DISCRETE MATHEMATICS

Complete Study Notes

UNIT I: Sets & Propositions • UNIT II: Relations & Functions
UNIT III: Trees • UNIT IV: Graph Theory • UNIT V: Algebraic Structures

Simple Words • Clear Examples • Exam Ready

UNIT I

Set and Propositions

Propositions, Logic, Sets & Set Operations

📐 Propositions & Logic

A Proposition is a statement that is either TRUE or FALSE.

TAUTOLOGY	A compound proposition that is ALWAYS TRUE for every combination of truth values of its variables. **Example:** *$P \vee \neg P$ is always true (either P is true, or not-P is true)*
CONTRADICTION	A compound proposition that is ALWAYS FALSE for every combination of truth values. **Example:** *$P \wedge \neg P$ is always false (P cannot be both true and false)*

Truth Table Example:

P	Q	P ∧ Q (Contradiction check)	P → Q (Tautology check)
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	T

✓ $P \rightarrow Q$ (Tautology if always T) | ✗ $P \wedge \neg P$ (Contradiction if always F)

Q2. What is Logical Equivalence?

Two propositions are Logically Equivalent if they have the same truth value for ALL possible inputs.
Written as: $P \equiv Q$

📌 **Example:** *$P \rightarrow Q \equiv \neg P \vee Q$ (These two always have the same truth values, so they are logically equivalent)*

🔑 *Think of it like two different routes to the same destination — same result, different path!*

Sets & Set Operations

Q3. Explain: Symmetric Difference, Union, Intersection, Subset — with examples.

Union $A \cup B$	All elements that are in A OR B or both. **Example:** *$A=\{1,2,3\}$, $B=\{3,4,5\} \rightarrow A\cup B=\{1,2,3,4,5\}$*
Intersection $A \cap B$	Only elements that are in BOTH A and B. **Example:** *$A=\{1,2,3\}$, $B=\{2,3,4\} \rightarrow A\cap B=\{2,3\}$*
Subset $A \subseteq B$	Every element of A is also in B. **Example:** *$A=\{1,2\}$, $B=\{1,2,3\} \rightarrow A\subseteq B$ ✓*
Symmetric Difference $A \triangle B$	Elements in A or B but NOT in both. $(A-B) \cup (B-A)$ **Example:** *$A=\{1,2,3\}$, $B=\{3,4,5\} \rightarrow A\triangle B=\{1,2,4,5\}$*

Q4. Explain: Universal Set, Complement, Power Set, Proper Subset.

Universal Set U	The set that contains ALL possible elements in a given context. **Example:** *$U = \{1,2,3,4,5,6,7,8,9,10\}$ (for single-digit numbers)*
Complement A'	All elements in Universal Set U that are NOT in A. **Example:** *$U=\{1,2,3,4,5\}$, $A=\{1,2\} \rightarrow A'=\{3,4,5\}$*
Power Set P(A)	The set of ALL possible subsets of A (including empty set and A itself). If $\|A\|=n$, then $\|P(A)\|=2^n$ **Example:** *$A=\{1,2\} \rightarrow P(A)=\{\emptyset,\{1\},\{2\},\{1,2\}\} \rightarrow$ 4 subsets*
Proper Subset $A \subset B$	A is a subset of B but $A \neq B$ (B has at least one more element). **Example:** *$A=\{1,2\}$, $B=\{1,2,3\} \rightarrow A\subset B$ (proper) | $A=\{1,2\}$, $B=\{1,2\} \rightarrow A\subseteq B$ (not proper)*

UNIT II
Relations and Functions
Equivalence Relations, Binary Relations & Types of Functions

🔗 Relations

Q1. What is Equivalence Relation? Explain properties of binary relations.

A Binary Relation R on a set A is a set of ordered pairs (a, b) where a, b ∈ A.

