1.2. Paths, Cycles, and Trails

In this section we return to the Königsberg Bridge Problem, determining when it is possible to traverse all the edges of a graph. We also we develop useful properties of connection, paths, and cycles.

Before embarking on this, we review an important technique of proof. Many statements in graph theory can be proved using the principle of induction. Readers unfamiliar with induction should read the material on this proof technique in Appendix A. Here we describe the form of induction that we will use most frequently, in order to familiarize the reader with a template for proof.

1.2.1. Theorem. (Strong Principle of Induction). Let $P(n)$ be a statement with an integer parameter n. If the following two conditions hold, then $P(n)$ is true for each positive integer n.

1) $P(1)$ is true.
2) For all $n > 1$, "$P(k)$ is true for $1 \le k < n$" implies "$P(n)$ is true".

Proof: We ASSUME the **Well Ordering Property** for the positive integers: every nonempty set of positive integers has a least element. Given this, suppose that $P(n)$ fails for some n. By the Well Ordering Property, there is a least n such that $P(n)$ fails. Statement (1) ensures that this value cannot be 1. Statement (2) ensures that this value cannot be greater than 1. The contradiction implies that $P(n)$ holds for every positive integer n. ■

In order to apply induction, we verify (1) and (2) for our sequence of statements. Verifying (1) is the **basis step** of the proof; verifying (2) is the **induction step**. The statement "$P(k)$ is true for all $k < n$" is the **induction hypothesis**, because it is the hypothesis of the implication proved in the induction step. The variable that indexes the sequence of statements is the **induction parameter**.

The induction parameter may be any integer function of the instances of our problem, such as the number of vertices or edges in a graph. We say that we are using "induction on n" when the induction parameter is n.

There are many ways to phrase inductive proofs. We can start at 0 to prove a statement for nonnegative integers. When our proof of $P(n)$ in the induction step makes use only of $P(n-1)$ from the induction hypothesis, the technique is called "ordinary" induction; making use of all previous statements is "strong" induction. We seldom distinguish between strong induction and ordinary induction; they are equivalent (see Appendix A).

Most students first learn ordinary induction in the following phrasing: 1) verify that $P(n)$ is true when $n = 1$, and 2) prove that if $P(n)$ is true when n is k, then $P(n)$ is also true when n is $k+1$. Proving $P(k+1)$ from $P(k)$ for $k \ge 1$ is equivalent to proving $P(n)$ from $P(n-1)$ for $n > 1$.

When we focus on proving the statement for the parameter value n in the induction step, we need not decide at the outset whether we are using strong induction or ordinary induction. The language is also simpler, since we avoid introducing a new name for the parameter. In Section 1.3 we will explain why this phrasing is also less prone to error.

CONNECTION IN GRAPHS

As defined in Definition 1.1.15, paths and cycles are graphs; a path *in* a graph G is a subgraph of G that is a path (similarly for cycles). We introduce further definitions to model other movements in graphs. A tourist wandering in a city (or a Königsberg pedestrian) may want to allow vertex repetitions but avoid edge repetitions. A mail carrier delivers mail to houses on both sides of the street and hence traverses each edge twice.

1.2.2. Definition. A **walk** is a list $v_0, e_1, v_1, \ldots, e_k, v_k$ of vertices and edges such that, for $1 \le i \le k$, the edge e_i has endpoints v_{i-1} and v_i. A **trail** is a walk with no repeated edge. A u,v-**walk** or u,v-**trail** has first vertex u and last vertex v; these are its **endpoints**. A u,v-**path** is a path whose vertices of degree 1 (its **endpoints**) are u and v; the others are **internal vertices**.

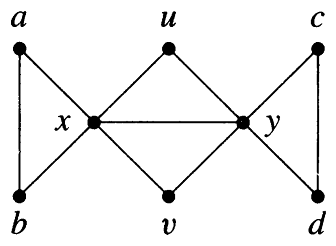
The **length** of a walk, trail, path, or cycle is its number of edges. A walk or trail is **closed** if its endpoints are the same.

1.2.3. Example. In the Königsberg graph (Example 1.1.1), the list $x, e_2, w, e_5, y, e_6, x, e_1, w, e_2, x$ is a closed walk of length 5; it repeats edge e_2 and hence is not a trail. Deleting the last edge and vertex yields a trail of length 4; it repeats vertices but not edges. The subgraph consisting of edges e_1, e_5, e_6 and vertices x, w, y is a cycle of length 3; deleting one of its edges yields a path. Two edges with the same endpoints (such as e_1 and e_2) form a cycle of length 2. A loop is a cycle of length 1. ■

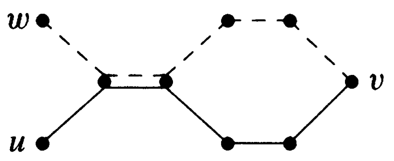
The reason for listing the edges in a walk is to distinguish among multiple edges when a graph is not simple. In a simple graph, a walk (or trail) is completely specified by its ordered list of vertices. We usually name a path, cycle, trail, or walk in a simple graph by listing only its vertices in order, even though it consists of both vertices and edges. When discussing a cycle, we can start at any vertex and do not repeat the first vertex at the end. We can use parentheses to clarify that this is a cycle and not a path.

