Chapter 1

Fundamental Concepts

1.1. What Is a Graph?

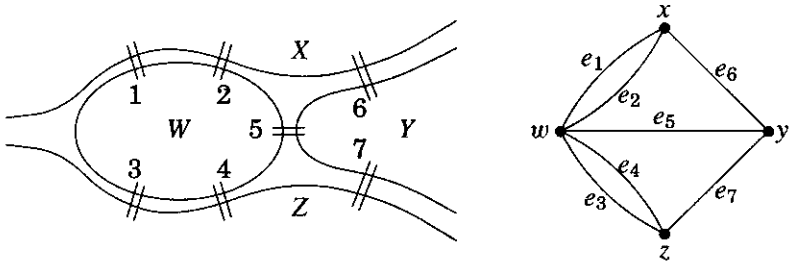
How can we lay cable at minimum cost to make every telephone reachable from every other? What is the fastest route from the national capital to each state capital? How can n jobs be filled by n people with maximum total utility? What is the maximum flow per unit time from source to sink in a network of pipes? How many layers does a computer chip need so that wires in the same layer don't cross? How can the season of a sports league be scheduled into the minimum number of weeks? In what order should a traveling salesman visit cities to minimize travel time? Can we color the regions of every map using four colors so that neighboring regions receive different colors?

These and many other practical problems involve graph theory. In this book, we develop the theory of graphs and apply it to such problems. Our starting point assumes the mathematical background in Appendix A, where basic objects and language of mathematics are discussed.

THE DEFINITION

The problem that is often said to have been the birth of graph theory will suggest our basic definition of a graph.

1.1.1. Example. *The Königsberg Bridge Problem.* The city of Königsberg was located on the Pregel river in Prussia. The city occupied two islands plus areas on both banks. These regions were linked by seven bridges as shown on the left below. The citizens wondered whether they could leave home, cross every bridge exactly once, and return home. The problem reduces to traversing the figure on the right, with heavy dots representing land masses and curves representing bridges.



The model on the right makes it easy to argue that the desired traversal does not exist. Each time we enter and leave a land mass, we use two bridges ending at it. We can also pair the first bridge with the last bridge on the land mass where we begin and end. Thus existence of the desired traversal requires that each land mass be involved in an even number of bridges. This necessary condition did not hold in Königsberg. ■

The Königsberg Bridge Problem becomes more interesting when we show in Section 1.2 which configurations have traversals. Meanwhile, the problem suggests a general model for discussing such questions.

1.1.2. Definition. A **graph** G is a triple consisting of a **vertex set** $V(G)$, an **edge set** $E(G)$, and a relation that associates with each edge two vertices (not necessarily distinct) called its **endpoints**.

We **draw** a graph on paper by placing each vertex at a point and representing each edge by a curve joining the locations of its endpoints.

1.1.3. Example. In the graph in Example 1.1.1, the vertex set is $\{x, y, z, w\}$, the edge set is $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, and the assignment of endpoints to edges can be read from the picture.

Note that edges e_1 and e_2 have the same endpoints, as do e_3 and e_4. Also, if we had a bridge over an inlet, then its ends would be in the same land mass, and we would draw it as a curve with both ends at the same point. We have appropriate terms for these types of edges in graphs. ■

1.1.4. Definition. A **loop** is an edge whose endpoints are equal. **Multiple edges** are edges having the same pair of endpoints.

A **simple graph** is a graph having no loops or multiple edges. We specify a simple graph by its vertex set and edge set, treating the edge set as a set of unordered pairs of vertices and writing $e = uv$ (or $e = vu$) for an edge e with endpoints u and v.

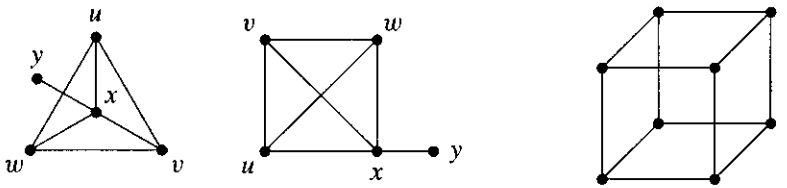
When u and v are the endpoints of an edge, they are **adjacent** and are **neighbors**. We write $u \leftrightarrow v$ for "u is adjacent to v".

In many important applications, loops and multiple edges do not arise, and we restrict our attention to simple graphs. In this case an edge is determined by

its endpoints, so we can *name* the edge by its endpoints, as stated in Definition 1.1.4. Thus in a *simple* graph we view an edge as an unordered pair of vertices and can ignore the formality of the relation associating endpoints to edges. This book emphasizes simple graphs.

1.1.5. Example. On the left below are two drawings of a simple graph. The vertex set is $\{u, v, w, x, y\}$, and the edge set is $\{uv, uw, ux, vx, vw, xw, xy\}$.

The terms "vertex" and "edge" arise from solid geometry. A cube has vertices and edges, and these form the vertex set and edge set of a graph. It is drawn on the right below, omitting the names of vertices and edges. ■



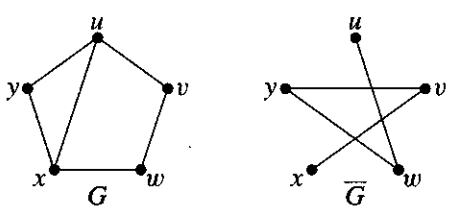
A graph is **finite** if its vertex set and edge set are finite. We adopt the convention that **every graph mentioned in this book is finite**, unless explicitly constructed otherwise.