Properties of Binary Relations:

Reflexive	Every element relates to itself. (a, a) ∈ R for all a ∈ A **Example:** *R on integers: a = a ✓ (reflexive)*
Symmetric	If (a,b) ∈ R then (b,a) ∈ R **Example:** *If a=b, then b=a ✓*
Transitive	If (a,b) ∈ R and (b,c) ∈ R, then (a,c) ∈ R **Example:** *If a=b and b=c, then a=c ✓*
Anti-symmetric	If (a,b) ∈ R and (b,a) ∈ R, then a = b **Example:** *≤ relation: if a≤b and b≤a then a=b*

Equivalence Relation = Reflexive + Symmetric + Transitive

📌 **Example:** *'Same age' relation among people: (1) Everyone is the same age as themselves, (2) If A is same age as B → B same age as A, (3) If A same as B and B same as C → A same as C. ✓ Equivalence!*

🔧 Functions

Q2. Explain the various types of functions.

A Function f: A → B maps every element of A to exactly one element of B.

One-to-One (Injective)	Different inputs give different outputs. No two elements in A map to same element in B. **Example:** *f(x)=2x: f(1)=2, f(2)=4 — unique outputs ✓*

Onto (Surjective)	Every element in B has at least one element from A mapping to it. Range = Co-domain. **Example:** *f: {1,2}⟶{a,b}, f(1)=a, f(2)=b ✓*
Bijective (One-to-one + Onto)	Both injective AND surjective. Perfect pairing between A and B. **Example:** *f(x)=x+1: Every output is unique and every value is covered ✓*
Constant Function	All inputs map to the same output. **Example:** *f(x)=5 for all x ∈ A*
Identity Function	Every element maps to itself. f(x) = x **Example:** *f(3)=3, f(7)=7 ✓*
Inverse Function f^{-1}	Reverses the mapping. Exists only for bijective functions. **Example:** *f(x)=2x → $f^{-1}(x)=x/2$*

UNIT III
Introduction to Trees
Tree Definitions, BST, Spanning Trees & Prim's Algorithm

Tree Definitions & Properties

Q1. Define: Forest, Fundamental Cutsets, Game Tree.

Forest	A graph with NO cycles. A collection of one or more disjoint trees. **Example:** *If tree T has components T1, T2, T3 each without cycles → together they form a Forest*
Fundamental Cutset	A minimal set of edges whose removal disconnects the graph. Each cutset corresponds to one tree edge. **Example:** *Remove edge (A-B) from spanning tree → splits graph into two parts → that edge's cutset = all edges crossing the split*
Game Tree	A tree that represents all possible moves in a game. Nodes = game states, Edges = moves. **Example:** *Tic-Tac-Toe tree: root = empty board, children = all first moves, leaves = win/lose/draw states*

Q2. Define: Level of a Tree, Height of a Tree, Fundamental Circuit.

Level of a Node	The distance (number of edges) from the root to that node. Root is at Level 0. **Example:** *Root=Level 0, Root's children=Level 1, Grandchildren=Level 2...*
Height of a Tree	The maximum level of any node in the tree = length of the longest path from root to leaf. **Example:** *If deepest leaf is at Level 4 → Height = 4*
Fundamental Circuit	A circuit formed by adding a non-tree edge to a spanning tree. Each non-tree edge creates exactly one circuit. **Example:** *Spanning tree T + one extra edge e → exactly one cycle containing e = fundamental circuit*

Q3. Define with example: Binary Tree, Ordered Tree, Eccentricity of Vertex.

Binary Tree	A rooted tree where each node has AT MOST 2 children (left and right). **Example:** *A / \ B C / \ D E*
Ordered Tree	A rooted tree where the children of each node are arranged in a specific order (left-right matters). **Example:** *Children (B, C) is different from (C, B) in an ordered tree*
Eccentricity of Vertex	The maximum distance from a vertex v to any other vertex in the graph. **Example:** *If vertex A has distances 1,2,3 to others → Eccentricity(A) = 3*

Q4. Define with example: Binary Search Tree, Spanning Tree.

