1 Graph Theory Basics

Definition 1.1 Graph .

A *graph* G is an ordered pair

$$G = (V, E),$$

where V is a finite nonempty set of *vertices*, and E is a finite multiset of unordered pairs of vertices called *edges*. An edge may join two distinct vertices or a vertex to itself.

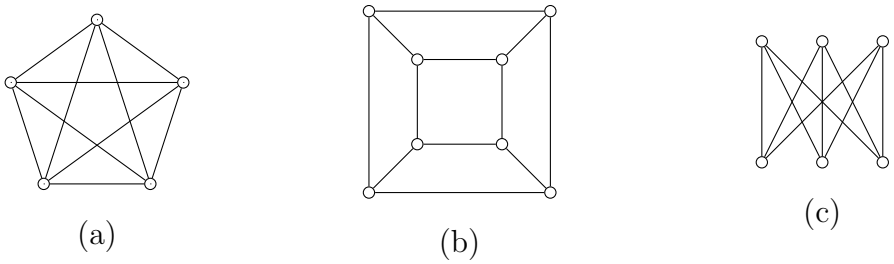
Edges joining a vertex to itself are called *loops*, and multiple edges are edges that join the same pair of vertices more than once.

Definition 1.2 Types of Graphs .

Different types of graphs include:

- A *simple graph* has no loops and no multiple edges.
- A *multigraph* has no loops but may have multiple edges.
- A *pseudograph* may have both loops and multiple edges.

Examples



(a) (b) (c)

Figure 1: (a) K_5; (b) the cube; (c) $K_{3,3}$

Definition 1.3 Order and Size .

The *order* of a graph G is $|V(G)|$, and the *size* of G is $|E(G)|$.

1.1 Graph Isomorphism

Definition 1.4 Isomorphism .

Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* if there exists a bijection

$$\varphi : V \to V'$$

such that for all $u, v \in V$,

$$uv \in E \iff \varphi(u)\varphi(v) \in E'.$$

Definition 1.5 Isomorphism Invariants .

Quantities preserved under isomorphism include:

- order and size,
- degree sequence,
- number of connected components,
- presence or absence of cycles.

1.2 Incidence and Adjacency Matrices

Let G be a graph with vertices $v_1, \ldots, v_n$ and edges $e_1, \ldots, e_m$.

Definition 1.6 Incidence Matrix .

The *incidence matrix* $B = (b_{ij})$ of G is

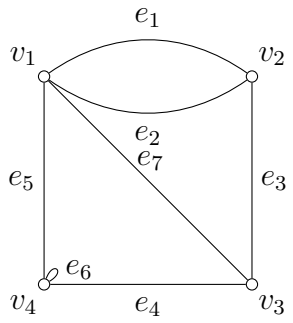
$$b_{ij} = \begin{cases} 1 & \text{if vertex } v_i \text{ is incident with edge } e_j, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 1.7 Adjacency Matrix .

The *adjacency matrix* $A = (a_{ij})$ of G is

$$a_{ij} = \text{number of edges joining } v_i \text{ and } v_j.$$

For a simple graph, A is symmetric and has zeros on the diagonal.



G

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
v_1	1	1	0	0	1	0	1
v_2	1	1	1	0	0	0	0
v_3	0	0	1	1	0	0	1
v_4	0	0	0	1	1	2	0

$\mathbf{M}(G)$

	v_1	v_2	v_3	v_4
v_1	0	2	1	1
v_2	2	0	1	0
v_3	1	1	0	1
v_4	1	0	1	1

$\mathbf{A}(G)$

Figure 2: Incidence and Adjacency Matrix

Theorem 1.1 Walk Enumeration .

For a graph with adjacency matrix A, the (i, j)-entry of A^k equals the number of walks of length k from vertex v_i to vertex v_j.

1.3 Subgraphs

Definition 1.8 Subgraph .

A graph H is a *subgraph* of G if

$$V(H) \subseteq V(G), \quad E(H) \subseteq E(G),$$

with incidence preserved.

Definition 1.9 Induced Subgraph .

For $S \subseteq V(G)$, the *induced subgraph* $G[S]$ has vertex set S and all edges of G with both ends in S.

Definition 1.10 Spanning Subgraph .

A subgraph H of G is *spanning* if $V(H) = V(G)$.