1.2.4. Example. We illustrate the simplified notation in a simple graph. In the graph below, $a, x, a, x, u, y, c, d, y, v, x, b, a$ specifies a closed walk of length 12. Omitting the first two steps yields a closed trail.

The graph has five cycles: (a, b, x), (c, y, d), (u, x, y), (x, y, v), (u, x, v, y). The u,v-trail u, y, c, d, y, x, v contains the edges of the u,v-path u, y, x, v, but not of the u,v-path u, y, v. ■



Suppose we follow a path from u to v in a graph and then follow a path from v to w. The result need not be a u,w-path, because the u,v-path and v,w-path may have a common internal vertex. Nevertheless, the list of vertices and edges that we visit does form a u,w-walk. In the illustration below, the u,w-walk contains a u,w-path. Saying that a walk W **contains** a path P means that the vertices and edges of P occur as a sublist of the vertices and edges of W, in order but not necessarily consecutive.

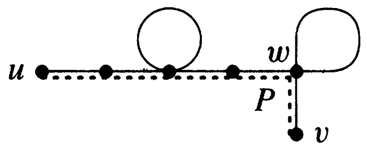


1.2.5. Lemma. Every u,v-walk contains a u,v-path.

Proof: We prove the statement by induction on the length l of a u,v-walk W.

Basis step: $l = 0$. Having no edge, W consists of a single vertex ($u = v$). This vertex is a u,v-path of length 0.

Induction step: $l \geq 1$. We suppose that the claim holds for walks of length less than l. If W has no repeated vertex, then its vertices and edges form a u,v-path. If W has a repeated vertex w, then deleting the edges and vertices between appearances of w (leaving one copy of w) yields a shorter u,v-walk W' contained in W. By the induction hypothesis, W' contains a u,v-path P, and this path P is contained in W. ■



Exercise 13b develops a shorter proof. We apply the lemma to properties of connection.

1.2.6. Definition. A graph G is **connected** if it has a u,v-path whenever $u, v \in V(G)$ (otherwise, G is **disconnected**). If G has a u,v-path, then u is **connected to** v in G. The **connection relation** on $V(G)$ consists of the ordered pairs (u, v) such that u is connected to v.

"Connected" is an adjective we apply only to graphs and to pairs of vertices (we never say "v is disconnected" when v is a vertex). The phrase "u is connected to v" is convenient when writing proofs, but in adopting it we must clarify the distinction between connection and adjacency:

G has a u, v-path	$uv \in E(G)$
u and v are connected	u and v are adjacent
u is connected to v	u is joined to v
	u is adjacent to v

1.2.7. Remark. By Lemma 1.2.5, we can prove that a graph is connected by showing that from each vertex there is a walk to one particular vertex.

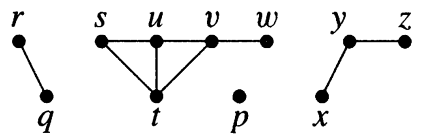
By Lemma 1.2.5, the connection relation is transitive: if G has a u, v-path and a v, w-path, then G has a u, w-path. It is also reflexive (paths of length 0) and symmetric (paths are reversible), so it is an equivalence relation. ■

Our next definition leads us to the equivalence classes of the connection relation. A *maximal* connected subgraph of G is a subgraph that is connected and is not contained in any other connected subgraph of G.

1.2.8. Definition. The **components** of a graph G are its maximal connected subgraphs. A component (or graph) is **trivial** if it has no edges; otherwise it is **nontrivial**. An **isolated vertex** is a vertex of degree 0.

The equivalence classes of the connection relation on $V(G)$ are the vertex sets of the components of G. An isolated vertex forms a trivial component, consisting of one vertex and no edge.

1.2.9. Example. The graph below has four components, one being an isolated vertex. The vertex sets of the components are $\{p\}$, $\{q, r\}$, $\{s, t, u, v, w\}$, and $\{x, y, z\}$; these are the equivalence classes of the connection relation. ■



1.2.10. Remark. Components are pairwise disjoint; no two share a vertex. Adding an edge with endpoints in distinct components combines them into one component. Thus adding an edge decreases the number of components by 0 or 1, and deleting an edge increases the number of components by 0 or 1. ■

1.2.11. Proposition. Every graph with n vertices and k edges has at least $n - k$ components.

Proof: An n-vertex graph with no edges has n components. By Remark 1.2.10, each edge added reduces this by at most 1, so when k edges have been added the number of components is still at least $n - k$. ■

Deleting a vertex or an edge can increase the number of components. Although deleting an edge can only increase the number of components by 1, deleting a vertex can increase it by many (consider the biclique $K_{1,m}$). When we obtain a subgraph by deleting a vertex, it must be a graph, so deleting the vertex also deletes all edges incident to it.

1.2.12. Definition. A **cut-edge** or **cut-vertex** of a graph is an edge or vertex whose deletion increases the number of components. We write $G - e$ or $G - M$ for the subgraph of G obtained by deleting an edge e or set of edges M. We write $G - v$ or $G - S$ for the subgraph obtained by deleting a vertex v or set of vertices S. An **induced subgraph** is a subgraph obtained by deleting a set of vertices. We write $G[T]$ for $G - \overline{T}$, where $\overline{T} = V(G) - T$; this is the subgraph of G **induced by** T.

When $T \subseteq V(G)$, the induced subgraph $G[T]$ consists of T and all edges whose endpoints are contained in T. The full graph is itself an induced subgraph, as are individual vertices. A set S of vertices is an independent set if and only if the subgraph induced by it has no edges.