Binary Search Tree (BST)	A binary tree where: Left child < Parent < Right child. Enables fast searching, insertion, deletion. **Example:** *Insert 5,3,7,1,4 → Root=5, Left=3(with 1,4), Right=7. Search 4: 5→3→4 ✓*
Spanning Tree	A subgraph that includes ALL vertices of the original graph and is a tree (connected + no cycles). If graph has n vertices → spanning tree has n-1 edges. **Example:** *Graph with 4 vertices A,B,C,D → Spanning tree uses 3 edges connecting all 4 with no cycles*

Q5. Explain: Cutset, Tree Properties, Prefix Code.

Cutset	A set of edges whose removal disconnects the graph. A minimal cutset is called a bond. **Example:** *Removing edges {AB, AC} might disconnect vertex A from the rest*

Tree Properties:

- A tree with n vertices has exactly n-1 edges
- There is exactly ONE path between any two vertices
- A tree is connected and has no cycles
- Adding any edge creates exactly one cycle
- Removing any edge disconnects the tree

Prefix Code (Huffman)	A set of binary codes where no code is a prefix of another. Used in data compression. **Example:** *a=0, b=10, c=110, d=111 → No code is prefix of another ✓*

Q6. Explain Complete Binary Tree.

Complete Binary Tree	A binary tree where ALL levels are completely filled EXCEPT possibly the last level, which is filled from LEFT to RIGHT. **Example:** *Level 0: [A], Level 1: [B,C], Level 2: [D,E,F,G] → Completely filled → Complete Binary Tree ✓*

🔑 *Key fact: A complete binary tree with n nodes has height = $\lfloor \log_2 n \rfloor$*

Q7. Explain Regular m-ary Tree with example.

Regular m-ary Tree	A rooted tree where EVERY internal node has exactly m children (not fewer, not more). **Example:** *m=2 (Binary): Every non-leaf has exactly 2 children m=3 (Ternary): Every non-leaf has exactly 3 children*

📌 **Example:** *A full binary tree with 3 levels: Root has 2 children, each child has 2 children → 4 leaf nodes. Total nodes = 1+2+4=7, Internal nodes=3, Leaves=4*

Q8. What is Minimum Spanning Tree? Explain Prim's Algorithm.

Minimum Spanning Tree (MST)	A spanning tree of a weighted graph with the MINIMUM total edge weight. **Example:** *Used in: network design, road planning, minimizing cable costs*

Prim's Algorithm — Step by Step:

- Step 1: Start with any vertex. Mark it as visited.
- Step 2: Find the minimum weight edge connecting visited and unvisited vertex.
- Step 3: Add that edge and the new vertex to the MST.
- Step 4: Repeat steps 2-3 until all vertices are included.

📌 **Example:** *Graph: A-B(4), A-C(2), B-C(1), B-D(5), C-D(8) Start at A → Pick A-C(2) → Pick C-B(1) → Pick B-D(5) MST = A-C, C-B, B-D. Total cost = 2+1+5 = 8*

UNIT IV

Introduction to Graph Theory

Graph Terminology, Euler/Hamilton, Planar Graphs & Isomorphism

Graph Terminology & Types

Q1. Define: Factor of Graph, Weighted Graph, Bipartite Graph.

Factor of Graph	A spanning subgraph of G where every vertex has degree ≥ 1. A 1-factor is a perfect matching. **Example:** *A 2-factor is a spanning subgraph where every vertex has degree exactly 2*
Weighted Graph	A graph where each edge has a numerical value (weight/cost) assigned to it. **Example:** *Road network: vertices=cities, edges=roads, weights=distances in km*
Bipartite Graph	A graph whose vertices can be split into two disjoint sets X and Y such that every edge connects a vertex in X to a vertex in Y (no edge within same set). **Example:** *Job assignment: X=workers, Y=jobs, edges=assignments. No two workers directly connected.*

Q2. Define: Complete Graph, Regular Graph, Complete Bipartite, Paths & Circuits, Acyclic Graph, Complement.

