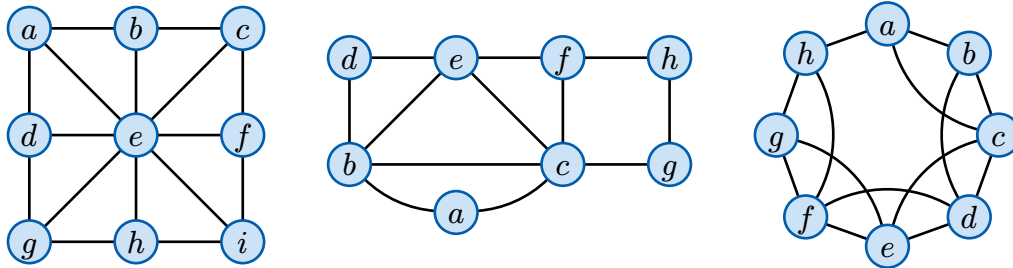


Problem 1: Graph Invariants Analysis

Core

For each of the following graphs, compute the requested metrics and properties.



1. Basic connectivity metrics:

- Minimum degree $\delta(G)$, maximum degree $\Delta(G)$
- Vertex connectivity $\kappa(G)$, edge connectivity $\lambda(G)$
- All $(\kappa + 1)$ -connected components, $(\lambda + 1)$ -edge-connected components
- Verify Whitney's inequality: $\kappa(G) \leq \lambda(G) \leq \delta(G)$

2. Distance metrics:

- $\text{ecc}(v)$ for every vertex $v \in V(G)$
- $\text{rad}(G)$, $\text{diam}(G)$, $\text{center}(G)$, $\text{girth}(G)$
- Verify the bounds: $\text{rad}(G) \leq \text{diam}(G) \leq 2 \cdot \text{rad}(G)$

3. Structural properties:

- Is the graph Eulerian? Hamiltonian? Bipartite? Justify each answer.
- Find: maximum clique $Q \subseteq V$, maximum stable set $S \subseteq V$, minimum dominating set $D \subseteq V$
- Find: maximum matching $M \subseteq E$. Is M perfect?
- Find: minimum vertex cover $R \subseteq V$, minimum edge cover $F \subseteq E$
- Find: minimum vertex coloring $C : V \rightarrow \{1, 2, \dots, \chi(G)\}$
- Find: minimum edge coloring $C : E \rightarrow \{1, 2, \dots, \chi'(G)\}$

Problem 2: Degree Sequences

Core

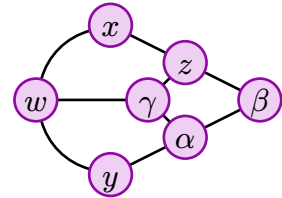
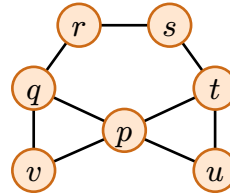
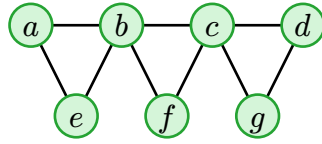
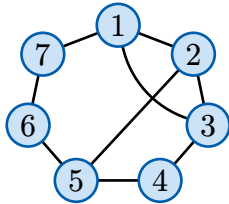
A sequence $d = (d_1, d_2, \dots, d_n)$ with $d_1 \geq d_2 \geq \dots \geq d_n \geq 0$ is *graphical* if there exists a simple graph G with this degree sequence.

- For each sequence below, determine whether it is graphical. If yes, construct a graph realizing it. If no, explain why (cite the Erdős–Gallai criterion or parity/sum arguments).
 - (5, 4, 3, 2, 2, 2)
 - (3, 3, 3, 3, 3, 3)
 - (4, 4, 3, 2, 1)
 - (6, 3, 3, 3, 3, 2, 2)
 - (1, 1, 1, 1, 1, 1)
- The total number of non-isomorphic simple undirected graphs on n vertices is given by the sequence (1, 2, 4, 11, 34, 156, ...) (OEIS A000088). The number of distinct degree sequences of length n is given by the sequence (1, 2, 4, 11, 31, 102, ...) (OEIS A004251). Explain why the second sequence grows more slowly than the first, and determine which degree sequences of length $n = 5$ are realizable by more than one graph.