1.2.13. Example. The graph of Example 1.2.9 has cut-vertices v and y. Its cut-edges are qr, vw, xy, and yz. (When we delete an edge, its endpoints remain.)

This graph has C_4 and P_5 as subgraphs but *not* as induced subgraphs. The subgraph induced by $\{s, t, u, v\}$ is a kite; the 4-vertex paths on these vertices are not induced subgraphs. The graph P_4 does occur as an induced subgraph; it is the subgraph induced by $\{s, t, v, w\}$ (also by $\{s, u, v, w\}$). ■

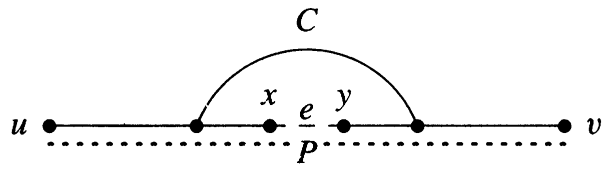
Next we characterize cut-edges in terms of cycles.

1.2.14. Theorem. An edge is a cut-edge if and only if it belongs to no cycle.

Proof: Let e be an edge in a graph G (with endpoints x, y), and let H be the component containing e. Since deletion of e affects no other component, it suffices to prove that $H - e$ is connected if and only if e belongs to a cycle.

First suppose that $H - e$ is connected. This implies that $H - e$ contains an x, y-path, and this path completes a cycle with e.

Now suppose that e lies in a cycle C. Choose $u, v \in V(H)$. Since H is connected, H has a u, v-path P. If P does not contain e, then P exists in $H - e$. If P contains e, suppose by symmetry that x is between u and y on P. Since $H - e$ contains a u, x-path along P, an x, y-path along C, and a y, v-path along P, the transitivity of the connection relation implies that $H - e$ has a u, v-path. We did this for all $u, v \in V(H)$, so $H - e$ is connected. ■



BIPARTITE GRAPHS

Our next goal is to characterize bipartite graphs using cycles. Characterizations are equivalence statements, like Theorem 1.2.14. When two conditions are equivalent, checking one also yields the other for free.

Characterizing a class $\mathbf{G}$ by a condition P means proving the equivalence "$G \in \mathbf{G}$ if and only if G satisfies P". In other words, P is both a **necessary** and a **sufficient** condition for membership in $\mathbf{G}$.

Necessity	Sufficiency
$G \in \mathbf{G}$ *only if* G satisfies P	$G \in \mathbf{G}$ *if* G satisfies P
$G \in \mathbf{G} \Rightarrow G$ satisfies P	G satisfies P $\Rightarrow G \in \mathbf{G}$

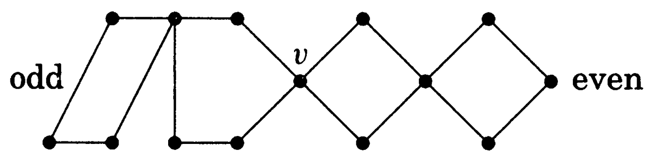
Recall that a loop is a cycle of length 1; also two distinct edges with the same endpoints form a cycle of length 2. A walk is **odd** or **even** as its length is odd or even. As in Lemma 1.2.5, a closed walk **contains** a cycle C if the vertices and edges of C occur as a sublist of W, in cyclic order but not necessarily consecutive. We can think of a closed walk or a cycle as starting at any vertex; the next lemma requires this viewpoint.

1.2.15. Lemma. Every closed odd walk contains an odd cycle.

Proof: We use induction on the length l of a closed odd walk W.

Basis step: $l = 1$. A closed walk of length 1 traverses a cycle of length 1.

Induction step: $l > 1$. Assume the claim for closed odd walks shorter than W. If W has no repeated vertex (other than first = last), then W itself forms a cycle of odd length. If vertex v is repeated in W, then we view W as starting at v and break W into two v, v-walks. Since W has odd length, one of these is odd and the other is even. The odd one is shorter than W. By the induction hypothesis, it contains an odd cycle, and this cycle appears in order in W. ■



1.2.16. Remark. A closed even walk need not contain a cycle; it may simply repeat. Nevertheless, if an edge e appears *exactly once* in a closed walk W, then W does contain a cycle through e. Let x, y be the endpoints of e. Deleting e from W leaves an x, y-walk that avoids e. By Lemma 1.2.5, this walk contains an x, y-path, and this path completes a cycle with e. (See Exercises 15–16.) ■

Lemma 1.2.15 will help us characterize bipartite graphs.

1.2.17. Definition. A **bipartition** of G is a specification of two disjoint independent sets in G whose union is $V(G)$. The statement "Let G be a bipartite graph with bipartition X, Y" specifies one such partition. An X, Y**-bigraph** is a bipartite graph with bipartition X, Y.

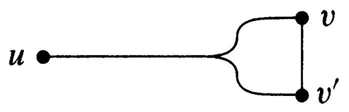
The sets of a bipartition are partite sets (Definition 1.1.10). A disconnected bipartite graph has more than one bipartition. A connected bipartite graph has only one bipartition, except for interchanging the two sets (Exercise 7).