Complete Graph K_n	A graph where EVERY pair of vertices is connected by an edge. Has n(n-1)/2 edges. **Example:** *K_4: 4 vertices, each connected to all others → 6 edges*
Regular Graph	A graph where EVERY vertex has the same degree (number of edges). **Example:** *K_3 is 2-regular (each vertex has degree 2)*
Complete Bipartite K_{mn}	Bipartite graph where every vertex in set X connects to every vertex in set Y. Has m×n edges. **Example:** *K_{23}: 2+3=5 vertices, 2×3=6 edges*
Path	A walk where no vertex is repeated. **Example:** *A→B→C→D (no vertex repeated) = Path*
Circuit	A closed path (starts and ends at same vertex, no repeated edges).

	Example: *A→B→C→A = Circuit of length 3*
Acyclic Graph	A graph with NO cycles. A connected acyclic graph = Tree. **Example:** *A tree is always acyclic*
Complement of Graph G'	A graph on same vertices where two vertices are adjacent in G' if and only if they are NOT adjacent in G. **Example:** *K_4 complement = empty graph (no edges)*

Q3. Define the graph Kn and Kmn.

K_n (Complete Graph)	Graph with n vertices where every pair of vertices has exactly one edge. Edges = n(n-1)/2. **Example:** *K_5: 5 vertices, 10 edges. Each vertex has degree 4.*
K_{mn} (Complete Bipartite)	Bipartite graph with m vertices in set X and n vertices in set Y. Every vertex in X connects to every vertex in Y. Edges = m×n. **Example:** *K_{34}: 3+4=7 vertices, 3×4=12 edges.*

Q4. Explain: Planar Graph, Eulerian Graph.

Planar Graph	A graph that can be drawn on a flat surface (plane) with NO edges crossing each other. **Example:** *K_4 is planar. K_5 and K_{33} are NOT planar (Kuratowski's theorem). Euler formula: V-E+F=2*
Eulerian Graph	A graph that contains an Euler Circuit — a path that visits EVERY EDGE exactly once and returns to start. **Example:** *Königsberg Bridge problem! A graph is Eulerian if all vertices have EVEN degree.*

Q5. Explain Colouring of Graph.

Graph Coloring = Assigning colors to vertices so that no two ADJACENT vertices have the same color.

Chromatic Number χ(G): Minimum number of colors needed to properly color graph G.

📌 **Example:** *Map coloring: Adjacent countries get different colors. Any map can be colored with at most 4 colors (Four Color Theorem).*

- $\chi(K_n) = n$ (complete graph needs n colors)

- χ(Bipartite) = 2 (only 2 colors needed)
- χ(Tree) = 2 (trees are bipartite)
- χ(Cycle with even nodes) = 2, odd nodes = 3

Q6. Explain: Adjacency Matrix and Incidence Matrix.

Adjacency Matrix	An n×n matrix A where A[i][j]=1 if there is an edge between vertex i and j, else 0. **Example:** *Graph: A-B, A-C, B-C → Row A: [0,1,1], Row B: [1,0,1], Row C: [1,1,0]*
Incidence Matrix	An n×m matrix M (n=vertices, m=edges) where M[i][j]=1 if vertex i is an endpoint of edge j, else 0. **Example:** *3 vertices, 3 edges → 3×3 matrix showing which vertex touches which edge*

Q7. Explain with example: Connected Graphs.

Connected Graph	A graph where there is a PATH between EVERY pair of vertices. **Example:** *A-B-C-D: Can reach any vertex from any other → Connected ✓*
Disconnected Graph	A graph where at least one pair of vertices has no path between them. **Example:** *Graph with isolated vertex E and connected component {A,B,C} → Disconnected*

🔑 A connected graph with n vertices has at least n-1 edges.

Q8. Explain: Edge Connectivity and Vertex Connectivity.

