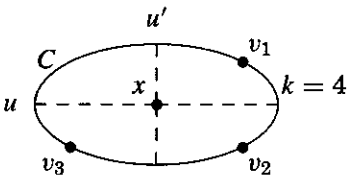
$x, V(C)$-fan of size k. We claim that again the fan has two paths forming a detour from C that includes x while keeping $S - \{x\}$. Let $v_1, \ldots, v_{k-1}$ be the vertices of $S - \{x\}$ in order on C, and let V_i be the portion of $V(C)$ from v_i up to but not including v_{i+1} (here $v_k = v_1$).

The sets $V_1, \ldots, V_{k-1}$ partition $V(C)$ into $k - 1$ disjoint sets. Since the $x, V(C)$-fan has k paths, two of them enter $V(C)$ in one of these sets, by the pigeonhole principle. Let u, u' be the vertices where these paths reach C. Replacing the u, u'-portion of C by the x, u-path and x, u'-path in the fan builds a new cycle that contains x and all of $S - \{x\}$. ■



Many applications of Menger's Theorem involve modeling a problem so that the desired objects correspond to paths in a graph or digraph, often by graph transformation arguments. For example, given sets $\mathbf{A} = A_1, \ldots, A_m$ with union X, a **system of distinct representatives** (SDR) is a set of distinct elements $x_1, \ldots, x_m$ such that $x_i \in A_i$. A necessary and sufficient condition for the existence of an SDR is that $\left|\bigcup_{i \in I} A_i\right| \geq |I|$ for all $I \subseteq [m]$. It is easy to prove this from Hall's Theorem by modeling $\mathbf{A}$ with an appropriate bipartite graph (Exercise 3.1.19). Indeed, Hall's Theorem was originally proved in the language of SDRs and is equivalent to Menger's Theorem (Exercise 23).

Ford and Fulkerson considered a more difficult problem. Let $\mathbf{A} = A_1, \ldots, A_m$ and $\mathbf{B} = B_1, \ldots, B_m$ be two families of sets. We may ask when there is a **common system of distinct representatives (CSDR)**, meaning a set of m elements that is both an SDR for $\mathbf{A}$ and an SDR for $\mathbf{B}$. They found a necessary and sufficient condition.

4.2.25.* Theorem. (Ford–Fulkerson [1958]) Families $\mathbf{A} = \{A_1, \ldots, A_m\}$ and $\mathbf{B} = \{B_1, \ldots, B_m\}$ have a common system of distinct representatives (CSDR) if and only if

$$\left|\left(\bigcup_{i \in I} A_i\right) \cap \left(\bigcup_{j \in J} B_j\right)\right| \geq |I| + |J| - m \quad \text{for each pair } I, J \subseteq [m].$$

Proof: We create a digraph G with vertices $a_1, \ldots, a_m$ and $b_1, \ldots, b_m$, plus a vertex for each element in the sets and special vertices s, t. The edges are

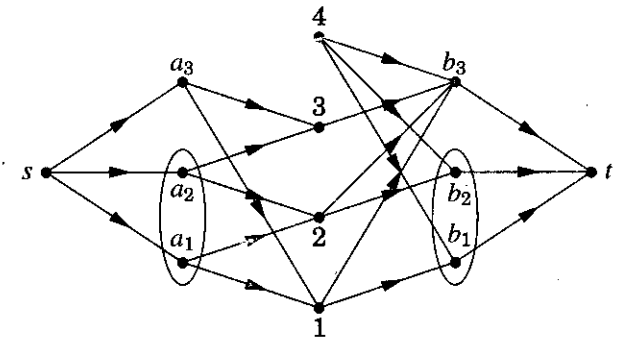
$$\{sa_i\colon A_i \in \mathbf{A}\} \quad \{a_i x\colon x \in A_i\}$$
$$\{b_j t\colon B_j \in \mathbf{B}\} \quad \{x b_j\colon x \in B_j\}$$

Each s, t-path selects a member of the intersection of some A_i and some B_j. There is a CSDR if and only if there is a set of m pairwise internally disjoint s, t-paths. By Menger's Theorem, it suffices to show that the stated condition

is equivalent to having no s,t-cut of size less than m. Given a set $R \subseteq V(G) - \{s,t\}$, let $I = \{a_i\} - R$ and $J = \{b_j\} - R$. The set R is an s,t-cut if and only if $(\bigcup_{i\in I} A_i) \cap (\bigcup_{j\in J} B_j) \subseteq R$. For an s,t-cut R, we thus have

$$|R| \geq \left|(\bigcup_{i\in I} A_i) \cap (\bigcup_{j\in J} B_j)\right| + (m - |I|) + (m - |J|).$$

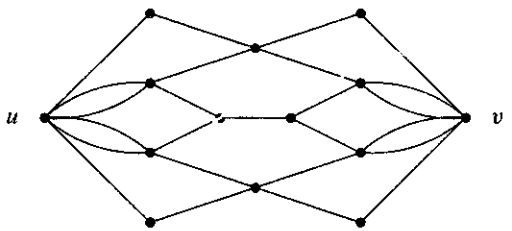
This lower bound is at least m for every s,t-cut if and only if the stated condition holds. ■



4.2.26.* Example. *Digraph for CSDR.* In the example above, the elements are $\{1,2,3,4\}$, $\mathbf{A} = \{12, 23, 31\}$, and $\mathbf{B} = \{14, 24, 1234\}$. Suppose that $R \cap \{a_i\} = \{a_1, a_2\}$ and $R \cap \{b_j\} = \{b_1, b_2\}$. In the argument, we set $I = \{a_3\}$ and $J = \{b_3\}$, and we observe that R is an s,t-cut if and only if it also contains $\{1,3\}$, which equals $(\bigcup_{i\in I} A_i) \cap (\bigcup_{j\in J} B_j)$. ■

EXERCISES

4.2.1. (−) Determine $\kappa(u,v)$ and $\kappa'(u,v)$ in the graph drawn below. (Hint: Use the dual problems to give short proofs of optimality.)