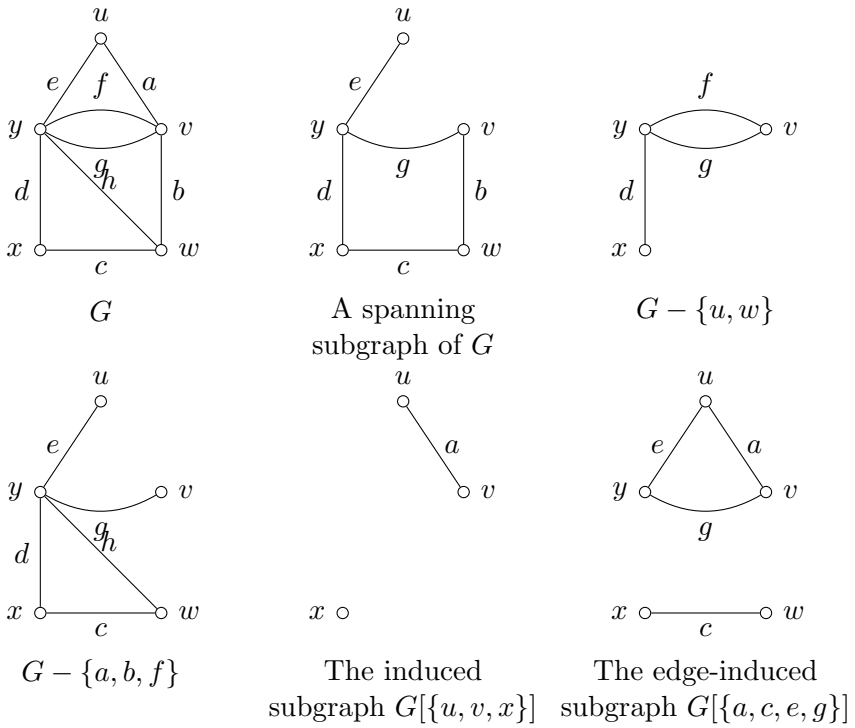


Figure 3: Spanning and Induced Graphs

1.4 Vertex Degrees

Definition 1.11 Degree .

The *degree* $d(v)$ of a vertex v is the number of edges incident with v, with each loop counted twice.

Theorem 1.2 Handshaking Lemma .

For any graph G,

$$\sum_{v \in V(G)} d(v) = 2|E(G)|.$$

Corollary 1.1 .

In any graph, the number of vertices with odd degree is even.

1.5 Paths and Connectivity

Definition 1.12 Walk, Trail, Path .

These are:

- *Walk*: sequence of vertices where consecutive vertices are adjacent.
- *Trail*: walk with no repeated edges.
- *Path*: walk with no repeated vertices.

Definition 1.13 Connected Graph .

A graph is *connected* if there is a path between every pair of vertices.

Theorem 1.3 .

A graph is connected if and only if it contains a spanning tree.

1.6 Cycles and Trees

Definition 1.14 Cycle .

A *cycle* is a closed path of length at least 3 with no repeated vertices except the first and last.

Definition 1.15 Tree .

A *tree* is a connected graph containing no cycles.

Theorem 1.4 .

A graph is acyclic if and only if it contains no cycles.

1.7 Applications

1.7.1 Shortest Path Problem

Problem 1.1 Shortest Path .

Given a connected graph with nonnegative edge weights, find a path of minimum total weight between two specified vertices.

Algorithm 1.1 Dijkstra's Algorithm .

The steps are as it follows:

1. Assign distance 0 to the source vertex and ∞ to all others.
2. Repeatedly select the unvisited vertex of minimum distance.
3. Update distances to adjacent vertices via relaxation.

As the algorithm proceeds, these labels are modified so that, at the end of stage i,

$$l(u) = d(u_0, u) \quad \text{for} \quad u \in S_i$$

and

$$l(v) = \min_{u \in S_{i-1}} \{d(u_0, u) + w(uv)\} \quad \text{for} \quad v \in \bar{S}_i$$

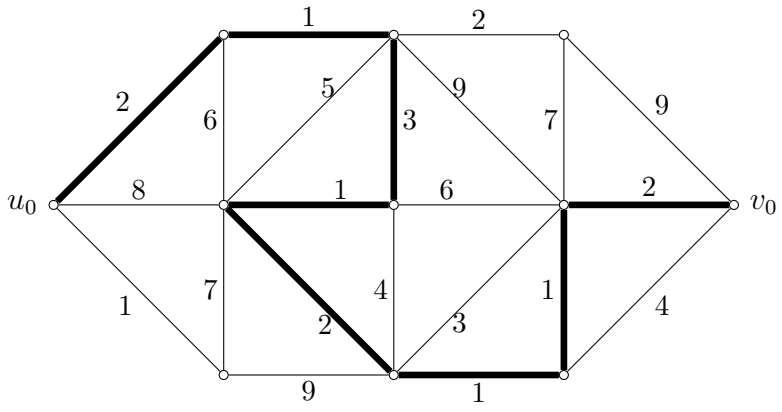


Figure 4: A (u_0, v_0)-path of minimum weight

1.8 Sperner's Lemma

Theorem 1.5 Sperner's Lemma .

Let a triangle be subdivided into smaller triangles with vertices labeled $\{1, 2, 3\}$ such that boundary vertices satisfy the Sperner labeling conditions. Then there exists at least one subtriangle whose vertices have all three labels.

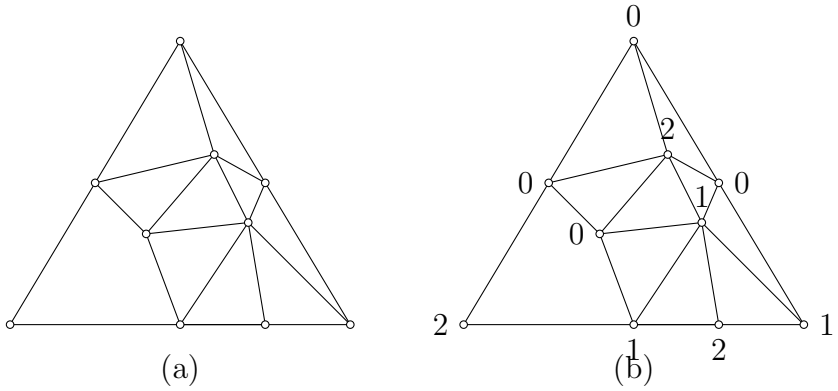


Figure 5: (a) A simplicial subdivision of a triangle; (b) a proper labelling of the subdivision

2 Trees

2.1 Trees

Definition 2.1 Tree .