1.1.6.* Remark. The **null graph** is the graph whose vertex set and edge set are empty. Extending general theorems to the null graph introduces unnecessary distractions, so we ignore it. All statements and exercises should be considered only for graphs with a nonempty set of vertices. ■

GRAPHS AS MODELS

Graphs arise in many settings. The applications suggest useful concepts and terminology about the structure of graphs.

1.1.7. Example. *Acquaintance relations and subgraphs.* Does every set of six people contain three mutual acquaintances or three mutual strangers? Since "acquaintance" is symmetric, we model it using a simple graph with a vertex for each person and an edge for each acquainted pair. The "nonacquaintance" relation on the same set yields another graph with the "complementary" set of edges. We introduce terms for these concepts. ■



1.1.8. Definition. The **complement** $\overline{G}$ of a simple graph G is the simple graph with vertex set $V(G)$ defined by $uv \in E(\overline{G})$ if and only if $uv \notin E(G)$. A **clique** in a graph is a set of pairwise adjacent vertices. An **independent set** (or **stable set**) in a graph is a set of pairwise nonadjacent vertices.

In the graph G of Example 1.1.7, $\{u, x, y\}$ is a clique of size 3 and $\{u, w\}$ is an independent set of size 2, and these are the largest such sets. These values reverse in the complement $\overline{G}$, since cliques become independent sets (and vice versa) under complementation. The question in Example 1.1.7 asks whether it is true that every 6-vertex graph has a clique of size 3 or an independent set of size 3 (Exercise 29). Deleting edge ux from G yields a 5-vertex graph having no clique or independent set of size 3.

1.1.9. Example. *Job assignments and bipartite graphs.* We have m jobs and n people, but not all people are qualified for all jobs. Can we fill the jobs with qualified people? We model this using a simple graph H with vertices for the jobs and people; job j is adjacent to person p if p can do j.

Each job is to be filled by exactly one person, and each person can hold at most one of the jobs. Thus we seek m pairwise disjoint edges in H (viewing edges as pairs of vertices). Chapter 3 shows how to test for this; it can't be done in the graph below.

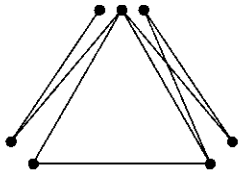
The use of graphs to model relations between two disjoint sets has many important applications. These are the graphs whose vertex sets can be partitioned into two independent sets; we need a name for them. ■

1.1.10. Definition. A graph G is **bipartite** if $V(G)$ is the union of two disjoint (possibly empty) independent sets called **partite sets** of G.

1.1.11. Example. *Scheduling and graph coloring.* Suppose we must schedule Senate committee meetings into designated weekly time periods. We cannot assign two committees to the same time if they have a common member. How many different time periods do we need?

We create a vertex for each committee, with two vertices adjacent when their committees have a common member. We must assign labels (time periods) to the vertices so the endpoints of each edge receive different labels. In the graph below, we can use one label for each of the three independent sets of vertices grouped closely together. The members of a clique must receive distinct labels, so in this example the minimum number of time periods is three.

Since we are only interested in partitioning the vertices, and the labels have no numerical value, it is convenient to call them **colors**. ■



1.1.12. Definition. The **chromatic number** of a graph G, written $\chi(G)$, is the minimum number of colors needed to label the vertices so that adjacent vertices receive different colors. A graph G is **k-partite** if $V(G)$ can be expressed as the union of k (possibly empty) independent sets.

This generalizes the idea of bipartite graphs, which are 2-partite. Vertices given the same color must form an independent set, so $\chi(G)$ is the minimum number of independent sets needed to partition $V(G)$. A graph is k-partite if and only if its chromatic number is at most k. We use the term "partite set" when referring to a set in a partition into independent sets.

We study chromatic number and graph colorings in Chapter 5. The most (in)famous problem in graph theory involves coloring of "maps".

1.1.13. Example. *Maps and coloring.* Roughly speaking, a **map** is a partition of the plane into connected regions. Can we color the regions of every map using at most four colors so that neighboring regions have different colors?

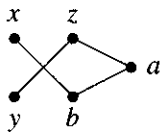
To relate map coloring to graph coloring, we introduce a vertex for each region and an edge for regions sharing a boundary. The map question asks whether the resulting graph must have chromatic number at most 4. The graph can be drawn in the plane without crossing edges; such graphs are **planar**. The graph before Definition 1.1.12 is planar; that drawing has a crossing, but another drawing has no crossings. We study planar graphs in Chapter 6. ■

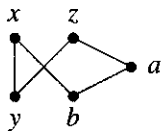
1.1.14. Example. *Routes in road networks.* We can model a road network using a graph with edges corresponding to road segments between intersections. We can assign edge weights to measure distance or travel time. In this context edges do represent physical links. How can we find the shortest route from x to y? We show how to compute this in Chapter 2.

If the vertices of the graph represent our house and other places to visit, then we may want to follow a route that visits every vertex exactly once, so as to visit everyone without overstaying our welcome. We consider the existence of such a route in Chapter 7.

