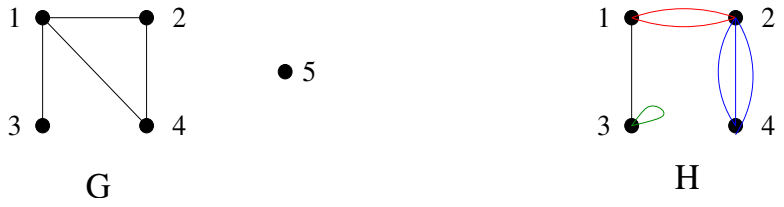
Math 725, Spring 2016
Lecture Notes

Contents

1. The Basics

1.1. Graphs.

Definition 1.1. A **graph** G is a pair (V, E), where V is a (finite), nonempty set of **vertices** and E is a (finite) set of **edges**. Each edge e is given by an unordered pair of two (possibly equal) vertices v, w, called its **endpoints**.



Equivalent statements:

- v, w are the **endpoints** of e; or
- v, w are **joined** by the edge e; or
- $e = vw$.

Technically, this last notation should only be used when e is the *only* edge joining v and w, but we often ignore that requirement for simplicity. Note that $e = wv$ is equivalent.

Sometimes, we don't want to bother to give the edge e a name; it is enough to know that there exists *some* edge joining v and w. Then we might say that v, w are **adjacent** or are **neighbors**. (It's tempting to say "connected" instead, but you should try to make a habit of resisting temptation, because that term properly means something else.)

Graphs can have **loops** (edges that join a vertex to itself) and **parallel edges** (edges with the same pairs of endpoints). Sometimes we want to exclude these possibilities, often because they are irrelevant. A graph with no loops and no parallel edges is called **simple**.

When studying graph theory, one quickly learns to be flexible about notation. For instance, when working with a single graph we want to use the concise symbols V and E for its vertex and edge sets; but if there are several different graphs around then it is clearer to write $V(G)$, $E(H)$, etc.

1.2. Isomorphisms and subgraphs.

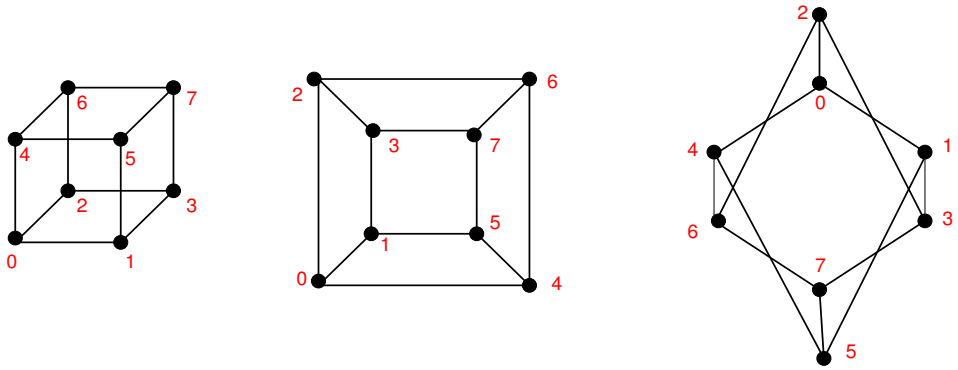
As in man fields of mathematics, one of our first orders of business is to say when two of the things we want to study are the same, and when one is a subthing of another thing.

Definition 1.2. Let G, H be graphs. An **isomorphism** is a bijection $f : V(G) \to V(H)$ such that for every $v, w \in V(G)$,

$$\#\{\text{edges of } G \text{ joining } v, w\} \;=\; \#\{\text{edges of } H \text{ joining } f(v), f(w)\}.$$

"$G \cong H$" means G, H are isomorphic.

Notice that this has nothing to do with what the graph looks like on the paper. **A drawing of a graph is not the same as the graph itself!** These three graphs are all isomorphic to each other; the red numbers indicate the isomorphism.



Think of an isomorphism as a relabeling, which doesn't really change the underlying structure of the graph.

Definition 1.3. An **isomorphism invariant** is a function ψ on graphs such that $\psi(G) = \psi(H)$ whenever $G \cong H$. (Equivalently, a function on equivalence classes of graphs.)

For example, the number of vertices is an invariant, as is the number of edges — but not the number of crossings when you draw the graph (although the minimum number of crossings among all possible drawings is indeed an invariant). Nor is a property like "Every edge has one vertex labeled with an odd number and one vertex labeled with an even number," since there's nothing to prevent me from shuffling the numbers to make this false. On the other hand, "The graph *can* be labeled so that every edge has one odd and one even label" is an invariant.

It is always possible to draw a given graph in lots of different ways, many geometrically inequivalent. It is crucial to remember that *a graph is a combinatorial object, not a geometric one.* That is, the structure of a graph really is given by its list of vertices, edges and incidences, *not* by a drawing composed of points and lines.

Definition 1.4. Let G be a graph. A **subgraph** of G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. For short we write $H \subseteq G$.

Note that it is *not* true that for every $X \subseteq V(G)$ and $F \subseteq E(G)$, the pair (X, F) is a subgraph of G, because it might not even be a graph—it needs to satisfy the condition that every endpoint of an edge in F belongs to X. (You can't have an edge dangling in the breeze — there needs to be a vertex on each end of it.)

Every subset $F \subseteq E(G)$ determines a subgraph whose vertex set is the set of all vertices that are endpoints of at least one edge in F. Also, every subset $X \subseteq V(G)$ determines a subgraph $G[X]$, the **induced subgraph**, whose edge set is the set of *all* edges of G with both endpoints in X. Being an induced subgraph is a stronger property than being a subgraph.

1.3. **Some applications of graph theory.** Graph theory has about a zillion applications. Here are a few.

Discrete optimization: a lot of discrete optimization problems can be modeled using graphs. For example, the TSP (traveling salesperson problem); the knapsack problem; matchings; cuts and flows.

Discrete geometry and linear optimization: the vertices and edges of a polytope P form a graph called its *1-skeleton*; when using the simplex method to solve a linear programming problem whose feasible region (i.e., the set of legal, although perhaps not optimal, solutions) is P, the 1-skeleton of P describes exactly the steps of the algorithm.

Algebra: the *Cayley graph* of a group G is a graph whose edges correspond to multiplication by one of a given set of generators; basic group-theoretic notions such as relations, conjugation, etc. now have natural descriptions in terms of the Cayley graph.

Topology: you can study an infinite and therefore complicated topological space by replacing it with a finite simplicial complex (a generalized kind of graph) from which you can calculate properties of the original space; also, deep graph-theoretic concepts such as deletion/contraction often have topological analogues.

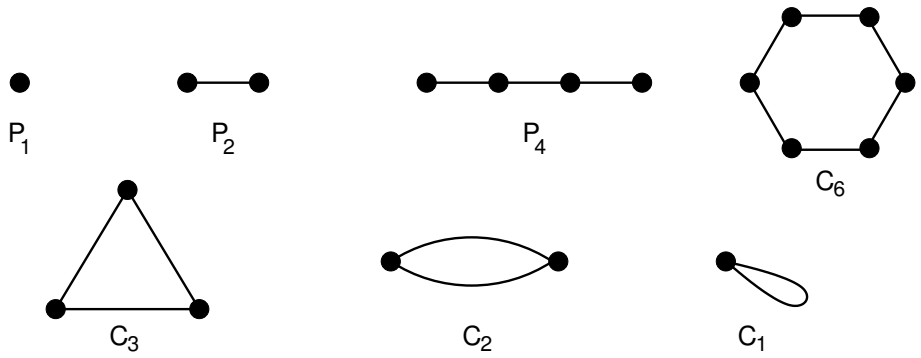
Theoretical computer science: many fundamental constructions such as finite automata are essentially glorified graphs, as are data structures such as binary search trees.

Chemistry: A molecule can be regarded as a graph in which vertices are atoms and edges are bonds. Amazingly, the chemical properties of a substance, such as its boiling point, can sometimes be predicted with great accuracy from the purely mathematical properties of the graph of the molecule!

Biology: More complicated structures like proteins can be modeled as graphs. The theory of rigidity of graphs has been used to understand how proteins fold and unfold.