1.2.18. Theorem. (König [1936]) A graph is bipartite if and only if it has no odd cycle.

Proof: *Necessity.* Let G be a bipartite graph. Every walk alternates between the two sets of a bipartition, so every return to the original partite set happens after an even number of steps. Hence G has no odd cycle.

Sufficiency. Let G be a graph with no odd cycle. We prove that G is bipartite by constructing a bipartition of each nontrivial component. Let u be a vertex in a nontrivial component H. For each $v \in V(H)$, let $f(v)$ be the minimum length of a u, v-path. Since H is connected, $f(v)$ is defined for each $v \in V(H)$.

Let $X = \{v \in V(H)\colon f(v) \text{ is even}\}$ and $Y = \{v \in V(H)\colon f(v) \text{ is odd}\}$. An edge v, v' within X or Y would create a closed odd walk using a shortest u, v-path, the edge vv', and the reverse of a shortest u, v'-path. By Lemma 1.2.15, such a walk must contain an odd cycle, which contradicts our hypothesis. Hence X and Y are independent sets. Also $X \cup Y = V(H)$, so H is an X, Y-bigraph. ■

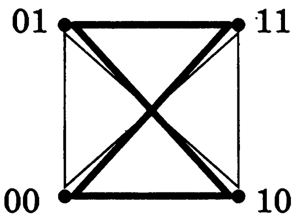


1.2.19. Remark. *Testing whether a graph is bipartite.* Theorem 1.2.18 implies that whenever a graph G is not bipartite, we can prove this statement by presenting an odd cycle in G. This is much easier than examining all possible bipartitions to prove that none work. When we want to prove that G *is* bipartite, we define a bipartition and prove that the two sets are independent; this is easier than examining all cycles. ■

We consider one application.

1.2.20. Definition. The **union** of graphs $G_1, \dots, G_k$, written $G_1 \cup \cdots \cup G_k$, is the graph with vertex set $\bigcup_{i=1}^{k} V(G_i)$ and edge set $\bigcup_{i=1}^{k} E(G_i)$.

1.2.21. Example. Below we show K_4 as the union of two 4-cycles. When a graph G is expressed as the union of two or more subgraphs, an edge of G can belong to many of them. This distinguishes union from decomposition, where each edge belongs to only one subgraph in the list. ■



1.2.22. Example. Consider an air traffic system with k airlines. Suppose that
1) direct service between two cities means round-trip direct service, and
2) each pair of cities has direct service from at least one airline.

Suppose also that no airline can schedule a cycle through an odd number of cities. In terms of k, what is the maximum number of cities in the system?

By Theorem 1.2.18, we seek the largest n such that K_n can be expressed as the union of k bipartite graphs, one for each airline. The answer is 2^k. ∎

1.2.23. Theorem. The complete graph K_n can be expressed as the union of k bipartite graphs if and only if $n \leq 2^k$.

Proof: We use induction on k. Basis step: $k = 1$. Since K_3 has an odd cycle and K_2 does not, K_n is itself a bipartite graph if and only if $n \leq 2$.

Induction step: $k > 1$. We prove each implication using the induction hypothesis. Suppose first that $K_n = G_1 \cup \cdots \cup G_k$, where each G_i is bipartite. We partition the vertex set into two sets X, Y such that G_k has no edge within X or within Y. The union of the other $k-1$ bipartite subgraphs must cover the complete subgraphs induced by X and by Y. Applying the induction hypothesis to each yields $|X| \leq 2^{k-1}$ and $|Y| \leq 2^{k-1}$, so $n \leq 2^{k-1} + 2^{k-1} = 2^k$.

Conversely, suppose that $n \leq 2^k$. We partition the vertex set into subsets X, Y, each of size at most 2^{k-1}. By the induction hypothesis, we can cover the complete subgraph induced by either subset with $k-1$ bipartite subgraphs. The union of the ith such subgraph on X with the ith such subgraph on Y is a bipartite graph. Hence we obtain $k-1$ bipartite graphs whose union consists of the complete subgraphs induced by X and Y. The remaining edges are those of the biclique with bipartition X, Y. Letting this be the kth bipartite subgraph completes the construction. ∎

This theorem can also be proved without induction by encoding the vertices as binary k-tuples (Exercise 31).

EULERIAN CIRCUITS

We return to our analysis of the Königsberg Bridge Problem. What the people of Königsberg wanted was a closed trail containing all the edges in a graph. As we have observed, a necessary condition for existence of such a trail is that all vertex degrees be even. Also it is necessary that all edges belong to the same component of the graph.

The Swiss mathematician Leonhard Euler (pronounced "oiler") stated [1736] that these conditions are also sufficient. In honor of his contribution, we associate his name with such graphs. Euler's paper appeared in 1741 but gave no proof that the obvious necessary conditions are sufficient. Hierholzer [1873] gave the first complete published proof. The graph we drew in Example 1.1.1 to model the city did not appear in print until 1894 (see Wilson [1986] for a discussion of the historical record).

1.2.24. Definition. A graph is **Eulerian** if it has a closed trail containing all edges. We call a closed trail a **circuit** when we do not specify the first vertex but keep the list in cyclic order. An **Eulerian circuit** or **Eulerian trail** in a graph is a circuit or trail containing all the edges.

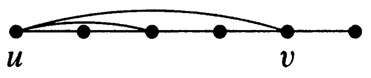
An **even graph** is a graph with vertex degrees all even. A vertex is **odd** [**even**] when its degree is odd [even].