A *tree* is a connected graph containing no cycles. Trees are the fundamental acyclic connected structures in graph theory and play a central role in both theory and applications.

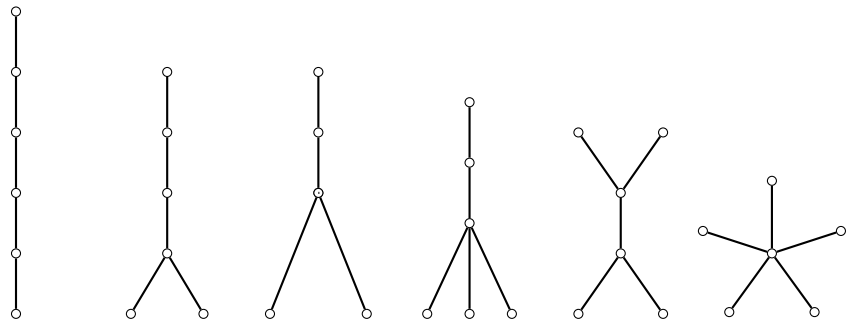


Figure 6: The trees on six vertices

Theorem 2.1 Equivalent Characterizations of Trees .

Let G be a graph with n vertices. The following statements are equivalent:

1. G is a tree.
2. G is connected and has $n-1$ edges.
3. G is acyclic and has $n-1$ edges.
4. G is connected, and the removal of any edge disconnects G.
5. G is acyclic, and the addition of any edge creates exactly one cycle.
6. There is a unique path between every pair of vertices of G.

Corollary 2.2 Every nontrivial tree has at least two vertices of degree one.

Proof Let G be a nontrivial tree. Then:

$$d(v) \geq 1 \quad \text{for all} \quad v \in V$$

Also, by theorems 1.1 and 2.2, we have

$$\sum_{v \in V} d(v) = 2\varepsilon = 2\nu - 2$$

It now follows that $d(v) = 1$ for at least two vertices v.

2.2 Cut Edges and Bonds

Definition 2.2 Cut Edge (Bridge) .

An edge e of a graph G is a *cut edge* if $G-e$ has more connected components than G.

Theorem 2.2 .

An edge of a graph is a cut edge if and only if it lies on no cycle.

Corollary 2.1 .

Every edge of a tree is a cut edge.

Definition 2.3 Bond .

A *bond* is a minimal nonempty set of edges whose deletion increases the number of connected components of the graph.

2.3 Cut Vertices

Definition 2.4 Cut Vertex (Articulation Point) .

A vertex v of a graph G is a *cut vertex* if $G-v$ has more connected components than G.

Theorem 2.3 .

A vertex v is a cut vertex if and only if there exist vertices $x, y \neq v$ such that every x–y path passes through v.
In a path P_n with $n \geq 3$, all internal vertices are cut vertices, while endpoints are not.

2.4 Cayley's Formula

Theorem 2.4 Cayley's Formula .

The number of distinct labeled trees on n vertices is

$$n^{n-2}.$$

For $n = 3$, there are $3^1 = 3$ labeled trees. For $n = 4$, there are $4^2 = 16$ labeled trees.

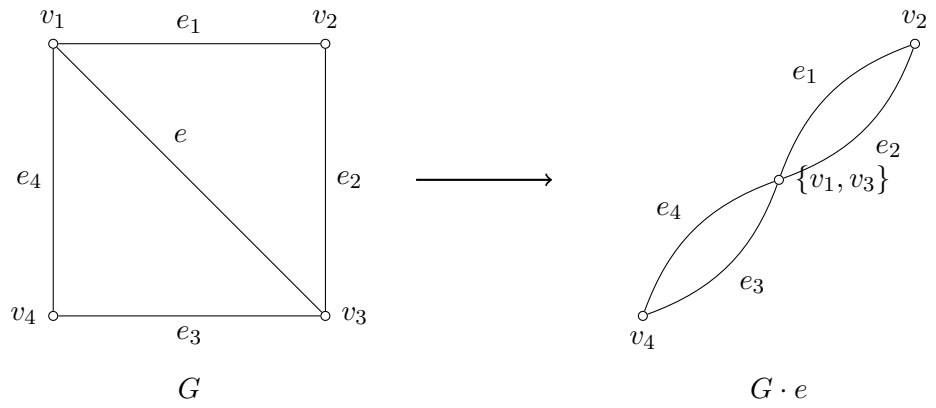


Figure 7: Contraction of an edge

Proof Idea (Prüfer Codes)

Each labeled tree on n vertices corresponds uniquely to a Prüfer sequence of length $n-2$, with entries in $\{1, \ldots, n\}$. This gives a bijection between labeled trees and sequences of length $n-2$ over n elements.

2.5 Applications

2.5.1 Spanning Trees

Definition 2.5 Spanning Tree .

A *spanning tree* of a connected graph G is a spanning subgraph of G that is a tree.

Theorem 2.5 .

Every connected graph has at least one spanning tree.
Removing one edge from each cycle of a connected graph produces a spanning tree.

2.5.2 Minimum Spanning Trees

Problem 2.1 Minimum Spanning Tree .

Given a connected graph with edge weights, find a spanning tree of minimum total weight.

Algorithm 2.1 Kruskal's Algorithm .

The steps are as follows:

1. Sort edges in nondecreasing order of weight.
2. Add edges one by one, skipping those that create a cycle.
3. Stop when $n - 1$ edges have been selected.