We need terms to describe these two types of routes in graphs. ■

1.1.15. Definition. A **path** is a simple graph whose vertices can be ordered so that two vertices are adjacent if and only if they are consecutive in the list. A **cycle** is a graph with an equal number of vertices and edges whose vertices can be placed around a circle so that two vertices are adjacent if and only if they appear consecutively along the circle.





Above we show a path and a cycle, as demonstrated by listing the vertices in the order x, b, a, z, y. Dropping one edge from a cycle produces a path. In studying the routes in road networks, we think of paths and cycles *contained* in the graph. Also, we hope that every vertex in the network can be reached from every other. The next definition makes these concepts precise.

1.1.16. Definition. A **subgraph** of a graph G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ and the assignment of endpoints to edges in H is the same as in G. We then write $H \subseteq G$ and say that "G **contains** H".

A graph G is **connected** if each pair of vertices in G belongs to a path; otherwise, G is **disconnected**.

The graph before Definition 1.1.12 has three subgraphs that are cycles. It is a connected graph, but the graph in Example 1.1.9 is not.

MATRICES AND ISOMORPHISM

How do we specify a graph? We can list the vertices and edges (with endpoints), but there are other useful representations. Saying that a graph is **loopless** means that multiple edges are allowed but loops are not.

1.1.17. Definition. Let G be a loopless graph with vertex set $V(G) = \{v_1, \ldots, v_n\}$ and edge set $E(G) = \{e_1, \ldots, e_m\}$. The **adjacency matrix** of G, written $A(G)$, is the n-by-n matrix in which entry $a_{i,j}$ is the number of edges in G with endpoints $\{v_i, v_j\}$. The **incidence matrix** $M(G)$ is the n-by-m matrix in which entry $m_{i,j}$ is 1 if v_i is an endpoint of e_j and otherwise is 0.

If vertex v is an endpoint of edge e, then v and e are **incident**. The **degree** of vertex v (in a loopless graph) is the number of incident edges.

The appropriate way to define adjacency matrix, incidence matrix, or vertex degrees for graphs with loops depends on the application; Sections 1.2 and 1.3 discuss this.

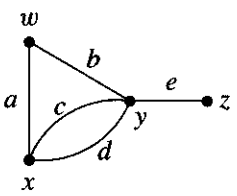
1.1.18. Remark. An adjacency matrix is determined by a vertex ordering. Every adjacency matrix is **symmetric** ($a_{i,j} = a_{j,i}$ for all i, j). An adjacency matrix of a simple graph G has entries 0 or 1, with 0s on the diagonal. The degree of v is the sum of the entries in the row for v in either $A(G)$ or $M(G)$. ■

1.1.19. Example. For the loopless graph G below, we exhibit the adjacency matrix and incidence matrix that result from the vertex ordering w, x, y, z and

the edge ordering a, b, c, d, e. The degree of y is 4, by viewing the graph or by summing the row for y in either matrix. ■

$$\begin{array}{c} \\ w \\ x \\ y \\ z \end{array}\!\!\begin{array}{c} \begin{array}{cccc} w & x & y & z \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$A(G)$



G

$$\begin{array}{c} \\ w \\ x \\ y \\ z \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

$M(G)$

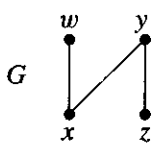
Presenting an adjacency matrix for a graph implicitly names the vertices by the order of the rows; the ith vertex corresponds to the ith row and column. Storing a graph in a computer requires naming the vertices.

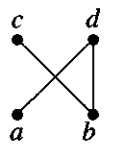
Nevertheless, we want to study properties (like connectedness) that do not depend on these names. Intuitively, the structural properties of G and H will be the same if we can rename the vertices of G using the vertices in H so that G will actually become H. We make the definition precise for simple graphs. The renaming is a function from $V(G)$ to $V(H)$ that assigns each element of $V(H)$ to one element of $V(G)$, thus pairing them up. Such a function is a *one-to-one correspondence* or **bijection** (see Appendix A). Saying that the renaming turns G into H is saying that the vertex bijection preserves the adjacency relation.

1.1.20. Definition. An **isomorphism** from a simple graph G to a simple graph H is a bijection $f\colon V(G) \to V(H)$ such that $uv \in E(G)$ if and only if $f(u)f(v) \in E(H)$. We say "G is **isomorphic to** H", written $G \cong H$, if there is an isomorphism from G to H.

1.1.21. Example. The graphs G and H drawn below are 4-vertex paths. Define the function $f\colon V(G) \to V(H)$ by $f(w) = a$, $f(x) = d$, $f(y) = b$, $f(z) = c$. To show that f is an isomorphism, we check that f preserves edges and non-edges. Note that rewriting $A(G)$ by placing the rows in the order w, y, z, x and the columns also in that order yields $A(H)$, as illustrated below; this verifies that f is an isomorphism.

Another isomorphism maps w, x, y, z to c, b, d, a, respectively. ■





$$\begin{array}{c} \\ w \\ x \\ y \\ z \end{array}\!\!\begin{array}{c} \begin{array}{cccc} w & x & y & z \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$\begin{array}{c} \\ w \\ y \\ z \\ x \end{array}\!\!\begin{array}{c} \begin{array}{cccc} w & y & z & x \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\!\!\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

1.1.22. Remark. *Finding isomorphisms.* As suggested in Example 1.1.21, presenting the adjacency matrices with vertices ordered so that the matrices are identical is one way to prove that two graphs are isomorphic. Applying a permutation σ to both the rows and the columns of $A(G)$ has the effect of reordering the vertices of G. If the new matrix equals $A(H)$, then the permutation yields an isomorphism. One can also verify preservation of the adjacency relation without writing out the matrices.

