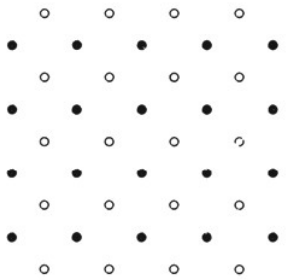
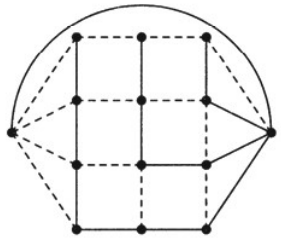
Bridges cannot cross. Therefore, every bridge that is played eliminates a potential move for the other player. Since every path from left to right crosses every path from top to bottom, the players cannot both win. Note also that the layout of the board is symmetric in the two players.

We argue that Player 2 cannot have a winning strategy. Suppose otherwise. Because the board is symmetric, Player 1 can start with any move and then followthe strategy of Player 2, making an arbitrary move if the strategy of Player 2 ever calls for a bridge that has already been played. Before Player 2 can win, Player 1 wins by using the same strategy.

If the game is played until no further moves are possible, then some player must have won (Exercise 70). Since Player 2 has no winning strategy, this implies that Player 1 has a winning strategy. Here we give an explicit strategythat Player 1 can use to win. (The argument holds more generally in the contextof "matroids"—see Theorem 8.2.46.)





2.1.17. Theorem. Player 1 has a winning strategy in Bridg-it.

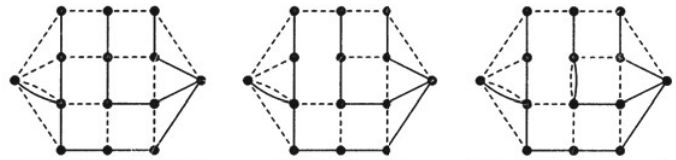
Proof: We form a graph of the potential connections for Player 1. Posts on the same end are equivalent, so we collect the (solid) posts from the end columns as single vertices. We add an auxiliary edge between the ends. The figure on the right above illustrates that this graph is the union of two edge-disjoint spanning trees; we omit a technical description of the two trees.

Together, the two trees contain edge-disjoint paths between the goal vertices. Since the auxiliary edge doesn't really exist, we pretend Player 2 moved first and took that edge. A move by Player 2 cuts one edge e in the graph and makes it no longer available. This cuts one of the trees into two components. By Proposition 2.1.6, some edge e' from the other tree reconnects it.

Player 1 chooses such an edge e'. This makes e' uncuttable, in effect putting e' in both spanning trees. After deleting e and making e' a double edge with one copy in each tree, our graph still consists of two edge-disjoint spanning trees. Since Player 2 cannot cut a double edge, Player 2 cannot cut both trees. Thus Player 1 can always defend. The figure below illustrates the strategy.

The process stops when Player 1 has won or when no single edges remain to be cut. In the latter case the remaining edges are double edges and form a

spanning tree of bridges built by Player 1. Thus in either case Player 1 has constructed a path connecting the special vertices. ∎



Player 1 reconnects Player 2 cuts Player 1 reconnects

EXERCISES

2.1.1. (–) For each k, list the isomorphism classes of trees with maximum degree k and at most six vertices. Do the same for diameter k. (Explain why there are no others.)

2.1.2. (–) Let G be a graph.
a) Prove that G is a tree if and only if G is connected and every edge is a cut-edge.
b) Prove that G is a tree if and only if adding any edge with endpoints in $V(G)$ creates exactly one cycle.

2.1.3. (–) Prove that a graph is a tree if and only if it is loopless and has exactly one spanning tree.

2.1.4. (–) Prove or disprove: Every graph with fewer edges than vertices has a component that is a tree.

2.1.5. (–) Let G be a graph. Prove that a maximal acyclic subgraph of G consists of a spanning tree from each component of G.

2.1.6. (–) Let T be a tree with average degree a. In terms of a, determine $n(T)$.

2.1.7. (–) Prove that every n-vertex graph with m edges has at least $m - n + 1$ cycles.

2.1.8. (–) Prove that each property below characterizes forests.
a) Every induced subgraph has a vertex of degree at most 1.
b) Every connected subgraph is an induced subgraph.
c) The number of components is the number of vertices minus the number of edges.

2.1.9. (–) For $2 \le k \le n-1$, prove that the n-vertex graph formed by adding one vertex adjacent to every vertex of P_{n-1} has a spanning tree with diameter k.

2.1.10. (–) Let u and v be vertices in a connected n-vertex simple graph. Prove that if $d(u,v) > 2$, then $d(u) + d(v) \le n + 1 - d(u,v)$. Construct examples to show that this can fail whenever $n \ge 3$ and $d(u,v) \le 2$.

2.1.11. (–) Let x and y be adjacent vertices in a graph G. For all $z \in V(G)$, prove that $|d_G(x,z) - d_G(y,z)| \le 1$.

2.1.12. (–) Compute the diameter and radius of the biclique $K_{m,n}$.

2.1.13. (–) Prove that every graph with diameter d has an independent set with at least $\lceil (1+d)/2 \rceil$ vertices.

2.1.14. (−) Suppose that the processors in a computer are named by binary k-tuples, and pairs can communicate directly if and only if their names are adjacent in the k-dimensional cube Q_k. A processor with name u wants to send a message to the processor with name v. How can it find the first step on a shortest path to v?

2.1.15. (−) Let G be a simple graph with diameter at least 4. Prove that $\overline{G}$ has diameter at most 2. (Hint: Use Theorem 2.1.11.)

2.1.16. (−) Given a simple graph G, define G' to be the simple graph on the same vertex set such that $xy \in E(G')$ if and only if x and y are adjacent in G or have a common neighbor in G. Prove that $\operatorname{diam}(G') = \lceil \operatorname{diam}(G)/2 \rceil$.

• • • • •

2.1.17. (!) Prove C ⇒ {A, B} in Theorem 2.1.4 by adding edges to connect components.

2.1.18. (!) Prove that every tree with maximum degree $\Delta > 1$ has at least Δ vertices of degree 1. Show that this is best possible by constructing an n-vertex tree with exactly Δ leaves, for each choice of n, Δ with $n > \Delta \geq 2$.

2.1.19. Prove or disprove: If n_i denotes the number of vertices of degree i in a tree T, then $\sum i n_i$ depends only on the number of vertices in T.