4.2.2. (−) Prove that if G is 2-edge-connected and G' is obtained from G by subdividing an edge of G, then G' is 2-edge-connected. Use this to prove that every graph having a closed-ear decomposition is 2-edge-connected. (Comment: This is an alternative proof of sufficiency for Theorem 4.2.10.)

4.2.3. (−) Let G be the digraph with vertex set [12] in which $i \to j$ if and only if i divides j. Determine $\kappa(1, 12)$ and $\kappa'(1, 12)$.

4.2.4. (−) Prove or disprove: If P is a u, v-path in a 2-connected graph G, then there is a u, v-path Q that is internally disjoint from P.

4.2.5. (−) Let G be a simple graph, and let $H(G)$ be the graph with vertex set $V(G)$ such that $uv \in E(H)$ if and only if u, v appear on a common cycle in G. Characterize the graphs G such that H is a clique.

4.2.6. (−) Use results of this section to prove that a simple graph G is 2-connected if and only if G can be obtained from C_3 by a sequence of edge additions and edge subdivisions.

• • • • •

4.2.7. Let xy be an edge in a digraph G. Prove that $\kappa(G - xy) \geq \kappa(G) - 1$.

4.2.8. Prove that a simple graph G is 2-connected if and only if for every triple (x, y, z) of distinct vertices, G has an x, z-path through y. (Chein [1968])

4.2.9. Prove that a graph G with at least four vertices is 2-connected if and only if for every pair X, Y of disjoint vertex subsets with $|X|, |Y| \geq 2$, there exist two completely disjoint paths P_1, P_2 in G such that each has an endpoint in X and an endpoint in Y and no internal vertex in X or Y.

4.2.10. **A greedy ear decomposition** of a 2-connected graph is an ear decomposition that begins with a longest cycle and iteratively adds a longest ear from the remaining graph. Use a greedy ear decomposition to prove that every 2-connected claw-free graph G has $\lfloor n(G)/3 \rfloor$ pairwise-disjoint copies of P_3. (Kaneko–Kelmans–Nishimura [2000])

4.2.11. (!) For a connected graph G with at least three vertices, prove that the following statements are equivalent (use of Menger's Theorem is permitted).

A) G is 2-edge-connected.
B) Every edge of G appears in a cycle.
C) G has a closed trail containing any specified pair of edges.
D) G has a closed trail containing any specified pair of vertices.

4.2.12. (!) Use Menger's Theorem to prove that $\kappa(G) = \kappa'(G)$ when G is 3-regular (Theorem 4.1.11).

4.2.13. (!) Let G be a 2-edge-connected graph. Define a relation R on $E(G)$ by $(e, f) \in R$ if $e = f$ or if $G - e - f$ is disconnected. (Lovász [1979, p277])

a) Prove that $(e, f) \in R$ if and only if e, f belong to the same cycles.
b) Prove that R is an equivalence relation on $E(G)$.
c) For each equivalence class F, prove that F is contained in a cycle.
d) For each equivalence class F, prove that $G - F$ has no cut-edge.

4.2.14. (!) A u, v-**necklace** is a list of cycles $C_1, \ldots, C_k$ such that $u \in C_1$, $v \in C_k$, consecutive cycles share one vertex, and nonconsecutive cycles are disjoint. Use induction on $d(u, v)$ to prove that a graph G is 2-edge-connected if and only if for all $u, v \in V(G)$ there is a u, v-necklace in G.

4.2.15. (+) Let v be a vertex of a 2-connected graph G. Prove that v has a neighbor u such that $G - u - v$ is connected. (Chartrand–Lesniak [1986, p51])

4.2.16. (+) Let G be a 2-connected graph. Prove that if T_1, T_2 are two spanning trees of G, then T_1 can be transformed into T_2 by a sequence of operations in which a leaf is removed and reattached using another edge of G.

4.2.17. Determine the smallest graph with connectivity 3 having a pair of nonadjacent vertices linked by four pairwise internally disjoint paths.

4.2.18. Let G be a graph without isolated vertices. Prove that if G has no even cycles, then every block of G is an edge or an odd cycle.

4.2.19. (!) *Membership in common cycles.*

a) Prove that two distinct edges lie in the same block of a graph if and only if they belong to a common cycle.

b) Given $e, f, g \in E(G)$, suppose that G has a cycle through e and f and a cycle through f and g. Prove that G also has a cycle through e and g. (Comment: This problem implies that for graphs without cut-edges, "belong to a common cycle" is an equivalence relation whose equivalence classes are the edge sets of blocks.)

4.2.20. Prove that the hypercube Q_k is k-connected by constructing k pairwise internally disjoint x, y-paths for each vertex pair $x, y \in V(Q_k)$.

4.2.21. (!) Let G be a $2k$-edge-connected graph with at most two vertices of odd degree. Prove that G has a k-edge-connected orientation. (Nash-Williams [1960])

4.2.22. (!) Suppose that $\kappa(G) = k$ and diam $G = d$. Prove that $n(G) \geq k(d-1) + 2$ and $\alpha(G) \geq \lceil (1+d)/2 \rceil$. For each $k \geq 1$ and $d \geq 2$, construct a graph for which equality holds in both bounds.

4.2.23. (!) Use Menger's Theorem ($\kappa(x,y) = \lambda(x,y)$ when $xy \notin E(G)$) to prove the König–Egerváry Theorem ($\alpha'(G) = \beta(G)$ when G is bipartite).

4.2.24. (!) Let G be a k-connected graph, and let S, T be disjoint subsets of $V(G)$ with size at least k. Prove that G has k pairwise disjoint S, T-paths.

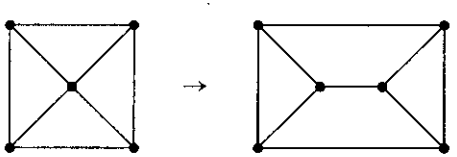
4.2.25. (*) Show that Theorem 4.2.24 is best possible by constructing for each k a k-connected graph having $k+1$ vertices that do not lie on a cycle.