In order for an explicit vertex bijection to be an isomorphism from G to H, the image in H of a vertex v in G must behave in H as v does in G. For example, they must have the same degree. ∎

1.1.23.* Remark. *Isomorphism for non-simple graphs.* The definition of isomorphism extends to graphs with loops and multiple edges, but the precise statement needs the language of Definition 1.1.2.

> An **isomorphism** from G to H is a bijection f that maps $V(G)$ to $V(H)$ and $E(G)$ to $E(H)$ such each edge of G with endpoints u and v is mapped to an edge with endpoints $f(u)$ and $f(v)$.

This technicality will not concern us, because we will study isomorphism only in the context of simple graphs. ∎

Since H is isomorphic to G whenever G is isomorphic to H, we often say "G and H are isomorphic" (meaning to each other). The adjective "isomorphic" applies only to pairs of graphs; "G is isomorphic" by itself has no meaning (we respond, "isomorphic to what?"). Similarly, we may say that a set of graphs is "pairwise isomorphic" (taken two at a time), but it doesn't make sense to say "this set of graphs is isomorphic".

A **relation** on a set S is a collection of ordered pairs from S. An **equivalence relation** is a relation that is reflexive, symmetric, and transitive (see Appendix A). For example, the adjacency relation on the set of vertices of a graph is symmetric, but it is not reflexive and rarely is transitive. On the other hand, the **isomorphism relation**, consisting of the set of ordered pairs (G, H) such that G is isomorphic to H, does have all three properties.

1.1.24. Proposition. The isomorphism relation is an equivalence relation on the set of (simple) graphs.

Proof: *Reflexive property.* The identity permutation on $V(G)$ is an isomorphism from G to itself. Thus $G \cong G$.

Symmetric property. If $f\colon V(G) \to V(H)$ is an isomorphism from G to H, then f^{-1} is an isomorphism from H to G, because the statement "$uv \in E(G)$ if and only if $f(u)f(v) \in E(H)$" yields "$xy \in E(H)$ if and only if $f^{-1}(x)f^{-1}(y) \in E(H)$". Thus $G \cong H$ implies $H \cong G$.

Transitive property. Suppose that $f\colon V(F) \to V(G)$ and $g\colon V(G) \to V(H)$ are isomorphisms. We are given "$uv \in E(F)$ if and only if $f(u)f(v) \in E(G)$" and "$xy \in E(G)$ if and only if $g(x)g(y) \in E(H)$". Since f is an isomorphism, for every $xy \in E(G)$ we can find $uv \in E(F)$ such that $f(u) = x$ and $f(v) = y$. This

yields "$uv \in E(F)$ if and only if $g(f(u))g(f(v)) \in E(H)$". Thus the composition $g \circ f$ is an isomorphism from F to H. We have proved that $F \cong G$ and $G \cong H$ together imply $F \cong H$. ∎

An equivalence relation partitions a set into **equivalence classes**; two elements satisfy the relation if and only if they lie in the same class.

1.1.25. Definition. An **isomorphism class** of graphs is an equivalence class of graphs under the isomorphism relation.

Paths with n vertices are pairwise isomorphic; the set of all n-vertex paths forms an isomorphism class.

1.1.26. Remark. *"Unlabeled" graphs and isomorphism classes.* When discussing a graph G, we have a fixed vertex set, but our structural comments apply also to every graph isomorphic to G. Our conclusions are independent of the names (labels) of the vertices. Thus, we use the informal expression "unlabeled graph" to mean an isomorphism class of graphs.

When we draw a graph, its vertices are named by their physical locations, even if we give them no other names. Hence a drawing of a graph is a member of its isomorphism class, and we just call it a graph. When we redraw a graph to display some structural aspect, we have chosen a more convenient member of the isomorphism class, still discussing the same "unlabeled graph". ∎

When discussing structure of graphs, it is convenient to have names and notation for important isomorphism classes. We want the flexibility to refer to the isomorphism class or to any representative of it.

1.1.27. Definition. The (unlabeled) path and cycle with n vertices are denoted P_n and C_n, respectively; an **n-cycle** is a cycle with n vertices. A **complete graph** is a simple graph whose vertices are pairwise adjacent; the (unlabeled) complete graph with n vertices is denoted K_n. A **complete bipartite graph** or **biclique** is a simple bipartite graph such that two vertices are adjacent if and only if they are in different partite sets. When the sets have sizes r and s, the (unlabeled) biclique is denoted $K_{r,s}$.

1.1.28.* Remark. We have defined a *complete graph* as a graph whose vertices are pairwise adjacent, while a *clique* is a set of pairwise adjacent vertices in a graph. Many authors use the terms interchangeably, but the distinction allows us to discuss cliques in the same language as independent sets.