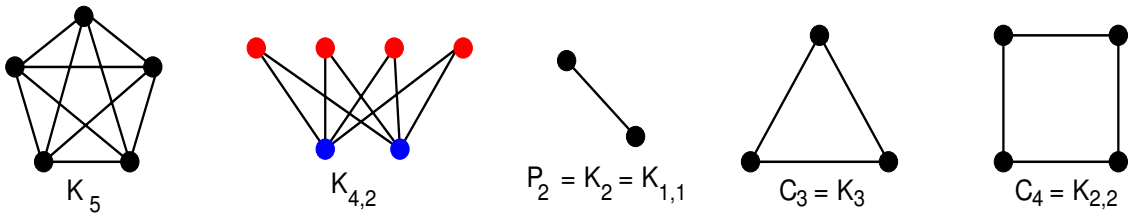
Not to mention the wonderful applicability of graphs to all manner of subjects including forestry, communications networks, efficient garbage collection, and evolutionary biology.

1.4. **Some important graphs and basic constructions.** The **path** P_n (Diestel: P^n) has n vertices and $n-1$ edges, connected sequentially. The **cycle** C_n (Diestel: C^n)has n vertices and n edges and can be drawn as a polygon.



The **complete graph** K_n (Diestel: K^n) has n vertices and one edge between each pair of vertices. Thus there are $\binom{n}{2} = \frac{n(n-1)}{2}$ edges in total. Often we assume that the vertex set is $[n] = \{1, 2, \ldots, n\}$. (This notation is standard in combinatorics.) A complete graph is also called a **clique**, particularly when it occurs inside another graph.

The **complete bipartite graph** $K_{p,q}$ has $p + q$ vertices, with p of them painted red and q painted blue, and an edge between each pair of differently colored vertices, for a total of pq edges.



The **empty graph** or $\bar{K}_n$ consists of n vertices and no edges. A copy of $\bar{K}_n$ appearing as an induced subgraph of a graph G is the same as a set of vertices of G of which no two are adjacent. Such a set is called a **coclique** (or **independent set** or **stable set**).

A few operations on graphs.

- If G is a simple graph, its **complement** $\bar{G}$ is the graph obtained by toggling adjacency and non-adjacency.
- The **underlying simple graph** G^{s} of any graph G is obtained by deleting all loops and all but one element of each parallel class of edges. Note that the connectivity relation on G^{s} is the same as that on G.
- The **disjoint union** $G + H$ is the union of G and H, assuming that the vertex sets are disjoint. For example, $\bar{K}_n + \bar{K}_m = \overline{K_{m+n}}$ and $\overline{K_{m,n}} = K_m + K_n$.
- The **join** $G * H$ also has vertex set $V(G) \cup V(H)$, but this time we add every possible edge between a vertex of G and a vertex of H.

1.5. **Vertex degrees and some counting formulas.** The number of vertices of a graph G is its *order*, often written $n(G)$. The number of edges is its *size*, written $e(G)$. Often when we are talking about a single graph G, we will just write n and e. Diestel uses $|G|$ for the order and $\|G\|$ for the size.

Definition 1.5. Let $G = (V, E)$ be a graph. The **degree** of a vertex v in G, written $d(v)$ or $d_G(v)$, is the number of edges of G having v as an endpoint (counting loops twice). The minimum and maximum degrees of a vertex in G are written $\delta(G)$ and $\Delta(G)$ (or δ and Δ).

Proposition 1.6 (Degree-Sum Formula / Handshaking Theorem)**.** *For every graph G,*

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

Proof. Each edge contributes 2 to each side of the equation. □

Corollary 1.7. *Every graph has an even number of vertices of odd degree.*

Corollary 1.8. *For every vertex v, $\delta(G) \leqslant d(v) \leqslant \Delta(G)$, so*

$$\delta \leqslant \frac{2e}{n} \leqslant \Delta.$$

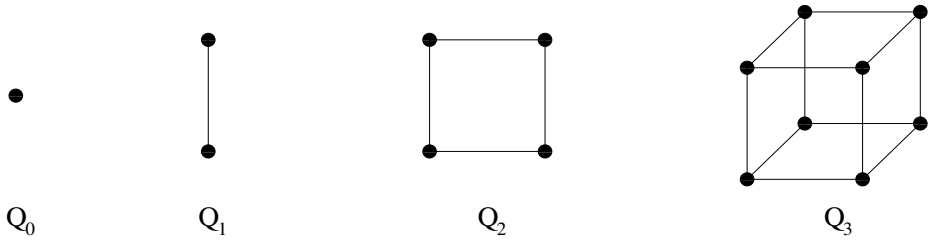
Definition 1.9. A graph G is ***d*-regular** if every vertex has degree d.

In this case equality holds in Corollary 1.8.

Corollary 1.10. *There are no regular graphs of odd degree and odd order.*

Example 1.11. The cycle C_n is 2-regular and the clique K_n is $(n-1)$-regular. An icosahedron has 12 vertices and is 5-regular, so $e = dn/2 = 5 \cdot 12/2 = 30$.

Example 1.12. The ***n*-dimensional cube** or **hypercube** Q_n is defined as follows. Let $V = 2^{[n]}$ be the power set of $[n]$ (so in particular $|V| = 2^n$), and let $E = \{ST \mid |S \triangle T| = 1\}$, where $\triangle$ denotes symmetric difference. This graph is called the *n-dimensional cube* or *hypercube* Q_n.



Note that $|V(Q_n)| = 2^n$ and it is regular of degree n (why?). Therefore, $|E(Q_n)| = n2^{n-1}$.

Equivalently, you can regard the vertices of Q_n as bit strings of length n, with two vertices adjacent if they agree in $n-1$ places. These two descriptions are isomorphic via associating a bit string $(b_1, \ldots, b_n)$ with the set $\{i \in [n] \mid b_i = 1\} \subseteq [n]$.

1.6. **Paths, trails, walks and cycles.**

Definition 1.13. Let $x, y \in V(G)$. A ***x*, *y*-walk** in G is an alternating sequence of vertices and edges

$$x = v_0,\ e_0,\ v_1,\ e_1,\ \ldots, v_{n-1},\ e_{n-1},\ y = v_n$$

where v_i, v_{i+1} are the endpoints of e_i for all i. The **length** of the walk is the number of edges, namely n. The vertices x, y are the **endpoints**; the other vertices are **internal** to the walk. The walk is **trivial** if $n = 0$.

It's not always necessary to specify all this data; e.g., we could just give the starting vertex and a sequence of edges. Or, if G has no parallel edges, we could just give the sequence of vertices.

Often we don't care about what the internal vertices are — in this case we can write just xWy (where technically W stands for $e_0, v_1, \ldots, v_{n-1}, e_{n-1}$). This makes it easy to concatenate walks: if xWy and $yW'z$ are walks, then so is $xWyW'z$. We'll write $\ell(W)$ for the length of W.

Definition 1.14. A walk is **closed** if $v_0 = v_n$. A **trail** is a walk with no repeated edges. A **path** is a walk with no repeated vertices. A **cycle** is a closed path.

These definitions of "path" and 'cycle" are consistent with the previous ones. A path in G of length n is the same thing as a subgraph of G isomorphic to P_{n+1}, and a cycle of length n is just a subgraph of G isomorphic to C_n.

Paths are the nicest kind of walks. Frequently, we are in a situation where we know how to walk from u to v, but what we really want is a u, v-path. Fortunately, if a walk is not a path, then it must contain some redundancy which can be eliminated, and repeating this process will eventually yield a path. To be precise:

Proposition 1.15. *If G has an x, y-walk, then it has an x, y-path.*

Proof. Let xWy be a walk. If some vertex z occurs more than once, then xWy has the form $xW'zW''zW'''y$, where W' and W''' may be trivial, but W'' is not. But then $xW'zW'''y$ is a strictly shorter x, y-walk (since its length is $\ell(W) - \ell(W'')$). Keep repeating this process until no further shortening is possible, which means that the walk is a path. □

Technically, the proof of Lemma 1.2.5 is an inductive argument, but I have phrased it instead as a recursive algorithm (which is really the same thing). The proof implies that every minimal-length walk is in fact a path.

Definition 1.16. Two vertices of G are **connected** if there is a path in G between them (equivalently, a walk). The graph G is **connected** if every pair of vertices u, v is connected. The **(connected) components** of G are its maximal connected subgraphs. The number of components is denoted $c(G)$.

Note that any two adjacent vertices are connected, but not every two connected vertices are adjacent.

