + ab.

We need to find an element e in S such that for any a in S, e * a = a. By plugging into our operation definition, we get e + a + ea = a.

a * e = a

a + e + ae = a

~~a~~ + e + ae = ~~a~~

e(1 + a) = 0

0 ∈ S

e * a = a

e + a + ea = a

e + ~~a~~ + ea = ~~a~~

e(1 + a) = 0

0 ∈ S

Solving for e in this equation, we would subtract 'a' from both sides to get e + ea = 0. If we factor out the 'e', we get e(1 + a) = 0.

Since we are excluding -1 from our set, 'a' will never be -1, so 1 + a will never be zero. Therefore, the only way for the product to be zero is if e = 0.

Hence, e = 0 is the identity element for this group. This means that for any real number a in S, 0 * a = a * 0 = a, confirming the identity property for our set S under the operation '*'.

Having an identity element in a group is crucial because it serves as a sort of "neutral" element, leaving other elements unchanged when the group operation is applied.

4) Existence of Identity Inverse:

The existence of inverses is another crucial property in group theory. It states that for every element in the group, there is another element such that when the group's operation is applied to the two elements, the result is the identity element of the group.

Formally, for a group G with an operation '*', and an identity element 'e', for each element 'a' in G, there exists an element 'b' in G such that a * b = b * a = e.

Our set is S = R - {-1}, the set of all real numbers excluding -1, and our operation is '*', defined by a * b = a + b + ab. The identity element 'e' of this group is 0.

We need to find an inverse for every element 'a' in S, which means we need to find another element 'b' in S such that a * b = 0. Plugging into our operation, this becomes a + b + ab = 0.

a * b = e

a + b + ab = 0

a (1 + b) + b = 0

$a = \frac{-b}{(1+b)} \in S$

b * a = e

b + a + ba = 0

b (1 + a) + b = 0

$b = \frac{-a}{(1+a)} \in S$

To find 'b', we can rearrange this equation as above to get b = -a / (1 + a).

Now, we need to ensure that 'b' is in S, which means 'b' needs to be a real number other than -1.

The only way 'b' could be -1 is if 'a' were -2. But -2 is a valid element of S, and plugging -2 into the equation gives us b = 2 / (1 - 2) = -2, which is not -1 and is thus also in S.

So for every 'a' in S, there exists a 'b' in S such that a * b = 0. Therefore, every element in S has an inverse, satisfying the requirement for the group.

The existence of inverse elements is crucial in group theory because it allows us to "undo" or reverse the group operation.

Conclusion :

Since all four fundamental properties of a group should hold (Closure, Associativity, Identity, Inverse) for the given set S = R - {-1} (the set of all real numbers excluding -1) under the operation *, we can determine the set S is indeed a group under the operation '*'.

In conclusion, the set S = R - {-1}, which encompasses all real numbers excluding -1, has been shown to form a group under the binary operation '*' defined as a * b = a + b + ab for any two elements a and b in S.

We have confirmed the key properties of a group for this set and operation. Firstly, the operation '*' is closed on S, meaning that any operation performed on elements of S will result in an element that also belongs to S.

Secondly, the operation '*' is associative, meaning the order of operations doesn't affect the outcome when three or more elements are combined.

Thirdly, we identified 0 as an identity element in the group, as any element from the set S, when operated with 0, yields the original element.

Lastly, we verified that every element in S has an inverse also within S, i.e., for every element 'a' in S, there is an element 'b' in S such that a * b equals the identity element, in this case, 0.

Therefore, given these properties, we can confirm that the set S is indeed a group under the operation '*'. This insight allows us to apply group theory techniques to analyze the structure and properties of this mathematical construct. This understanding is fundamental to many areas of mathematical study, including algebra and number theory, and has implications in fields as diverse as physics, computer science, and cryptography.