Edge Connectivity λ(G)	Minimum number of edges that must be REMOVED to disconnect the graph. **Example:** *$\lambda(G)=2$ means you need to remove at least 2 edges to disconnect G*
Vertex Connectivity κ(G)	Minimum number of vertices that must be REMOVED to disconnect the graph (or make it trivial). **Example:** *$\kappa(G)=1$ means there is a single vertex (cut vertex) whose removal disconnects G*

Whitney's Theorem: $\kappa(G) \leq \lambda(G) \leq \delta(G)$ where δ = minimum degree

Q9. Explain: 'Every graph with chromatic number 2 is a bipartite graph.'

Statement: $\chi(G) = 2 \iff G$ is bipartite (and non-empty).

Proof Logic:

- If $\chi(G) = 2$: We can 2-color G. Let Set X = vertices with color 1, Set Y = vertices with color 2.
- Since adjacent vertices have different colors, every edge goes from X to Y.
- So G is bipartite by definition! ✓

📌 **Example:** *Cycle C_4 (4-node cycle): Color alternately Red-Blue-Red-Blue → χ=2 → It IS bipartite (sets {1,3} and {2,4}). Cycle C_3 (triangle): Cannot be 2-colored → χ=3 → NOT bipartite ✓*

🔄 Graph Isomorphism

Q10. What is Graph Isomorphism?

Two graphs G_1 and G_2 are Isomorphic if there exists a one-to-one correspondence (bijection) between their vertices that preserves all edges.
They look different but are structurally the SAME graph.

Conditions for Isomorphism:

- Same number of vertices
- Same number of edges
- Same degree sequence (sorted list of degrees)
- Same number of cycles of each length

📌 **Example:** *G_1: A-B-C-D-A (cycle) and G_2: W-X-Y-Z-W (cycle) → Both are C_4 → They are isomorphic!*

Euler & Hamilton Paths

Q11. Conditions for Hamiltonian Path and Eulerian Path.

Eulerian Conditions:

Euler Circuit	Exists if: Graph is connected AND every vertex has EVEN degree. **Example:** *All vertices even degree → Euler Circuit exists*
Euler Path (not circuit)	Exists if: Graph is connected AND EXACTLY 2 vertices have ODD degree (these are start/end). **Example:** *Exactly 2 odd-degree vertices → Euler Path exists (not circuit)*

Hamiltonian Conditions:

Hamilton Path	A path visiting every VERTEX exactly once. No simple necessary & sufficient condition (NP-complete problem). **Example:** *Ore's Theorem: If deg(u)+deg(v)≥n-1 for all non-adjacent pairs → Ham. Path exists*
Hamilton Circuit	Hamilton Path that returns to start. Ore's Theorem: deg(u)+deg(v)≥n for all non-adjacent pairs. **Example:** *Dirac's Condition: If every vertex has degree ≥ n/2 → Hamilton Circuit exists*

Q12. Explain with example: Euler Path & Circuit, Hamilton Path & Circuit.

Euler Path	A trail (no repeated edges) that visits EVERY EDGE exactly once. **Example:** *Graph: A-B-C-A-D-B → Euler Path: visits each edge once*
Euler Circuit	An Euler Path that starts and ends at the SAME vertex. **Example:** *Königsberg: Not possible because 4 vertices have odd degree*
Hamilton Path	A path that visits every VERTEX exactly once (edges can be skipped). **Example:** *Graph K_4: A→B→C→D visits all 4 vertices once = Hamilton Path*
Hamilton Circuit	A Hamilton Path that returns to starting vertex. **Example:** *A→B→C→D→A in K_4 = Hamilton Circuit*

🔑 **Key Difference: Euler = all EDGES once | Hamilton = all VERTICES once**

Q13. What is a Planar Graph?