Algorithm 2.2 Prim's Algorithm .

The steps are as follows:

1. Start with a single vertex.
2. Repeatedly add the minimum-weight edge joining the tree to a new vertex.

Theorem 2.6 .

Both Kruskal's and Prim's algorithms produce a minimum spanning tree.

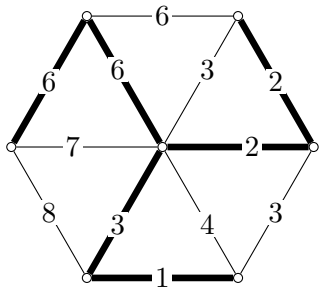


Figure 8: An optimal tree in a weighted graph

2.6 The Connector Problem

Problem 2.2 Connector Problem .

Given a connected graph with weighted edges, find a minimum-weight set of edges that keeps the graph connected.

Theorem 2.7 .

The solution to the connector problem is a minimum spanning tree.
Designing a minimum-cost communication or road network that connects all locations without redundancy is an instance of the connector problem.

2.7 Structural Properties of Trees

Theorem 2.8 .

Every tree with at least two vertices has at least two vertices of degree 1.

Proof Sketch. Assume all vertices have degree at least 2. Then the sum of degrees is at least $2n$, contradicting the handshaking lemma for a tree with $n - 1$ edges.

Corollary 2.2 .

Every tree can be constructed by successively attaching leaves.

2.8 Summary of Chapter 2

- Trees are connected acyclic graphs.
- They have exactly $n - 1$ edges.
- Every edge of a tree is a cut edge.
- Cayley's formula counts labeled trees.
- Spanning trees connect general graphs to tree theory.
- Minimum spanning trees solve optimization problems on networks.

3 Connectivity

3.1 Definitions of Connectivity

Definition 3.1 Vertex Cut .

Let $G = (V, E)$ be a graph. A subset $S \subset V$ is a *vertex cut* (or separating set) if the induced subgraph $G - S$ is disconnected or contains only a single vertex.

Definition 3.2 Vertex-Connectivity .

The *vertex-connectivity* of a graph G, denoted by $\kappa(G)$, is the minimum number of vertices whose removal disconnects G or leaves a trivial graph.

- For a non-complete graph, $\kappa(G)$ is the size of the smallest vertex cut.
- For a complete graph K_n, we define $\kappa(K_n) = n - 1$.

Definition 3.3 Edge Cut .

A subset of edges $F \subseteq E(G)$ is an *edge cut* if $G - F$ has more connected components than G.

Definition 3.4 Edge-Connectivity .

The *edge-connectivity* of G, denoted by $\lambda(G)$, is the minimum size of an edge cut. If G is disconnected or has only one vertex, $\lambda(G) = 0$.

Definition 3.5 k-Connectedness .

A graph G is said to be:

- **k-connected** if $\kappa(G) \geq k$. This implies the graph remains connected after removing any $k - 1$ vertices.
- **k-edge-connected** if $\lambda(G) \geq k$. This implies the graph remains connected after removing any $k - 1$ edges.

3.2 Fundamental Inequalities

Theorem 3.1 Whitney, 1932 .

For any nontrivial graph G, the following relationship holds:

$$\kappa(G) \leq \lambda(G) \leq \delta(G)$$

where $\delta(G)$ represents the minimum degree of the vertices in G.

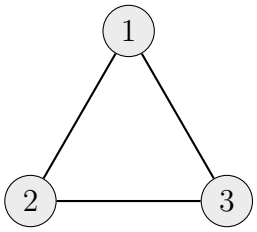
Proof Sketch. 1. To show $\lambda(G) \leq \delta(G)$: Let v be a vertex with $deg(v) = \delta(G)$. Deleting all edges incident to v isolates v, thus disconnecting the graph using exactly $\delta(G)$ edges. 2. To show $\kappa(G) \leq \lambda(G)$: Removing one endpoint from each edge in a minimum edge cut creates a vertex set that disconnects the graph (though technical care is needed for cases where edges share endpoints). □

3.2.1 The Cycle Graph C_n

For any cycle C_n where $n \geq 3$:

- $\delta(C_n) = 2$ (each vertex has two neighbors).
- $\lambda(C_n) = 2$ (cutting one edge leaves a path; cutting two edges disconnects it).
- $\kappa(C_n) = 2$ (removing one vertex leaves a path; removing two disconnects it).

$$\kappa(C_n) = \lambda(C_n) = \delta(C_n) = 2$$



3.3 Vertex Cuts and Separating Sets

Definition 3.6 Separating Set .

While any set S that disconnects G is a vertex cut, a *separating set* is specifically a vertex cut of minimum cardinality $\kappa(G)$.

Theorem 3.2 .

A connected graph G with at least three vertices is 2-connected if and only if it contains no *cut vertices* (also known as articulation points).

Path P_n vs. Cycle C_n
In a path P_n ($n \geq 3$), any "inner" vertex is a cut vertex because its removal splits the path into two or more components. Consequently, $\kappa(P_n) = 1$, making it 1-connected but not 2-connected.

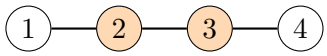


Figure 9: P_4: Orange nodes are cut-vertices

3.4 Menger's Theorem (Vertex Version)

Menger's Theorem is a cornerstone of network reliability. It establishes a "min-max" relationship: the difficulty of disconnecting two points is exactly equal to the number of independent "highways" between them.