2.1.20. A *saturated hydrocarbon* is a molecule formed from k carbon atoms and l hydrogen atoms by adding bonds between atoms such that each carbon atom is in four bonds, each hydrogen atom is in one bond, and no sequence of bonds forms a cycle of atoms. Prove that $l = 2k + 2$. (Bondy–Murty [1976, p27])

2.1.21. Let G be an n-vertex simple graph having a decomposition into k spanning trees. Suppose also that $\Delta(G) = \delta(G) + 1$. For $2k \geq n$, show that this is impossible. For $2k < n$, determine the degree sequence of G in terms of n and k.

2.1.22. Let T be an n-vertex tree having one vertex of each degree i with $2 \leq i \leq k$; the remaining $n - k + 1$ vertices are leaves. Determine n in terms of k.

2.1.23. Let T be a tree in which every vertex has degree 1 or degree k. Determine the possible values of $n(T)$.

2.1.24. Prove that every nontrivial tree has at least two maximal independent sets, with equality only for stars. (Note: maximal ≠ maximum.)

2.1.25. Prove that among trees with n vertices, the star has the most independent sets.

2.1.26. (!) For $n \geq 3$, let G be an n-vertex graph such that every graph obtained by deleting one vertex is a tree. Determine $e(G)$, and use this to determine G itself.

2.1.27. (!) Let $d_1, \ldots, d_n$ be positive integers, with $n \geq 2$. Prove that there exists a tree with vertex degrees $d_1, \ldots, d_n$ if and only if $\sum d_i = 2n - 2$.

2.1.28. Let $d_1 \geq \cdots \geq d_n$ be nonnegative integers. Prove that there exists a connected graph (loops and multiple edges allowed) with degree sequence $d_1, \ldots, d_n$ if and only if $\sum d_i$ is even, $d_n \geq 1$, and $\sum d_i \geq 2n - 2$. (Hint: Consider a realization with the fewest components.) Is the statement true for simple graphs?

2.1.29. (!) Every tree is bipartite. Prove that every tree has a leaf in its larger partite set (in both if they have equal size).

2.1.30. Let T be a tree in which all vertices adjacent to leaves have degree at least 3. Prove that T has some pair of leaves with a common neighbor.

2.1.31. Prove that a simple connected graph having exactly two vertices that are not cut-vertices is a path.

2.1.32. Prove that an edge e of a connected graph G is a cut-edge if and only if e belongs to every spanning tree. Prove that e is a loop if and only if e belongs to no spanning tree.

2.1.33. (!) Let G be a connected n-vertex graph. Prove that G has exactly one cycle if and only if G has exactly n edges.

2.1.34. (!) Let T be a tree with k edges, and let G be a simple n-vertex graph with more than $n(k-1) - \binom{k}{2}$ edges. Use Proposition 2.1.8 to prove that $T \subseteq G$ if $n > k$.

2.1.35. (!) Let T be a tree. Prove that the vertices of T all have odd degree if and only if for all $e \in E(T)$, both components of $T - e$ have odd order.

2.1.36. (!) Let T be a tree of even order. Prove that T has exactly one spanning subgraph in which every vertex has odd degree.

2.1.37. (!) Let T, T' be two spanning trees of a connected graph G. For $e \in E(T) - E(T')$, prove that there is an edge $e' \in E(T') - E(T)$ such that $T' + e - e'$ and $T - e + e'$ are both spanning trees of G.

2.1.38. Let T, T' be two trees on the same vertex set such that $d_T(v) = d_{T'}(v)$ for each vertex v. Prove that T' can be obtained from T using 2-switches (Definition 1.3.32) so that every graph along the way is also a tree. (Kelmans [1998])

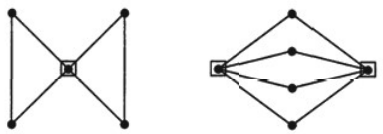
2.1.39. (!) Let G be a tree with $2k$ vertices of odd degree. Prove that G decomposes into k paths. (Hint: Prove the stronger result that the claim holds for all forests.)

2.1.40. (!) Let G be a tree with k leaves. Prove that G is the union of paths $P_1, \ldots, P_{\lceil k/2 \rceil}$ such that $P_i \cap P_j \neq \emptyset$ for all $i \neq j$. (Ando–Kaneko–Gervacio [1996])

2.1.41. For $n \geq 4$, let G be a simple n-vertex graph with $e(G) \geq 2n - 3$. Prove that G has two cycles of equal length. (Chen–Jacobson–Lehel–Shreve [1999] strengthens this.)

2.1.42. Let G be a connected Eulerian graph with at least three vertices. A vertex v in G is *extendible* if every trail beginning at v can be extended to form an Eulerian circuit of G. For example, in the graphs below only the marked vertices are extendible. Prove the following statements about G (adapted from Chartrand–Lesniak [1986, p61]).

a) A vertex $v \in V(G)$ is extendible if and only if $G - v$ is a forest. (Ore [1951])
b) If v is extendible, then $d(v) = \Delta(G)$. (Bäbler [1953])
c) All vertices of G are extendible if and only if G is a cycle.
d) If G is not a cycle, then G has at most two extendible vertices.



2.1.43. Let u be a vertex in a connected graph G. Prove that it is possible to select shortest paths from u to all other vertices of G so that the union of the paths is a tree.

2.1.44. (!) Prove or disprove: If a simple graph with diameter 2 has a cut-vertex, then its complement has an isolated vertex.

2.1.45. Let G be a graph having spanning trees with diameter 2 and diameter l. For $2 < k < l$, prove that G also has a spanning tree with diameter k. (Galvin)

2.1.46. (!) Prove that the trees with diameter 3 are the **double-stars** (two central vertices plus leaves). Count the isomorphism classes of double-stars with n vertices.

2.1.47. (!) *Diameter and radius.*
a) Prove that the distance function $d(u, v)$ on pairs of vertices of a graph satisfies the triangle inequality: $d(u, v) + d(v, w) \geq d(u, w)$.
b) Use part (a) to prove that $\operatorname{diam} G \leq 2 \operatorname{rad} G$ for every graph G.
c) For all positive integers r and d that satisfy $r \leq d \leq 2r$, construct a simple graph with radius r and diameter d. (Hint: Build a suitable graph with one cycle.)

2.1.48. (!) For $n \geq 4$, prove that the minimum number of edges in an n-vertex graph with diameter 2 and maximum degree $n-2$ is $2n-4$.

2.1.49. Let G be a simple graph. Prove that $\operatorname{rad} G \geq 3 \Rightarrow \operatorname{rad} \overline{G} \leq 2$.