Proposition 1.17. *The relation "u is connected to v" is an equivalence relation on $V(G)$, whose equivalence classes are the vertex sets of the connected components of G.*

Proof. Connectedness is reflexive (consider the trivial walk), symmetric (walks can be reversed), and transitive (walks can be concatenated). □

Proposition 1.18. *Let G be connected on n vertices. Then the vertices can be labeled $v_1, \ldots, v_n$ so that every induced subgraph $G_j := G[v_1, \ldots, v_j]$ is connected, for $1 \leqslant j \leqslant n$. In addition, v_1 can be chosen arbitrarily.*

Proof. Choose v_1 arbitrarily. Clearly $G_1 \cong K_1$ is connected. To construct G_{j+1} from G_j, choose any vertex $x \notin \{v_1, \ldots, v_j\}$ and find a path from v_1 to x. Take v_{j+1} to be the first vertex on this path not in G_j. □

Again, I have chosen to express the proof as an algorithm rather than a formal proof by induction.

Corollary 1.19. *If G is connected, then $e(G) \geqslant n(G) - 1$. More generally, $c(G) \geqslant n(G) - e(G)$ for all G.*

Proposition 1.20. *Let $a \in E(G)$, and let $G - a$ denote the graph obtained by removing a. If a belongs to a cycle in G, then $c(G - a) = c(G)$. Otherwise, $c(G - a) = c(G) + 1$. In the latter case, a is called a **bridge** or **cut-edge** or **isthmus** or **coloop** of G.*

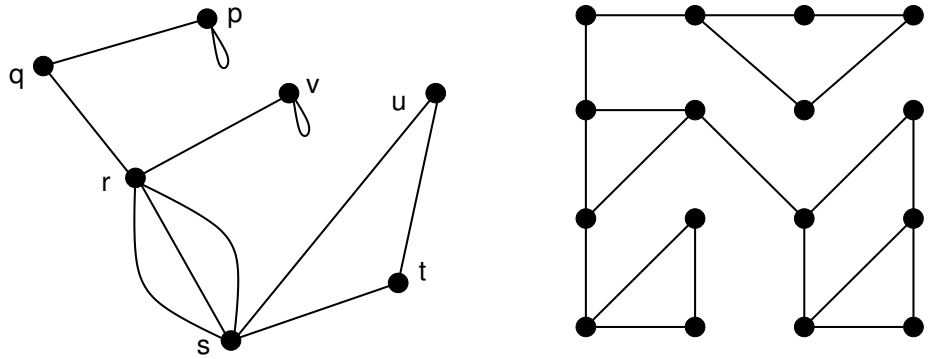
Proof. First, it is clear that every two vertices connected in $G - a$ are connected in G, so $c(G) \leqslant c(G - a)$.

Suppose that a belongs to a cycle, and let P be the path that constitutes the rest of the cycle. Then any two vertices that are connected in G are connected in $G - a$, because a can be replaced with P in any walk. Therefore the connectivity relations on G and $G - a$ are the same, and $c(G) = c(G - a)$.

Now suppose that a does not belong to any cycle. Then its two endpoints cannot be connected by any path $P \subseteq G - a$, for then $P \cup a$ would be a cycle in G containing a. So $c(G) > c(G - a)$. On the other hand, adding a to $G - a$ can only join two components into one. So $c(G) = c(G - a) + 1$. □

By the way, a *cut-vertex* is a vertex v such that $c(G - v) > c(G)$. (Synonyms: cutpoint, articulation point.) Here $G - v$ means the graph obtained by deleting all v and all its incident edges; equivalently, $G - v = G[V(G) \backslash v]$.

Example 1.21. In the connected graph G on the left below, q, r and s are cut-vertices; the others aren't. Note that $c(G - q) = c(G - s) = 2$ but $c(G - r) = 3$. The bridges are pq, qr, rv. We have $c(G - a) = 2$ for each bridge a.



Note that a loop cannot be a bridge, nor can any edge that has another parallel edge.

Example 1.22. A cycle has no cut-vertices or bridges. On the other hand, every internal vertex of a path (but not either of the endpoints) is a cut-vertex, and every edge is a bridge.

1.7. **Trees and forests.**

Definition 1.23. A graph is **acyclic**, or a **forest**, if it has no cycles. By Proposition 1.20, this is equivalent to the condition that every edge is a bridge. A connected forest is a **tree**.

Proposition 1.24. *A graph G is acyclic if and only if $c(G) = n(G) - e(G)$. In particular, every tree T has $n(T) - e(T) = 1$.*

Proof. Start with the vertex set $V(G)$ and no edges. This is certainly acyclic. and $c = n$ and $e = 0$. Now add edges one by one. Each time you do so, e increases by 1 and c might or might not decrease by 1. If c ever stays constant, you just created a cycle. Otherwise, every edge is a bridge, which means that you didn't create a cycle. □

The following corollary will be very useful (although not immediately).

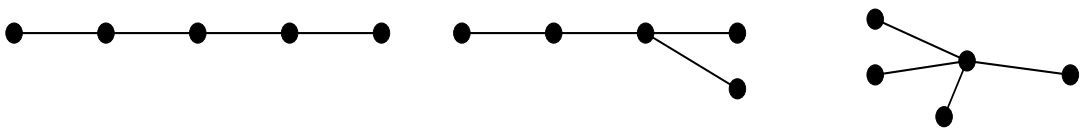
Corollary 1.25. *Every tree T with $n \geqslant 2$ vertices has exactly at least two **leaves** (vertices of degree 1).*

Proof. Handshaking says that

$$\sum_{v \in V(T)} d_T(v) = 2e(T) = 2n - 2.$$

If a sum of n positive integers equals $2n - 2$, then at least two of the summands must equal 1. □

Here are the three isomorphism classes of trees on 5 vertices:



Theorem 1.26. *(Characterizations of trees; Diestel Thm. 1.5.1) Let $G = (V, E)$ with $n = |V|$, $e = |E|$. TFAE:*

(1) G is a tree (i.e., connected and acyclic).
(2) G is connected and $e = n - 1$.
(2') G is minimally connected, i.e., $G - a$ is disconnected for every $a \in E$.
(3) G is acyclic and $e = n - 1$.
(3') G is maximally acyclic, i.e., $G + xy$ has a cycle for every nonadjacent $x, y \in V$.
(4) G has no loops, and for every $v, w \in V(G)$, there is exactly *one v, w-path in G.*

Proof. We've already proved that G is acyclic if and only if $c = n - e$. It's actually easy to prove from this that (1), (2), (3) are equivalent:

- If G is acyclic and connected, then $c = 1 = n - e$.
- If G is acyclic and $e = n - 1$, then $c = 1$, i.e., G is connected.
- If G is connected and $e = n - 1$, then in fact $c = 1 = n - e$, which means that G is acyclic.

The proof of (4 $\iff$ 1) is left as an exercise. □

Definition 1.27. Let G be connected. A **spanning tree** is a tree $T \subset G$ with $V(T) = V(G)$. (More generally, a **spanning subgraph** of G is a subgraph with the same vertex set, i.e., a subgraph obtained by deleting edges but not deleting any vertices.)

Every connected graph has at least one spanning tree. For example, you can find one by labeling the vertices as in Prop. 1.18 and keeping only the $n-1$ edges that join v_{j+1} to a previous vertex, for each $j \in [n-1]$. Or, you can repeatedly delete non-bridge edges until only a tree is left.

Some natural questions:

(1) How many spanning trees does a given graph have? This number $\tau(G)$ is an interesting measure of the complexity of the graph, and for many graphs there are amazing formulas for $\tau(G)$.
(2) How can you find the best spanning tree? Suppose each edge has a particular cost and you want to find the spanning tree that minimizes total cost.

We will come back to these things.

Frequently we want to think of one of the vertices r of a tree T as the **root**. In this case there is a partial order on vertices of the tree: $x \geqslant y$ if y lies on the unique xPr in T (i.e., xPr factors as $xP'yP''r$). For every $x \neq r$, the vertex adjacent to x in xPr is called its **parent**, denoted $p(x)$.

Theorem 1.28. *Let G be a connected simple graph and let $r \in V(G)$. There exists a spanning tree T with the property that for every $x \in V(G)$, the rPx in T is of minimum length over all r,x-paths in G. (Such a tree is called **normal** with respect to r, or a **breadth-first search tree**.)*

Proof. Here is some notation that will be useful. For each $x \in V(G)$, let $N(x)$ denote the set of all vertices adjacent to x, and let $N[x] = N(x) \cup \{x\}$. (The letter N stands for "neighborhood"; the parentheses and square brackets are intended to suggest open and closed neighborhoods respectively.) In addition, define

$$N^0[x] = \{x\}, N^2[x] = \bigcup_{y \in N[x]} N[y]$$

$$N^3[x] = \bigcup_{y \in N^2[x]} N[y]$$

$$\cdots$$

$$N^k[x] = \bigcup_{y \in N^{k-1}[x]} N[y]$$

$$\cdots$$

Equivalently, $N^k[x]$ is the set of vertices that are at distance at most k from r (i.e., are connected to r by a path of length at most k). Since G is connected and finite, we have $N^k[x] = V(G)$ for sufficiently large k.