Our discussion of Eulerian circuits applies also to graphs with loops; we extend the notion of vertex degree to graphs with loops by letting each loop contribute 2 to the degree of its vertex. This does not change the parity of the degree, and the presence of a loop does not affect whether a graph has an Eulerian circuit unless it is a loop in a component with one vertex.

Our proof of the characterization of Eulerian graphs uses a lemma. A **maximal path** in a graph G is a path P in G that is not contained in a longer path. When a graph is finite, no path can extend forever, so maximal (non-extendible) paths exist.

1.2.25. Lemma. If every vertex of a graph G has degree at least 2, then G contains a cycle.

Proof: Let P be a maximal path in G, and let u be an endpoint of P. Since P cannot be extended, every neighbor of u must already be a vertex of P. Since u has degree at least 2, it has a neighbor v in $V(P)$ via an edge not in P. The edge uv completes a cycle with the portion of P from v to u. ∎



Note the importance of finiteness. If $V(G) = \mathbb{Z}$ and $E(G) = \{ij\colon |i - j| = 1\}$, then every vertex of G has degree 2, but G has no cycle (and no non-extendible path). We avoid such examples by assuming that all graphs in this book are finite, with rare explicit exceptions.

1.2.26. Theorem. A graph G is Eulerian if and only if it has at most one nontrivial component and its vertices all have even degree.

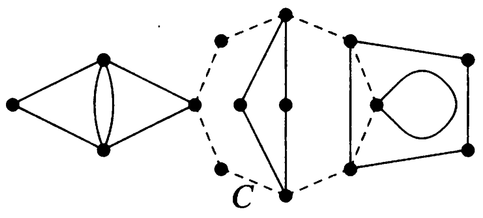
Proof: *Necessity.* Suppose that G has an Eulerian circuit C. Each passage of C through a vertex uses two incident edges, and the first edge is paired with the last at the first vertex. Hence every vertex has even degree. Also, two edges can be in the same trail only when they lie in the same component, so there is at most one nontrivial component.

Sufficiency. Assuming that the condition holds, we obtain an Eulerian circuit using induction on the number of edges, m.

Basis step: $m = 0$. A closed trail consisting of one vertex suffices.

Induction step: $m > 0$. With even degrees, each vertex in the nontrivial component of G has degree at least 2. By Lemma 1.2.25, the nontrivial component has a cycle C. Let G' be the graph obtained from G by deleting $E(C)$.

Since C has 0 or 2 edges at each vertex, each component of G' is also an even graph. Since each component also is connected and has fewer than m edges, we can apply the induction hypothesis to conclude that each component of G' has an Eulerian circuit. To combine these into an Eulerian circuit of G, we traverse C, but when a component of G' is entered for the first time we detour along an Eulerian circuit of that component. This circuit ends at the vertex where we began the detour. When we complete the traversal of C, we have completed an Eulerian circuit of G. ■



Perhaps as important as the characterization of Eulerian graphs is what the method of proof says about even graphs.

1.2.27. Proposition. Every even graph decomposes into cycles.

Proof: In the proof of Theorem 1.2.26, we noted that every even nontrivial graph has a cycle, and that the deletion of a cycle leaves an even graph. Thus this proposition follows by induction on the number of edges. ■

In the characterization of Eulerian circuits, the necessity of the condition is easy to see. This also holds for the characterization of bipartite graphs by absence of odd cycles and for many other characterizations. Nash-Williams and others popularized a mnemonic for such theorems: **TONCAS**, meaning "The Obvious Necessary Conditions are Also Sufficient".

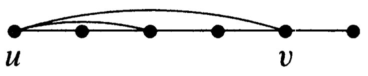
The proof of Lemma 1.2.25 is an example of an important technique of proof in graph theory that we call **extremality**. When considering structures of a given type, choosing an example that is extreme in some sense may yield useful additional information. For example, since a maximal path P cannot be extended, we obtain the extra information that every neighbor of an endpoint of P belongs to $V(P)$.

In a sense, making an extremal choice goes directly to the important case. In Lemma 1.2.25, we could start with any path. If it is extendible, then we extend it. If not, then something important happens. We illustrate the technique with several examples, and Exercises 37–42 also use extremality. We begin by strengthening Lemma 1.2.25 for simple graphs.

1.2.28. Proposition. If G is a simple graph in which every vertex has degree at least k, then G contains a path of length at least k. If $k \geq 2$, then G also contains a cycle of length at least $k + 1$.

Proof: Let u be an endpoint of a maximal path P in G. Since P does not extend, every neighbor of u is in $V(P)$. Since u has at least k neighbors and G is simple,

P therefore has at least k vertices other than u and has length at least k. If $k \geq 2$, then the edge from u to its farthest neighbor v along P completes a sufficiently long cycle with the portion of P from v to u. ■



1.2.29. Proposition. Every graph with a nonloop edge has at least two vertices that are not cut-vertices.

Proof: If u is an endpoint of a maximal path P in G, then the neighbors of u lie on P. Since $P - u$ is connected in $G - u$, the neighbors of u belong to a single component of $G - u$, and u is not a cut-vertex. ■

1.2.30. Remark. Note the difference between "maximal" and "maximum". As adjectives, **maximum** means "maximum-sized", and **maximal** means "no larger one contains this one". Every maximum path is a maximal path, but maximal paths need not have maximum length. Similarly, the biclique $K_{r,s}$ has two maximal independent sets, but when $r \neq s$ it has only one maximum independent set. When describing numbers rather than containment, the meanings are the same; maximum vertex degree = maximal vertex degree.