Planar Graph	A graph that can be drawn in a plane such that NO two edges cross each other (except at shared vertices). **Example:** *K_4 is planar. K_5 and K_{33} are non-planar.*

Euler's Formula for Planar Graphs: V - E + F = 2

(V = vertices, E = edges, F = faces including outer face)

📌 **Example:** *Cube graph: V=8, E=12, F=6 → 8-12+6=2 ✓*

- Non-planar test: K_5 (5 vertices, 10 edges) violates E ≤ 3V-6
- K_{33} (6 vertices, 9 edges) violates E ≤ 2V-4 (bipartite check)

UNIT V
Counting Principles & Algebraic Structures
Groups, Rings, Fields, Homomorphism & Codes

🏛 Algebraic Structures

Q1. Define with examples: Groupoid, Semigroup, Monoid, Abelian Group, Subgroup.

Algebraic Structure = A set with one or more binary operations satisfying certain properties.

Groupoid	A set with a binary operation (closure only). No other properties required. **Example:** *(ℤ, -): Integers under subtraction. 3-5=-2 ∈ ℤ ✓ (closed)*
Semigroup	Groupoid + Associativity: (a*b)*c = a*(b*c) **Example:** *(ℤ, +): (1+2)+3 = 1+(2+3) = 6 ✓*
Monoid	Semigroup + Identity element (e such that a*e = e*a = a) **Example:** *(ℤ, +): Identity = 0. a+0=0+a=a ✓*
Group	Monoid + Inverse (for every a, ∃ a^{-1} such that $a*a^{-1} = e$) **Example:** *(ℤ, +): Inverse of 3 is -3. 3+(-3)=0 ✓*
Abelian Group	Group + Commutativity: a*b = b*a **Example:** *(ℤ, +): 3+5 = 5+3 = 8 ✓ | Matrix multiplication is NOT abelian*
Subgroup	A subset H of group G that is itself a group under the same operation. **Example:** *G=(ℤ,+), H=(even integers,+) → H is a subgroup of G ✓*

Q2. Define: Properties of Binary Operation, Ring with Unity, Field, Integral Domain.

Binary Operation *: A×A → A (takes two elements, gives one result from same set).

Properties:

- Closure: a*b ∈ A
- Associativity: (a*b)*c = a*(b*c)
- Commutativity: a*b = b*a
- Identity: a*e = e*a = a
- Inverse: $a*a^{-1} = e$

Ring	A set with TWO operations (+, ×): (R,+) is Abelian group, (R,×) is semigroup, × distributes over +. **Example:** *($\mathbb{Z}$, +, ×): Integers form a ring*
Ring with Unity	A ring that has a multiplicative identity element 1 (where 1×a = a×1 = a). **Example:** *($\mathbb{Z}$, +, ×): 1 is the unity since 1×n = n ✓*
Integral Domain	A commutative ring with unity and NO zero divisors (if a×b=0 then a=0 or b=0). **Example:** *($\mathbb{Z}$, +, ×) is an integral domain. No two non-zero integers multiply to 0.*
Field	A commutative ring with unity where EVERY non-zero element has a multiplicative inverse. **Example:** *($\mathbb{Q}$, +, ×): Rationals form a field. 1/3 has inverse 3 ✓. $\mathbb{Z}$ is NOT a field (2 has no integer inverse).*

Q3. Define with examples: Cyclic Group and Cosets.

Cyclic Group	A group G that can be generated by a single element a. Every element is a power of a. G = <a> = $\{a^n : n \in \mathbb{Z}\}$ **Example:** *$\mathbb{Z}_4$ = {0,1,2,3} under addition mod 4. Generator = 1: 1,2,3,0,1,2... ✓*
Coset	For subgroup H of G and element g ∈ G: Left coset = gH = {g*h : h ∈ H}. Right coset = Hg. **Example:** *G=$\mathbb{Z}_6$, H={0,3}. Cosets: 0+H={0,3}, 1+H={1,4}, 2+H={2,5} — these partition G!*

🔑 Lagrange's Theorem: |H| divides |G| — order of subgroup divides order of group.

Q4. Define with examples: Group Codes.

Group Code	A linear error-correcting code where the set of valid codewords forms a GROUP under binary addition (XOR). Used for error detection and correction. **Example:** *Binary code {000, 011, 101, 110}: XOR of any two codewords gives another codeword → forms a group!*

Key Property: Group codes can detect and correct transmission errors.