4.2.26. For $k \geq 2$, prove that a graph with at least $k+1$ vertices is k-connected if and only if for every $T \subseteq S \subseteq V(G)$ with $|S| = k$ and $|T| = 2$, there is a cycle in G that contains T and avoids $S - T$. (Lick [1973])

4.2.27. A **vertex k-split** of a graph G is a graph H obtained from G by replacing one vertex $x \in V(G)$ by two adjacent vertices x_1, x_2 such that $d_H(x_i) \geq k$ and that $N_H(x_1) \cup N_H(x_2) = N_G(x) \cup \{x_1, x_2\}$.

a) Prove that every vertex k-split of a k-connected graph is k-connected.

b) Conclude that any graph obtained from a "wheel" $W_n = K_1 \vee C_{n-1}$ (Definition 3.3.6) by a sequence of edge additions and vertex 3-splits on vertices of degree at least 4 is 3-connected. (Comment: Tutte [1961b] proved also that every 3-connected graph arises in this way. The characterization does not extend easily for $k > 3$.)



4.2.28. (!) Let X and Y be disjoint sets of vertices in a k-connected graph G. Let $u(x)$ for $x \in X$ and $w(y)$ for $y \in Y$ be nonnegative integers such that $\sum_{x \in X} u(x) = \sum_{y \in Y} w(y) = k$. Prove that G has k pairwise internally disjoint X, Y-paths so that $u(x)$ of them start at x and $w(y)$ of them end at y, for $x \in X$ and $y \in Y$.

4.2.29. Given a graph G, let D be the digraph obtained by replacing each edge with two oppositely-directed edges having the same endpoints (thus D is the symmetric digraph with underlying graph G). Assume that for all $x, y \in V(D)$ both $\kappa'_D(x, y) = \lambda'_D(x, y)$ and $\kappa_D(x, y) = \lambda_D(x, y)$ hold, the latter applying only when $x \not\to y$. Use this hypothesis to prove that also $\kappa'_G(x, y) = \lambda'_G(x, y)$ and $\kappa_G(x, y) = \lambda_G(x, y)$, the latter for $x \not\leftrightarrow y$.

4.2.30. (!) Prove that applying the expansion operation of Example 1.3.26 to a 3-connected graph yields a 3-connected graph. Obtain the Petersen graph from K_4 by expansions. (Comment: Tutte [1966a] proved that a 3-regular graph is 3-connected if and only if it arises from K_4 by a sequence of these operations.)

4.2.31. Let G be a k-connected simple graph.

a) Let C and D be two cycles in G of maximum length. For $k = 2$ and $k = 3$, prove that C and D share at least k vertices. (Hint: If they don't, construct a longer cycle.)

b) For each $k \geq 2$, construct a k-connected graph that has distinct longest cycles with only k common vertices. (Hint: $K_{2,4}$ works for $k = 2$.)

4.2.32. *Graph splices.* Let G_1 and G_2 be disjoint k-connected graphs with $k \geq 2$. Choose $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$. Let B be a bipartite graph with partite sets $N_{G_1}(v_1)$ and $N_{G_2}(v_2)$ that has no isolated vertex and has a matching of size at least k. Prove that $(G_1 - v_1) \cup (G_2 - v_2) \cup B$ is k-connected.

4.2.33. (*) Prove Hall's Theorem from Theorem 4.2.25.

4.2.34. A k-connected graph G is **minimally k-connected** if for every $e \in E(G)$, the graph $G - e$ is not k-connected. Halin [1969] proved that $\delta(G) = k$ when G is minimally k-connected. Use ear decomposition to prove this for $k = 2$. Conclude that a minimally 2-connected graph G with at least 4 vertices has at most $2n(G) - 4$ edges, with equality only for $K_{2,n-2}$. (Dirac [1967])

4.2.35. Prove that if G is 2-connected, then $G - xy$ is 2-connected if and only if x and y lie on a cycle in $G - xy$. Conclude that a 2-connected graph is minimally 2-connected if and only if every cycle is an induced subgraph. (Dirac [1967], Plummer [1968])

4.2.36. (!) For $S \subseteq V(G)$, let $d(S) = \left|[S, \overline{S}]\right|$. Let X and Y be nonempty proper vertex subsets of G. Prove that $d(X \cap Y) + d(X \cup Y) \leq d(X) + d(Y)$. (Hint: Draw a picture and consider contributions from various types of edges.)

4.2.37. (+) A k-edge-connected graph G is **minimally k-edge-connected** if for every $e \in E(G)$ the graph $G - e$ is not k-edge-connected. Prove that $\delta(G) = k$ when G is minimally k-edge-connected. (Hint: Consider a minimal set S such that $\left|[S, \overline{S}]\right| = k$. If $|S| \neq 1$, use $G - e$ for some $e \in E(G[S])$ to obtain another set T with $\left|[T, \overline{T}]\right| = k$ such that S, T contradict Exercise 4.2.36.) (Mader [1971]; see also Lovász [1979, p285])

4.2.38. Mader [1978] proved the following: "If z is a vertex of a graph G such that $d_G(z) \notin \{0, 1, 3\}$ and z is incident to no cut-edge, then z has neighbors x and y such that $\kappa_{G-xz-yz+xy}(u, v) = \kappa_G(u, v)$ for all $u, v \in V(G) - \{z\}$." Use Mader's Theorem and Exercise 4.2.37 to prove Nash-Williams' Orientation Theorem: every $2k$-edge-connected graph has a k-edge-connected orientation. (Comment: A weaker version of Mader's Theorem, given in Lovász [1979, p286–288], also yields Nash-Williams' Theorem in the same way.)