2.1.50. *Radius and eccentricity.*
a) Prove that the eccentricities of adjacent vertices differ by at most 1.
b) In terms of the radius r, determine the maximum possible distance from a vertex of eccentricity $r+1$ to the center of G. (Hint: Use a graph with one cycle.)

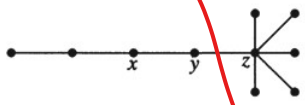
2.1.51. Let x and y be distinct neighbors of a vertex v in a graph G.
a) Prove that if G is a tree, then $2\epsilon(v) \leq \epsilon(x) + \epsilon(y)$.
b) Determine the smallest graph where this inequality can fail.

2.1.52. Let x be a vertex in a graph G, and suppose that $\epsilon(x) > \operatorname{rad} G$.
a) Prove that if G is a tree, then x has a neighbor with eccentricity $k-1$.
b) Show that part (a) does not hold for all graphs by constructing, for each even r that is at least 4, a graph with radius r in which x has eccentricity $r+2$ and has no neighbor with eccentricity $r+1$. (Hint: Use a graph with one cycle.)

2.1.53. Prove that the center of a graph can be disconnected and can have components arbitrarily far apart by constructing a graph where the center consists of two vertices and the distance between these two vertices is k.

2.1.54. *Centers of trees.* Let T be a tree.
a) Give a noninductive proof that the center of T is a vertex or an edge.
b) Prove that the center of T is one vertex if and only if $\operatorname{diam} T = 2 \operatorname{rad} T$.
c) Use part (a) to prove that if $n(T)$ is odd, then every automorphism of T maps some vertex to itself.

2.1.55. Given $x \in V(G)$, let $s(x) = \sum_{v \in V(G)} d(x, v)$. The **barycenter** of G is the subgraph induced by the set of vertices minimizing $s(x)$ (the set is also called the **median**).
a) Prove that the barycenter of a tree is a single vertex or an edge. (Hint: Study $s(u) - s(v)$ when u and v are adjacent.) (Jordan [1869])
b) Determine the maximum distance between the center and the barycenter in a tree of diameter d. (Example: in the tree below, the center is the edge xy, the barycenter contains only z, and the distance between them is 1.)



2.1.56. Let T be a tree. Prove that T has a vertex v such that for all $e \in E(T)$, the component of $T - e$ containing v has at least $\lceil n(T)/2 \rceil$ vertices. Prove that either v is unique or there are just two adjacent such vertices.

2.1.57. Let $n_1, \ldots, n_k$ be positive integers with sum $n-1$.
a) By counting edges in complete graphs, prove that $\sum_{i=1}^{k} \binom{n_i}{2} \leq \binom{n-1}{2}$.
b) Use part (a) to prove that $\sum_{v \in V(T)} d(u, v) \leq \binom{n}{2}$ when u is a vertex of a tree T. (Hint: Use strong induction on the number of vertices.)

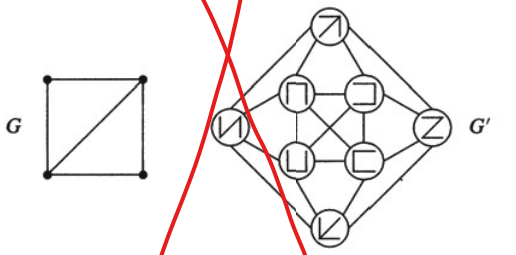
2.1.58. (+) Let S and T be trees with leaves $\{x_1, \ldots, x_k\}$ and $\{y_1, \ldots, y_k\}$, respectively. Suppose that $d_S(x_i, x_j) = d_T(y_i, y_j)$ for each pair i, j. Prove that S and T are isomorphic. (Smolenskii [1962])

2.1.59. (!) Let G be a tree with n vertices, k leaves, and maximum degree k.
a) Prove that G is the union of k paths with a common endpoint.
b) Determine the maximum and minimum possible values of $\operatorname{diam} G$.

2.1.60. Let G be a graph with diameter d and maximum degree k. Prove that $n(G) \leq 1 + [(k-1)^d - 1]k/(k-2)$. (Comment: Equality holds for the Petersen graph.)

2.1.61. (+) Let G be a graph with smallest order among k-regular graphs with girth at least g (Exercise 1.3.16 establishes the existence of such graphs). Prove that G has diameter at most g. (Hint: If $d_G(x, y) > g$, modify G to obtain a smaller k-regular graph with girth at least g.) (Erdős–Sachs [1963])

2.1.62. (!) Let G be a connected graph with n vertices. Define a new graph G' having one vertex for each spanning tree of G, with vertices adjacent in G' if and only if the corresponding trees have exactly $n(G) - 2$ common edges. Prove that G' is connected. Determine the diameter of G'. An example appears below.



2.1.63. (!) Prove that every n-vertex graph with $n+1$ edges has girth at most $\lfloor (2n+2)/3 \rfloor$. For each n, construct an example achieving this bound.

2.1.64. (!) Let G be a connected graph that is not a tree. Prove that some cycle in G has length at most $2(\operatorname{diam} G) + 1$. For each $k \in \mathbb{N}$, show that this is best possible by exhibiting a graph with diameter k and girth $2k+1$.

2.1.65. (+) Let G be a connected n-vertex graph with minimum degree k, where $k \geq 2$ and $n - 2 \geq 2(k+1)$. Prove that $\operatorname{diam} G \leq 3(n-2)/(k+1) - 1$. Whenever $k \geq 2$ and $(n-2)/(k+1)$ is an integer greater than 1, construct a graph where the bound holds with equality. (Moon [1965b])

2.1.66. Let $F_1, \ldots, F_m$ be forests whose union is G. Prove that $m \geq \max_{H \subseteq G} \left\lceil \frac{e(H)}{n(H)-1} \right\rceil$. (Comment: Nash-Williams [1964] and Edmonds [1965b] proved that this bound is always achieveable—Corollary 8.2.57).

2.1.67. Prove that the following is a necessary condition for the existence of k pairwise edge-disjoint spanning trees in G: for every partition of the vertices of G into r sets, there are at least $k(r-1)$ edges of G whose endpoints are in different sets of the partition. (Comment: Corollary 8.2.59 shows that this condition is also sufficient - Tutte [1961a], Nash-Williams [1961], Edmonds [1965c].)

2.1.68. Can the graph below be decomposed into edge-disjoint spanning trees? Into isomorphic edge-disjoint spanning trees?