Now, construct a spanning tree T with root r by the following algorithm.

0. Start by putting r in T.
1. For every $x \in N[r] \backslash \{r\}$, add the edge rx. (So $p(x) = r$ for all such x.)
2. Each $x \in N^2[r] \backslash N[r]$ has a neighbor y in $N[r]$. Add the edge xy, so that $p(x) = y$.
3. Each $x \in N^3[r] \backslash N^2[r]$ has a neighbor y in $N^2[r]$. Add the edge xy, so that $p(x) = y$.

$\cdots$

By induction on k, the vertices added at step k are exactly those at distance k from r. In other words, T is normal with respect to r. □

Some remarks:

(1) This definition of distance in fact makes G into a metric space.

(2) This algorithm can be souped up by assigning every edge e a positive real number $\ell(e)$ (think of this as "length" in a metric sense), and then defining the distance between two vertices to be the shortest possible total length of a path between them. In this form it is known as *Dijkstra's algorithm*, and is fundamental in computer science and discrete optimization. It is a theoretically efficient algorithm in the sense that its run time is polynomial in the numbers of vertices and edges.

1.8. **Bipartite graphs.**

Definition 1.29. A graph G is **bipartite** if $V(G) = X \cup Y$. where X, Y are cocliques. That is, every edge has one endpoint in each of X, Y. The pair X, Y is called a **bipartition** and the sets X, Y themselves are **partite sets** or **color classes**. Also, we might say for short that G is an ***X*, *Y*-bigraph.**

More generally, a graph G is ***k*-partite** if its vertex set is the disjoint union of k cocliques (also called partite sets).

- A graph is bipartite if and only if every one of its components is bipartite.
- A bipartite graph can't contain any loops (parallel edges are OK).
- Any subgraph of a bipartite graph is bipartite.
- Even cycles are bipartite but odd cycles are not.
- Q_n is bipartite. Remember that the edges of Q_n are pairs $S, T \in 2^{[n]}$ with $|S \triangle T| = 1$. For this to happen, one of S, T must have even cardinality and the other odd, so parity gives a bipartition.

Proposition 1.30 (Bipartite Handshaking)**.** *Let G be an X, Y-bigraph. Then*

$$\sum_{v \in X} d(v) = \sum_{v \in Y} d(v) = e(G).$$

Corollary 1.31. *If G is a regular X, Y-bigraph, then $|X| = |Y|$ (and in particular $|V(G)|$ is even).*

Bipartite graphs arise in lots of real-world applications, notably matching problems:

$X = \{\text{workers } w\}$, $Y = \{\text{shifts } s\}$,
$E = \{(w, s) \ : \text{worker } w \text{ is able to work shift } s\}$

$X = \{\text{job applicants } a\}$, $Y = \{\text{available jobs } p\}$,
$E = \{(a, p) \ : a \text{ is qualified for job } p\}$.

$V = \{\text{people}\}$, $E = \{\text{brother-sister pairs}\}$:
$X = \text{women}$, $Y = \text{men}$

Lemma 1.32. *Every closed odd walk contains an odd cycle.*

Proof. Suppose that we have a closed odd walk that is not itself an odd cycle. Then it has some repeated vertex, so it has the form $xWxW'x$. But $\ell(W) + \ell(W')$ is odd, so exactly one of W or W' is odd (say W), which means that xWx is a shorter closed odd walk. Repeating this, we eventually obtain an odd cycle. □

Proposition 1.33. *A graph is bipartite if and only if it contains no odd cycle.*

Proof. Odd cycles are non-bipartite, so no bipartite graph can contain an odd cycle.

Now suppose that G contains no odd cycle. We may as well assume that G is connected. Fix a vertex v and define

$$X = \{x \in V(G) \mid G \text{ has an even path } vPx\},$$
$$Y = \{y \in V(G) \mid G \text{ has an odd path } vP'y\}.$$

Then $X \cup Y = V(G)$ because G is connected. If $x \in X \cap Y$, then we have a closed walk $vPxP'v$ with $\ell(P)$ even and $\ell(P')$ odd, but by the Lemma, this means that G has an odd cycle, which is impossible. Hence $V(G) = X \cup Y$.

Suppose that two vertices $x, x' \in X$ are adjacent via an edge a. Then again we have a closed walk $vPxax'P'v$, of odd length $\ell(P) + \ell(P') + 1$ (since P, P' are even), which again is a contradiction. Hence X is a coclique. The same argument implies that Y is a coclique (here P, P' are both odd so again $\ell(P)+\ell(P')+1$ is odd). □

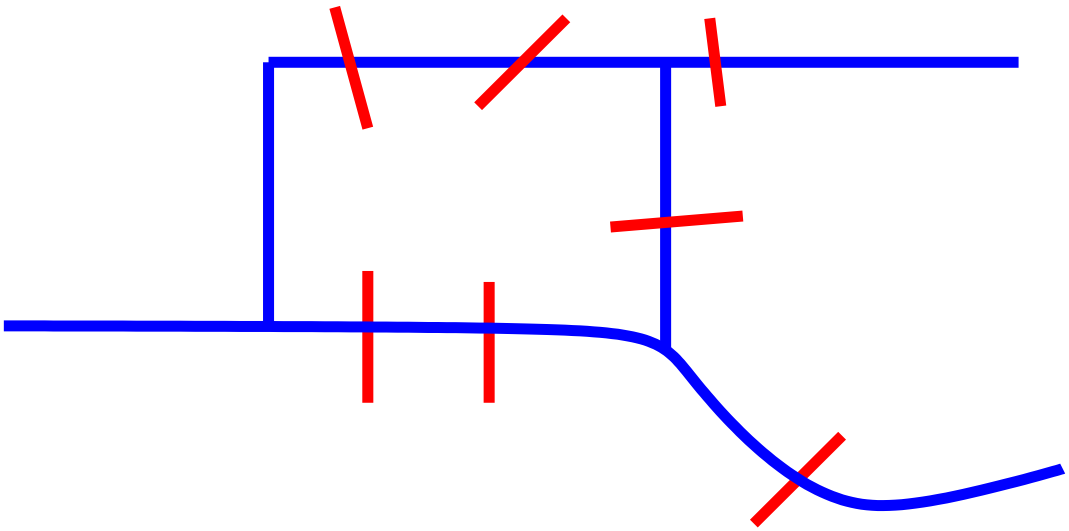
Note that this proof is essentially constructive: if G is bipartite, you can construct it by picking a starting vertex, coloring it blue, and walking around the graph, toggling your color between blue and red at every step. (For that matter, you can test bipartiteness easily by doing exactly this and seeing if it works.)

One way to think about this: Odd cycles are the minimal obstructions to being bipartite.

Corollary 1.34. *Acyclic graphs are bipartite.*

Proof. If you have no cycles, you certainly have no odd cycles! □

1.9. **Eulerian Graphs.** Königsberg Bridge Problem (Euler, 1737)



Definition 1.35. A **circuit** (or *tour*) in a graph is a closed trail, i.e., a walk that ends where it started and does not repeat every edge. An **Euler circuit** of a graph is a circuit using every edge. A graph is **Eulerian** if it has an Euler circuit.

Example: K_4 is not Eulerian. K_5 is.

— Removing or adding loops does not affect whether or not a graph is Eulerian.
— If G is Eulerian and disconnected, then it has at most one nontrivial component.
— So from now on, suppose that G is loopless and connected.

Theorem 1.36. *A connected graph G is Eulerian if and only if it is an* even graph*, i.e., every vertex has even degree.*

Proof. ($\implies$) Let W be an Euler tour. Then W leaves and enters each vertex the same number of times, and since it traverses each edge exactly once, every vertex must therefore have even degree.

($\Longleftarrow$) Let $W = x \cdots y$ be a trail of greatest possible length. I claim that W is in fact a circuit, i.e., $x = y$. Indeed, if $x \neq y$, then the number of edges of W incident to y is odd, but by the assumption that G is even, there must be at least one edge of $G \backslash W$ incident to y, which contradicts the assumption that W is of maximum length.