4.3. Network Flow Problems

Consider a network of pipes where valves allow flow in only one direction. Each pipe has a capacity per unit time. We model this with a vertex for each junction and a (directed) edge for each pipe, weighted by the capacity. We also assume that flow cannot accumulate at a junction. Given two locations s, t in the network, we may ask "what is the maximum flow (per unit time) from s to t?"

This question arises in many contexts. The network may represent roads with traffic capacities, or links in a computer network with data transmission capacities, or currents in an electrical network. There are applications in industrial settings and to combinatorial min-max theorems. The seminal book on the subject is Ford–Fulkerson [1962]. More recently, Ahuja–Magnanti–Orlin [1993] presents a thorough treatment of network flow problems.

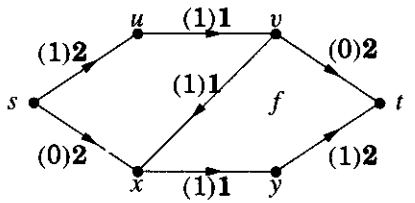
4.3.1. Definition. A **network** is a digraph with a nonnegative **capacity** $c(e)$ on each edge e and a distinguished **source vertex** s and **sink vertex** t. Vertices are also called **nodes**. A **flow** f assigns a value $f(e)$ to each edge e. We write $f^+(v)$ for the total flow on edges leaving v and $f^-(v)$ for the total flow on edges entering v. A flow is **feasible** if it satisfies the **capacity constraints** $0 \le f(e) \le c(e)$ for each edge and the **conservation constraints** $f^+(v) = f^-(v)$ for each node $v \notin \{s, t\}$.

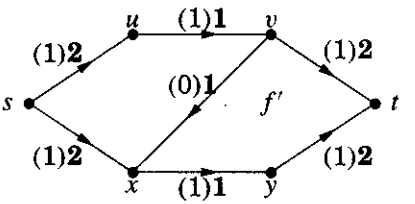
MAXIMUM NETWORK FLOW

We consider first the problem of maximizing the net flow into the sink.

4.3.2. Definition. The **value** $\mathrm{val}(f)$ of a flow f is the net flow $f^-(t) - f^+(t)$ into the sink. A **maximum flow** is a feasible flow of maximum value.

4.3.3. Example. The **zero flow** assigns flow 0 to each edge; this is feasible. In the network below we illustrate a nonzero feasible flow. Each capacities are shown in bold, flow values in parentheses. Our flow f assigns $f(sx) = f(vt) = 0$, and $f(e) = 1$ for every other edge e. This is a feasible flow of value 1.





A path from the source to the sink with excess capacity would allow us to increase flow. In this example, no path remains with excess capacity, but the

flow f' with $f'(vx) = 0$ and $f'(e) = 1$ for $e \neq vx$ has value 2. The flow f is "maximal" in that no other feasible flow can be found by increasing the flow on some edges, but f is not a maximum flow.

We need a more general way to increase flow. In addition to traveling forward along edges with excess capacity, we allow traveling backward (against the arrow) along edges where the flow is nonzero. In this example, we can travel from s to x to v to t. Increasing the flow by 1 on sx and vt and decreasing it by one on vx changes f into f'. ∎

4.3.4. Definition. When f is a feasible flow in a network N, an **f-augmenting path** is a source-to-sink path P in the underlying graph G such that for each $e \in E(P)$,

a) if P follows e in the forward direction, then $f(e) < c(e)$.

b) if P follows e in the backward direction, then $f(e) > 0$.

Let $\epsilon(e) = c(e) - f(e)$ when e is forward on P, and let $\epsilon(e) = f(e)$ when e is backward on P. The **tolerance** of P is $\min_{e \in E(P)} \epsilon(e)$.

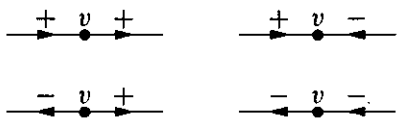
As in Example 4.3.3, an f-augmenting path leads to a flow with larger value. The definition of f-augmenting path ensures that the tolerance is positive; this amount is the increase in the flow value.

4.3.5. Lemma. If P is an f-augmenting path with tolerance z, then changing flow by $+z$ on edges followed forward by P and by $-z$ on edges followed backward by P produces a feasible flow f' with $\mathrm{val}(f') = \mathrm{val}(f) + z$.

Proof: The definition of tolerance ensures that $0 \leq f'(e) \leq c(e)$ for every edge e, so the capacity constraints hold. For the conservation constraints we need only check vertices of P, since flow elsewhere has not changed.

The edges of P incident to an internal vertex v of P occur in one of the four ways shown below. In each case, the change to the flow out of v is the same as the change to the flow into v, so the net flow out of v remains 0 in f'.

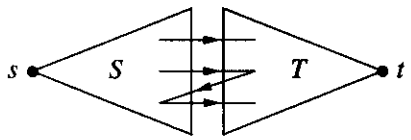
Finally, the net flow into the sink t increases by z. ∎



The flow on backward edges did not disappear; it was redirected. In effect, the augmentation in Example 4.3.3 cuts the flow path and extends each portion to become a new flow path. We will soon describe an algorithm to find augmenting paths.

Meanwhile, we would like a quick way to know when our present flow is a maximum flow. In Example 4.3.3, the central edges seem to form a "bottleneck"; we only have capacity 2 from the left half of the network to the right half. This observation will give us a PROOF that the flow value can be no larger.

4.3.6. Definition. In a network, a **source/sink cut** $[S, T]$ consists of the edges from a **source set** S to a **sink set** T, where S and T partition the set of nodes, with $s \in S$ and $t \in T$. The **capacity** of the cut $[S, T]$, written $\text{cap}(S, T)$, is the total of the capacities on the edges of $[S, T]$.



Keep in mind that in a digraph $[S, T]$ denotes the set of edges with tail in S and head in T. Thus the capacity of a cut $[S, T]$ is completely unaffected by edges from T to S.