Problem 3: Graph Isomorphism

Core

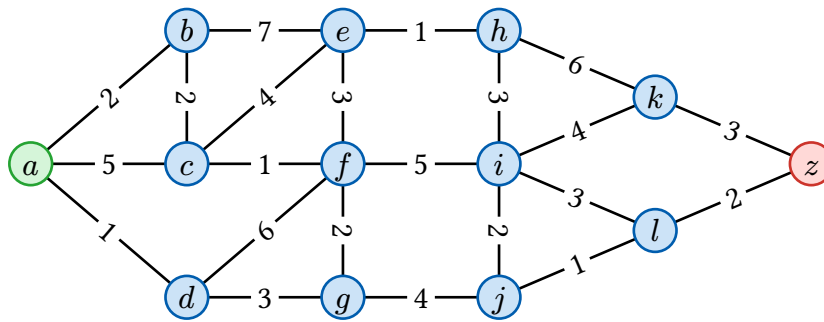
Determine which pairs of graphs below are *isomorphic*. For each isomorphic pair (G, H) , exhibit an explicit bijection $f : V(G) \rightarrow V(H)$ that preserves adjacency. For each non-isomorphic pair, identify a distinguishing invariant (degree sequence, girth, number of triangles, *etc.*).



Problem 4: Dijkstra's Shortest Path Algorithm

Essential

Apply **Dijkstra's algorithm** to find a shortest path from a to z in the weighted graph below.



- Trace the algorithm** step-by-step:
 - Maintain a table showing, after each iteration: the visited set V_{visited} , the current distance labels $d(v)$ for all $v \in V$, and the predecessor pointers $\pi(v)$.
 - Reconstruct a shortest path from a to z , and compute its total weight.
- Correctness & Structure:**
 - State the key *invariant* that Dijkstra maintains during execution: what is guaranteed to be true about the distances $d(v)$ for visited or yet unvisited vertices?
 - In a graph where some shortest paths are not unique, the predecessor pointers form a **shortest path DAG** rooted at a . Draw this DAG (include only edges corresponding to some $\pi(v)$, and mark vertex z). How many different shortest paths from a to z are in this DAG?
 - Dijkstra fails on graphs with negative-weight edges. Give a small example (3–4 vertices, one negative edge) where the algorithm produces an incorrect result, and explain *why* the invariant from (a) is violated.

Problem 5: Find the Error

Core

The following “proof” contains a subtle error. Identify the error and explain why the conclusion is false and the claim is not valid.

False Claim: Every tree with n vertices has a path of length $n - 1$.

“Proof:”

Base case: A tree with one vertex clearly has a path of length $0 = 1 - 1$.

Inductive step: Assume that every tree with n vertices has a path of length $n - 1$, which terminate at some leaf u . Add a new vertex v and connect it to u with an edge. The resulting tree has $n + 1$ vertices and contains a path of length n , which is $(n + 1) - 1$. ■

Problem 6: Trees

Essential

The following are six fundamental characterizations of trees. Prove that they are all equivalent for a graph $G = \langle V, E \rangle$ with $n = |V|$ vertices and $m = |E|$ edges by establishing a cycle of implications:

$$(1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (4) \Rightarrow (5) \Rightarrow (6) \Rightarrow (1)$$

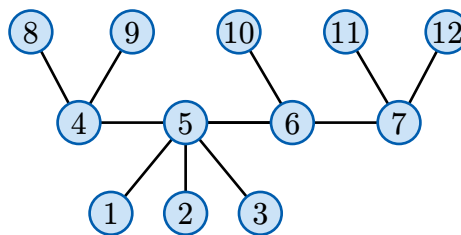
Theorem: The following are equivalent:

1. G is a tree (connected and acyclic).
2. G is connected and $m = n - 1$.
3. G is acyclic and $m = n - 1$.
4. For every pair of vertices $u, v \in V$, there exists a *unique* path from u to v .
5. G is connected, but removing any edge disconnects it.
6. G is acyclic, but adding any new edge creates exactly one cycle.