Definition 3.7 Internally Disjoint Paths .

Two x, y-paths are *internally disjoint* if they do not share any vertices other than the start (x) and the end (y). These represent parallel routes that do not rely on the same intermediate "hubs."

Theorem 3.3 Menger's Theorem — Vertex Form .

Let x and y be distinct nonadjacent vertices of G. The maximum number of pairwise internally disjoint x, y-paths equals the minimum number of vertices whose removal separates x from y.

Corollary 3.1 .

A graph G is k-connected if and only if every pair of vertices has at least k internally disjoint paths between them.

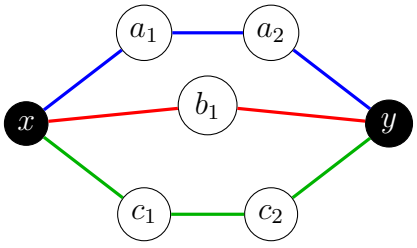


Figure 10: Three internally disjoint paths $\implies$ 3-connected locally

3.5 Blocks and Decomposition

Definition 3.8 Block .

A *block* (or biconnected component) of a graph G is a maximal subgraph such that any two vertices in the block can be connected by at least two internally disjoint paths.

In practical terms, a block is either:

- A bridge (a single edge that, if removed, disconnects the graph).
- A maximal 2-connected subgraph (like a cycle or a complex cluster).

Theorem 3.4 Block Decomposition .

Any connected graph G is the union of its blocks. While edges belong to exactly one block, vertices can belong to multiple blocks. A vertex belongs to more than one block if and only if it is a *cut vertex*.

Tree Decomposition
In a tree, every single edge is its own block. The cut vertices are all nodes with a degree > 1.

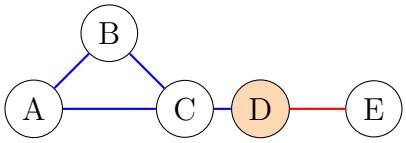


Figure 11: Vertex D is the "hinge" between two blocks.

3.6 The Block-Cut Tree

Definition 3.9 Block-Cut Tree .

The structure of how blocks are connected can be represented by a *Block-Cut Tree* (or Block Graph). This is a bipartite graph where:

- One set of nodes represents the **Blocks** $(B_1, B_2, \dots)$.
- The other set represents the **Cut Vertices** $(v_1, v_2, \dots)$.
- An edge exists between a block node and a cut vertex node if the vertex belongs to that block in the original graph.

3.7 Applications: Network Reliability

In network design, connectivity is a proxy for **fault tolerance**.

- **Vertex-Connectivity (κ):** Measures resistance to node failure (e.g., a router crashing).
- **Edge-Connectivity (λ):** Measures resistance to link failure (e.g., a fiber cable being cut).

3.7.1 Comparison of Topologies

Consider a network with n nodes:

Topology	$\kappa(G)$	$\lambda(G)$	**Reliability**	**Cost (Edges)**
Path (P_n)	1	1	Very Low	$n-1$
Cycle (C_n)	2	2	Medium	n
Complete (K_n)	$n-1$	$n-1$	Extremely High	$n(n-1)/2$

3.8 Harary Graphs and Minimum Edges

We denote by $f(m, n)$ the minimum number of edges that an m-connected graph on n vertices can have (where $m < n$). According to the relationship between connectivity and minimum degree ($\kappa(G) \leq \delta(G)$) and the Handshaking Lemma, we have:

$$f(m, n) \geq \left\lceil \frac{mn}{2} \right\rceil$$

Equality holds for all $m < n$. We demonstrate this by constructing the **Harary Graph** $H_{m,n}$, which is m-connected and has exactly $\lceil mn/2 \rceil$ edges. The construction depends on the parities of m and n.

3.8.1 Case 1: m is even

Let $m = 2r$. The graph $H_{2r,n}$ is constructed by placing n vertices in a circle, labeled $0, 1, \ldots, n-1$. Two vertices i and j are adjacent if:

$$|i - j| \leq r \pmod{n}$$

Essentially, each vertex is connected to its r nearest neighbors on both sides.

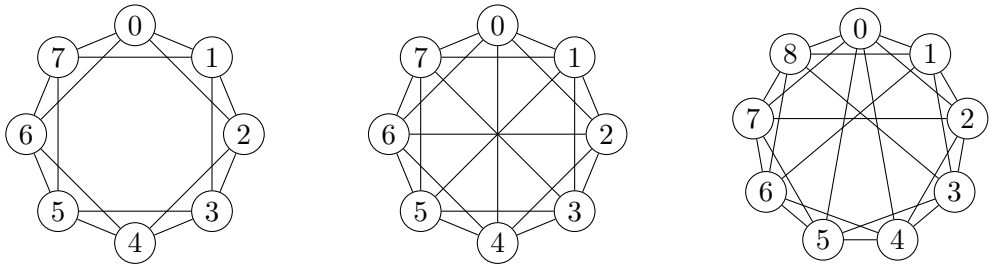
3.8.2 Case 2: m is odd, n is even

Let $m = 2r + 1$. We start with the graph $H_{2r,n}$ from Case 1. We then add "diametric" edges by joining each vertex i to its opposite vertex $i + n/2$ for all $0 \leq i < n/2$.

3.8.3 Case 3: m is odd, n is odd

Let $m = 2r + 1$. We start with $H_{2r,n}$ and add edges joining:

- Vertex 0 to vertices $(n-1)/2$ and $(n+1)/2$.
- Vertex i to vertex $i + (n+1)/2$ for $1 \leq i < (n-1)/2$.