Given a cut $[S, T]$, every s, t-path uses at least one edge of $[S, T]$, so intuition suggests that the value of a feasible flow should be bounded by $\text{cap}(S, T)$. To make this precise, we extend the notion of net flow to sets of nodes. Let $f^+(U)$ denote the total flow on edges leaving U, and let $f^-(U)$ be the total flow on edges entering U. The net flow out of U is then $f^+(U) - f^-(U)$.

4.3.7. Lemma. If U is a set of nodes in a network, then the net flow out of U is the sum of the net flows out of the nodes in U. In particular, if f is a feasible flow and $[S, T]$ is a source/sink cut, then the net flow out of S and net flow into T equal $\text{val}(f)$.

Proof: The stated claim is that

$$f^+(U) - f^-(U) = \textstyle\sum_{v \in U} [f^+(v) - f^-(v)].$$

We consider the contribution of the flow $f(xy)$ on an edge xy to each side of the formula. If $x, y \in U$, then $f(xy)$ is not counted on the left, but it contributes positively (via $f^+(x)$) and negatively (via $f^-(y)$) on the right. If $x, y \notin U$, then $f(xy)$ contributes to neither sum. If $xy \in [U, \overline{U}]$, then it contributes positively to each sum. If $xy \in [\overline{U}, U]$, then it contributes negatively to each sum. Summing over all edges yields the equality.

When $[S, T]$ is a source/sink cut and f is a feasible flow, net flow from nodes of S sums to $f^+(s) - f^-(s)$, and net flow from nodes of T sums to $f^+(t) - f^-(t)$, which equals $-\text{val}(f)$. Hence the net flow across any source/sink cut equals both the net flow out of s and the net flow into t. ∎

4.3.8. Corollary. (Weak duality) If f is a feasible flow and $[S, T]$ is a source/sink cut, then $\text{val}(f) \leq \text{cap}(S, T)$.

Proof: By the lemma, the value of f equals the net flow out of S. Thus

$$\text{val}(f) = f^+(S) - f^-(S) \leq f^+(S),$$

since the flow into S is no less than 0. Since the capacity constraints require $f^+(S) \leq \text{cap}(S, T)$, we obtain $\text{val}(f) \leq \text{cap}(S, T)$. ∎

Among source/sink cuts, one with minimum capacity yields the best bound on the value of a flow. This defines the **minimum cut** problem. The max flow and min cut problems on a network are dual optimization problems.† Given a flow with value α and a cut with value α, the duality inequality in Corollary 4.3.8 PROVES that the cut is a minimum cut and the flow is a maximum flow.

If every instance has solutions with the same value to both the max problem and the min problem ("strong duality"), then a short proof of optimality always exists. This does not hold for all dual pairs of problems (recall matching and covering in general graphs), but it holds for max flow and min cut.

The Ford–Fulkerson algorithm seeks an augmenting path to increase the flow value. If it does not find such a path, then it finds a cut with the same value (capacity) as this flow; by Corollary 4.3.8, both are optimal. If no infinite sequence of augmentations is possible, then the iteration leads to equality between the maximum flow value and the minimum cut capacity.

4.3.9. Algorithm. (Ford–Fulkerson labeling algorithm)
Input: A feasible flow f in a network.
Output: An f-augmenting path or a cut with capacity $\mathrm{val}(f)$.
Idea: Find the nodes reachable from s by paths with positive tolerance. Reaching t completes an f-augmenting path. During the search, R is the set of nodes labeled *Reached*, and S is the subset of R labeled *Searched*.
Initialization: $R = \{s\}$, $S = \emptyset$.
Iteration: Choose $v \in R - S$.

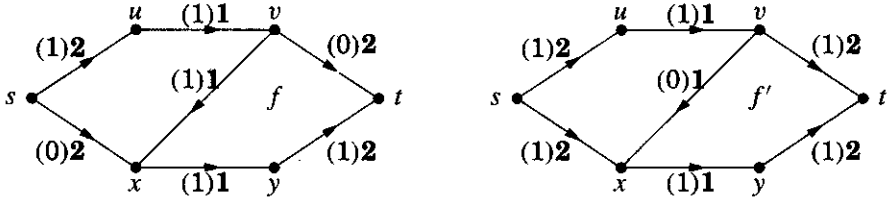
For each exiting edge vw with $f(vw) < c(vw)$ and $w \notin R$, add w to R.

For each entering edge uv with $f(uv > 0)$ and $u \notin R$, add u to R.

Label each vertex added to R as "reached", and record v as the vertex reaching it. After exploring all edges at v, add v to S.

If the sink t has been reached (put in R), then trace the path reaching t to report an f-augmenting path and terminate. If $R = S$, then return the cut $[S, \overline{S}]$ and terminate. Otherwise, iterate. ■

4.3.10. Example. On the left below is the network of Example 4.3.3 with the flow f. We run the labeling algorithm. First we search from s and find excess capacity to u and x, labeling them reached. Now we have $u, v \in R - S$. There is no excess capacity on uv or xy, so searching from u reaches nothing, and also



†The precise notion of "dual problem" comes from linear programming. For our purposes, dual problems are a maximization problem and a minimization problem such that $a \leq b$ whenever a and b are the values of feasible solutions to the max problem and min problem, respectively. See Section 8.1 for further discussion.

searching from x does not reach y. However, there is nonzero flow on vx. Thus we label v from x. Now v is the only element of $R - S$, and searching from v reaches t. We labeled t from v, v from x, and x from s, so we have found the augmenting path s, x, v, t.

The tolerance on this path is 1, so the augmentation increases the flow value by 1. In the new flow f' shown on the right, every edge has unit flow except $f'(vx) = 0$. When we run the labeling algorithm again, we have excess capacity on su and sx and can label $\{u, x\}$, but from these nodes we can label no others. We terminate with $R = S = \{s, u, x\}$. The capacity of the resulting cut $[S, \overline{S}]$ is 2, which equals $\mathrm{val}(f')$ and proves that f' is a maximum flow. ∎

Repeated use of the labeling algorithm allows us to solve the maximum flow problem and prove the strong duality relationship.