Besides maximal or maximum paths or independent sets, other extremal aspects include vertices of minimum or maximum degree, the first vertex where two paths diverge, maximal connected subgraphs (components), etc. In a connected graph G with disjoint sets $S, T \subset V(G)$, we can obtain a path from S to T having only its endpoints in $S \cup T$ by choosing a shortest path from S to T; Exercise 40 applies this. Exercise 37 uses extremality for a short proof of the transitivity of the connection relation. ■

Many proofs using induction can be phrased using extremality, and many proofs using extremality can be done by induction. To underscore the interplay, we reprove the characterization of Eulerian graphs using extremality directly.

1.2.31. Lemma. In an even graph, every maximal trail is closed.

Proof: Let T be a maximal trail in an even graph. Every passage of T through a vertex v uses two edges at v, none repeated. Thus when arriving at a vertex v other than its initial vertex, T has used an odd number of edges incident to v. Since v has even degree, there remains an edge on which T can continue.

Hence T can only end at its initial vertex. In a finite graph, T must indeed end. We conclude that a maximal trail must be closed. ■

1.2.32. Theorem 1.2.26—Second Proof. We prove TONCAS. In a graph G satisfying the conditions, let T be a trail of maximum length; T must also be a maximal trail. By Lemma 1.2.31, T is closed.

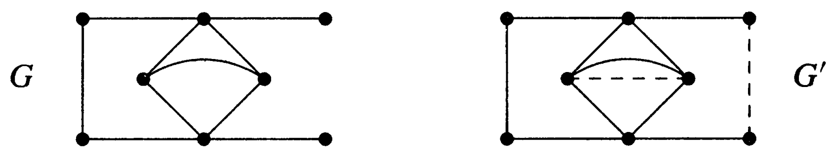
Suppose that T omits some edge e of G. Since G has only one nontrivial component, G has a shortest path from e to the vertex set of T. Hence some edge e' not in T is incident to some vertex v of T.

Since T is closed, there is a trail T' that starts and ends at v and uses the same edges as T. We now extend T' along e' to obtain a longer trail than T. This contradicts the choice of T, and hence T traverses all edges of G. ■

This proof and the resulting construction procedure (Exercise 12) are similar to those of Hierholzer [1873]. Exercise 35 develops another proof.

Later chapters contain several applications of the statement that every connected even graph has an Eulerian circuit. Here we give a simple one. When drawing a figure G on paper, how many times must we stop and move the pen? We are not allowed to repeat segments of the drawing, so each visit to the paper contributes a trail. Thus we seek a decomposition of G into the minimum number of trails. We may reduce the problem to connected graphs, since the number of trails needed to draw G is the sum of the number needed to draw each component.

For example, the graph G below has four odd vertices and decomposes into two trails. Adding the dashed edges on the right makes it Eulerian.



1.2.33. Theorem. For a connected nontrivial graph with exactly $2k$ odd vertices, the minimum number of trails that decompose it is $\max\{k, 1\}$.

Proof: A trail contributes even degree to every vertex, except that a non-closed trail contributes odd degree to its endpoints. Therefore, a partition of the edges into trails must have some non-closed trail ending at each odd vertex. Since each trail has only two ends, we must use at least k trails to satisfy $2k$ odd vertices. We also need at least one trail since G has an edge, and Theorem 1.2.26 implies that one trail suffices when $k = 0$.

It remains to prove that k trails suffice when $k > 0$. Given such a graph G, we pair up the odd vertices in G (in any way) and form G' by adding for each pair an edge joining its two vertices, as illustrated above. The resulting graph G' is connected and even, so by Theorem 1.2.26 it has an Eulerian circuit C. As we traverse C in G', we start a new trail in G each time we traverse an edge of $G' - E(G)$. This yields k trails decomposing G. ■

We prove theorems in general contexts to avoid work. The proof of Theorem 1.2.33 illustrates this; by transforming G into a graph where Theorem 1.2.26 applies, we avoid repeating the basic argument of Theorem 1.2.26. Exercise 33 requests a proof of Theorem 1.2.33 directly by induction.

Note that Theorem 1.2.33 considers only graphs having an even number of vertices of odd degree. Our first result in the next section explains why.

EXERCISES

Most problems in this book require proofs. Words like “construct”, “show”, “obtain”, “determine”, etc., explicitly state that proof is required. Disproof by providing a counterexample requires confirming that it is a counterexample.

1.2.1. (−) Determine whether the statements below are true or false.

a) Every disconnected graph has an isolated vertex.

b) A graph is connected if and only if some vertex is connected to all other vertices.

c) The edge set of every closed trail can be partitioned into edge sets of cycles.

d) If a maximal trail in a graph is not closed, then its endpoints have odd degree.

1.2.2. (−) Determine whether K_4 contains the following (give an example or a proof of non-existence).

a) A walk that is not a trail.

b) A trail that is not closed and is not a path.

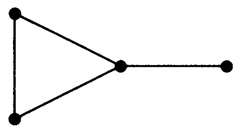
c) A closed trail that is not a cycle.