2.1.69. (∗) Consider the graph before Theorem 2.1.17 with 12 vertical edges and 16 edges that are horizontal or slanted. Let $g_{i,j}$ be the ith edge from the top in the jth column of vertical edges. Let $h_{i,j}$ be the jth edge from the left in the ith row of horizontal/diagonal edges. Suppose that Player 1 follows the strategy of Theorem 2.1.17 and first takes $h_{1,1}$. Player 2 deletes $g_{2,2}$, and Player 1 takes $h_{2,3}$. Next Player 2 deletes $v_{3,2}$, and Player 1 takes $h_{4,2}$. Draw the two spanning trees at this point. Given that Player 2 next deletes $g_{2,1}$, list all moves available to Player 1 within the strategy. (Pritikin)

2.1.70. (∗) Prove that Bridg-it cannot end in a tie no matter how the moves are made. That is, prove that when no further moves can be made, one of the players must have built a path connecting his/her goals.

2.1.71. (∗) The players change the rules of Bridg-it so that a player with path between friendly ends is the *loser*. It is forbidden to stall by building a bridge joining end posts or joining posts already connected by a path. Show that Player 2 has a strategy that forces Player 1 to lose. (Hint: Use Proposition 2.1.7 instead of Proposition 2.1.6.) (Pritikin)

2.1.72. (+) Prove that if $G_1, \ldots, G_k$ are pairwise-intersecting subtrees of a tree G, then G has a vertex that belongs to all of $G_1, \ldots, G_k$. (Hint: Use induction on k. Comment: This result is the **Helly property** for trees.)

2.1.73. (+) Prove that a simple graph G is a forest if and only if for every pairwise intersecting family of paths in G, the paths have a common vertex. (Hint: For sufficiency, use induction on the size of the family of paths.)

2.1.74. Let G be a simple n-vertex graph having $n-2$ edges. Prove that G has an isolated vertex or has two components that are nontrivial trees. Use this to prove inductively that G is a subgraph of $\overline{G}$. (Comment: The claim is not true for all graphs with $n-1$ edges.) (Burns–Schuster [1977])

2.1.75. (+) Prove that every n-vertex *tree other* than $K_{1,n-1}$ is contained in its complement. (Hint: Use induction on n to prove a stronger statement: if T is an n-vertex tree other than a star, then K_n contains two edge-disjoint copies of T in which the two copies of each non-leaf vertex of T appear at distinct vertices.) (Burns–Schuster [1978])

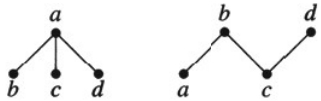
2.1.76. (+) Let S be an n-element set, and let $\{A_1, \ldots, A_n\}$ be n distinct subsets of S. Prove that S has an element x such that the sets $A_1 \cup \{x\}, \ldots, A_n \cup \{x\}$ are distinct. (Hint: Define a graph with vertices $a_1, \ldots, a_n$ such that $a_i \leftrightarrow a_j$ if and only if one of $\{A_i, A_j\}$ is obtained from the other by adding a single element y. Use y as a label on the edge. Prove that there is a forest consisting of one edge with each label used. Use this to obtain the desired x.) (Bondy [1972a])

2.2. Spanning Trees and Enumeration

There are $2^{\binom{n}{2}}$ simple graphs with vertex set $[n] = \{1, \ldots, n\}$, since each pair may or may not form an edge. How many of these are trees? In this section, we solve this counting problem, count spanning trees in arbitrary graphs, and discuss several applications.

ENUMERATION OF TREES

With one or two vertices, only one tree can be formed. With three vertices there is still only one isomorphism class, but the adjacency matrix is determined by which vertex is the center. Thus there are three trees with vertex set [3]. With vertex set [4], there are four stars and 12 paths, yielding 16 trees. With vertex set [5], a careful study yields 125 trees.



Now we may see a pattern. With vertex set $[n]$, there are n^{n-2} trees; this is **Cayley's Formula**. Prüfer, Kirchhoff, Pólya, Renyi, and others found proofs. J.W. Moon [1970] wrote a book about enumerating classes of trees. We present a bijective proof, establishing a one-to-one correspondence between the set of trees with vertex set $[n]$ and a set of known size.

Given a set S of n numbers, there are exactly n^{n-2} ways to form a list of length $n-2$ with entries in S. The set of lists is denoted S^{n-2} (see Appendix A). We use S^{n-2} to encode the trees with vertex set S. The list that results from a tree is its **Prüfer code**.

2.2.1. Algorithm. (Prüfer code) Production of $f(T) = (a_1, \ldots, a_{n-2})$.
Input: A tree T with vertex set $S \subseteq \mathbb{N}$.
Iteration: At the ith step, delete the least remaining leaf, and let a_i be the *neighbor* of this leaf. ■

2.2.2. Example. After $n-2$ iterations, only one of the original $n-1$ edges remains, and we have produced a list $f(T)$ of length $n-2$ with entries in S. In the tree below, the least leaf is 2; we delete it and record 7. After deleting 3 and 5 and recording 4 each time, the least leaf in the remaining 5-vertex tree is 4. The full code is (744171), and the vertices remaining at the end are 1 and 8. After the first step, the remainder of the Prüfer code is the Prüfer code of the subtree T' with vertex set $[8] - \{2\}$.

EXERCISES

2.2.1. (−) Determine which trees have Prüfer codes that (a) contain only one value, (b) contain exactly two values, or (c) have distinct values in all positions.

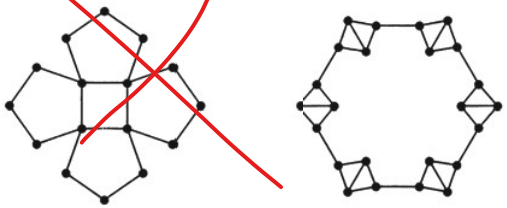
2.2.2. (−) Count the spanning trees in the graph on the left below. (Proposition 2.2.8 provides a systematic approach, and then Remark 2.2.10 and Example 2.2.6 can be used to shorten the computation.)

2.2.3. (−) Let G be the graph on the right above. Use the Matrix Tree Theorem to find a matrix whose determinant is $\tau(G)$. Compute $\tau(G)$.

2.2.4. (−) Let G be a simple graph with m edges. Prove that if G has a graceful labeling, then K_{2m+1} decomposes into copies of G. (Hint: Follow the proof of Theorem 2.2.16.)

• • • • •

2.2.5. The graph on the left below was the logo of the 9th Quadrennial International Conference in Graph Theory, held in Kalamazoo in 2000. Count its spanning trees.