In the bipartite setting, we simply use "biclique" to abbreviate "complete bipartite graph". The alternative name "biclique" is a reminder that a complete bipartite graph is generally *not* a complete graph (Exercise 1). ■

1.1.29. Remark. *A graph by any other name . . .* When we name a graph without naming its vertices, we often mean its isomorphism class. Technically, "H is a subgraph of G" means that some subgraph of G is isomorphic to H (we say "G contains a **copy** of H"). Thus C_3 is a subgraph of K_5 (every complete graph with 5 vertices has 10 subgraphs isomorphic to C_3) but not of $K_{2,3}$.

Similarly, asking whether G "is" C_n means asking whether G is isomorphic to a cycle with n vertices. ■

The structural properties of a graph are determined by its adjacency relation and hence are preserved by isomorphism. We can prove that G and H are *not* isomorphic by finding some structural property in which they differ. If they have different number of edges, or different subgraphs, or different complements, etc., then they cannot be isomorphic.

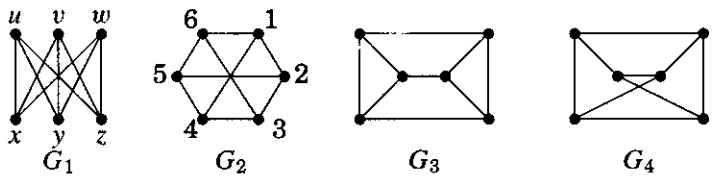
On the other hand, checking that a few structural properties are the same does not imply that $G \cong H$. To prove that $G \cong H$, we must present a bijection $f\colon V(G) \to V(H)$ that preserves the adjacency relation.

1.1.30. Example. *Isomorphic or not?* Each graph below has six vertices and nine edges and is connected, but these graphs are not pairwise isomorphic.

To prove that $G_1 \cong G_2$, we give names to the vertices, specify a bijection, and check that it preserves the adjacency relation. As labeled below, the bijection that sends u, v, w, x, y, z to 1, 3, 5, 2, 4, 6, respectively, is an isomorphism from G_1 to G_2. The map sending u, v, w, x, y, z to 6, 4, 2, 1, 3, 5, respectively, is another isomorphism.

Both G_1 and G_2 are bipartite; they are drawings of $K_{3,3}$ (as is G_4). The graph G_3 contains K_3, so its vertices cannot be partitioned into two independent sets. Thus G_3 is not isomorphic to the others.

Sometimes we can test isomorphism quickly using the complements. Simple graphs G and H are isomorphic if and only if their complements are isomorphic (Exercise 4). Here $\overline{G}_1, \overline{G}_2, \overline{G}_4$ all consist of two disjoint 3-cycles and are not connected, but $\overline{G}_3$ is a 6-cycle and is connected. ■



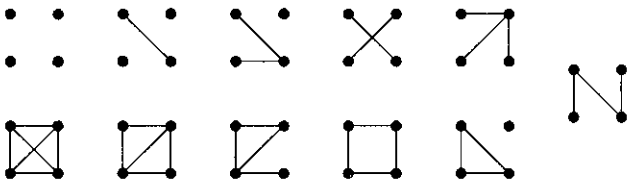
G_1 G_2 G_3 G_4

1.1.31. Example. *The number of n-vertex graphs.* When choosing two vertices from a set of size n, we can pick one and then the other but don't care about the order, so the number of ways is $n(n-1)/2$. (The notation for the number of ways

to choose k elements from n elements is $\binom{n}{k}$, read "n choose k". These numbers are called **binomial coefficients**; see Appendix A for further background.)

In a simple graph with a vertex set X of size n, each vertex pair may form an edge or may not. Making the choice for each pair specifies the graph, so the number of simple graphs with vertex set X is $2^{\binom{n}{2}}$.

For example, there are 64 simple graphs on a fixed set of four vertices. These graphs form only 11 isomorphism classes. The classes appear below in complementary pairs; only P_4 is isomorphic to its complement. Isomorphism classes have different sizes, so we cannot count the isomorphism classes of n-vertex simple graphs by dividing $2^{\binom{n}{2}}$ by the size of a class. ■



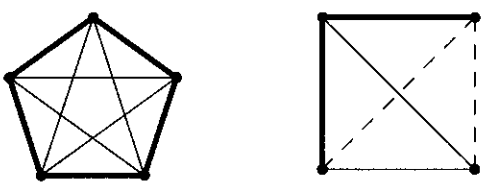
DECOMPOSITION AND SPECIAL GRAPHS

The example $P_4 \cong \overline{P}_4$ suggests a family of graph problems.

1.1.32. Definition. A graph is **self-complementary** if it is isomorphic to its complement. A **decomposition** of a graph is a list of subgraphs such that each edge appears in exactly one subgraph in the list.

An n-vertex graph H is self-complementary if and only if K_n has a decomposition consisting of two copies of H.

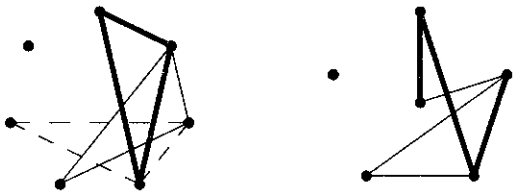
1.1.33. Example. We can decompose K_5 into two 5-cycles, and thus the 5-cycle is self-complementary. Any n-vertex graph and its complement decompose K_n. Also $K_{1,n-1}$ and K_{n-1} decompose K_n, even though one of these subgraphs omits a vertex. On the right below we show a decomposition of K_4 using three copies of P_3. Exercises 31–39 consider graph decompositions. ■



1.1.34.* Example. The question of which complete graphs decompose into copies of K_3 is a fundamental question in the theory of combinatorial designs.