1.2.3. (−) Let G be the graph with vertex set $\{1, \ldots, 15\}$ in which i and j are adjacent if and only if their greatest common factor exceeds 1. Count the components of G and determine the maximum length of a path in G.

1.2.4. (−) Let G be a graph. For $v \in V(G)$ and $e \in E(G)$, describe the adjacency and incidence matrices of $G - v$ and $G - e$ in terms of the corresponding matrices for G.

1.2.5. (−) Let v be a vertex of a connected simple graph G. Prove that v has a neighbor in every component of $G - v$. Conclude that no graph has a cut-vertex of degree 1.

1.2.6. (−) In the graph below (the paw), find all the maximal paths, maximal cliques, and maximal independent sets. Also find all the maximum paths, maximum cliques, and maximum independent sets.



1.2.7. (−) Prove that a bipartite graph has a unique bipartition (except for interchanging the two partite sets) if and only if it is connected.

1.2.8. (−) Determine the values of m and n such that $K_{m,n}$ is Eulerian.

1.2.9. (−) What is the minimum number of trails needed to decompose the Petersen graph? Is there a decomposition into this many trails using only paths?

1.2.10. (−) Prove or disprove:

a) Every Eulerian bipartite graph has an even number of edges.

b) Every Eulerian simple graph with an even number of vertices has an even number of edges.

1.2.11. (−) Prove or disprove: If G is an Eulerian graph with edges e, f that share a vertex, then G has an Eulerian circuit in which e, f appear consecutively.

1.2.12. (−) Convert the proof at 1.2.32 to an procedure for finding an Eulerian circuit in a connected even graph.

• • • • •

1.2.13. *Alternative proofs that every u, v-walk contains a u, v-path (Lemma 1.2.5).*

a) (ordinary induction) Given that every walk of length $l - 1$ contains a path from its first vertex to its last, prove that every walk of length l also satisfies this.

b) (extremality) Given a u, v-walk W, consider a shortest u, v-walk contained in W.

1.2.14. Prove or disprove the following statements about simple graphs. (Comment: "Distinct" does not mean "disjoint".)

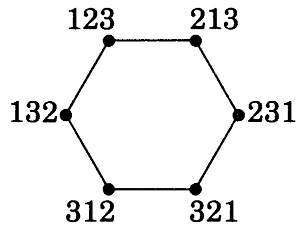
a) The union of the edge sets of distinct u, v-walks must contain a cycle.

b) The union of the edge sets of distinct u, v-paths must contain a cycle.

1.2.15. (!) Let W be a closed walk of length at least 1 that does not contain a cycle. Prove that some edge of W repeats immediately (once in each direction).

1.2.16. Let e be an edge appearing an odd number of times in a closed walk W. Prove that W contains the edges of a cycle through e.

1.2.17. (!) Let G_n be the graph whose vertices are the permutations of $\{1, \dots, n\}$, with two permutations $a_1, \dots, a_n$ and $b_1, \dots, b_n$ adjacent if they differ by interchanging a pair of adjacent entries (G_3 shown below). Prove that G_n is connected.



1.2.18. (!) Let G be the graph whose vertex set is the set of k-tuples with elements in $\{0, 1\}$, with x adjacent to y if x and y differ in exactly two positions. Determine the number of components of G.

1.2.19. Let r and s be natural numbers. Let G be the simple graph with vertex set $v_0, \dots, v_{n-1}$ such that $v_i \leftrightarrow v_j$ if and only if $|j - i| \in \{r, s\}$. Prove that S has exactly k components, where k is the greatest common divisor of $\{n, r, s\}$.

1.2.20. (!) Let v be a cut-vertex of a simple graph G. Prove that $\overline{G} - v$ is connected.

1.2.21. Let G be a self-complementary graph. Prove that G has a cut-vertex if and only if G has a vertex of degree 1. (Akiyama–Harary [1981])

1.2.22. Prove that a graph is connected if and only if for every partition of its vertices into two nonempty sets, there is an edge with endpoints in both sets.

1.2.23. For each statement below, determine whether it is true for every connected simple graph G that is not a complete graph.

a) Every vertex of G belongs to an induced subgraph isomorphic to P_3.

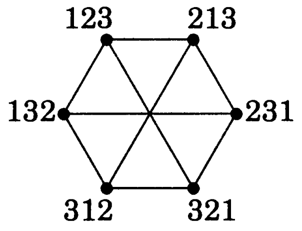
b) Every edge of G belongs to an induced subgraph isomorphic to P_3.

1.2.24. Let G be a simple graph having no isolated vertex and no induced subgraph with exactly two edges. Prove that G is a complete graph.

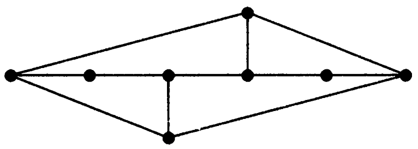
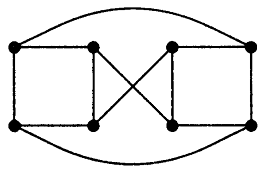
1.2.25. (!) Use ordinary induction on the number of edges to prove that absence of odd cycles is a sufficient condition for a graph to be bipartite.

1.2.26. (!) Prove that a graph G is bipartite if and only if every subgraph H of G has an independent set consisting of at least half of $V(H)$.