2.2.6. (!) Let G be the 3-regular graph with $4m$ vertices formed from m pairwise disjoint kites by adding m edges to link them in a ring, as shown on the right above for $m = 6$. Prove that $\tau(G) = 2m8^m$

2.2.7. (!) Use Cayley's Formula to prove that the graph obtained from K_n by deleting an edge has $(n-2)n^{n-3}$ spanning trees.

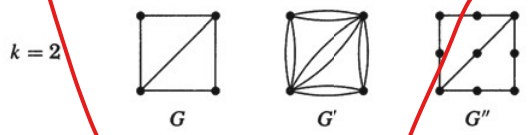
2.2.8. Count the following sets of trees with vertex set $[n]$, giving two proofs for each: one using the Prüfer correspondence and one by direct counting arguments.
a) trees that have 2 leaves.
b) trees that have $n-2$ leaves.

2.2.9. Let $S(m,r)$ denote the number of partitions of an m-element set into r nonempty subsets. In terms of these numbers, count the trees with vertex set $\{v_1,\dots,v_n\}$ that have exactly k leaves. (Rényi [1959])

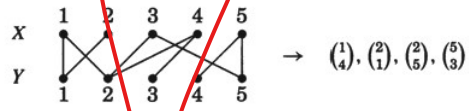
2.2.10. Compute $\tau(K_{2,m})$. Also compute the number of isomorphism classes of spanning trees of $K_{2,m}$.

2.2.11. (+) Compute $\tau(K_{3,m})$.

2.2.12. From a graph G we define two new graphs. Let G' be the graph obtained by replacing each edge of G with k copies of that edge. Let G'' be the graph obtained by replacing each edge $uv \in E(G)$ with a u,v-path of length k through $k-1$ new vertices. Determine $\tau(G')$ and $\tau(G'')$ in terms of $\tau(G)$ and k.



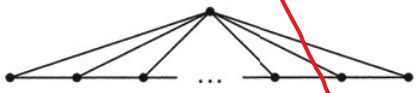
2.2.13. Consider $K_{n,n}$ with bipartition X, Y, where $X = \{x_1,\dots,x_n\}$ and $Y = \{y_1,\dots,y_n\}$. For each spanning tree T, we form a list $f(T)$ of ordered pairs (written vertically). Having generated part of the list, let u be the least-indexed leaf in X in the remaining subtree, and similarly let v be the least-indexed leaf in Y. Append the pair $\binom{a}{b}$ to the list, where a is the index of the neighbor of u and b is the index of the neighbor of v. Delete $\{u,v\}$. Iterate until $n-1$ pairs have been generated to form $f(T)$ (one edge remains). Part (a) shows that f is well-defined.
a) Prove that every spanning tree of $K_{n,n}$ has a leaf in each partite set.
b) Prove that f is a bijection from the set of spanning trees of $K_{n,n}$ to $([n]\times[n])^{n-1}$
Thus $K_{n,n}$ has n^{2n-2} spanning trees. (Rényi [1966], Kelmans [1992], Pritikin [1995])



2.2.14. (+) Let $f(r,s)$ be the number of trees with vertex $[n]$ that have partite sets of sizes r and s (with $r+s=n$). Prove that $f(r,s) = \binom{r+s}{s}s^{r-1}r^{s-1}$ if $r \neq s$. What is the formula when $r = s$? (Hint: First show that the Prüfer sequence for such a tree will have $r-1$ of its terms from the partite set of size s and $s-1$ of its terms from the partite set of size r.) (Scoins [1962], Glicksman [1963])

2.2.15. Let G_n be the graph with $2n$ vertices and $3n-2$ edges pictured below, for $n \geq 1$. Prove for $n > 2$ that $\tau(G_n) = 4\tau(G_{n-1}) - \tau(G_{n-2})$. (Kelmans [1967a])

2.2.16. For $n \geq 1$, let a_n be the number of spanning trees in the graph formed from P_n by adding one vertex adjacent to all of $V(P_n)$. For example, $a_1 = 1$, $a_2 = 3$, and $a_3 = 8$. Prove for $n > 1$ that $a_n = a_{n-1} + 1 + \sum_{i=1}^{n-1} a_i$. Use this to prove for $n > 2$ that $a_n = 3a_{n-1} - a_{n-2}$. (Comment: It is also possible to argue directly that $a_n = 3a_{n-1} - a_{n-2}$.)



2.2.17. Use the Matrix Tree Theorem to prove Cayley's Formula.

2.2.18. Use the Matrix Tree Theorem to compute $\tau(K_{r,s})$. (Lovász [1979, p223]—see Kelmans [1965] for a generalization)

2.2.19. (+) Prove combinatorially that the number t_n of trees with vertex set $[n]$ satisfies the recurrence $t_n = \sum_{k=1}^{n-1} k\binom{n-2}{k-1} t_k t_{n-k}$. (Comment: Since $t_n = n^{n-2}$, this proves the identity $n^{n-2} = \sum_{k=1}^{n-1} \binom{n-2}{k-1} k^{k-1}(n-k)^{n-k-2}$.) (Dzioḃek [1917]; see Lovász [1979, p219])

2.2.20. (!) Prove that a d-regular simple graph G has a decomposition into copies of $K_{1,d}$ if and only if it is bipartite.

2.2.21. (+) Prove that $K_{2m-1,2m}$ decomposes into m spanning paths.

2.2.22. Let G be an n-vertex simple graph that decomposes into k spanning trees. Given also that $\Delta(G) = \delta(G) + 1$, determine the degree sequence of G in terms of n and k.

2.2.23. (!) Prove that if the Graceful Tree Conjecture is true and T is a tree with m edges, then K_{2m} decomposes into $2m-1$ copies of T. (Hint: Apply the cyclically invariant decomposition of K_{2m-1} for trees with $m-1$ edges from the proof of Theorem 2.2.16.)

2.2.24. Of the n^{n-2} trees with vertex set $\{0, \ldots, n-1\}$, how many are gracefully labeled by their vertex names?

2.2.25. (!) Prove that if a graph G is graceful and Eulerian, then $e(G)$ is congruent to 0 or 3 mod 4. (Hint: Sum the absolute edge differences (mod 2) in two different ways.)

2.2.26. (+) Prove that C_n is graceful if and only if 4 divides n or $n+1$. (Frucht [1979])

2.2.27. (+) Let G be the graph consisting of k 4-cycles with one common vertex. Prove that G is graceful. (Hint: Put 0 at the vertex of degree $2k$.)