(a) (b) (c)

Figure 12: (a) $H_{4,8}$; (b) $H_{5,8}$; (c) $H_{5,9}$

3.9 Whitney's Theorem and Ear Decompositions

Whitney's Theorem provides a structural characterization of 2-connected graphs, showing they can be built up from a simple cycle by adding "ears."

Definition 3.10 Ear .

An *ear* of a graph H is a path P whose endpoints belong to H, but whose internal vertices (and edges) do not.

- If the endpoints of the path are distinct, it is an **open ear**.
- If the endpoints are the same vertex, it is a **closed ear** (a cycle attached at one point).

Theorem 3.5 Whitney, 1932 .

A graph G is 2-connected if and only if it has an *ear decomposition*. That is, there exists a sequence of subgraphs $G_0, G_1, \ldots, G_k = G$ such that:

1. G_0 is a cycle.
2. G_{i+1} is obtained by adding an open ear to G_i for all $0 \leq i < k$.

Building a 2-connected Graph
Start with a cycle C_4. By adding a path (ear) between two non-adjacent vertices, we create a graph that remains 2-connected. If we added a "closed ear" (a loop at a single vertex), the graph would only be 1-connected because that single vertex would become a cut-vertex.

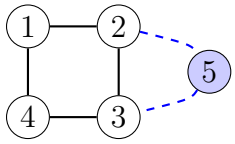


Figure 13: Cycle G_0 with an added dashed blue **ear** (G_1)

3.10 Intuition behind the Theorem

The presence of an ear ensures that there are always at least two internally disjoint paths between any two vertices.

- In the base cycle, there are two paths between any two nodes.
- Adding an *open* ear creates new paths without creating a "bottleneck" (cut-vertex).
- A **cut-vertex** only appears if we attach a piece of the graph to only one point; the ear decomposition prevents this by requiring two distinct attachment points.

3.11 Summary of Chapter 3

- Vertex- and edge-connectivity measure robustness of graphs.
- Connectivity is bounded by minimum degree.
- Menger's Theorem links connectivity to disjoint paths.
- Blocks describe the 2-connected structure of graphs.
- Block graphs form trees and provide a global decomposition.
- Connectivity theory underpins network reliability.

4 Euler Tours and Hamilton Cycles

4.1 Walks, Trails, and Tours

To understand Eulerian properties, we must distinguish between different ways of "moving" through a graph.

Definition 4.1 Trail and Closed Trail .

A *trail* is a walk in which all edges are distinct (no edge is repeated). A *closed trail* is a trail that starts and ends at the same vertex.

Definition 4.2 Euler Trail and Euler Tour .

Let G be a graph.

- An **Euler trail** is a trail that traverses every edge of G exactly once.
- An **Euler tour** (or *Euler circuit*) is a closed trail that traverses every edge of G exactly once.

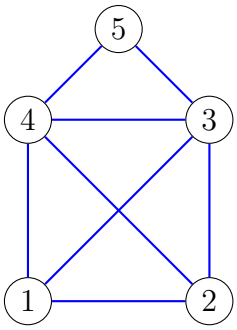


Figure 14: An Euler Tour: Every vertex has degree 2 or 4.

4.2 Characterization of Eulerian Graphs

Definition 4.3 Eulerian Graph .

A connected graph is **Eulerian** if it contains an Euler tour.

Theorem 4.1 Euler's Theorem, 1736 .

A connected graph G is Eulerian if and only if every vertex of G has an **even degree**.

Intuition. Whenever the tour enters a vertex via one edge, it must leave that vertex via a different, unused edge. Therefore, edges must come in pairs (one "in" and one "out"). This applies to every vertex, including the start/end vertex because the tour is closed. □

Theorem 4.2 Euler Trail Condition .

A connected graph has an **Euler trail** (but no tour) if and only if it has **exactly two** vertices of odd degree. These two vertices will be the start and end points of the trail.

Comparison of Common Graphs

- **Cycles (C_n):** Always Eulerian since every vertex has degree 2.
- **Complete Graphs (K_n):** K_n is Eulerian if and only if n is odd (so that each vertex has degree $n-1$, which is even).
- **Bipartite Graphs ($K_{r,s}$):** Eulerian if and only if both r and s are even.

4.3 The Seven Bridges of Königsberg

The study of graph theory began with this problem. The city had seven bridges. Representing landmasses as vertices and bridges as edges, the resulting multigraph had four vertices with odd degrees ($3, 3, 3$, and 5). Since more than two vertices had odd degrees, Euler proved it was impossible to cross every bridge exactly once and return to the start.

4.4 Algorithm for Finding Euler Tours

While Fleury's Algorithm is another option, **Hierholzer's Algorithm** is generally preferred for its efficiency ($O(E)$ time complexity).

Algorithm 4.1 Hierholzer's Algorithm .

Given a connected graph G where all vertices have even degree:

1. **Initial Circuit:** Start at any vertex v. Follow unused edges until you return to v. Since all degrees are even, you are guaranteed never to get "stuck" at a vertex other than the start. Label this circuit C.
2. **Find Sub-circuits:** If there are edges in G not in C, pick a vertex u belonging to C that is incident to an unused edge. Start a new trail from u, again following unused edges until returning to u.
3. **Splicing:** Integrate the new sub-circuit into the original circuit C at vertex u.
4. **Repeat:** Continue until all edges are included in the tour.