On the left below we suggest such a decomposition for K_7. Rotating the triangle through seven positions uses each edge exactly once.

On the right we suggest a decomposition of K_6 into copies of P_4. Placing one vertex in the center groups the edges into three types: the outer 5-cycle, the inner (crossing) 5-cycle on those vertices, and the edges involving the central vertex. Each 4-vertex path in the decomposition uses one edge of each type; we rotate the picture to get the next path. ∎

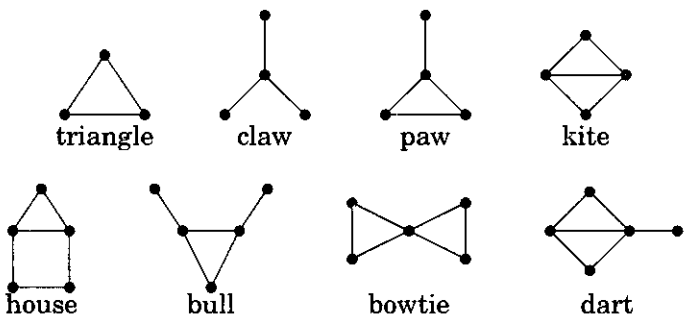


We referred to a copy of K_3 as a *triangle*. Short names for graphs that arise frequently in structural discussions can be convenient.

1.1.35. Example. *The Graph Menagerie.* A catchy "name" for a graph usually comes from some drawing of the graph. We also use such a name for all other drawings, and hence it is best viewed as a name for the isomorphism class. Below we give names to several graphs with at most five vertices.

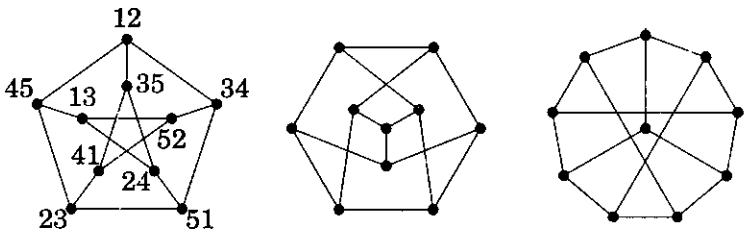
Among these the most important are the **triangle** (K_3) and the **claw** ($K_{1,3}$). We also sometimes discuss the **paw** ($K_{1,3}+e$)and the **kite** (K_4-e); the others arise less frequently.

The complements of the graphs in the first row are disconnected. The complement of the house is P_5, and the bull is self-complementary. Exercise 39 asks which of these graphs can be used to decompose K_6. ∎



In order to decompose H into copies of G, the number of edges of G must divide the number of edges of H. This is not sufficient, since K_5 does not decompose into two copies of the kite.

1.1.36. Definition. The **Petersen graph** is the simple graph whose vertices are the 2-element subsets of a 5-element set and whose edges are the pairs of disjoint 2-element subsets.



We have drawn the Petersen graph in three ways above. It is a useful example so often that an entire book was devoted to it (Holton–Sheehan [1993]). Its properties follow from the statement of its adjacency relation that we have used as the definition.

1.1.37. Example. *Structure of the Petersen graph.* Using $[5] = \{1, 2, 3, 4, 5\}$ as our 5-element set, we write the pair $\{a, b\}$ as ab or ba. Since 12 and 34 are disjoint, they are adjacent vertices when we form the graph, but 12 and 23 are not. For each 2-set ab, there are three ways to pick a 2-set from the remaining three elements of [5], so every vertex has degree 3.

The Petersen graph consists of two disjoint 5-cycles plus edges that pair up vertices on the two 5-cycles. The disjointness definition tells us that 12, 34, 51, 23, 45 in order are the vertices of a 5-cycle, and similarly this holds for the remaining vertices 13, 52, 41, 35, 24. Also 13 is adjacent to 45, and 52 is adjacent to 34, and so on, as shown on the left above.

We use this name even when we do not specify the vertex labeling; in essence, we use "Petersen graph" to name an isomorphism class. To show that the graphs above are pairwise isomorphic, it suffices to name the vertices of each using the 2-element subsets of [5] so that in each case the adjacency relation is disjointness (Exercise 24). ∎

1.1.38. Proposition. If two vertices are nonadjacent in the Petersen graph, then they have exactly one common neighbor.

Proof: Nonadjacent vertices are 2-sets sharing one element; their union S has size 3. A vertex adjacent to both is a 2-set disjoint from both. Since the 2-sets are chosen from $\{1, 2, 3, 4, 5\}$, there is exactly one 2-set disjoint from S. ∎

1.1.39. Definition. The **girth** of a graph with a cycle is the length of its shortest cycle. A graph with no cycle has infinite girth.

1.1.40. Corollary. The Petersen graph has girth 5.

Proof: The graph is simple, so it has no 1-cycle or 2-cycle. A 3-cycle would require three pairwise-disjoint 2-sets, which can't occur among 5 elements.