2.2.28. Let $d_1, \ldots, d_n$ be positive integers. Prove directly that there exists a caterpillar with vertex degrees $d_1, \ldots, d_n$ if and only if $\sum d_i = 2n-2$.

2.2.29. Prove that every tree can be turned into a caterpillar with the same degree sequence using 2-switches (Definition 1.3.32) such that each intermediate graph is a tree.

2.2.30. A bipartite graph is *drawn on a channel* if the vertices of one partite set are placed on one line in the plane (in some order) and the vertices of the other partite set are placed on a line parallel to it and the edges are drawn as straight-line segments between them. Prove that a connected graph G can be drawn on a channel without edge crossings if and only if G is a caterpillar.

2.2.31. (!) An *up/down labeling* is a graceful labeling for which there exists a *critical value* α such that every edge joins vertices with labels above and below α. Prove that every caterpillar has an up/down labeling. Prove that the 7-vertex tree that is not a caterpillar has no up/down-labeling.

2.2.32. (+) Prove that the number of isomorphism classes of n-vertex caterpillars is $2^{n-4} + 2^{\lfloor n/2 \rfloor - 2}$ if $n \geq 3$. (Harary–Schwenk [1973], Kimble–Schwenk [1981])

2.2.33. (!) Let T be an orientation of a tree such that the heads of the edges are all distinct; the one vertex that is not a head is the *root*. Prove that T is a union of paths from the root. Prove that for each vertex of T, exactly one path reaches it from the root.

2.2.34. (*) Use Theorem 2.2.26 to prove that the algorithm below generates a binary deBruijn cycle of length 2^n (the cycle in Application 1.4.25 arises in this way).

Start with n 0's. Subsequently, append a 1 if doing so does not repeat a previous string of length n, otherwise append a 0.

2.2.35. (*) *Tarry's Algorithm* (as presented by D.G. Hoffman). Consider a castle with finitely many rooms and corridors. Each corridor has two ends; each end has a door into a room. Each room has door(s), each of which leads to a corridor. Each room can be reached from any other by traversing corridors and rooms. Initially, no doors have marks. A robot started in some room will explore the castle using the following rules.

1) After entering a corridor, traverse it and enter the room at the other end.
2) Upon entering a room with all doors unmarked, mark I on the door of entry.
3) In a room with an unmarked door, mark O on such a door and use it.
4) In a room with all doors marked, exit via a door not marked O if one exists.
5) In a room with all doors marked O, stop.

Prove that the robot traverses each corridor exactly twice, once in each direction, and then stops. (Hint: Prove that this holds for the corridors at every reached vertex, and prove that every vertex is reached. Comment: All decisions are completely local; the robot sees nothing other than the current room or corridor. Tarry's Algorithm [1895] and others are described by König [1936, p35–56] and by Fleischner [1983, 1991].)

2.3. Optimization and Trees

"The best spanning tree" may have various meanings. A **weighted graph** is a graph with numerical labels on the edges. When building links to connect locations, the costs of potential links yield a weighted graph. The minimum cost of connecting the system is the minimum total weight of its spanning trees.

Alternatively, the weights may represent distances. In these case we define the length of a path to be the sum of its edge weights. We may seek a spanning tree with small distances. When discussing weighted graphs, **we consider only nonnegative edge weights**.

We also study a problem about finding good trees to encode messages.

MINIMUM SPANNING TREE

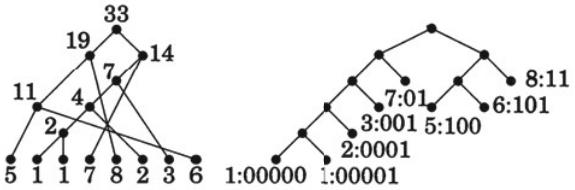
In a connected weighted graph of possible communication links, all spanning trees have $n-1$ edges; we seek one that minimizes or maximizes the sum of the edge weights. For these problems, the most naive heuristic quickly produces an optimal solution.

2.3.1. Algorithm. (Kruskal's Algorithm - for minimum spanning trees.)
Input: A weighted connected graph.
Idea: Maintain an acyclic spanning subgraph H, enlarging it by edges with low weight to form a spanning tree. Consider edges in nondecreasing order of weight, breaking ties arbitrarily.
Initialization: Set $E(H) = \emptyset$.
Iteration: If the next cheapest edge joins two components of H, then include it; otherwise, discard it. Terminate when H is connected. ■

Recursion: When $n > 2$, replace the two least likely items p, p' with a single item q of weight $p + p'$. Treat the smaller set as a problem with $n - 1$ items. After solving it, give children with weights p, p' to the resulting leaf with weight q. Equivalently, replace the code computed for the combined item with its extensions by 1 and 0, assigned to the items that were replaced. ■

2.3.14. Example. *Huffman coding.* Consider eight items with frequencies 5, 1, 1, 7, 8, 2, 3, 6. Algorithm 2.3.13 combines items according to the tree on the left below, working from the bottom up. First the two items of weight 1 combine to form one of weight 2. Now this and the original item of weight 2 are the least likely and combine to form an item of weight 4. The 3 and 4 now combine, after which the least likely elements are the original items of weights 5 and 6. The remaining combinations in order are $5 + 6 = 11$, $7 + 7 = 14$, $8 + 11 = 19$, and $14 + 19 = 33$.

From the drawing of this tree on the right, we obtain code words. In their original order, the items have code words 100, 00000, 00001, 01, 11, 0001, 001, and 101. The expected length is $\sum p_i l_i = 90/33$. This is less than 3, which would be the expected length of a code using the eight words of length 3. ■



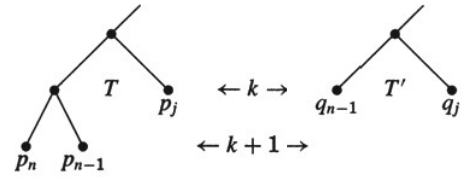
2.3.15. Theorem. Given a probability distribution $\{p_i\}$ on n items, Huffman's Algorithm produces the prefix-free code with minimum expected length.

Proof: We use induction on n. Basis step: $n = 2$. We must send a bit to send a message, and the algorithm encodes each item as a single bit, so the optimum is expected length 1.

Induction step: $n > 2$. Suppose that the algorithm computes the optimal code when given a distribution for $n - 1$ items. Every code assigns items to the leaves of a binary tree. Given a fixed tree with n leaves, we minimize the expected length by greedily assigning the messages with probabilities $p_1 \geq \cdots \geq p_n$ to leaves in increasing order of depth. Thus every optimal code has least likely messages assigned to leaves of greatest depth. Since every leaf at maximum depth has another leaf as its sibling and permuting the items at a given depth does not change the expected length, we may assume that two least likely messages appear as siblings at greatest depth.