Algorithm 4.2 Fleury's Algorithm .

This algorithm builds a single path by following one rule: **Never cross a bridge unless you have no other choice.**

1. Start at any vertex v.
2. At each step, choose an unused edge incident to the current vertex such that its removal does not disconnect the graph of remaining edges (i.e., do not pick a bridge of the remaining subgraph).
3. If only a bridge is available, you must take it.
4. Terminate when all edges have been traversed.

4.5 Comparison of Algorithms

While both algorithms are mathematically sound, they serve different purposes:

Feature	Hierholzer's	Fleury's
Strategy	Splicing cycles together	Extending a single trail
Complexity	$O(E)$ (Linear)	$O(E^2)$ (due to bridge testing)
Use Case	Computer implementations	Manual/human solving
Philosophy	Modular construction	Greedy with foresight

Applying Fleury's Rule
Imagine a graph shaped like a figure-8 (two triangles joined at a single vertex v). If you start at v and traverse one triangle, the last edge you use to return to v becomes a bridge to the second triangle. Fleury's rule ensures you finish the first triangle completely before "crossing over" to the second.

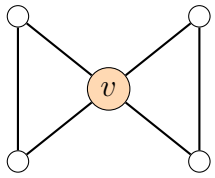


Figure 15: In a Figure-8, v is the critical "pivot" for Fleury's Rule.

4.6 Applications: The Chinese Postman Problem (CPP)

The **Route Inspection Problem**, popularly known as the Chinese Postman Problem (coined by Jack Edmonds in honor of Mei-Ko Kwan), asks for the shortest path that covers every edge.

Problem 4.1 The CPP Challenge .

If a graph is Eulerian, any Euler tour is an optimal solution. However, if the graph has $2k$ vertices of odd degree, some edges *must* be traversed more than once. The goal is to choose these "duplicated" edges such that their total weight is minimized.

Theorem 4.3 Optimal Strategy .

To solve the CPP on a non-Eulerian graph:

1. Identify all vertices with **odd degree**. (There will always be an even number of them).
2. Find a **minimum weight matching** between these odd-degree vertices. This involves finding paths between pairs of odd vertices such that the sum of the path weights is minimized.
3. "Duplicate" the edges along these shortest paths. This effectively makes every vertex have an even degree.
4. Find an Euler tour in this new augmented graph.

4.6.1 Postman on a Non-Eulerian Grid

Imagine a rectangular grid where only the four corner vertices have degree 2 and the rest have degree 3 or 4.

- **Odd Vertices:** Identify the "dead ends" or intersections with 3 streets.
- **Edge Duplication:** A postman might have to walk down "Main Street" twice to return to the depot after visiting a dead-end alley.

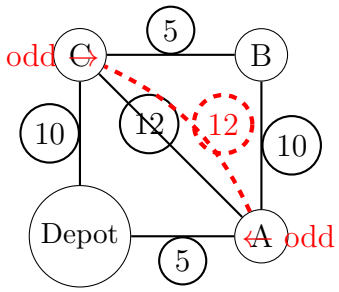


Figure 16: In the example above, the postman duplicates the edge AC (weight 12) to make the graph Eulerian.

4.7 Hamilton Paths and Cycles

While Eulerian properties focus on traversing every **edge**, Hamiltonian properties focus on visiting every **vertex**. Interestingly, while checking if a graph is Eulerian is easy ($O(E)$), checking if a graph is Hamiltonian is an NP-complete problem.

Definition 4.4 Hamilton Path and Hamilton Cycle .

Let G be a graph.

- A **Hamilton path** is a path that visits every vertex in G exactly once.
- A **Hamilton cycle** is a cycle that visits every vertex in G exactly once.

Definition 4.5 Hamiltonian Graph .

A graph is **Hamiltonian** if it contains at least one Hamilton cycle.

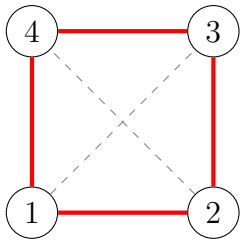


Figure 17: Red cycle visits all vertices exactly once.

4.8 Sufficient Conditions for Hamiltonicity

Since finding a Hamilton cycle is hard, we rely on **sufficient conditions**—rules that say "if the graph is dense enough, it must be Hamiltonian." Note that these are not *necessary*; a graph can be Hamiltonian without meeting these criteria.

Theorem 4.4 Dirac's Theorem, 1952 .

Let G be a simple graph with $n \geq 3$ vertices. If every vertex has a degree of at least $n/2$:

$$\delta(G) \geq \frac{n}{2}$$

then G is Hamiltonian.

Theorem 4.5 Ore's Theorem, 1960 .

Let G be a simple graph with $n \geq 3$ vertices. If for every pair of non-adjacent vertices u and v:

$$\deg(u) + \deg(v) \geq n$$

then G is Hamiltonian. (Note: Dirac's Theorem is a special case of Ore's).

Hamiltonian vs. Non-Hamiltonian

- **Complete Graphs (K_n):** Always Hamiltonian for $n \geq 3$.
- **Cycles (C_n):** Always Hamiltonian by definition.
- **Petersen Graph:** This is a famous example of a graph that is **not** Hamiltonian, even though it is highly symmetrical and 3-regular.
- **Bipartite Graphs ($K_{m,n}$):** Hamiltonian only if $m = n$. If the sets are unequal, you will eventually get "stuck" in the larger set.