A 4-cycle in the absence of 3-cycles would require nonadjacent vertices with two common neighbors, which Proposition 1.1.38 forbids. Finally, the vertices 12, 34, 51, 23, 45 yield a 5-cycle, so the girth is 5. ■

The Petersen graph is highly symmetric. Every permutation of $\{1, 2, 3, 4, 5\}$ generates a permutation of the 2-subsets that preserves the disjointness relation. Thus there are at least $5! = 120$ isomorphisms from the Petersen graph to itself. Exercise 43 confirms that there are no others.

1.1.41.* Definition. An **automorphism** of G is an isomorphism from G to G. A graph G is **vertex-transitive** if for every pair $u, v \in V(G)$ there is an automorphism that maps u to v.

The automorphisms of G are the permutations of $V(G)$ that can be applied to both the rows and the columns of $A(G)$ without changing $A(G)$.

1.1.42.* Example. *Automorphisms.* Let G be the path with vertex set $\{1, 2, 3, 4\}$ and edge set $\{12, 23, 34\}$. This graph has two automorphisms: the identity permutation and the permutation that switches 1 with 4 and switches 2 with 3. Interchanging vertices 1 and 2 is not an automorphism of G, although G is isomorphic to the graph with vertex set $\{1, 2, 3, 4\}$ and edge set $\{21, 13, 34\}$.

In $K_{r,s}$, permuting the vertices of one partite set does not change the adjacency matrix; this leads to $r!s!$ automorphisms. When $r = s$, we can also interchange the partite sets; $K_{t,t}$ has $2(t!)^2$ automorphisms.

The biclique $K_{r,s}$ is vertex-transitive if and only if $r = s$. If $n > 2$, then P_n is not vertex-transitive, but every cycle is vertex-transitive. The Petersen graph is vertex-transitive. ■

We can prove a statement for every vertex in a vertex-transitive graph by proving it for one vertex. Vertex-transitivity guarantees that the graph "looks the same" from each vertex.

EXERCISES

Solutions to problems generally require clear explanations written in sentences. The designations on problems have the following meanings:

"(−)" = easier or shorter than most,
"(+)" = harder or longer than most,
"(!)" = particularly useful or instructive,
"(∗)" = involves concepts marked optional in the text.

The exercise sections begin with easier problems to check understanding, ending with a line of dots. The remaining problems roughly follow the order of material in the text.

1.1.1. (−) Determine which complete bipartite graphs are complete graphs.

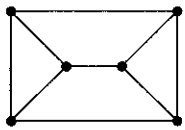
1.1.2. (−) Write down all possible adjacency matrices and incidence matrices for a 3-vertex path. Also write down an adjacency matrix for a path with six vertices and for a cycle with six vertices.

1.1.3. (−) Using rectangular blocks whose entries are all equal, write down an adjacency matrix for $K_{m,n}$.

1.1.4. (−) From the definition of isomorphism, prove that $G \cong H$ if and only if $\overline{G} \cong \overline{H}$.

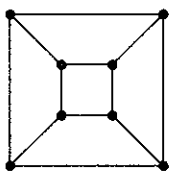
1.1.5. (−) Prove or disprove: If every vertex of a simple graph G has degree 2, then G is a cycle.

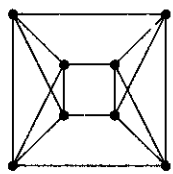
1.1.6. (−) Determine whether the graph below decomposes into copies of P_4.



1.1.7. (−) Prove that a graph with more than six vertices of odd degree cannot be decomposed into three paths.

1.1.8. (−) Prove that the 8-vertex graph on the left below decomposes into copies of $K_{1,3}$ and also into copies of P_4.



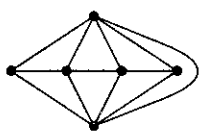


1.1.9. (−) Prove that the graph on the right above is isomorphic to the complement of the graph on the left.

1.1.10. (−) Prove or disprove: The complement of a simple disconnected graph must be connected.

• • • • •

1.1.11. Determine the maximum size of a clique and the maximum size of an independent set in the graph below.



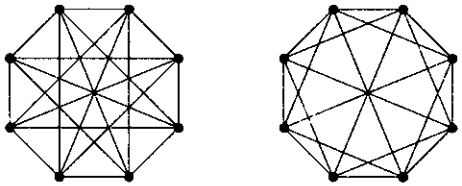
1.1.12. Determine whether the Petersen graph is bipartite, and find the size of its largest independent set.

1.1.13. Let G be the graph whose vertex set is the set of k-tuples with coordinates in $\{0, 1\}$, with x adjacent to y when x and y differ in exactly one position. Determine whether G is bipartite.

1.1.14. (!) Prove that removing opposite corner squares from an 8-by-8 checkerboard leaves a subboard that cannot be partitioned into 1-by-2 and 2-by-1 rectangles. Using the same argument, make a general statement about all bipartite graphs.

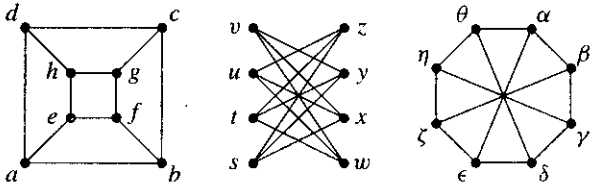
1.1.15. Consider the following four families of graphs: $A = \{\text{paths}\}$, $B = \{\text{cycles}\}$, $C = \{\text{complete graphs}\}$, $D = \{\text{bipartite graphs}\}$. For each pair of these families, determine all isomorphism classes of graphs that belong to both families.