Now, suppose that W is not an Euler tour. Then there is some vertex v that has at least one edge in W and at least one edge $e = vu$ not in W. Say $W = xW'vW''x$; then $uevxW''xW'v$ is a trail — but it is longer than W which is a contradiction. □

Another method of proof is a little more constructive. By induction on the number of edges, every even graph decomposes as a (edge-)disjoint union of cycles (since erasing the edges in a cycle preserves evenness), and the cycles can be glued together to producer an Euler tour.

There is a simple method, called Fleury's algorithm, for constructing an Euler tour in an even connected graph. Start at any vertex and start taking a walk, erasing each edge after you traverse it. There is only one rule: cross a bridge only if it is the only option open to you.

1.10. **Matrices associated with graphs. [One note: Prove that every tree has at least two leaves. I think I forgot this last time.]**

Let G be loopless, $V(G) = \{v_1, \ldots, v_n\}$, and $E(G) = \{e_1, \ldots, e_r\}$.

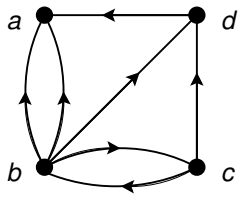
Definition 1.37. The **adjacency matrix** is the $n \times n$ matrix $A = A(G) = [a_{ij}]$, where a_{ij} is the number of edges joining vertices i and j. Note that $A^T = A$.

Fix an **orientation** on $E(G)$. That is, for each edge, call one of its vertices the **head** and the other the **tail**. What we have is now a **directed graph**, or **digraph**, which can be drawn by replacing each edge with an arrow pointing from the tail to the head.

Definition 1.38. The **incidence matrix** is the $n \times r$ matrix $B = B(G) = [b_{ve}]$, where

$$b_{ve} = \begin{cases} 1 & \text{if } v = \text{head}(e), \\ -1 & \text{if } v = \text{tail}(e), \\ 0 & \text{otherwise.} \end{cases}$$

Example 1.39. Let G be the graph as follows (actually, it's the Königsberg bridge graph):



Then

$$A(G) = \begin{bmatrix} 0 & 2 & 0 & 1 \\ 2 & 0 & 2 & 1 \\ 0 & 2 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \qquad B(G) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 1 \end{bmatrix},$$

Warning: Diestel defines these matrices A and B as living over $\mathbb{Z}_2$ instead of $\mathbb{R}$. This doesn't affect the behavior of A or B appreciably, and it has the advantage of making the orientation irrelevant (since $1 = -1 \mod 2$). However, in order to work with L you really have to work over $\mathbb{R}$.

Theorem 1.40. *Let $G = (V, E)$ be a connected graph, $H = (V, S)$ a spanning subgraph, and let $B(H) = \{B_e \mid e \in H\}$ be the corresponding set of columns of $B = B(G)$.*

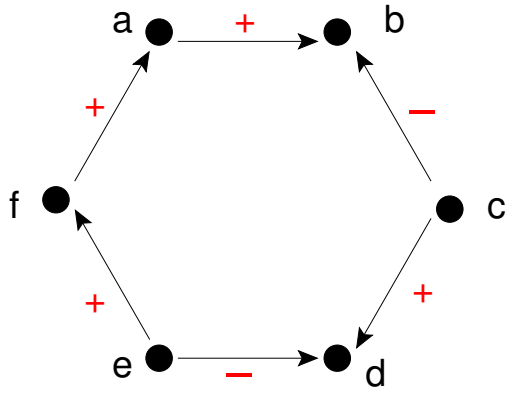
- *H acyclic $\iff$ $B(H)$ linearly independent;*
- *H connected $\iff$ $B(H)$ spans the column space of B;*
- *H is a spanning tree $\iff$ $B(H)$ is a column basis for B;*

Proof. First, notice that all of these properties are independent of the choice of orientation (since reorienting simply multiplies one or more columns by -1 without changing which sets of columns are linearly (in)dependent).

Suppose that C is a cycle. Traverse the cycle starting at any point, and keep track of whether you walk forward or backward (i.e., with or against the arrows). Then

$$\sum_{\text{forward } e \in C} B_e - \sum_{\text{backward } e \in C} B_e = 0.$$

For example, consider the 6-cycle shown below, traversed clockwise.



$$\begin{array}{c} \begin{array}{cccccc} +ab & -bc & +cd & -de & +ef & +fa \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \end{array}$$

Now, suppose that H has only one edge e incident to some vertex x. Then B_e is the only column in $B(H)$ to have a nonzero entry in the x row. Therefore, it is linearly independent in $B(H)$. By induction, it follows that if if H is acyclic, then $B(H)$ is a linearly independent set (remove leaves one by one).

In particular, if H is a spanning tree then the rank of $B(H)$ is $n - 1$. On the other hand, the rank of the entire incidence matrix B is no more than $n - 1$, since it has n rows and they are not linearly independent — their sum is zero. Hence every spanning tree corresponds to a column basis, and any edge set containing a spanning tree spans the column space.

Suppose that H has c components $H_1, \ldots, H_c$. Then the column spaces of $B(H_1), \ldots, B(H_c)$ are disjoint, so

$$\operatorname{rank} B(H) = \sum_{i=1}^{c} \operatorname{rank} B(H_i) = \sum_{i=1}^{c} n(H_i) - 1 = n - c$$

(since each $B(H_i)$ is connected). We have seen that $\operatorname{rank} B = n - 1$, so $B(H)$ is a spanning set[1] if and only if H is connected. □

[1]Unfortunately, the term "spanning" can cause problems. In the graph theory context it refers to a subgraph that contains all the vertices of its parent graph; in the linear algebra it refers to a collection of vectors that span a subspace. Be careful.

2. Counting Spanning Trees (Not in Diestel)

2.1. **Deletion and contraction.** Let $\mathscr{T}(G)$ denote the set of spanning trees of G, and let $\tau(G) = |\mathscr{T}(G)|$ be the number of spanning trees.

G	$\tau(G)$
any tree	1
C_n	n
K_3	3
K_4	16
$K_{2,3}$	12
$K_{3,3}$	81
Q_3	384
Pete	2000

Definition 2.1. Let G be a graph and $e \in E(G)$ an edge. The **deletion** $G - e$ is the graph obtained by erasing e, leaving its endpoints (and everything else) intact. The **contraction** G/e is obtained by erasing e and merging its endpoints into a single vertex. (Contraction is not defined if e is a loop.) Note:

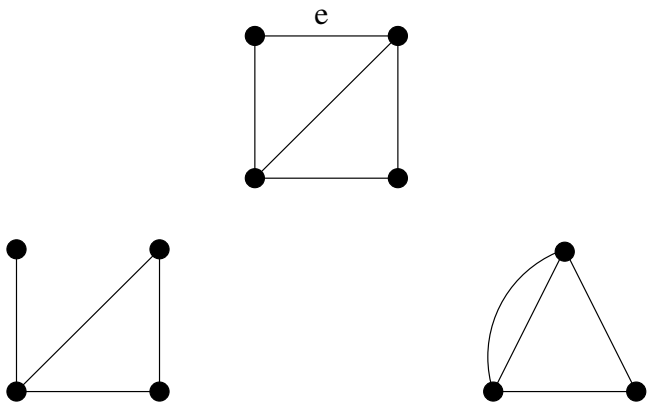
$$n(G-e) = n(G), \qquad n(G/e) = n(G) - 1,$$
$$e(G-e) = e(G) - 1, \qquad e(G/e) = e(G) - 1.$$

Two kinds of edges are special:

- If e is a loop, then it can't belong to any spanning tree. So $\mathscr{T}(G) = \mathscr{T}(G-e)$ and $\tau(G) = \tau(G-e)$.
- If e is a bridge, then it belongs to every spanning tree (since you can't have a connected spanning subgraph without it). In fact $\tau(G) = \tau(G/e)$.

By contrast, each "ordinary" edge (one that is neither a loop nor a bridge) belongs to at least one spanning tree, but not to all spanning trees. More specifically:

Theorem 2.2. *If $e \in E(G)$ is not a loop, then $\tau(G) = \tau(G-e) + \tau(G/e)$.*



Proof. We will find bijections

$$\{T \in \mathscr{T}(G) : e \notin E(T)\} \to \mathscr{T}(G-e) \qquad \text{and}$$
$$\{T \in \mathscr{T}(G) : e \in E(T)\} \to \mathscr{T}(G/e).$$

The first bijection is the easy one: a spanning tree of G not containing e is the same thing as a spanning tree of $G - e$.