Let T be an optimal tree for $p_1, \ldots, p_n$, with the least likely items p_n and p_{n-1} located as sibling leaves at greatest depth. Let T' be the tree obtained from T by deleting these leaves, and let $q_1, \ldots, q_{n-1}$ be the probability distribution obtained by replacing $\{p_{n-1}, p_n\}$ by $q_{n-1} = p_{n-1} + p_n$. The tree T' yields a code for $\{q_i\}$. The expected length for T is the expected length for T' plus q_{n-1}, since if k is the depth of the leaf assigned q_{n-1}, we lose kq_{n-1} and gain $(k+1)(p_{n-1} + p_n)$ in moving from T' to T.

This holds for each choice of T', so it is best to use the tree T' that is optimal for $\{q_i\}$. By the induction hypothesis, the optimal choice for T' is obtained by applying Huffman's Algorithm to $\{q_i\}$. Since the replacement of $\{p_{n-1}, p_n\}$ by q_{n-1} is the first step of Huffman's Algorithm for $\{p_i\}$, we conclude that Huffman's Algorithm generates the optimal tree T for $\{p_i\}$. ■



Huffman's Algorithm computes an optimal prefix-free code, and its expected length is close to the optimum over all types of binary codes. Shannon [1948] proved that for every code with binary digits, the expected length is at least the **entropy** of the discrete probability distribution $\{p_i\}$, defined to be $-\sum p_i \lg p_i$ (Exercise 31). When each p_i is a power of 1/2, the Huffman code meets this bound exactly (Exercise 30).

EXERCISES

2.3.1. (–) Assign integer weights to the edges of K_n. Prove that the total weight on every cycle is even if and only if the total weight on every triangle is even.

2.3.2. (–) Prove or disprove: If T is a minimum-weight spanning tree of a weighted graph G, then the u, v-path in T is a minimum-weight u, v-path in G.

2.3.3. (–) There are five cities in a network. The cost of building a road directly between i and j is the entry $a_{i,j}$ in the matrix below. An infinite entry indicates that there is a mountain in the way and the road cannot be built. Determine the least cost of making all the cities reachable from each other.

$$\begin{pmatrix} 0 & 3 & 5 & 11 & 9 \\ 3 & 0 & 3 & 9 & 8 \\ 5 & 3 & 0 & \infty & 10 \\ 11 & 9 & \infty & 0 & 7 \\ 9 & 8 & 10 & 7 & 0 \end{pmatrix}$$

2.3.4. (–) In the graph below, assign weights (1, 1, 2, 2, 3, 3, 4, 4) to the edges in two ways: one way so that the minimum-weight spanning tree is unique, and another way so that the minimum-weight spanning tree is not unique.

2.3.5. (–) There are five cities in a network. The travel time for traveling directly from i to j is the entry $a_{i,j}$ in the matrix below. The matrix is not symmetric (use directed

graphs), and $a_{i,j} = \infty$ indicates that there is no direct route. Determine the least travel time and quickest route from i to j for each pair i, j.

$$\begin{pmatrix} 0 & 10 & 20 & \infty & 17 \\ 7 & 0 & 5 & 22 & 33 \\ 14 & 13 & 0 & 15 & 27 \\ 30 & \infty & 17 & 0 & 10 \\ \infty & 15 & 12 & 8 & 0 \end{pmatrix}$$

• • • • •

2.3.6. (!) Assign integer weights to the edges of K_n. Prove that on every cycle the total weight is even if and only if the subgraph consisting of the edges with odd weight is a spanning complete bipartite subgraph. (Hint: Show that every component of the subgraph consisting of the edges with even weight is a complete graph.)

2.3.7. Let G be a weighted connected graph with distinct edge weights. Without using Kruskal's Algorithm, prove that G has only one minimum-weight spanning tree. (Hint: Use Exercise 2.1.34.)

2.3.8. Let G be a weighted connected graph. Prove that no matter how ties are broken in choosing the next edge for Kruskal's Algorithm, the list of weights of a minimum spanning tree (in nondecreasing order) is unique.

2.3.9. Let F be a spanning forest of a connected weighted graph G. Among all edges of G having endpoints in different components of F, let e be one of minimum weight. Prove that among all the spanning trees of G that contain F, there is one of minimum weight that contains e. Use this to give another proof that Kruskal's Algorithm works.

2.3.10. (!) **Prim's Algorithm** grows a spanning tree from a given vertex of a connected weighted graph G, iteratively adding the cheapest edge from a vertex already reached to a vertex not yet reached, finishing when all the vertices of G have been reached. (Ties are broken arbitrarily.) Prove that Prim's Algorithm produces a minimum-weight spanning tree of G. (Jarník [1930], Prim [1957], Dijkstra [1959], independently).

2.3.11. For a spanning tree T in a weighted graph, let $m(T)$ denote the maximum among the weights of the edges in T. Let x denote the minimum of $m(T)$ over all spanning trees of a weighted graph G. Prove that if T is a spanning tree in G with minimum total weight, then $m(T) = x$ (in other words, T also minimizes the maximum weight). Construct an example to show that the converse is false. (Comment: A tree that minimizes the maximum weight is called a **bottleneck** or **minimax** spanning tree.)

2.3.12. In a weighted complete graph, iteratively select the edge of least weight such that the edges selected so far form a disjoint union of paths. After $n - 1$ steps, the result is a spanning path. Prove that this algorithm always gives a minimum-weight spanning path, or give an infinite family of counterexamples where it fails.

2.3.13. (!) Let T be a minimum-weight spanning tree in G, and let T' be another spanning tree in G. Prove that T' can be transformed into T by a list of steps that exchange one edge of T' for one edge of T, such that the edge set is always a spanning tree and the total weight never increases.

2.3.14. (!) Let C be a cycle in a connected weighted graph. Let e be an edge of maximum weight on C. Prove that there is a minimum spanning tree not containing e. Use this to prove that iteratively deleting a heaviest non-cut-edge until the remaining graph is acyclic produces a minimum-weight spanning tree.

2.3.15. Let T be a minimum-weight spanning tree in a weighted connected graph G. Prove that T omits some heaviest edge from every cycle in G.