Problem 7: Prüfer Code

Core

The Prüfer sequence provides a bijection between labeled trees on n vertices and sequences of length $n - 2$ with entries in $\{1, 2, \dots, n\}$.



1. Encode the labeled tree above into its Prüfer sequence.
2. Decode the Prüfer sequence $(3, 3, 3, 7, 7, 5)$ back into a labeled tree. Draw the result.
3. Prove that in the Prüfer sequence the number i appears exactly $\deg(i) - 1$ times.

Problem 8: Minimum Spanning Trees

Essential

Consider the weighted graph from Problem 4.

1. Use **Kruskal's algorithm** to find a minimum spanning tree (MST). Show the edges in the order they are considered, and mark which are added to the MST.
2. Use **Prim's algorithm** starting from vertex a . Show how the tree grows step by step.
3. Prove: If all edge weights in a connected graph are *distinct*, then the MST is unique.

Problem 9: Eulerian Graphs

Essential

1. For which values of $n \geq 3$ does the complete graph K_n have an Euler circuit? An Euler trail (but not a circuit)? Prove your answers.

- For which pairs (m, n) with $m, n \geq 1$ does the complete bipartite graph $K_{m,n}$ have an Euler circuit? An Euler trail?
- State and prove Euler's criterion: A connected graph has an Euler circuit if and only if every vertex has even degree.

Problem 10: Hamiltonian Graphs

Challenge

- Prove that $K_{m,n}$ is Hamiltonian if and only if $m = n \geq 2$.
- The n -dimensional **hypercube graph** Q_n has vertex set $\{0, 1\}^n$ (all binary strings of length n), and two vertices are adjacent iff they differ in exactly one coordinate. Prove that Q_n has a Hamiltonian cycle for all $n \geq 2$.
- Construct a 2-connected non-Hamiltonian graph with at least 6 vertices.

Problem 11: Bipartite Graphs

Essential

Prove the following fundamental properties of bipartite graphs.

- A graph G is bipartite if and only if it contains no odd cycle.
- If G is bipartite and d -regular with $d \geq 1$, then G has a perfect matching.
- In any bipartite graph, the size of a maximum matching equals the size of a minimum vertex cover.

Problem 12: Hall's Marriage Theorem

Essential

A dance school has 6 leaders and 6 followers. Each leader is willing to dance with certain followers, as shown in the table below ($\times$ indicates willingness).

	F_1	F_2	F_3	F_4	F_5	F_6
L_1	$\times$	$\times$				$\times$
L_2	$\times$		$\times$			
L_3		$\times$		$\times$		
L_4			$\times$		$\times$	
L_5				$\times$	$\times$	
L_6	$\times$					$\times$

- Determine whether a perfect matching exists by verifying Hall's condition.
- Find a maximum matching. Is it perfect?
- Now suppose L_2 becomes more selective and will **only** dance with F_1 . Does a perfect matching still exist? If Hall's condition fails, identify the violating subset S and explain why no perfect matching exists.