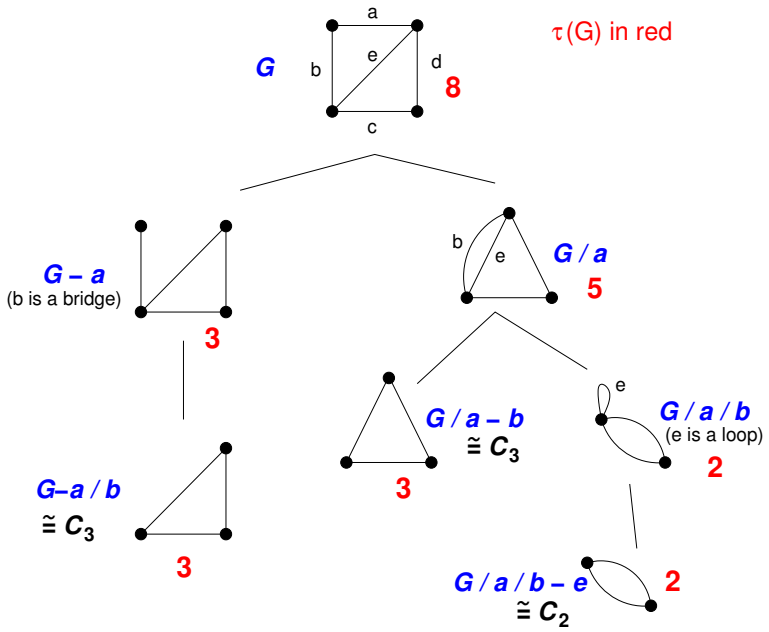
For the second bijection, if T is a spanning tree of G containing e, then $T' = T/e$ is a spanning tree of G/e. Indeed, T' is connected because T is, and

$$e(T') = e(T) - 1 = (n(G) - 1) - 1 = n(G/e) - 1.$$

On the other hand, given any spanning tree $T' \in \mathscr{T}(G/e)$, the corresponding edges of T, together with e itself, form a spanning tree of G. □

Remark 2.3. The recurrence even works when e is a bridge (because $G - e$ is disconnected, hence has zero spanning trees) or even if e is a loop (well, in a silly way: G/e is undefined, so it isn't even a graph and then doesn't have any spanning trees).

Example: By repeatedly applying deletion/contraction, we can calculate $\tau(G)$ of any graph. Here's the calculation for the "diamond graph" obtained by removing an edge from K_4.



We start by applying deletion/contraction to edge a. On the left side, $G - a$ has a bridge b, and contracting it gives a 3-cycle, which we know has $\tau = 3$. On the right side, G/a has neither a loop nor a bridge, so we recurse again, deleting and contracting edge b. The deletion is another 3-cycle, and contracting gives a 2-cycle plus a loop. So $\tau(G/a) = \tau(G/a - b) + \tau(G/a/b) = 3 + 2 = 5$, and then $\tau(G) = \tau(G - a) + \tau(G/a) = 3 + 5 = 8$.

The bad news is that computing $\tau(G)$ by deletion/contraction takes exponential time, essentially $O(2^{e(G)})$, because each instance of the recursion contributes a factor of 2. So this is not a good way to compute $\tau(G)$ in practice, although there are some families of graphs for which you can find interesting recurrences for $\tau(G)$ (see problem set). In the next section we will see a computationally efficient way of calculating $\tau(G)$: the Matrix-Tree Theorem, which exploits linear algebra.

2.2. **The Matrix-Tree Theorem.**

Definition 2.4. The **Laplacian matrix** is the $n \times n$ matrix

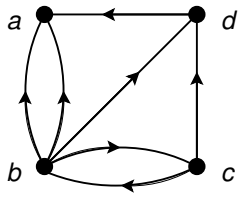
$$L = L(G) = BB^T$$

where B is the incidence matrix. Note that $L = D - A$, where D is the diagonal matrix of vertex degrees. Also, the choice of orientation does not affect the Laplacian.

Example: The graph $G = K_4 - e$ has

$$A = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ -1 & -1 & 0 & 2 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 \\ 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 \end{bmatrix}, \qquad L = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ -1 & -1 & 0 & 2 \end{bmatrix}.$$

For the Königsberg bridge graph, A, B, L are as follows:



$$A = \begin{bmatrix} 0 & 2 & 0 & 1 \\ 2 & 0 & 2 & 1 \\ 0 & 2 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 1 \end{bmatrix}, \qquad L(G) = \begin{bmatrix} 3 & -2 & 0 & -1 \\ -2 & 5 & -2 & -1 \\ 0 & -2 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}.$$

In general, for any loopless graph G, $L(G) = BB^T$ is a symmetric, positive-semi-definite matrix of rank $n(G) - c(G)$, with entries as follows:

$$\ell_{ij} \quad = \quad \text{dot product of } i^{th} \text{ and } j^{th} \text{ rows of } B \quad = \quad \begin{cases} d_G(i) & \text{if } i = j, \\ -m_{ij} & \text{if } i, j \text{ share } m_{ij} \text{ edges}, \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 2.5 (Matrix-Tree Theorem, Kirchhoff 1845)**.** *Let $n \geqslant 2$, let G be a loopless graph on vertex set $[n]$, let $i, j \in V(G)$, and let $L^{i,j}$ be the "reduced Laplacian" matrix obtained by deleting the i^{th} row and j^{th} column of $L(G)$. Then:*

(1) $\tau(G) = (-1)^{i+j} \det L^{i,j}$. (In particular, if $i = j$, then the sign is $+1$.)
(2) Let the nonzero eigenvalues of G be $\lambda_1, \ldots, \lambda_{n-1}$. Then

$$\tau(G) \quad = \quad \frac{\lambda_1 \cdots \lambda_{n-1}}{n}.$$

Example: For $G = K_4 - e$, we have

$$L^{1,1}(G) \quad = \quad \begin{bmatrix} 3 & -1 & -1 \\ -1 & 2 & 0 \\ -1 & 0 & 2 \end{bmatrix}$$

and $\det L^{1,1} = 8 = \tau(G)$, which is the answer we had gotten by deletion-contraction.

OTOH, if you go ahead and diagonalize L (which you always can, because it's symmetric), you get

$$\begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

i.e., the eigenvalues are 4,4,2,0, which says that the number of spanning trees is $4 * 4 * 2/4 = 8$.

Note: The number of nonzero eigenvalues, counting multiplicities, is always $n - 1$ (provided G is connected), and they are always positive real numbers (because $L(G)$ is symmetric). But they don't have to be integers.

Example 2.6. *Cayley's formula* says that $\tau(K_n) = n^{n-2}$. This can be proven by the Matrix-Tree Theorem.

$$L = L(K_n) = \begin{bmatrix} n-1 & -1 & \cdots & -1 \\ -1 & n-1 & \cdots & -1 \\ \vdots & \vdots & & \vdots \\ -1 & -1 & \cdots & n-1 \end{bmatrix}_{n \times n}$$

and $L^{n,n}$ looks the same, only it's an $(n-1) \times (n-1)$ matrix. What is $\det L^{n,n}$? Don't expand the determinant! Instead, think about what the eigenvectors might look like. The all-1's vector is an eigenvector, with eigenvalue 1. And any vector with a 1 in one place and a -1 in another place is an eigenvector with eigenvalue $(n-1) - (-1) = n$. These span an eigenspace of dimension $n-2$. Hence, by the MTT,

$$\tau(K_n) = \det L^{n,n} = n^{n-2}.$$

There are other graphs for which all eigenvalues are integers, such as Q_n and $K_{m,n}$. But it's a rare property.

Proof #1 of Matrix-Tree Theorem (i). Proceed by double induction on $n = n(G)$ and $r = e(G)$.

If $n = 2$, then G consists of r parallel edges, any one of which is a spanning tree. So $\tau(G) = r$. Meanwhile,

$$L = \begin{bmatrix} r & -r \\ -r & r \end{bmatrix}, \qquad L^{1,1} = L^{2,2} = [r], \qquad L^{1,2} = L^{2,1} = [-r].$$

If $r = 0$, then G is not connected, so $\tau(G) = 0$, and meanwhile $L(G)$ is the zero matrix.