Part 4

Question 1

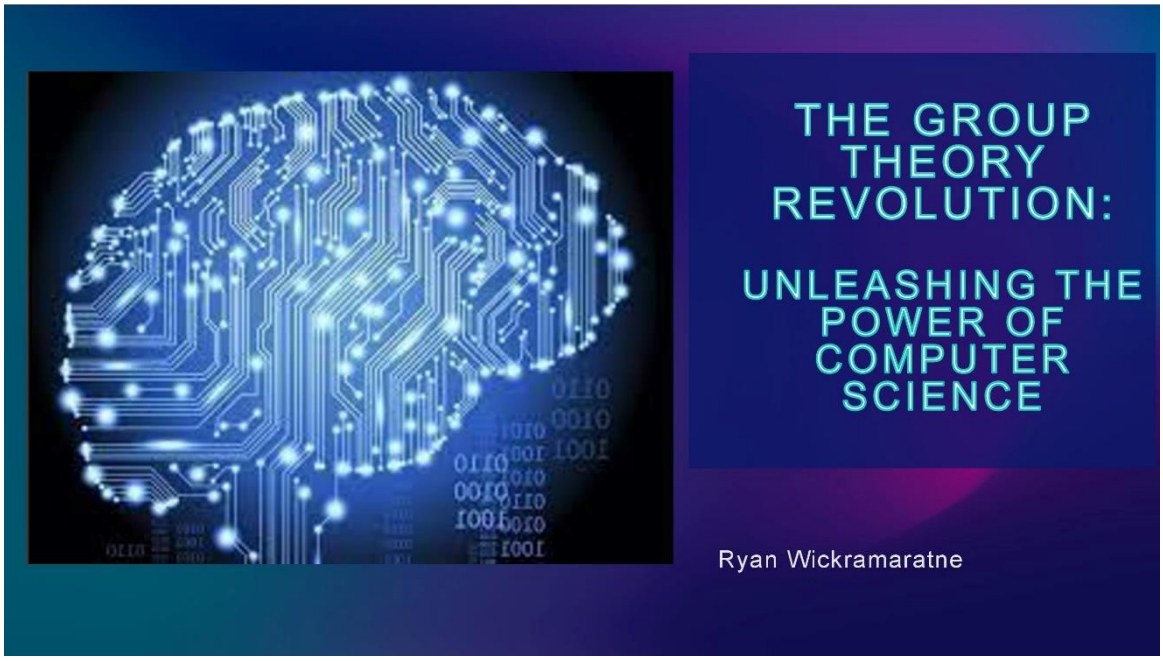


Figure 4. 1 Presentation slide 1

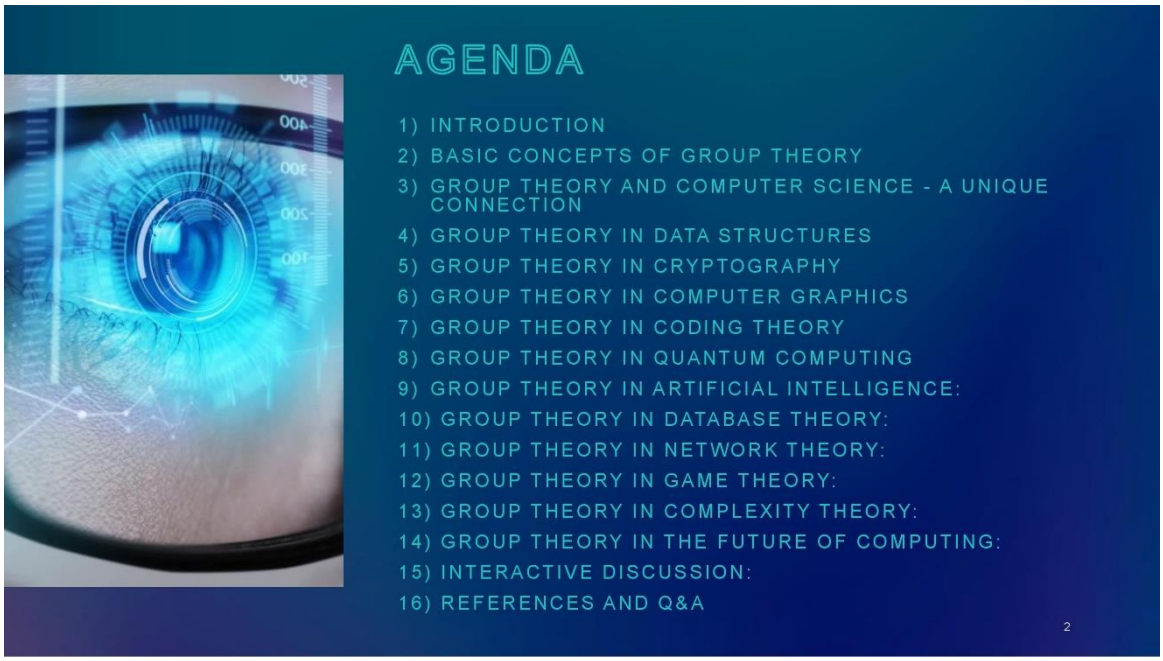


Figure 4. 2 Presentation slide 2

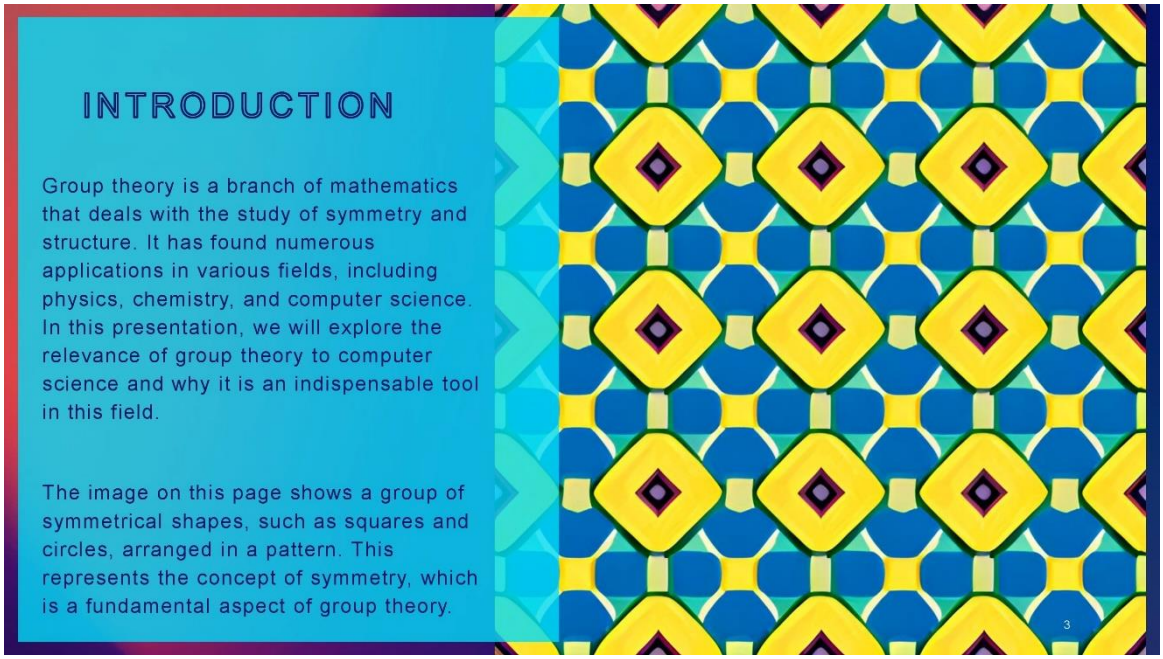


Figure 4. 3 Presentation slide 3

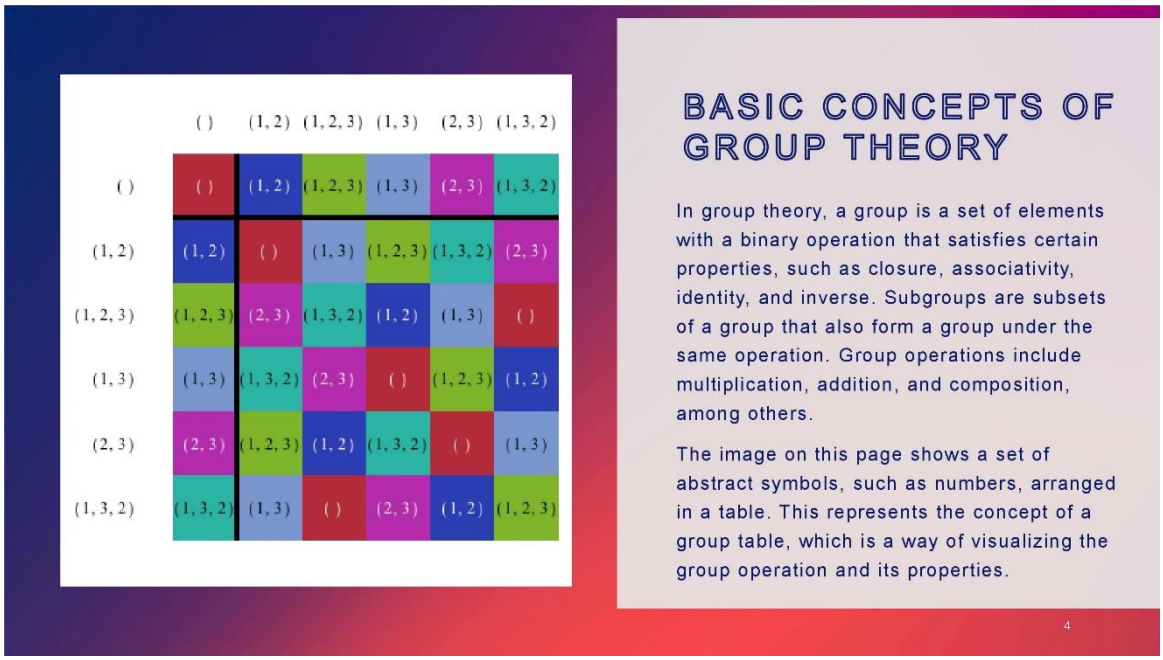


Figure 4. 4 Presentation slide 4

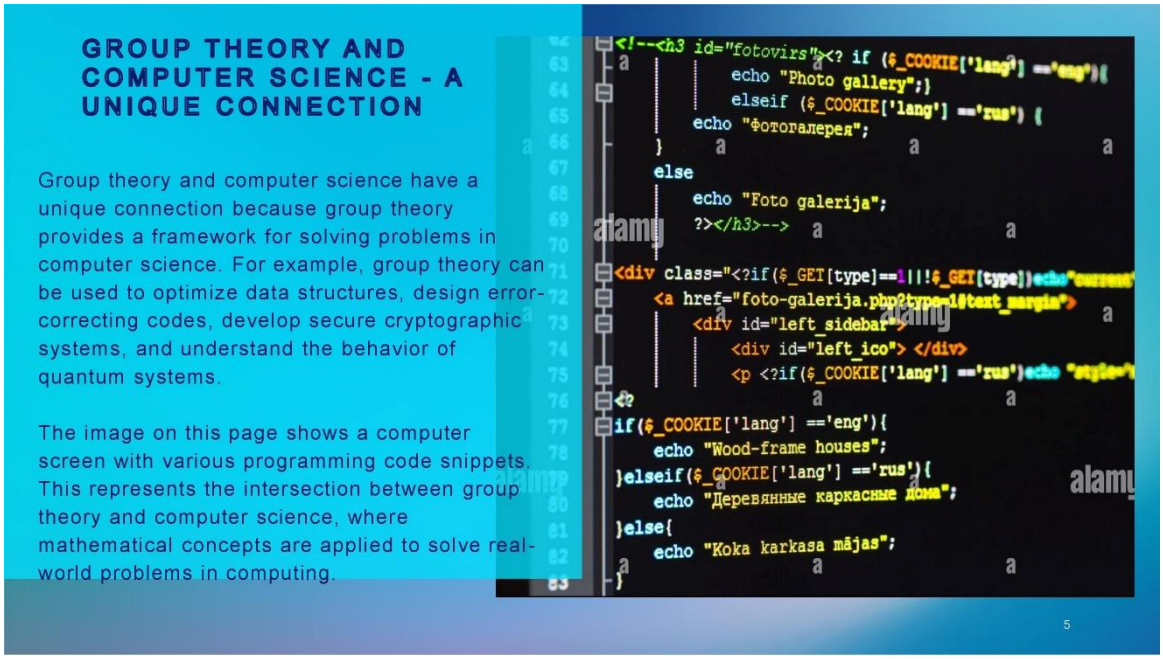


Figure 4. 5 Presentation slide 5

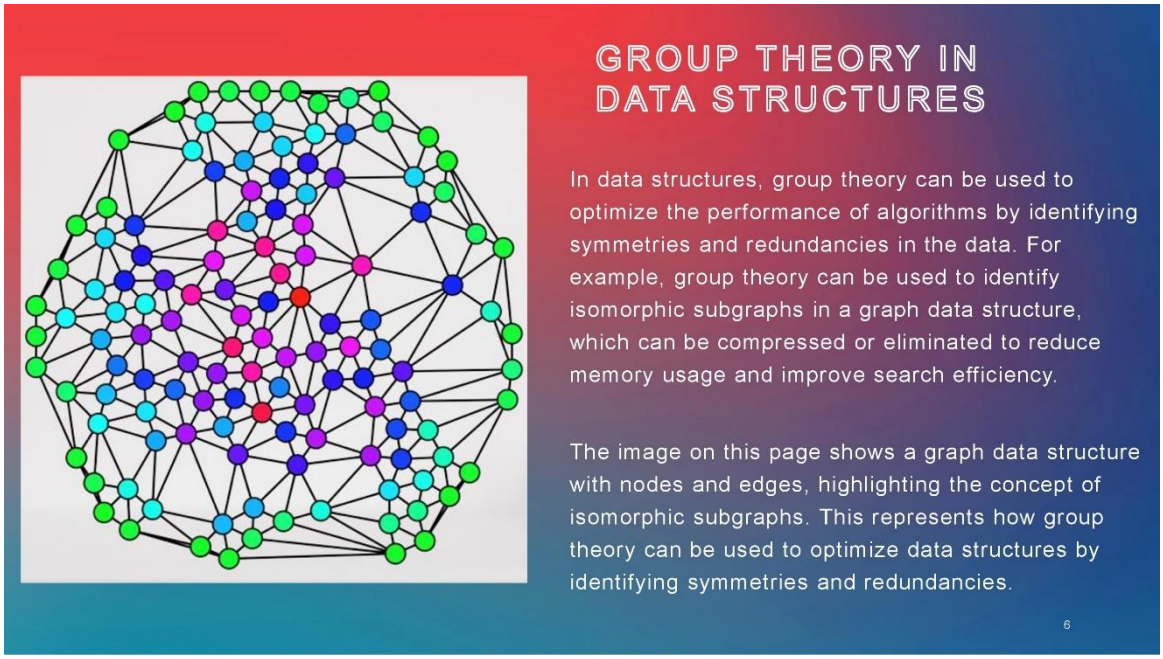


Figure 4. 6 Presentation slide 6

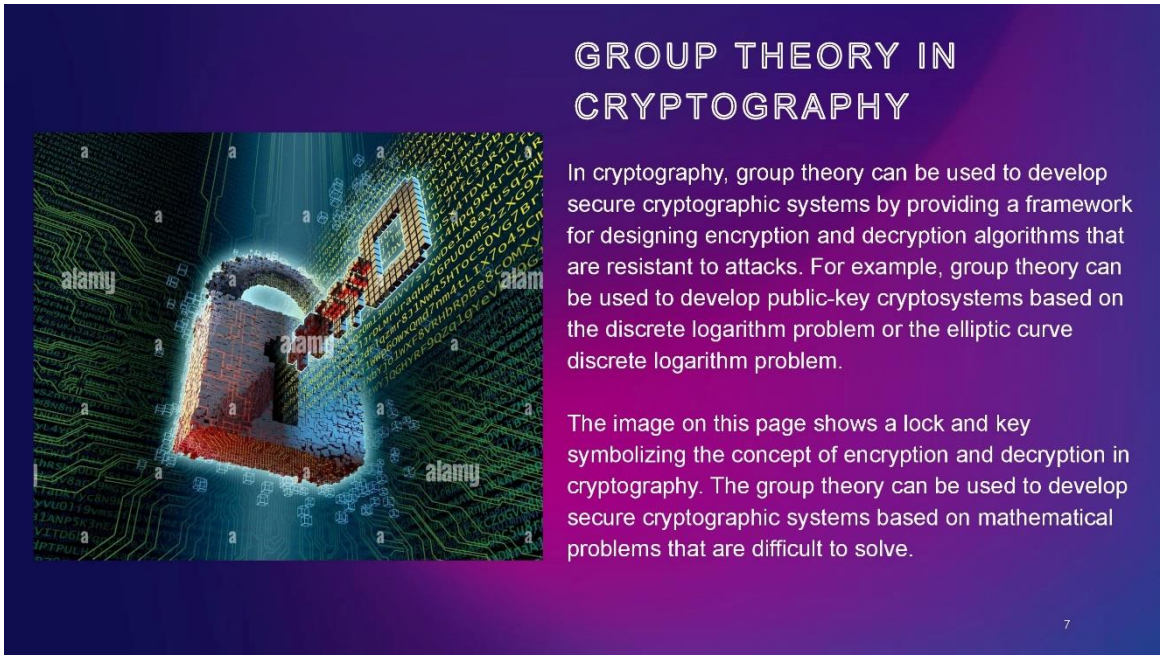


Figure 4. 7 Presentation slide 7

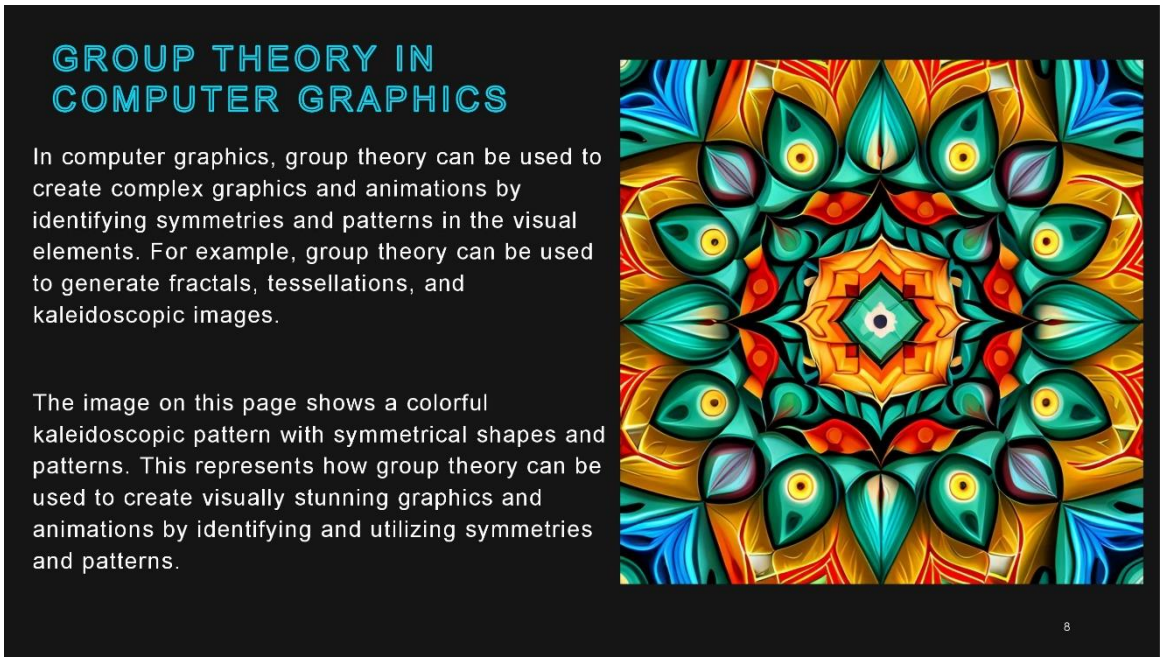


Figure 4. 8 Presentation slide 8

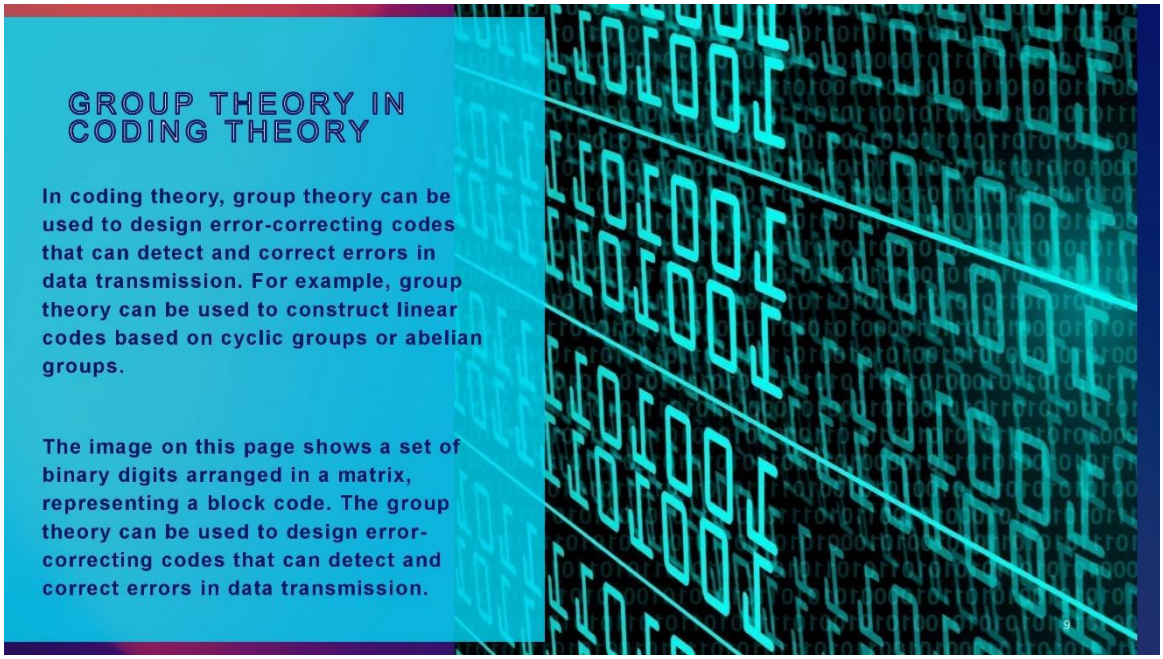


Figure 4. 9 Presentation slide 9

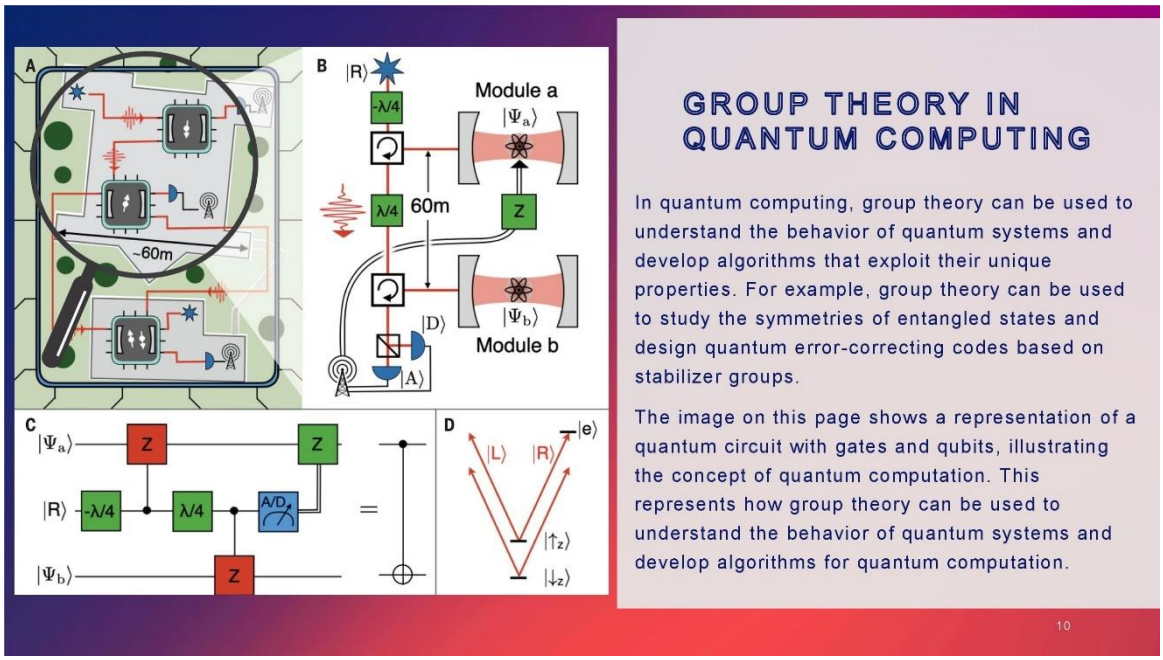


Figure 4. 10 Presentation slide 10

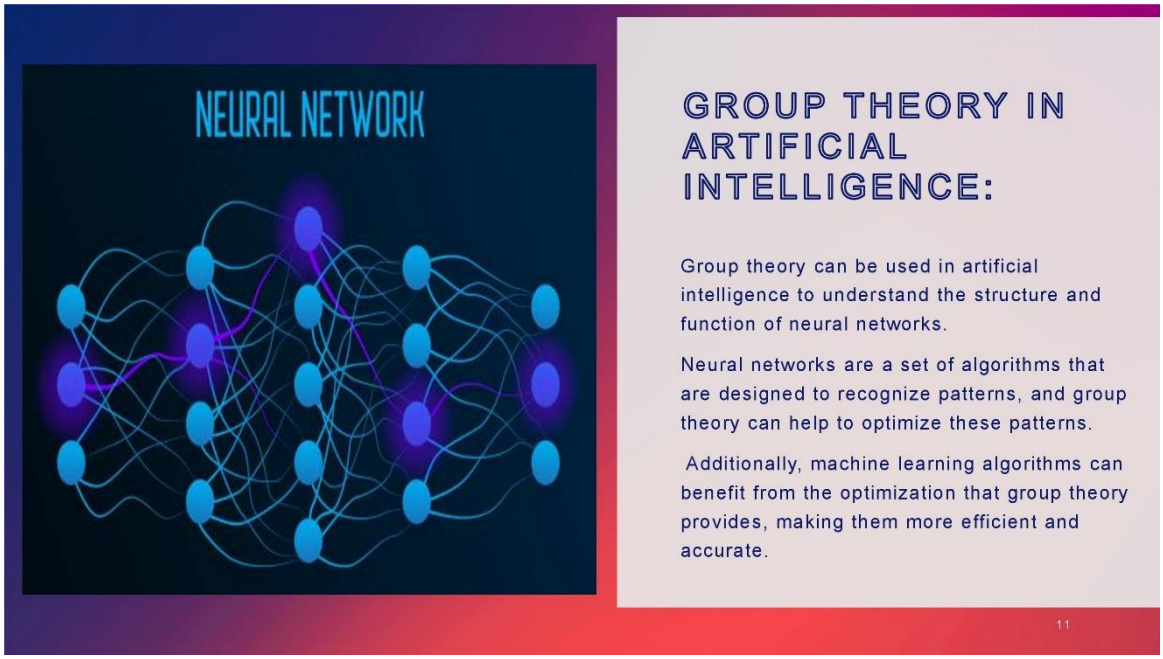


Figure 4. 11 Presentation slide 11