2.3.16. Four people must cross a canyon at night on a fragile bridge. At most two people can be on the bridge at once. Crossing requires carrying a flashlight, and there is only one flashlight (which can cross only by being carried). Alone, the four people cross in 10, 5, 2, 1 minutes, respectively. When two cross together, they move at the speed of the slower person. In 18 minutes, a flash flood coming down the canyon will wash away the bridge. Can the four people get across in time? Prove your answer without using graph theory and describe how the answer can be found using graph theory.

2.3.17. Given a starting vertex u in an unweighted graph or digraph G, prove directly (without Dijkstra's Algorithm) that Algorithm 2.3.8 computes $d(u, z)$ for all $z \in V(G)$.

2.3.18. Explain how to use Breadth-First Search to compute the girth of a graph.

2.3.19. (+) Prove that the following algorithm correctly finds the diameter of a tree. First, run BFS from an arbitrary vertex w to find a vertex u at maximum distance from w. Next, run BFS from u to reach a vertex v at maximum distance from u. Report diam $T = d(u, v)$. (Cormen–Leiserson–Rivest [1990, p476])

2.3.20. *Minimum diameter spanning tree.* An MDST is a spanning tree where the maximum length of a path is as small as possible. Intuition suggests that running Dijkstra's Algorithm from a vertex of minimum eccentricity (a center) will produce an MDST, but this may fail.

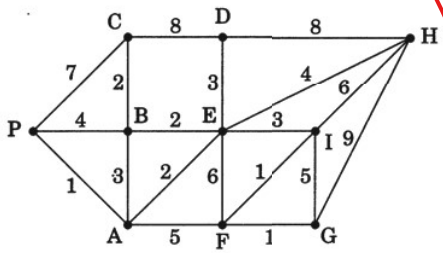
a) Construct a 5-vertex example of an unweighted graph (edge weights all equal 1) such that Dijkstra's Algorithm can be run from some vertex of minimum eccentricity and produce a spanning tree that does not have minimum diameter.

b) Construct a 4-vertex example of a weighted graph such that Dijkstra's algorithm cannot produce an MDST when run from any vertex.

2.3.21. Develop a fast algorithm to test whether a graph is bipartite. The graph is given by its adjacency matrix or by lists of vertices and their neighbors. The algorithm should not need to consider an edge more than twice.

2.3.22. (−) Solve the Chinese Postman Problem in the k-dimensional cube Q_k under the condition that every edge has weight 1.

2.3.23. Every morning the Lazy Postman takes the bus to the Post Office. From there, he chooses a route to reach home as quickly as possible (**NOT** ending at the Post Office). Below is a map of the streets along which he must deliver mail, giving the number of minutes required to walk each block whether delivering or not. P denotes the post office and H denotes home. What must the edges traveled more than once satisfy? How many times will each edge be traversed in the optimal route?



2.3.24. (−) Explain why the optimal trails pairing up odd vertices in an optimal solution to the Chinese Postman Problem may be assumed to be paths. Construct a weighted graph with four odd vertices where the optimal solution to the Chinese Postman Problem requires duplicating the edges on two paths that have a common vertex.

2.3.25. Let G be a rooted tree where every vertex has 0 or k children. Given k, for what values of $n(G)$ is this possible?

2.3.26. Find a recurrence relation to count the binary trees with $n+1$ leaves (here each non-leaf vertex has exactly two children, and the left-to-right order of children matters). When $n = 2$, the possibilities are the two trees below.

2.3.27. Find a recurrence relation for the number of rooted plane trees with n vertices. (As in a rooted binary tree, the subtrees obtained by deleting the root of a rooted plane tree are distinguished by their order from left to right.)

2.3.28. (−) Compute a code with minimum expected length for a set of ten messages whose relative frequencies are 1, 2, 3, 4, 5, 5, 6, 7, 8, 9. What is the expected length of a message in this optimal code?

2.3.29. (−) The game of *Scrabble* has 100 tiles as listed below. This does not agree with English; "S" is less frequent here, for example, to improve the game. Pretend that these are the relative frequencies in English, and compute a prefix-free code of minimum expected length for transmitting messages. Give the answer by listing the relative frequency for each length of code word. Compute the expected length of the code (per text character). (Comment: ASCII coding uses five bits per letter; this code will beat that. Of course, ASCII suffers the handicap of including codes for punctuation.)

A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	Ø
9	2	2	4	12	2	3	2	9	1	1	4	2	6	8	2	1	6	4	6	4	2	2	1	2	1	2

2.3.30. Consider n messages occurring with probabilities $p_1, \ldots, p_n$, such that each p_i is a power of $1/2$ (each $p_i \geq 0$ and $\sum p_i = 1$).

a) Prove that the two least likely messages have equal probability.

b) Prove that the expected message length of the Huffman code for this distribution is $-\sum p_i \lg p_i$.

2.3.31. (+) Suppose that n messages occur with probabilities $p_1, \ldots, p_n$ and that the words are assigned distinct binary code words. Prove that for every code, the expected length of a code word with respect to this distribution is at least $-\sum p_i \lg p_i$. (Hint: Use induction on n.) (Shannon [1948])

Chapter 3

Matchings and Factors

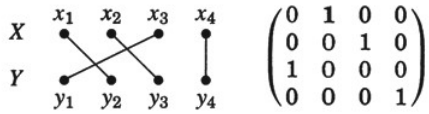
3.1. Matchings and Covers

Within a set of people, some pairs are compatible as roommates; under what conditions can we pair them all up? Many applications of graphs involve such pairings. In Example 1.1.9 we considered the problem of filling jobs with qualified applicants. Bipartite graphs have a natural vertex partition into two sets, and we want to know whether the two sets can be paired using edges. In the roommate question, the graph need not be bipartite.

3.1.1. Definition. A **matching** in a graph G is a set of non-loop edges with no shared endpoints. The vertices incident to the edges of a matching M are **saturated** by M; the others are **unsaturated** (we say *M-saturated* and *M-unsaturated*). A **perfect matching** in a graph is a matching that saturates every vertex.

3.1.2. Example. *Perfect matchings in $K_{n,n}$.* Consider $K_{n,n}$ with partite sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$. A perfect matching defines a bijection from X to Y. Successively finding mates for $x_1, x_2, \ldots$ yields $n!$ perfect matchings.

Each matching is represented by a permutation of $[n]$, mapping i to j when x_i is matched to y_j. We can express the matchings as matrices. With X and Y indexing the rows and columns, we let position i, j be 1 for each edge $x_i y_j$ in a matching M to obtain the corresponding matrix. There is one 1 in each row and each column. ■



$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$