4.3.11. Theorem. (Max-flow Min-cut Theorem—Ford and Fulkerson [1956]) In every network, the maximum value of a feasible flow equals the minimum capacity of a source/sink cut.

Proof: In the max-flow problem, the zero flow ($f(e) = 0$ for all e) is always a feasible flow and gives us a place to start. Given a feasible flow, we apply the labeling algorithm. It iteratively adds vertices to S (each vertex at most once) and terminates with $t \in R$ ("breakthrough") or with $S = R$.

In the breakthrough case, we have an f-augmenting path and increase the flow value. We then repeat the labeling algorithm. When the capacities are rational, each augmentation increases the flow by a multiple of $1/a$, where a is the least common multiple of the denominators, so after finitely many augmentations the capacity of some cut is reached. The labeling algorithm then terminates with $S = R$.

When terminating this way, we claim that $[S, T]$ is a source/sink cut with capacity $\mathrm{val}(f)$, where $T = \overline{S}$ and f is the present flow. It is a cut because $s \in S$ and $t \notin R = S$. Since applying the labeling algorithm to the flow f introduces no node of T into R, no edge from S to T has excess capacity, and no edge from T to S has nonzero flow in f. Hence $f^+(S) = \mathrm{cap}(S, T)$ and $f^-(S) = 0$.

Since the net flow out of any set containing the source but not the sink is $\mathrm{val}(f)$, we have proved

$$\mathrm{val}(f) = f^+(S) - f^-(S) = f^+(S) = \mathrm{cap}(S, T). \qquad \blacksquare$$

This proof of Theorem 4.3.11 requires rational capacities; otherwise, Algorithm 4.3.9 may yield augmenting paths forever! Ford and Fulkerson provided an example of this with only ten vertices (see Papadimitriou–Steiglitz [1982, p126-128]). Edmonds and Karp [1972] modified the labeling algorithm to use at most $(n^3 - n)/4$ augmentations in an n-vertex network and work for all real capacities. As in the bipartite matching problem (Theorem 3.2.22), this is done by searching always for shortest augmenting paths. Faster algorithms are now known; again we cite Ahuja–Magnanti–Orlin [1993] for a thorough discussion.

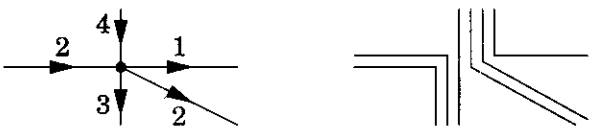
INTEGRAL FLOWS

In combinatorial applications, we typically have integer capacities and want a solution in which the flow on each edge is an integer.

4.3.12. Corollary. (Integrality Theorem) If all capacities in a network are integers, then there is a maximum flow assigning integral flow to each edge. Furthermore, some maximum flow can be partitioned into flows of unit value along paths from source to sink.

Proof: In the labeling algorithm of Ford and Fulkerson, the change in flow value when an augmenting path is found is always a flow value or the difference between a flow value and a capacity. When these are integers, the difference is also an integer. Starting with the zero flow, this implies that there is no first time when a noninteger flow appears.

The algorithm thus produces a maximum flow with integer flow on each edge. At each internal node, we now match units of entering flow to units of exiting flow. This forms s, t-paths and perhaps cycles. If a cycle arises, then we decrease flow on its edges by 1 to eliminate it without changing the flow value. This leaves $\mathrm{val}(f)$ paths from s to t, each corresponding to a unit of flow. ∎



The integrality theorem yields paths of unit flow. In applications, we build networks where these units of flow have meaning.

The next two remarks show that the Max-flow Min-cut Theorem for networks with integer capacities is almost the same statement as Menger's Theorem for edge-disjoint paths in digraphs.

4.3.13. Remark. *Menger from Max-flow Min-cut.* When x, y are vertices in a digraph D, we can view D as a network with source x and sink y and capacity 1 on every edge. Capacity 1 ensures that units of flow from x to y correspond to pairwise edge-disjoint x, y-paths in D. Thus a flow of value k yields a set of k such paths.

Similarly, every source/sink partition S, T defines a set of edges whose deletion makes y unreachable from x: the set $[S, T]$. Since every capacity is 1, the size of this set is $\mathrm{cap}(S, T)$.

The paths and the edge cut we have obtained might not be optimal, but by the Max-flow Min-cut Theorem we have

$$\lambda'_D(x, y) \geq \max \mathrm{val}(f) = \min \mathrm{cap}(S, T) \geq \kappa'_D(x, y).$$

Since always $\kappa'(x, y) \geq \lambda'(x, y)$, equality now holds. ∎

4.3.14. Remark. *Max-flow Min-cut from Menger.* To show that Menger's Theorem implies the Max-flow Min-cut Theorem for rational capacities, we take an arbitrary network and transform it into a digraph where we apply Menger's Theorem. By multiplying all capacities by the least common denominator, we may assume that the capacities are integers.

Given a network N with integer capacities, we form a digraph D by splitting each edge of capacity j into j edges with the same endpoints. For N, duality yields $\max \mathrm{val}(f) \leq \min \mathrm{cap}(S, T)$. This time we want to use Menger's Theorem on D to obtain the reverse inquality, so in contrast to Remark 4.3.13 our desired computation is

$$\max \mathrm{val}(f) \geq \lambda'_D(s,t) = \kappa'_D(s,t) \geq \min \mathrm{cap}(S, T).$$

A set of $\lambda'(s,t)$ pairwise edge-disjoint s,t-paths in D collapses into a flow of value $\lambda'(s,t)$ in N, since the number of copies of each edge in D equals the capacity of the edge in N. Thus $\max \mathrm{val}(f) \geq \lambda'(s,t)$.