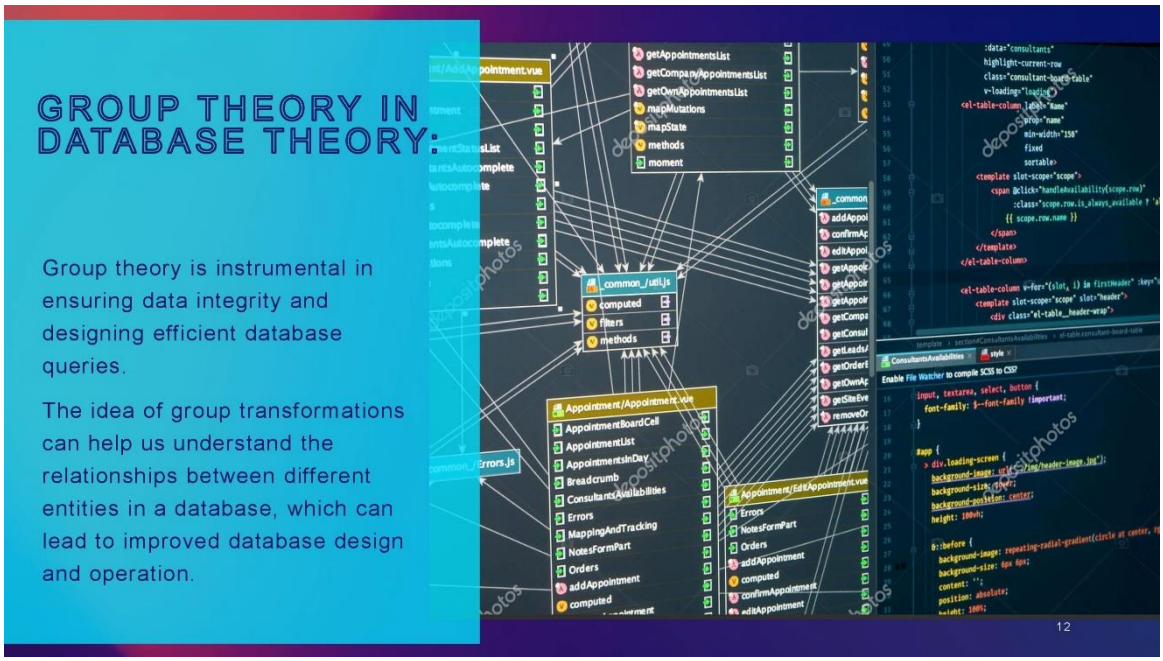


Figure 4. 12 Presentation slide 12

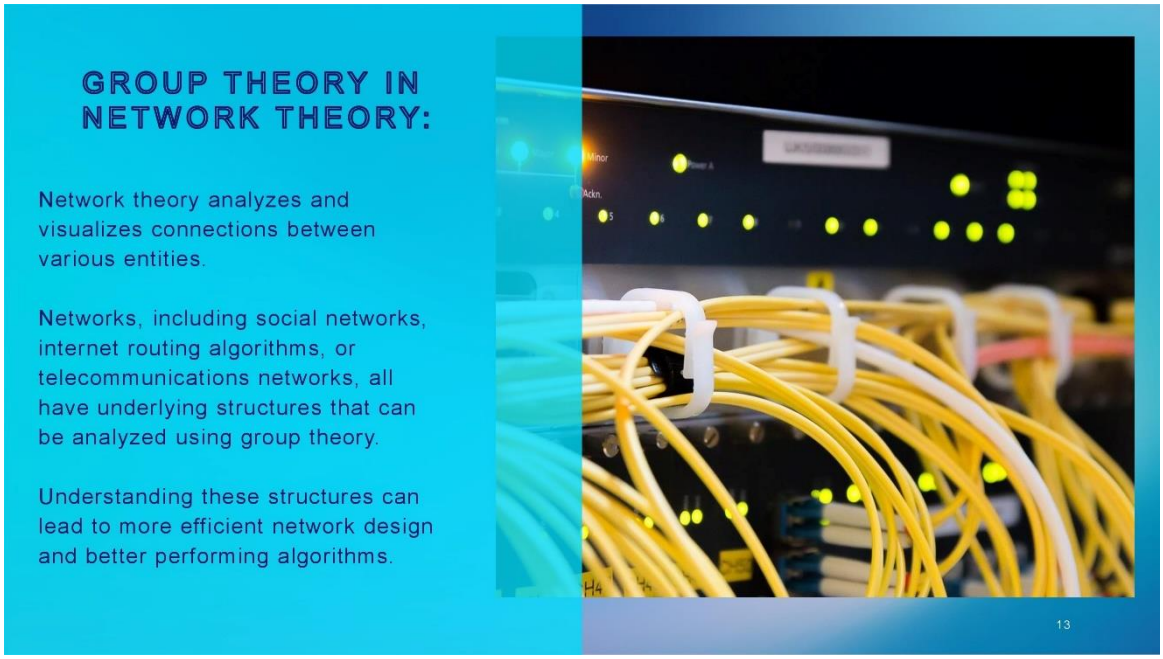


Figure 4. 13 Presentation slide 13

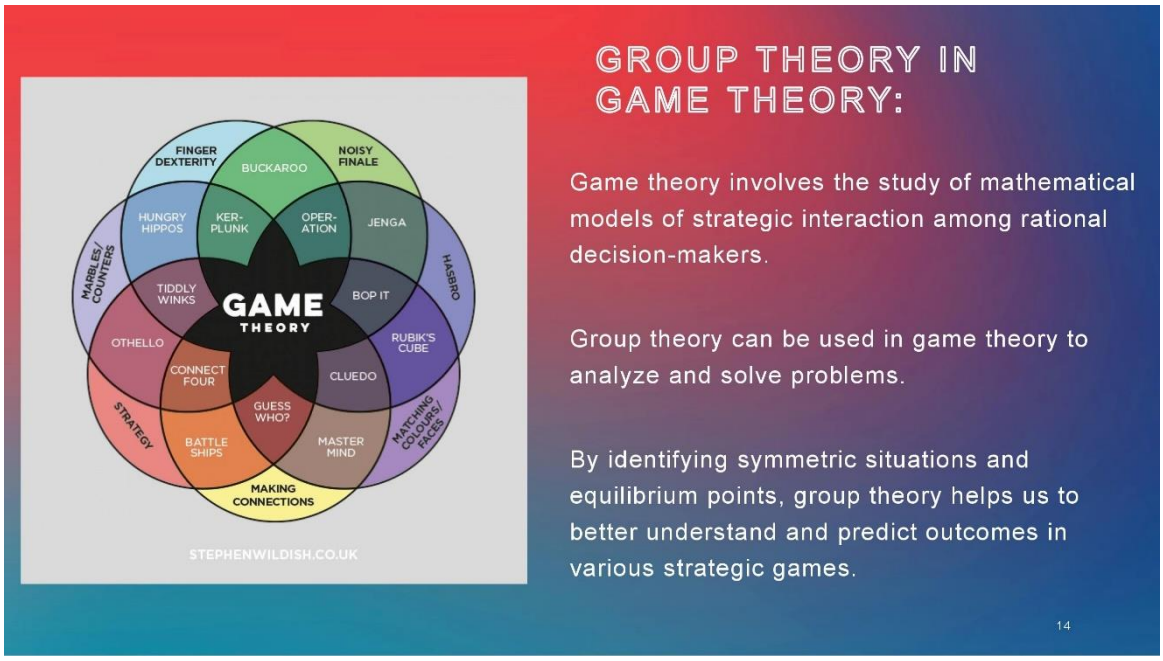


Figure 4. 14 Presentation slide 14

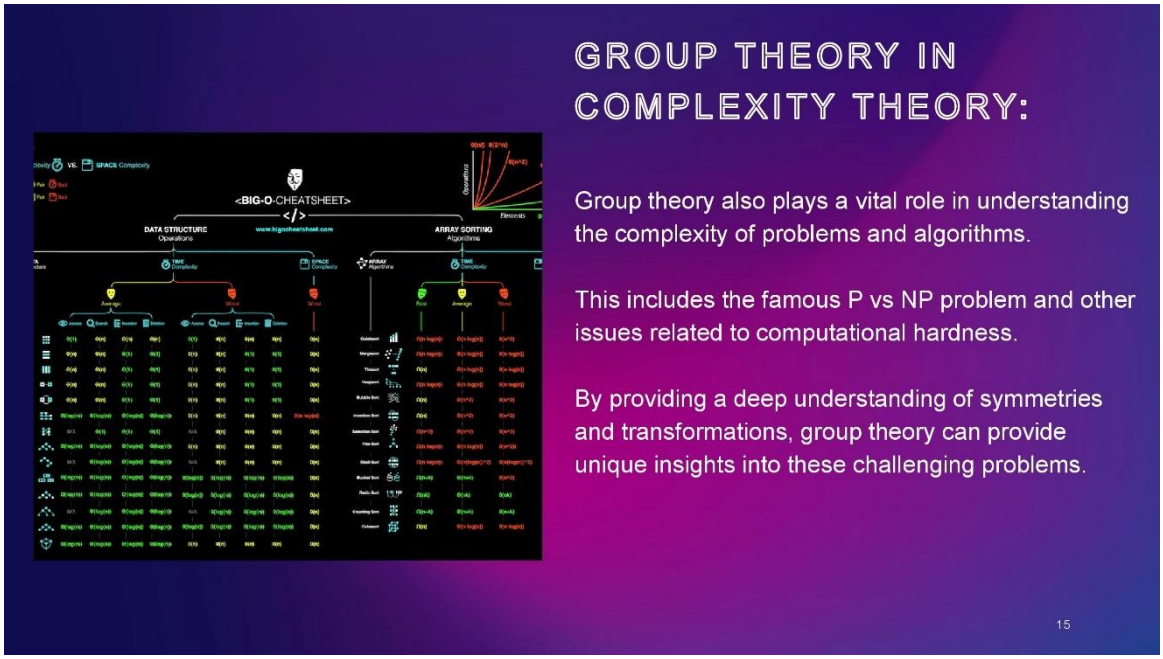


Figure 4. 15 Presentation slide 15

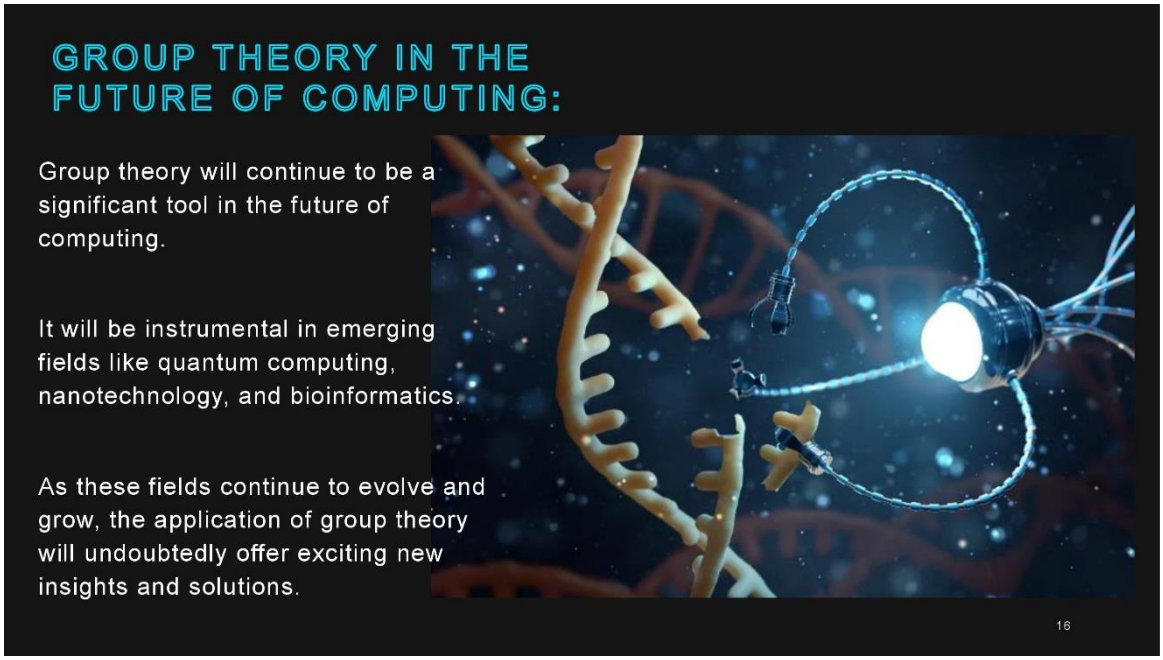


Figure 4. 16 Presentation slide 16

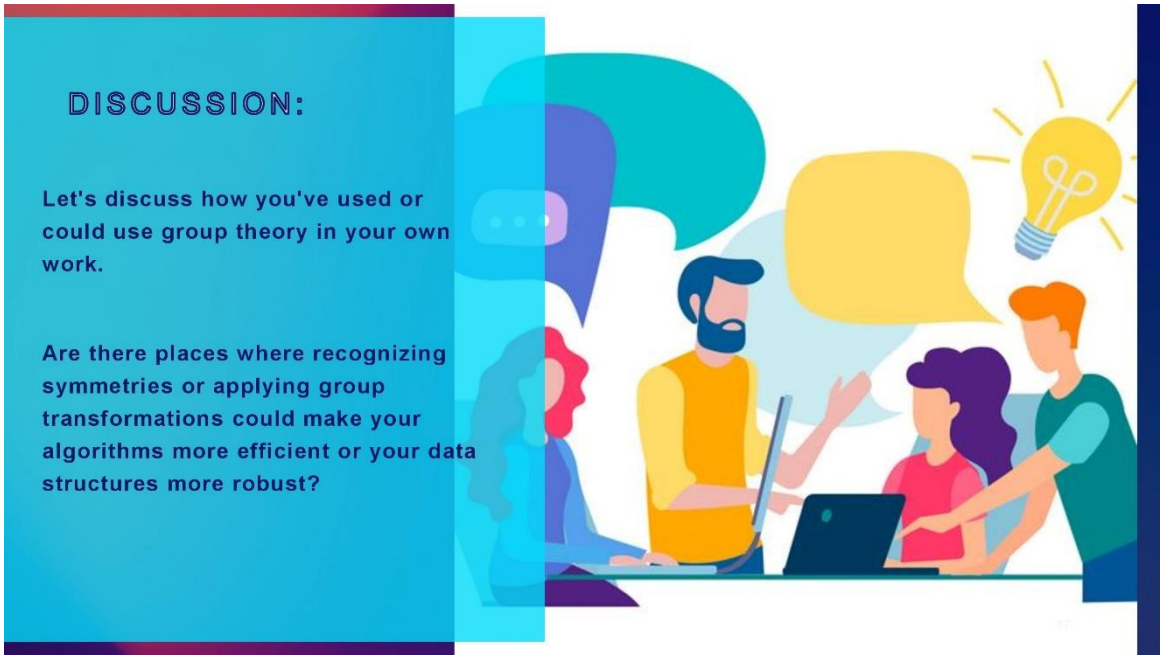


Figure 4. 17 Presentation slide 17

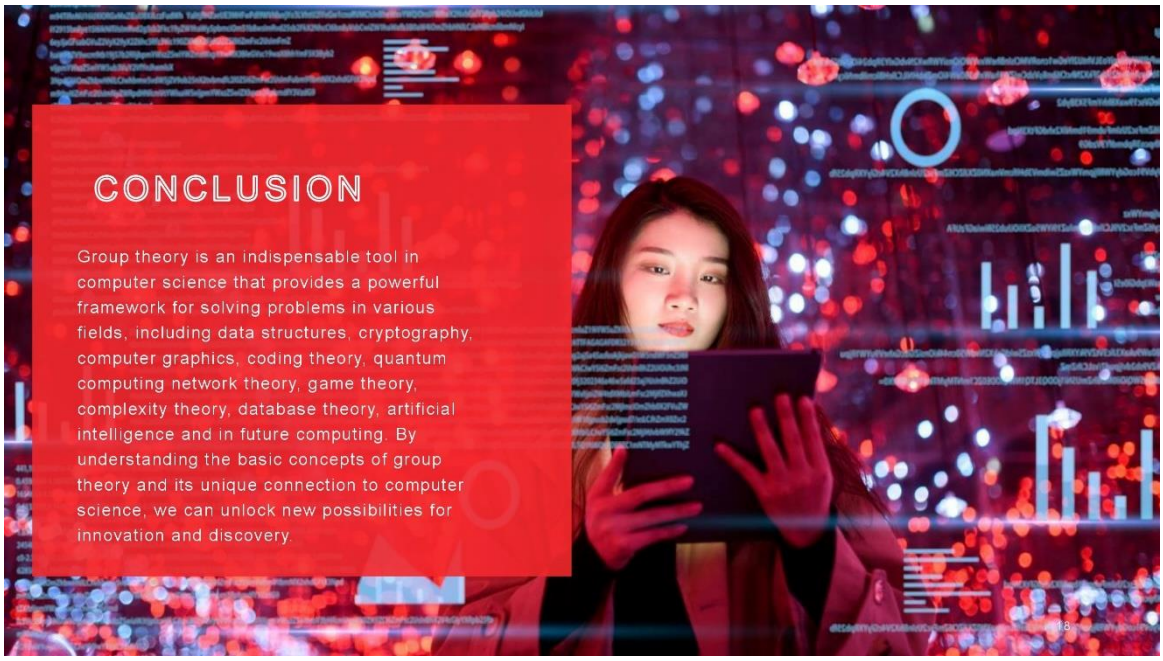


Figure 4. 18 Presentation slide 18

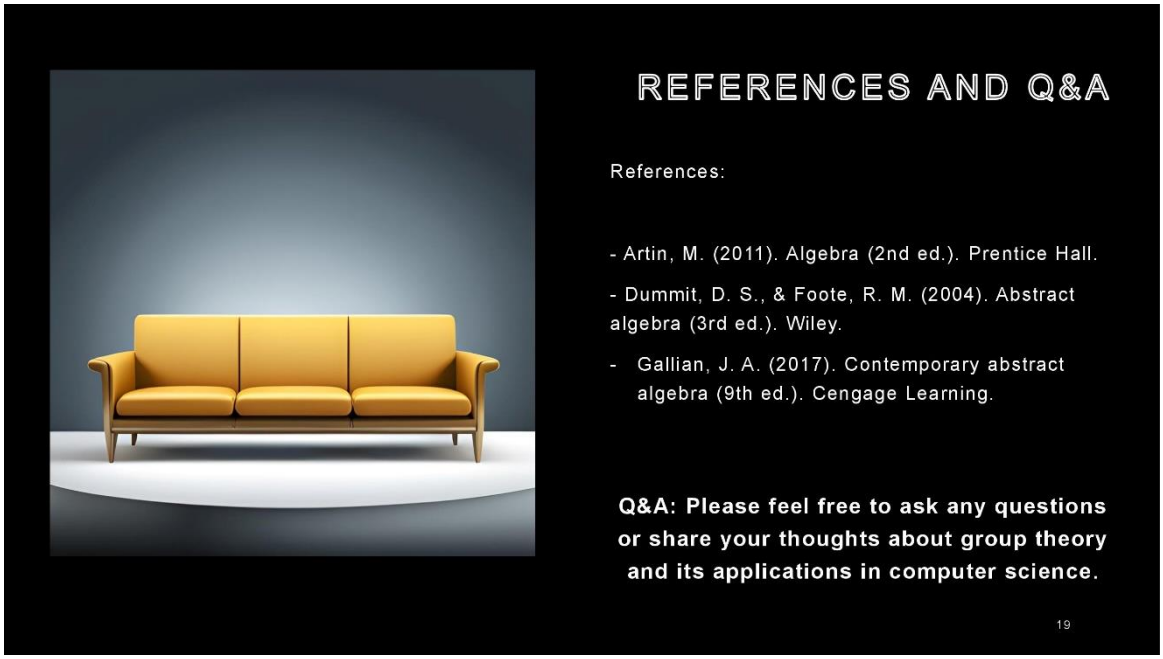


Figure 4. 19 Presentation slide 19

Conclusion

This assignment extensively explores mathematical theory, focusing on set theory, function analysis, and number theory. It includes an examination of prime factorization, multiplicities, and cardinalities of multisets. Functions, their