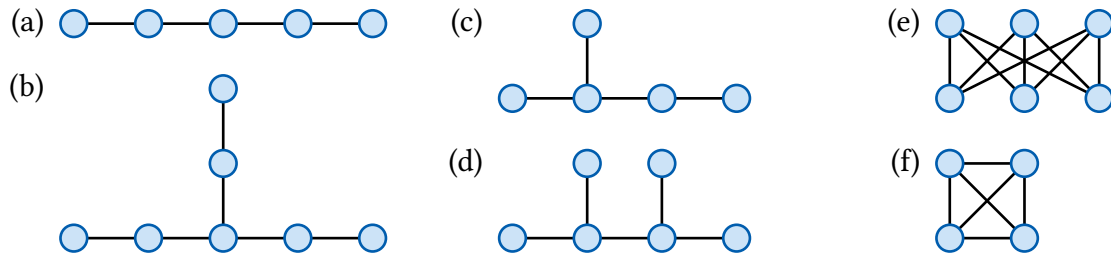
Problem 13: Graceful Graphs

Challenge

A **graceful labeling** of a graph G with m edges is an injective function $f : V(G) \rightarrow \{0, 1, 2, \dots, m\}$ such that the induced edge labels $|f(u) - f(v)|$ for each edge $(u, v) \in E(G)$ are distinct and cover

the set $\{1, 2, \dots, m\}$. In other words, the absolute differences of the labels of adjacent vertices are all different and cover the integers from 1 to m . A graph is called *graceful* if it has a graceful labeling.

Show that the following graphs are graceful (by explicitly constructing a graceful labeling for each).



Problem 14: Fundamental Theorems

Challenge

Prove each of the following theorems *rigorously*.

1. **(Harary)** Every block of a block graph is a clique.

A *block graph* $H = B(G)$ is an intersection graph of all blocks (biconnected components) of G , i.e. each vertex $v \in V(H)$ corresponds to a block of G , and there is an edge $\{u, v\} \in E(H)$ iff “blocks” u and v share a cut vertex.

2. **(Gallai)** For every finite graph G , we have:

$$\alpha(G) + \tau(G) = |V(G)|$$

where $\alpha(G)$ is the size of a maximum independent set and $\tau(G)$ is the minimum vertex cover size.

3. **(Erdős–Gallai)** The sequence $d_1 \geq d_2 \geq \dots \geq d_n \geq 0$ is *graphical* iff $d_1 + \dots + d_n$ is even and

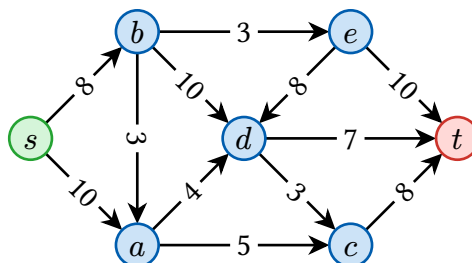
$$\sum_{i=1}^k d_i \leq k(k-1) + \sum_{j=k+1}^n \min(d_j, k)$$

holds for every k in $1 \leq k \leq n$.

Problem 15: Network Flows

Essential

Consider the following flow network $N = (V, E, s, t, c)$ with source s and sink t .



1. **Ford–Fulkerson execution.** Starting from zero flow, perform augmentations until no augmenting path exists.
 - At each iteration, specify the chosen augmenting path and its bottleneck.
 - Give the updated flow value $|f|$ and describe all changed residual capacities.
 - Your solution must include at least one iteration that uses a backward residual edge.

2. **Max-flow/min-cut certificate.**

- Compute a maximum flow value.
- Extract one minimum s - t cut (S, T) from the final residual network.
- Verify numerically that $|f| = c(S, T)$.

3. **Sensitivity analysis.**

- Increase the capacity of edge (d, t) by 2 and recompute the new maximum flow value.
- Identify one edge whose +1 capacity increase does **not** change the maximum flow, and justify via minimum cuts.

4. Prove the **integrality theorem**: if all capacities are integers, then there exists a maximum flow with integer values on all edges.

Notes on Tags System

The problems in this assignment are *tagged*:

Core are mandatory problems that must be completed *before the test*.

Essential are main problems that have to be submitted *before the announced deadline*.

Challenge are slightly more challenging problems with *a flexible deadline*.