Now, let F be a set of $\kappa'(s,t)$ edges disconnecting t from s in D. If $e \in F$, then the minimality of F implies that $D - (F - e)$ has an s,t-path P through e. If some other copy e' of the edge $e = uv$ is not in F, then P can be rerouted along e' to obtain an s,t-path in $D - F$. Therefore, F contains all copies or no copies of each multiple edge in D. Hence $\kappa'(s,t)$ is the sum of the capacities on a set of edges that disconnects t from s in N. Letting S be the set of vertices reachable from s in $D - F$, we have $\mathrm{cap}(S, T) = \kappa'(s,t)$. The minimum cut has at most this capacity, so $\min \mathrm{cap}(S, T) \leq \kappa'(s,t)$, and we have proved all the needed inequalities. ∎

For combinatorial applications, Menger's Theorem may yield simpler proofs than the Max-flow Min-cut Theorem (compare Theorem 4.2.25 with ??). Nevertheless, our proof of Menger's Theorem in Section 4.2 is awkward to implement algorithmically. For large-scale computations, network flow and the Ford–Fulkerson labeling algorithm are more appropriate. Indeed, most algorithms that compute connectivity in graphs and digraphs use network flow methods (Stoer–Wagner [1994] presents a different approach).

We present other network models for combinatorial problems. For example, the other local versions of Menger's Theorem can also be obtained directly.

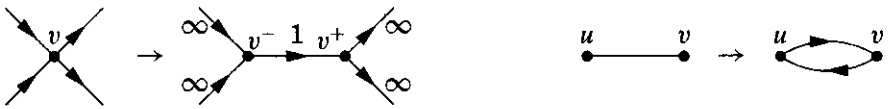
4.3.15. Remark. *Other transformations.* For each version of Menger's Theorem, we encode the path problem using network flows with integer capacities.

To obtain a network model for the problem of internally disjoint paths in a digraph D, we must prevent two units of flow from passing through a vertex. This can be done by replacing each vertex v with two vertices v^-, v^+ that inherit the entering and exiting edges at v. By adding an edge of unit capacity from v^- to v^+, we obtain the effect of limiting flow through v to one unit. By putting very large capacity (essentially infinite) on the edges that were in D, we ensure that a minimum cut will count only edges of the form v^-v^+

To obtain a network model for the problem of edge-disjoint paths in a graph G, we must permit flow to pass either way in an edge. This can be done by

replacing each edge uv with two directed edges uv and vu. When the network sends unit flow in both directions, in effect the edge is not being used at all.

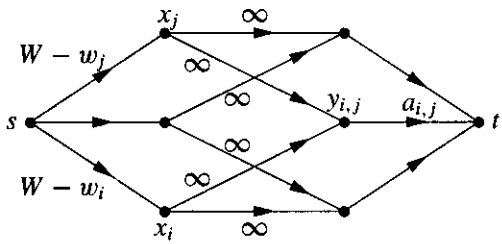
In each case, a flow in the network provides a set of paths, and a minimum cut leads to a separating set of vertices or edges. As in Remark 4.3.13, duality then gives us the desired equality in Menger's Theorem. To model the problem of internally disjoint paths in a graph, we need both of these transformations. Exercises 5–7 request the details of these proofs. ∎



4.3.16. Application. *Baseball Elimination Problem* (Schwartz [1966]). At some time during the season, we may wonder whether team X can still win the championship. In other words, can winners be assigned for the remaining games so that no team ends with more victories than X? If so, then such an assignment exists with X winning all its remaining games, reaching W wins. We want to know whether winners can be chosen for other games so that no team obtains more than W wins. To test this, we create a network where units of flow correspond to the remaining games.

Let $X_1, \ldots, X_n$ be the other teams. Include nodes $x_1, \ldots, x_n$ for the n teams, nodes $y_{i,j}$ for the $\binom{n}{2}$ pairs of teams, and a source s and sink t. Put an edge from s to each team node and an edge from each pair node to t. Each pair node $y_{i,j}$ is entered by edges from x_i and x_j.

The capacities model the constraints. The capacity on edge $y_{i,j}t$ is $a_{i,j}$, the number of remaining games between X_i and X_j. Given that X_i has won w_i games already, the capacity on edge sx_i is $W - w_i$ to keep X in contention. The capacity on edges $x_i y_{i,j}$ and $x_j y_{i,j}$ is ∞ (the number of games x_i can win from x_j is constrained by the capacity on $y_{i,j}t$).



By the integrality theorem, a maximum flow breaks into flow units. Each unit corresponds to one game; the first edge specifies the winner, and the last edge specifies the pair. The network has a flow of value $\sum_{i,j} a_{i,j}$ if and only if *all* remaining games can be played with no team exceeding W wins; this is the condition for X remaining in contention.

By the Max-flow Min-cut Theorem, there is a flow of value $\sum a_{i,j}$ if and only if every cut has capacity at least $\sum a_{i,j}$. Let S, T be a cut with finite capacity,

and let $Z = \{i: x_i \in T\}$. Since $c(x_i y_{i,j}) = \infty$, we cannot have $x_i \in S$ and $y_{i,j} \in T$; thus $y_{i,j} \in S$ whenever i or j is not in Z. To minimize capacity, we put $y_{i,j} \in T$ whenever $\{i, j\} \subseteq Z$. Now $\text{cap}(S, T) = \sum_{i \in Z}(W - w_i) + \sum_{\{i,j\} \not\subseteq Z} a_{i,j}$. The condition that every cut have capacity at least $\sum a_{i,j}$ becomes

$$\sum_{i \in Z}(W - w_i) \geq \sum_{\{i,j\} \subseteq Z} a_{i,j} \qquad \text{for all } Z \subseteq [n].$$

Note that this condition is obviously necessary; it states that we need enough leeway in the total wins among teams indexed by Z in order to accommodate winners for all the games among these teams. We have proved TONCAS. ∎

Combinatorial applications of network flow usually involve showing that the desired configuration exists if and only if a related network has a large enough flow. As in Application 4.3.16, the Max-flow Min-cut Theorem then yields a necessary and sufficient condition for its existence. Other examples include most of Exercises 5– and also Exercise 13 and Theorems 4.3.17–4.3.18.