characteristics, injectivity, surjectivity, and invertibility were studied across quadratic, reciprocal, and trigonometric types. Additionally, we delved into set theory, offering a formal proof of De Morgan's laws and the distributive laws for finite sets. Lastly, graph theory was discussed, particularly focusing on Hamiltonian cycles. The assignment underscored thorough explanations, rigorous proofs, and precise formulations to ensure a comprehensive understanding of the principles involved.

References

BYJU'S. (n.d.). Basics of Set Theory: Sets, Types of Sets, Set Theory Formulas.

Available at: https://byjus.com/maths/basics-set-theory/

[Accessed 7 July 2023].

BYJU'S. (n.d.). Set Operations: Types, Union, Intersection & Difference Of Sets.

Available at: https://byjus.com/maths/set-operations/

[Accessed 7 July 2023].

Math24. (n.d.). Set Identities.

Available at: https://math24.net/set-identities.html#:~:text=The%20basic%20method%20to%20prove,hand%20side%20and%20vice%20versa

[Accessed 7 July 2023].

Brilliant. (n.d.). Multiset.

Available at: https://brilliant.org/wiki/multiset/

[Accessed 7 July 2023].

Cuemath. (n.d.). Domain and Range of a Function.

Available at: https://www.cuemath.com/calculus/domain-and-range-of-a-function/

[Accessed 7 July 2023].

Cuemath. (n.d.). One-to-One Function.

Available at: https://www.cuemath.com/algebra/one-to-one-function/

[Accessed 7 July 2023].

BYJU'S. (n.d.). Functions and its Types.

Available at: https://byjus.com/jee/functions-and-its-types/

[Accessed 7 July 2023].

Third Space Learning. (n.d.). Inverse Functions Explained.

Available at: https://thirdspacelearning.com/gcse-maths/algebra/inverse-functions/#:~:text=Inverse%20functions%20are%20functions%20which,input%20from%20a%20known%20output