Bonus are *optional* problems that can be attempted for ~~extra credit~~ *memorable experience*.

Optional Bonus Problems

The following problems are “*optional*”. They are more challenging and are intended for students who wish to explore deeper topics in graph theory. These problems will *not* count toward your grade but may earn bonus points.

Problem A: Ramsey Theory

Bonus

The *Ramsey number* $R(r, s)$ is the minimum n such that any 2-coloring of the edges of K_n (complete graph) contains either a red K_r or a blue K_s .

1. Prove that $R(3, 3) = 6$ by showing:
 - Any edge-coloring of K_5 with two colors can avoid monochromatic triangles.
 - Any edge-coloring of K_6 with two colors must contain a monochromatic triangle.
2. Show that $R(3, 4) \leq 10$ by using a similar argument.
3. Prove the *Ramsey recurrence*: For all $r, s \geq 2$,

$$R(r, s) \leq R(r - 1, s) + R(r, s - 1)$$

4. Use the recurrence to show that $R(4, 4) \leq 18$.

Problem B: The Friendship Theorem

Bonus

The **Friendship Theorem** (Erdős–Rényi–Sós, 1966) states:

Let G be a finite simple graph in which every pair of distinct vertices has exactly one common neighbor (the **friendship condition**). Then G is a *friendship graph* (also called a *windmill graph* $W_{3,k}$), consisting of k triangles that share exactly one common vertex.

1. Verify the theorem for small cases. Try to construct non-windmill graphs satisfying the friendship condition with $n \leq 8$ vertices.
2. Consider the claim: “Every graph satisfying the friendship condition is regular.” Decide whether this statement is *true* or *false*, and justify your answer by a proof or a counterexample.
3. Prove the **corrected lemma**: If a graph G satisfies the friendship condition and has *no* universal vertex (that is, no vertex adjacent to all other vertices), then G is regular.
4. Using the corrected lemma, complete the proof of the theorem. Show that a finite graph satisfying the friendship condition must have a universal vertex, and therefore is a windmill graph $W_{3,k}$.

Problem C: Max-Flow Solver

Bonus

Design and implement a production-quality max-flow solver.

1. Implement **Edmonds–Karp** (BFS-based Ford–Fulkerson) for directed graphs with non-negative integer edge capacities.

Your code must produce three deliverables per run:

- (1) **Maximum flow value** $|f^*|$ from source s to sink t
- (2) **One minimum cut** (S, T) extracted from the residual network
- (3) **Flow decomposition**: express the solution as a sum of edge-disjoint s - t paths (with multiplicities) and identify any residual cycles

2. Test on three progressively complex instances:

- The flow network from Problem 15
- Random sparse network: $|V| = 30$, $|E| \approx 60$ with random integer capacities $c(e) \in [1..20]$
- Random dense network: $|V| = 30$, $|E| \approx 300$ with same capacity distribution

For each test, verify the *max-flow/min-cut duality*: $|f^*| = c(S, T)$ numerically.

3. Across varying sizes ($|V| \in \{10, 20, \dots, 100\}$), measure and report:

- **Execution time** (wall-clock seconds, average of 3 runs per size)
- **Iteration count**: number of augmenting paths found until termination
- **Empirical scaling**: does runtime grow as $\mathcal{O}(|V| \cdot |E|^2)$ or faster?
Compare fit quality to theoretical bound.
- **Runtime plots**: present as a figure (runtime vs $|V|$ with fitted curve)

4. Implement **Dinic’s algorithm** (blocking flows; theoretical bound $\mathcal{O}(|V|^2 |E|)$) and benchmark against Edmonds–Karp on the same test suite. Which dominates on sparse vs dense graphs? Explain the observed crossover point.

5. Formulate one real-world problem (e.g., airline crew scheduling, data routing, bipartite matching) as a max-flow instance with 20+ vertices. Solve it using your implementation and interpret the flow decomposition.