SUPPLIES AND DEMANDS (optional)

Next we consider a more general network model. We allow multiple sources and sinks, and also we associate with each source x_i a **supply** $\sigma(x_i)$ and with each sink y_j a **demand** $\partial(y_j)$. To the capacity constraints for edges and conservation constraints for internal nodes, we add **transportation constraints** for the sources and sinks.

$$f^+(x_i) - f^-(x_i) \leq \sigma(x_i) \text{ for each source } x_i$$
$$f^-(y_j) - f^+(y_j) \geq \partial(y_j) \text{ for each sink } y_j$$

The resulting configuration is a **transportation network**. With positive values for the demands, the zero flow is not feasible. We seek a feasible flow satisfying these additional constraints. The "supply/demand" terminology suggests the constraints; we must satisfy the demands at the sinks without exceeding the available supply at any source. This model is appropriate when a company has multiple distribution centers (sources) and retail outlets (sinks).

Let X and Y denote the sets of sources and sinks, respectively. Let $\sigma(A) = \sum_{v \in A} \sigma(v)$ and $\partial(B) = \sum_{v \in B} \partial(v)$ denote the total supply or demand at a set $A \subseteq X$ or $B \subseteq Y$. For a set F of edges, let $c(F) = \sum_{e \in F} c(e)$. Given a set T of vertices, the **net demand** $\partial(Y \cap T) - \sigma(X \cap T)$ must be satisfied by flow from the remaining vertices. Hence it is necessary that $c([\overline{T}, T])$ be at least this large. Satisfying this for every set T is also sufficient for a feasible flow (TONCAS).

4.3.17. Theorem. (Gale [1957]) In a transportation network N with sources X and sinks Y, a feasible flow exists if and only if

$$c([S, T]) \geq \partial(Y \cap T) - \sigma(X \cap T)$$

for every partition of the vertices of N into sets S and T.

Proof: We have already observed the necessity of the condition. For sufficiency, construct a new network N' by adding a supersource s and a supersink t, with an edge of capacity $\sigma(x_i)$ from s to each $x_i \in X$ and an edge of capacity $\partial(y_j)$ from each $y_j \in Y$ to t. The transportation network N has a feasible flow if and only if N' has a flow saturating each edge to t (a flow of value $\partial(Y)$).

By the Ford–Fulkerson Theorem, we know that N' has a flow of value $\partial(Y)$ if and only if $\mathrm{cap}(S \cup s, T \cup t) \geq \partial(Y)$ for each partition S, T of $V(N)$. The cut $[S \cup s, T \cup t]$ in N' consists of $[S, T]$ from N, plus edges from s to T and edges from S to t in N'. Hence

$$\mathrm{cap}(S \cup s, T \cup t) = c(S, T) + \sigma(T \cap X) + \partial(S \cap Y).$$

We now have $\mathrm{cap}(S \cup s, T \cup t) \geq \partial(Y)$ if and only if

$$c(S, T) + \sigma(X \cap T) \geq \partial(Y) - \partial(Y \cap S) = \partial(Y \cap T),$$

which is the condition assumed. ∎

For specific instances, the construction of N' is the key point, because we produce a feasible flow in N (when it exists) by running the Ford–Fulkerson algorithm on the network N'. When costs (per unit flow) are attached to the edges, we have the Min-cost Flow Problem, which generalizes the Transportation Problem of Application 3.2.14. Solution algorithms for the Min-cost Flow Problem appear in Ford–Fulkerson [1962] and in Ahuja–Magnanti–Orlin [1993].

We discuss several applications of Gale's condition. A pair of integer lists $p = (p_1, \ldots, p_m)$ and $q = (q_1, \ldots, q_n)$ is **bigraphic** (Exercise 1.4.31) if there is a simple X, Y-bigraph such that the vertices of X have degrees $p_1, \ldots, p_m$ and the vertices of Y have degrees $q_1, \ldots, q_n$. Clearly $\sum p_i = \sum q_j$ is necessary, but this condition is not sufficient. To test whether (p, q) is bigraphic, we create a network in which units of flows will correspond to edges in the desired graph. The result is a bipartite analogue of the Erdős–Gallai condition for graphic sequences (Exercise 3.3.28).

4.3.18. Theorem. (Gale [1957], Ryser [1957]) If p, q are lists of nonnegative integers with $p_1 \geq \cdots \geq p_m$ and $q_1 \geq \cdots \geq q_n$, then (p, q) is bigraphic if and only if $\sum_{i=1}^{m} \min\{p_i, k\} \geq \sum_{j=1}^{k} q_j$ for $1 \leq k \leq n$.

Proof: *Necessity.* Let G be a simple X, Y-bigraph realizing (p, q). Consider the edges incident to a set of k vertices in Y. Because G is simple, each $x_i \in X$ is incident to at most k of these edges, and also x_i is incident to at most p_i of these edges. Hence $\sum_{i=1}^{m} \min\{p_i, k\}$ is an upper bound on the number of edges incident to any k vertices of Y, such as those with degrees $q_1, \ldots, q_k$.

Sufficiency. Given (p, q), create a network N with an edge of capacity 1 from x_i to y_j for each i, j, and let $\sigma(x_i) = p_i$ and $\partial(y_j) = q_j$. Unit capacity prevents multiple edges, and (p, q) is realizable if and only if N has a feasible flow.

It suffices to show that the stated condition on p and q implies the condition of Theorem 4.3.17. For $S \subseteq V(N)$, let $I(S) = \{i\colon x_i \in S\}$ and $J(S) = \{j\colon y_j \in S\}$. For a partition S, T of $V(N)$, we now have $\sigma(X \cap T) = \sum_{i \in I(T)} p_i$ and $\partial(Y \cap T) = \sum_{j \in J(T)} q_j$, and we have $c([S, T]) = |I(S)| \cdot |J(T)|$.