[Accessed 7 July 2023].

Britannica. (n.d.). Graph Theory.

Available at: https://www.britannica.com/topic/graph-theory

[Accessed 7 July 2023].

Tutorials Point. (n.d.). Graph Theory Trees.

Available at: https://www.tutorialspoint.com/graph_theory/graph_theory_trees.htm

[Accessed 7 July 2023].

Programiz. (n.d.). Spanning Tree and Minimum Spanning Tree.

Available at: https://www.programiz.com/dsa/spanning-tree-and-minimum-spanning-tree#:~:text=A%20spanning%20tree%20is%20a,have%20weights%20assigned%20to%20them

[Accessed 7 July 2023].

freeCodeCamp. (n.d.). Dijkstra's Shortest Path Algorithm: A Visual Introduction.

Available at: https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/#:~:text=Dijkstra's%20Algorithm%20finds%20the%20shortest,node%20and%20all%20other%20nodes

[Accessed 7 July 2023].

UL. (n.d.). Euler and Hamilton Paths.

Available at: https://ulsites.ul.ie/cemtl/sites/default/files/cemtl_graph_euler_hamilton.pdf

[Accessed 7 July 2023].

Mera Calculator. (n.d.). KMAP Solver (Karnaugh Map).

Available at: https://www.meracalculator.com/math/kmap-solver.php

[Accessed 7 July 2023].

32x8. (n.d.). Online Karnaugh Map Solver.

Available at: http://www.32x8.com/index.html

[Accessed 7 July 2023].

Boolean Algebra. (n.d.). Boolean Algebra.

Available at: https://www.boolean-algebra.com/

[Accessed 7 July 2023].

Math Only Math. (n.d.). Proof of De Morgan's Law.

Available at: https://www.math-only-math.com/proof-of-de-morgans-law.html

[Accessed 7 July 2023].

GeoGebra. (n.d.). Graphing Calculator.

Available at: https://www.geogebra.org/graphing?lang=en

[Accessed 7 July 2023].

Easy Calculation. (n.d.). Distributive Law.

Available at: https://www.easycalculation.com/theorems/distributive-law.php

[Accessed 7 July 2023].

McGill University. (n.d.). Graph Coloring.

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

[Accessed 7 July 2023].

Geeks for Geeks. (n.d.). Set Theory Formulas.

Available at: https://www.geeksforgeeks.org/set-theory-formulas/

[Accessed 7 July 2023].

Brilliant. (n.d.). Logic Gates.

Available at: https://brilliant.org/wiki/logic-gates/#:~:text=Logic%20gates%20are%20devices%20that,inputs%20and%20only%20one%20output

[Accessed 7 July 2023].

All About Circuits. (n.d.). Binary Math.

Available at: https://www.allaboutcircuits.com/worksheets/binary-math/

[Accessed 7 July 2023].

BYJU'S. (n.d.). Boolean Algebra: Properties & Laws.

Available at: https://byjus.com/maths/boolean-algebra/

[Accessed 7 July 2023].

Electronics Hub. (n.d.). Boolean Logic: SOP Form, POS Form.

Available at: https://www.electronicshub.org/boolean-logic-sop-form-pos-form/

[Accessed 7 July 2023].