4.8.1 Famous Examples: Hamiltonian and Non-Hamiltonian Graphs

The Petersen graph is perhaps the most famous graph in this context. It is 3-regular, 3-connected, and has 10 vertices, yet it is **not Hamiltonian**. It does, however, contain a Hamilton path.

The dodecahedron graph represents the edges and vertices of a dodecahedron. Unlike the Petersen graph, the dodecahedron **is Hamiltonian**.

4.9 Summary: Euler vs. Hamilton

Feature	**Eulerian**	**Hamiltonian**
Requirement	Visit every **edge** once	Visit every **vertex** once
Key Criteria	Even degrees	Graph density (Dirac/Ore)
Complexity	P (Easy to solve)	NP-Complete (Hard to solve)

4.10 Closure of a Graph

The concept of a closure allows us to simplify a graph while preserving its "Hamiltonian nature." It essentially fills in the "obvious" edges that don't change whether a Hamilton cycle exists.

Definition 4.6 Closure .

The *closure* of a graph G, denoted $\text{cl}(G)$, is the graph obtained from G by recursively joining pairs of non-adjacent vertices u, v whose degree sum satisfies:

$$d(u) + d(v) \geq n$$

where $n = |V(G)|$. This process continues until no such pair of vertices remains.

Theorem 4.6 Bondy–Chvátal Theorem, 1976 .

A graph G is Hamiltonian if and only if its closure $\text{cl}(G)$ is Hamiltonian.

Corollary 4.1 .

If the closure of a graph G is a complete graph K_n, then G is Hamiltonian. This provides a much stronger condition than Ore's Theorem.

4.11 The Travelling Salesman Problem (TSP)

While the Hamilton Cycle problem asks "Can we visit every city?", the **Travelling Salesman Problem** asks "What is the cheapest way to visit every city and return home?"

Problem 4.2 Travelling Salesman Problem .

Given a complete graph K_n where each edge e has a weight $w(e)$, find a Hamilton cycle C such that the total weight $\sum_{e \in C} w(e)$ is minimized.

Unlike the Chinese Postman Problem (which is P), the TSP is **NP-hard**. As n increases, the number of possible cycles $(n-1)!/2$ grows factorially, making brute-force search impossible for large networks.

4.11.1 Metric TSP and the Triangle Inequality

In many real-world applications, the weights satisfy the **Triangle Inequality**:

$$w(u, v) \leq w(u, x) + w(x, v)$$

This means a direct flight is never more expensive than a layover. For "Metric TSP," we can find approximate solutions (like the Double Tree Algorithm) that are guaranteed to be within a certain factor of the true optimum.

Real-World Modeling

- **Logistics:** Minimizing fuel consumption for delivery trucks.
- **Manufacturing:** A robotic drill must visit 1,000 hole locations on a circuit board in the shortest time possible.
- **Genetics:** Reconstructing DNA sequences by finding the shortest overlap between fragments.

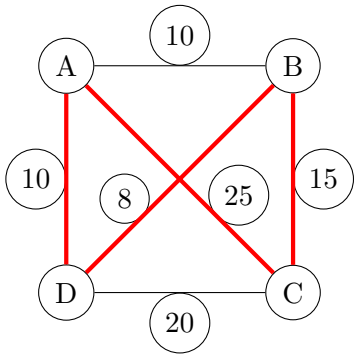


Figure 18: Optimal Route: $10 + 8 + 15 + 25 = 58$

4.11.2 Case Studies: The Petersen, Dodecahedron, and Herschel Graphs

The Petersen Graph
The Petersen graph is a 3-regular (cubic) graph with 10 vertices and 15 edges. It is famously **non-Hamiltonian**, making it the most prominent counterexample to many early conjectures about symmetrical, highly-connected graphs. Despite having no Hamilton cycle, it is 3-connected and contains a Hamilton path between any two vertices.

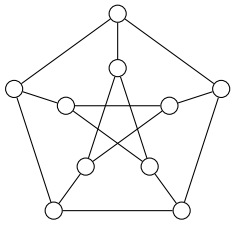


Figure 19: The Petersen Graph: Symmetrical but non-Hamiltonian.

Dodecahedron and Herschel Graphs
While both are 3-connected and planar, these two graphs differ fundamentally in their Hamiltonicity. The Dodecahedron is the classic example of a Hamiltonian polyhedral graph, while the Herschel graph is the smallest polyhedral graph that is non-Hamiltonian.

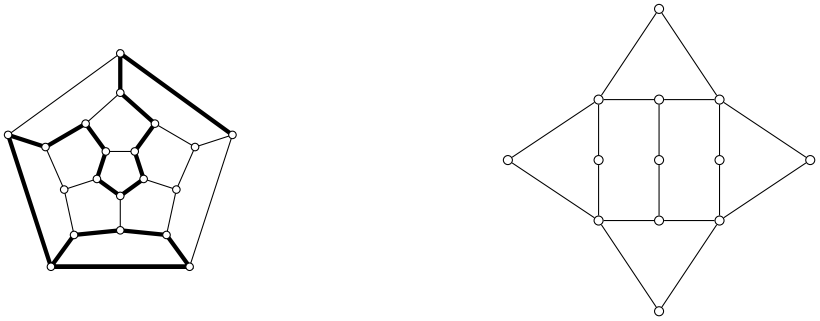


Figure 20: The Dodecahedron Hamiltonian (20 vertices) - The Herschel Graph Non-Hamiltonian (11 vertices)