Now suppose that $n > 2$ and $r > 0$, and that the MTT holds for all graphs with either fewer vertices, or n vertices and fewer edges. Let $e \in E(G)$; assume WLOG that its endpoints are $1, n$. Note that $L(G)$ and $L(G-e)$ are almost the same:

$$\ell_{ij}^{G-e} \;=\; \big[L(G-e)\big]_{ij} \;=\; \begin{cases} \ell_{ij}^{G} - 1 & \text{if } i = j = 1 \text{ or } i = j = n, \\ \ell_{ij}^{G} + 1 & \text{if } \{i,j\} = \{1,n\}, \\ \ell_{ij}^{G} & \text{otherwise.} \end{cases}$$

When we delete the n^{th} row and column, we obtain reduced Laplacians that differ only in one entry, namely

$$\ell_{1,1}^{G} \;=\; \ell_{1,1}^{G-e} + 1.$$

Therefore, if we evaluate each of $\det \tilde{L}(G)$ and $\det \tilde{L}(G-e)$ by expanding on the top row, then the calculations are almost the same; the difference is

$$\det \tilde{L}(G) - \det \tilde{L}(G-e) \;=\; \det \begin{bmatrix} \ell_{2,2}^{G} & \cdots & \ell_{2,n-1}^{G} \\ \vdots & & \vdots \\ \ell_{n-1,2}^{G} & \cdots & \ell_{n-1,n-1}^{G} \end{bmatrix}.$$

But that matrix is precisely the reduced Laplacian of G/e obtained by deleting the row and column indexed by the merged vertex. (The degrees of nonmerged vertices aren't affected by the contraction, nor are edges between two nonmerged vertices.) Thus

$$\det \tilde{L}(G) = \tau(G-e) + \tau(G/e) = \tau(G)$$

by induction and the deletion-contraction recurrence (Theorem 2.2). □

Proof #2 of Matrix-Tree Theorem (ii). This proof uses the *Binet-Cauchy Theorem*, a linear algebra fact that we will use as a black box.

The Binet-Cauchy Formula

Let $m \geqslant p$, $A \in \mathbb{R}^{p\times m}$, $B \in \mathbb{R}^{m\times p}$, so $AB \in \mathbb{R}^{p\times p}$.
For $S \subset [m]$, $|S| = p$, let
$A_S = p \times p$ submatrix of A with columns S
$B_S = p \times p$ submatrix of B with rows S
Then

$$\det AB = \sum_S (\det A_S)(\det B_S).$$

Let N be the "reduced incidence matrix" formed by deleting a row from the signed incidence matrix M. Observe that $NN^T = L^{1,1}$.

Let S be a set of $n-1$ edges of G, and consider the corresponding columns of N. Note that S either contains a cycle or is a spanning tree. As noted before:

- If S contains a cycle, then the columns are linearly dependent.
- If S is acyclic (hence is a spanning tree), then every $(n-1)\times(n-1)$ submatrix of N with columns S has determinant ± 1.

Now we can apply Binet-Cauchy with $p = n-1$, $m = e$, $A = N$, $B = N^T$.

$$\begin{aligned}
\det L^{1,1}(G) \;=\; \det NN^T \;&=\; \sum_{S\subseteq E(G):\ |S|=n-1} (\det N_S)(\det N_S^T) \qquad \text{(Binet-Cauchy)} \\
&=\; \sum_S (\det N_S)^2 \\
&=\; \sum_S \begin{cases} 1 & \text{if } S \text{ is a tree} \\ 0 & \text{if it isn't} \end{cases} \;=\; \tau(G).
\end{aligned}$$

□

2.3. **The Prüfer Code.**

Theorem 2.7. *There is a bijection*

$$P:\ \mathscr{T}_n \to [n]^{n-2} = \{(p_1,\ldots,p_{n-2}):\ p_i \in [n]\},$$

called the ***Prüfer code****, such that for every vertex v,*

$$\deg_T(v) = 1 + \#\{i \in [n-2]:\ p_i = v\}.$$

Cayley's formula $\tau(K_n) = n^{n-2}$ is an immediate corollary, as is an even more refined count of trees called the **Cayley-Prüfer formula**. Here's the idea:

- Peel off leaves, one by one, choosing the smallest available leaf each time.
- Keeping track of which leaf is deleted is not enough information to recover the tree; we need to keep track of the stem (unique neighbor of the deleted leaf)
- The list of stems is enough information to recover the tree.

Here is a pseudocode algorithm for computing $P(T)$:

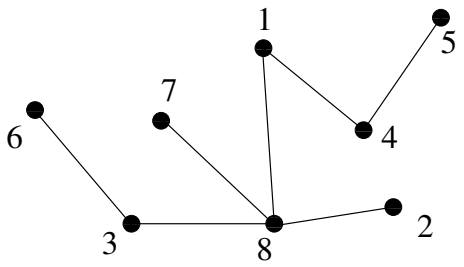
Input: $T \in \mathscr{T}_n$
Output: $P(T) \in [n]^{n-2}$

$T_0 := T$
for i from 1 to $n-2$ do
{
    $y_i :=$ smallest leaf of T_{i-1}
    $p_i :=$ unique neighbor ("stem") of y_i
    $T_i := T_{i-1} - p_i$
}
$P(T) = (p_1, \ldots, p_{n-2})$

Here it is in Sage:

```
def PruferCode(T):  ##  assume that T is a tree on 2 or more vertices
    U = deepcopy(T)
    P = []
    while U.num_verts() > 2:
        Leaves = [v for v in U.vertices() if U.degree(v) == 1]
        y = min(Leaves)
        p = U.neighbors(y)[0]
        P.append(p)
        U.delete_vertex(y)
    return P
```

Before proving that this algorithm gives a bijection, let's do an example. Let $n = 8$ and let T be the tree shown.



Step 1:	Leaves: $2, 5, 6, 7$.	Delete $y_1 = 2$, write down $\ell_1 = 8$.
Step 2:	Leaves: $5, 6, 7$.	Delete $y_2 = 5$, write down $\ell_2 = 4$.
Step 3:	Leaves: $4, 6, 7$.	Delete $y_3 = 4$, write down $\ell_3 = 1$.
Step 4:	Leaves: $1, 6, 7$.	Delete $y_4 = 1$, write down $\ell_4 = 8$.
Step 5:	Leaves: $6, 7$.	Delete $y_5 = 6$, write down $\ell_5 = 3$.
Step 6:	Leaves: $3, 7$.	Delete $y_6 = 3$, write down $\ell_6 = 8$.

There's only one edge left, so we are done; the Prüfer code is (8, 4, 1, 8, 3, 8).

Lemma 2.8. *For all $v \in V(T)$, we have* $\deg_T(v) = 1 + \#\{i : \ p_i = v\}$.

Proof. Every vertex v eventually becomes a leaf (either it is deleted, or it is one of the two remaining ones). To make v into a leaf, we need to remove $\deg_T(v) - 1$ of its neighbors, so v will occur exactly that many times in $P(T)$. □

Proof of Theorem 2.7. Now suppose you are given the Prüfer code of a tree $T \in \mathscr{T}_8$. I claim that you can reconstruct the leaves $\ell_1, \ldots, \ell_n$, hence T. We'll do this with the running example $P(T) = (p_1, \ldots, p_6) = (8, 4, 1, 8, 3, 8)$.

- The first leaf deleted must have been $\ell_1 = 2$, because it is the smallest vertex not in $P(T)$. I.e.,
$$\ell_1 = \min\left([n]\backslash\{p_1, p_2, \ldots, p_{n-2}\}\right).$$

- The second leaf deleted must have been $\ell_2 = 5$, because it is the smallest vertex that is not ℓ_1 (hence is a vertex of $T - \ell_1 p_1$) and does not appear in $\{p_2, \ldots, p_{n-2}\}$ (hence is a leaf of $T - \ell_1 p_1$). That is,
$$\ell_2 = \min\left([n]\backslash\{\ell_1, p_2, \ldots, p_{n-2}\}\right).$$

By the same reasoning, the third leaf deleted must have been
$$\ell_3 = \min\left([n]\backslash\{\ell_1, \ell_2, p_3, \ldots, p_{n-2}\}\right).$$
and in general, for all $i \in [n-2]$, we have
$$\boxed{\ell_i = \min\left([n]\backslash\{\ell_1, \ldots, \ell_i, p_{i+1}, \ldots, p_{n-2}\}\right).}$$

- Thus $\ell_1, \ldots, \ell_{n-2}$ are all distinct vertices, and the edges $\ell_i p_i$ include all but one of the edges of T. The other edge is the one left when the algorithm finishes; its endpoints are the two vertices that were never deleted, i.e., the elements of $[n]\backslash\{\ell_1, \ldots, \ell_{n-2}\}$. So we can recover T from $P(T)$. On the other hand, starting with an arbitrary sequence $(p_i) \in [n]^{n-2}$ and constructing the sequence (ℓ_i) yields a tree T such that $P(T) = (p_i)$, so we have a bijection. □