1.2.27. Let G_n be the graph whose vertices are the permutations of $\{1, \dots, n\}$, with two permutations $a_1, \dots, a_n$ and $b_1, \dots, b_n$ adjacent if they differ by switching two entries. Prove that G_n is bipartite (G_3 shown below). (Hint: For each permutation a, count the pairs i, j such that $i < j$ and $a_i > a_j$; these are called **inversions**.)



1.2.28. (!) In each graph below, find a bipartite subgraph with the maximum number of edges. Prove that this is the maximum, and determine whether this is the only bipartite subgraph with this many edges.

1.2.29. (!) Let G be a connected simple graph not having P_4 or C_3 as an induced subgraph. Prove that G is a biclique (complete bipartite graph).

1.2.30. Let G be a simple graph with vertices $v_1, \dots, v_n$. Let A^k denote the kth power of the adjacency matrix of G under matrix multiplication. Prove that entry i, j of A^k is the number of v_i, v_j-walks of length k in G. Prove that G is bipartite if and only if, for the odd integer r nearest to n, the diagonal entries of A^r are all 0. (Reminder: A walk is an **ordered** list of vertices and edges.)

1.2.31. (!) *Non-inductive proof of Theorem 1.2.23* (see Example 1.2.21).

a) Given $n \le 2^k$, encode the vertices of K_n as distinct binary k-tuples. Use this to construct k bipartite graphs whose union is K_n.

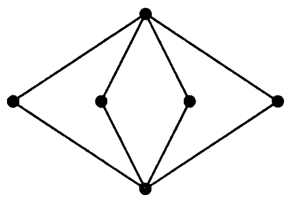
b) Given that K_n is a union of bipartite graphs $G_1, \dots, G_k$, encode the vertices of K_n as distinct binary k-tuples. Use this to prove that $n \le 2^k$.

1.2.32. The statement below is false. Add a hypothesis to correct it, and prove the corrected statement.

"Every maximal trail in an even graph is an Eulerian circuit."

1.2.33. Use ordinary induction on k or on the number of edges (one by one) to prove that a connected graph with $2k$ odd vertices decomposes into k trails if $k > 0$. Does this remain true without the connectedness hypothesis?

1.2.34. Two Eulerian circuits are *equivalent* if they have the same unordered pairs of consecutive edges, viewed cyclically (the starting point and direction are unimportant). A cycle, for example, has only one equivalence class of Eulerian circuits. How many equivalence classes of Eulerian circuits are there in the graph drawn below?



1.2.35. *Tucker's Algorithm.* Let G be a connected even graph. At each vertex, partition the incident edges into pairs (each edge appears in a pair for each of its endpoints). Starting along a given edge e, form a trail by leaving each vertex along the edge paired with the edge just used to enter it, ending with the edge paired with e. This decomposes G into closed trails. As long as there is more than one trail in the decomposition, find two trails with a common vertex and combine them into a longer trail by changing the pairing at a common vertex. Prove that this procedure works and produces an Eulerian circuit as its final trail. (Tucker [1976])

1.2.36. (+) *Alternative characterization of Eulerian graphs.*

a) Prove that if G is Eulerian and $G' = G - uv$, then G' has an odd number of u, v-trails that visit v only at the end. Prove also that the number of the trails in this list that are not paths is even. (Toida [1973])

b) Let v be a vertex of odd degree in a graph. For each edge e incident to v, let $c(e)$ be the number of cycles containing e. Use $\sum_e c(e)$ to prove that $c(e)$ is even for some e incident to v. (McKee [1984])

c) Use part (a) and part (b) to conclude that a nontrivial connected graph is Eulerian if and only if every edge belongs to an odd number of cycles.

1.2.37. (!) Use extremality to prove that the connection relation is transitive. (Hint: Given a u, v-path P and a v, w-path Q, consider the first vertex of P in Q.)

1.2.38. (!) Prove that every n-vertex graph with at least n edges contains a cycle.

1.2.39. Suppose that every vertex of a loopless graph G has degree at least 3. Prove that G has a cycle of even length. (Hint: Consider a maximal path.) (P. Kwok)

1.2.40. (!) Let P and Q be paths of maximum length in a connected graph G. Prove that P and Q have a common vertex.

1.2.41. Let G be a connected graph with at least three vertices. Prove that G has two vertices x, y such that 1) $G - \{x, y\}$ is connected and 2) x, y are adjacent or have a common neighbor. (Hint: Consider a longest path.) (Chung [1978a])

1.2.42. Let G be a connected simple graph that does not have P_4 or C_4 as an induced subgraph. Prove that G has a vertex adjacent to all other vertices. (Hint: Consider a vertex of maximum degree.) (Wolk [1965])

1.2.43. (+) Use induction on k to prove that every connected simple graph with an even number of edges decomposes into paths of length 2. Does the conclusion remain true if the hypothesis of connectedness is omitted?

1.3. Vertex Degrees and Counting

The degrees of the vertices are fundamental parameters of a graph. We repeat the definition in order to introduce important notation.

1.3.1. Definition. The **degree** of vertex v in a graph G, written $d_G(v)$ or $d(v)$, is the number of edges incident to v, except that each loop at v counts twice. The maximum degree is $\Delta(G)$, the minimum degree is $\delta(G)$, and G is **regular** if $\Delta(G) = \delta(G)$. It is **k-regular** if the common degree is k. The **neighborhood** of v, written $N_G(v)$ or $N(v)$, is the set of vertices adjacent to v.