1.1.16. Determine whether the graphs below are isomorphic.

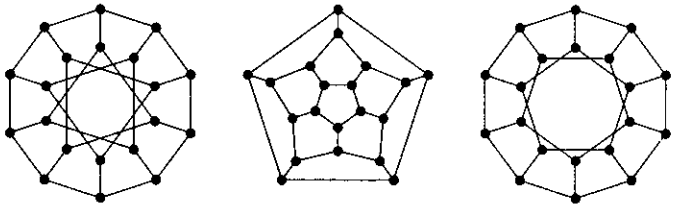


1.1.17. Determine the number of isomorphism classes of simple 7-vertex graphs in which every vertex has degree 4.

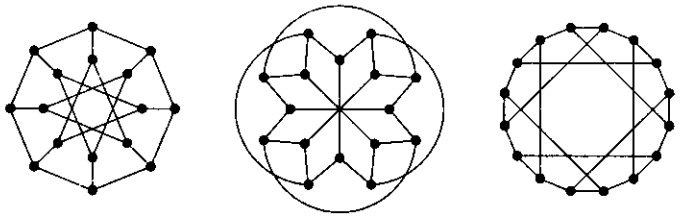
1.1.18. Determine which pairs of graphs below are isomorphic.



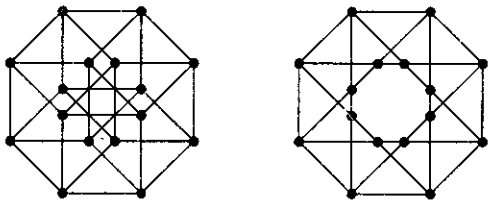
1.1.19. Determine which pairs of graphs below are isomorphic.



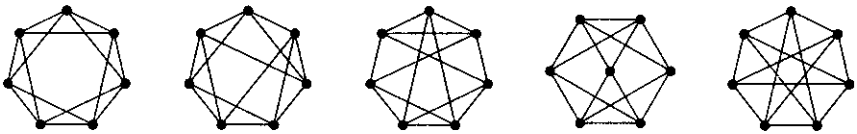
1.1.20. Determine which pairs of graphs below are isomorphic.



1.1.21. Determine whether the graphs below are bipartite and whether they are isomorphic. (The graph on the left appears on the cover of Wilson–Watkins [1990].)



1.1.22. (!) Determine which pairs of graphs below are isomorphic, presenting the proof by testing the smallest possible number of pairs.

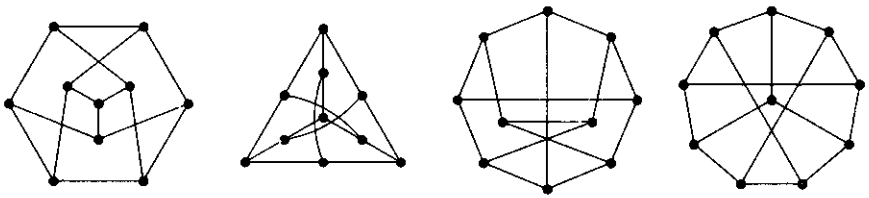


1.1.23. In each class below, determine the smallest n such that there exist nonisomorphic n-vertex graphs having the same list of vertex degrees.

(a) all graphs, (b) loopless graphs, (c) simple graphs.

(Hint: Since each class contains the next, the answers form a nondecreasing triple. For part (c), use the list of isomorphism classes in Example 1.1.31.)

1.1.24. (!) Prove that the graphs below are all drawings of the Petersen graph (Definition 1.1.36). (Hint: Use the disjointness definition of adjacency.)



1.1.25. (!) Prove that the Petersen graph has no cycle of length 7.

1.1.26. (!) Let G be a graph with girth 4 in which every vertex has degree k. Prove that G has at least $2k$ vertices. Determine all such graphs with exactly $2k$ vertices.

1.1.27. (!) Let G be a graph with girth 5. Prove that if every vertex of G has degree at least k, then G has at least $k^2 + 1$ vertices. For $k = 2$ and $k = 3$, find one such graph with exactly $k^2 + 1$ vertices.

1.1.28. (+) *The Odd Graph* O_k. The vertices of the graph O_k are the k-element subsets of $\{1, 2, \ldots, 2k + 1\}$. Two vertices are adjacent if and only if they are disjoint sets. Thus O_2 is the Petersen graph. Prove that the girth of O_k is 6 if $k \geq 3$.

1.1.29. Prove that every set of six people contains (at least) three mutual acquaintances or three mutual strangers.

1.1.30. Let G be a simple graph with adjacency matrix A and incidence matrix M. Prove that the degree of v_i is the ith diagonal entry in A^2 and in MM^T. What do the entries in position (i, j) of A^2 and MM^T say about G?

1.1.31. (!) Prove that a self-complementary graph with n vertices exists if and only if n or $n - 1$ is divisible by 4. (Hint: When n is divisible by 4, generalize the structure of P_4 by splitting the vertices into four groups. For $n \equiv 1 \bmod 4$, add one vertex to the graph constructed for $n - 1$.)

1.1.32. Determine which bicliques decompose into two isomorphic subgraphs.

1.1.33. For $n = 5$, $n = 7$, and $n = 9$, decompose K_n into copies of C_n.