- Minimum distance d determines error detection/correction capability
- Can detect d-1 errors and correct ⌊(d-1)/2⌋ errors

Q5. Explain Algebraic System and Properties of Binary Operations.

Algebraic System	A non-empty set S together with one or more binary operations defined on it. **Example:** *(ℤ, +, ×) is an algebraic system with two operations*

Summary of Properties:

Property	Definition	Example
Closure	$a*b \in S$	$2+3=5 \in \mathbb{Z}$ ✓
Associativity	$(a*b)*c = a*(b*c)$	$(1+2)+3=1+(2+3)$ ✓
Commutativity	$a*b = b*a$	$2+3=3+2$ ✓
Identity	$a*e = e*a = a$	$a+0=a$ ✓
Inverse	$a*a^{-1} = e$	$3+(-3)=0$ ✓
Idempotent	$a*a = a$	$1\wedge 1=1$ (Boolean) ✓

Q6. Explain: Identity and Inverse (Properties of Algebraic Structure).

Identity Element	An element e in set S such that $a*e = e*a = a$ for all $a \in S$. It leaves every element unchanged. **Example:** *Addition: 0 is identity ($a+0=a$). Multiplication: 1 is identity ($a\times 1=a$). String concatenation: '' (empty string) is identity.*
Inverse Element	For element a, its inverse a^{-1} is the element such that $a*a^{-1} = a^{-1}*a = e$ (identity). **Example:** *Addition: Inverse of 5 is -5 ($5+(-5)=0$). Multiplication: Inverse of 2 is 1/2 ($2\times 1/2=1$).*

Q7. Explain: Group and Homomorphism of Groups.

Group (G, *)	A set G with binary operation * satisfying: Closure, Associativity, Identity, Inverse. **Example:** *(ℤ, +): Closed ✓, Associative ✓, Identity=0 ✓, Inverse=-a ✓→ Group!*
Group Homomorphism	A function $f: G\to H$ between two groups that preserves the group operation: $f(a*b) = f(a)\circ f(b)$ **Example:** *$f: (\mathbb{Z},+)\to(\mathbb{Z},+)$ defined by $f(x)=2x$. $f(a+b)=2(a+b)=2a+2b=f(a)+f(b)$ ✓*

Special Homomorphisms:

- Monomorphism: Injective (one-to-one) homomorphism
- Epimorphism: Surjective (onto) homomorphism

- Isomorphism: Bijective homomorphism (groups are structurally identical)
- Automorphism: Isomorphism from G to itself

Q8. What is Hamming Distance?

Hamming Distance d(u,v)	The number of positions at which two binary strings of equal length DIFFER. **Example:** *u=1010, v=1101 → Differ at positions 2,3,4 → d(u,v) = 3*

Why it matters:

- Minimum Hamming Distance d_{min} of a code determines its error-handling ability
- Can detect up to $d_{min} - 1$ errors
- Can correct up to $\lfloor(d_{min} - 1)/2\rfloor$ errors

📌 **Example:** *Code {000, 111}: d_{min} = 3. Can detect 2 errors, correct 1 error. Receive 001 → closest codeword is 000 → corrected! ✓*

Q9. Define: Ring and Ring Homomorphism.

Ring (R, +, ×)	A set R with two operations: (R,+) is Abelian group, (R,×) is semigroup, multiplication distributes over addition. **Example:** *(ℤ, +, ×): Integers under addition and multiplication form a ring ✓*
Ring Homomorphism	A function f: R→S between two rings that preserves BOTH operations: f(a+b) = f(a)+f(b) AND f(a×b) = f(a)×f(b) **Example:** *f: $\mathbb{Z} \to \mathbb{Z}_n$ defined by f(a) = a mod n. Preserves both + and × mod n ✓*

Special Types:

- Ring Isomorphism: Bijective ring homomorphism
- Kernel of f: {a ∈ R : f(a) = 0} — forms an ideal