Corollary 2.9 (Cayley-Prüfer Formula)**.**
$$\sum_{T \in \mathscr{T}_n} \prod_{j \in [n]} x_j^{d_T(j)} \;=\; x_1 \cdots x_n \left(x_1 + \cdots + x_n\right)^{n-2}.$$

Proof. This is a straight calculation using Lemma 2.8.
$$\begin{aligned}
\sum_{T \in \mathscr{T}_n} \prod_{i \in [n]} x_i^{d_T(i)} &= x_1 \cdots x_n \sum_{P=(p_1,\ldots,p_{n-2}) \in [n]^{n-2}} \prod_{i=1}^{n-2} x_{p_i} \\
&= x_1 \cdots x_n \sum_{p_1 \in [n]} \sum_{p_2 \in [n]} \cdots \sum_{p_{n-2} \in [n]} \prod_{i=1}^{n-2} x_{p_i} \\
&= x_1 \cdots x_n \left(\sum_{p_1 \in [n]} x_{p_1}\right)\left(\sum_{p_2 \in [n]} x_{p_1}\right) \cdots \left(\sum_{p_{n-2} \in [n]} x_{p_{n-2}}\right) \\
&= x_1 \cdots x_n \left(x_1 + \cdots + x_n\right)^{n-2}.
\end{aligned}$$
□

The idea of the Prüfer code can be extended to many other general kinds of graphs: complete bipartite graphs, complete multipartite graphs, and more. For a very general construction, see A. Kelmans, "Spanning trees of extended graphs", *Combinatorica* **12** (1992), 45–51.

2.4. **MSTs and Kruskal's algorithm.** Let $G = (V, E)$ be a loopless graph equipped with a weight function $\mathrm{wt} : E \to \mathbb{R}_{\geqslant 0}$. For a subset $A \subseteq E$, define
$$\mathrm{wt}(A) = \sum_{e \in A} \mathrm{wt}(e).$$
How do we find a spanning tree of minimum total weight?

The most naive algorithm is as follows. Find the cheapest edge and color it green. Then find the next cheapest edge and color it green (provided it isn't parallel to the first edge). Then find the next cheapest edge and color it green (provided it isn't parallel to either of the first two edges, or complete a C_3). Keep coloring the cheapest edge available green, provided you never complete a cycle.

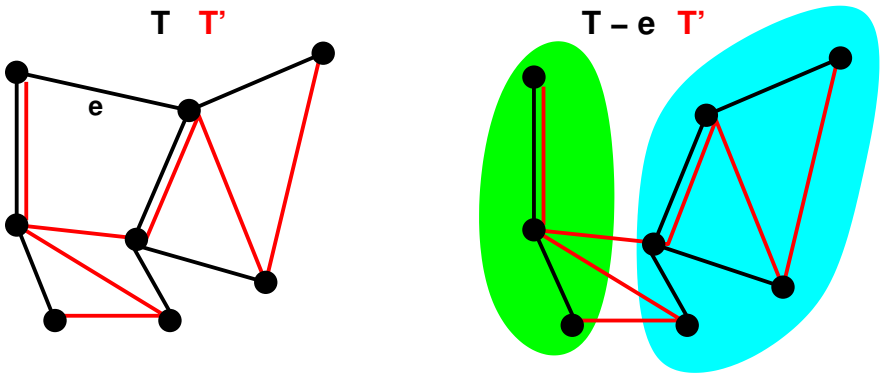
This procedure, called *Kruskal's algorithm*, is very easy to understand and implement, but it is not clear that it works. But, amazingly, it does. The key to the proof is understanding the structure of the family of spanning trees of a graph G. We already know that all spanning trees have the same number of edges, but not just any family of sets of the same size can be $\mathscr{T}(G)$ for some G. In fact, any two trees interact in a very specific way: his procedure, called *Kruskal's algorithm*, is very easy to understand and implement, but it is not clear that it works. But, amazingly, it does.

Proposition 2.10 (Exchange rules for spanning trees)**.** *Let G be connected on n vertices, and let T, T' be spanning trees of G. Then:*

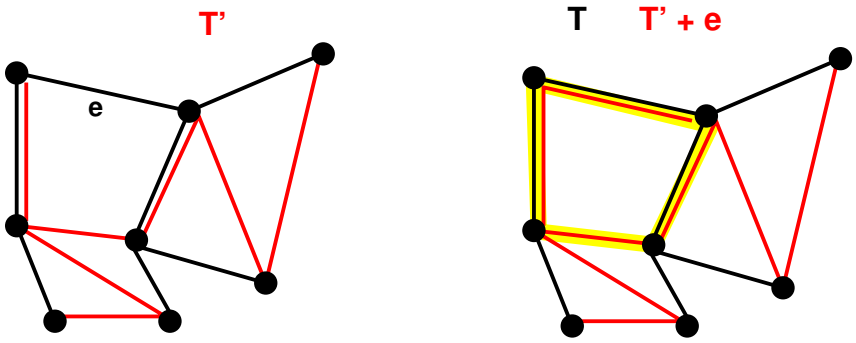
(1) For each $e \in E(T) - E(T')$, there exists $e' \in E(T') - E(T)$ such that $T - e + e'$ is a spanning tree.
(2) For each $e \in E(T) - E(T')$, there exists $e' \in E(T') - E(T)$ such that $T' + e - e'$ is a spanning tree.

(Note: I am using $+$ for the operation of adding an edge. This is different from $G + H$ which would denote the disjoint union of two graphs.)

Proof. (1): $T - e$ has exactly two components (shown in green and blue in the figure below). It suffices to choose e' to be an edge of T' with one endpoint in each component of $T - e$. Such an edge must exist because T' is connected.



(2): $T' + e$ has a cycle (since trees are maximally acyclic graphs); call it C. It is shown in yellow below. Then $C \nsubseteq T$ (because T is acyclic), so pick $e' \in C \backslash T$. Then $T' + e - e'$ is still connected and has $n - 1$ edges, hence is a spanning tree. □



You'll actually prove a stronger fact on HW #2: given $e \in E(T) - E(T')$, there exists $e' \in E(T') - E(T)$ such that $T - e + e'$ and $T' + e - e'$ are *both* spanning trees of G.

Here is a precise statement of Kruskal's algorithm.

Input: connected graph G with weight function w
Output: a MST T

T_0 := $\varnothing$
$i = 0$
A := E # available edges
while (V, T_0) is disconnected and $A \neq \varnothing$ do
    Choose $e \in A$ of minimum weight
    A := $A - e$
    if $T_i + e$ is acyclic:
        Set T_{i+1} := $T_i + e$
        Set i := $i + 1$
Output $T = T_i$

Theorem 2.11. *The output T of Kruskal's algorithm is a MST of G.*

Proof. First, we check that the output is really a spanning tree. By construction, T is acyclic. If it is disconnected, then the algorithm made a mistake — pick any edge e that is a bridge of $T + e$; that was still true at whatever stage in the algorithm e was considered (since $T_i \subset T$) so e should have been added, but wasn't.

Now comes the clever part. Let T^* be some MST (certainly one must exist since $\mathscr{T}(G)$ is finite). If $T^* = T$, then we're done.

Otherwise, let e be the first edge chosen for T that is not in T^*. Let

$$F := \{f \in E(T) : \ f \text{ was added earlier than } e\}.$$

- By the choice of e we have $F \subseteq E(T^*)$.
- By Prop. 2.1.7, we can choose $e' \in E(T^*) - E(T)$ so that $T^{**} = T^* - e' + e$ is a spanning tree.
- Note that e' could not have been available at the stage of the algorithm when e was added to T. Otherwise it would have been added to T (since $F + e' \subset E(T^*)$ is acyclic) and then would be in F (by the choice of e), hence in T^*, but it isn't.
- Therefore, e' was considered after e, so $\mathrm{wt}(e') \geqslant \mathrm{wt}(e)$.
- In particular, $\mathrm{wt}(T^{**}) \leqslant \mathrm{wt}(T^*)$.
- But T^* is a MST, so equality must hold, which means that in fact T^{**} is a minimum spanning tree.

We have shown: if G has a MST T^* with $|T \cap T^*| = k < |T|$,
then it has a MST T^{**} with $|T \cap T^{**}| = k + 1$.

By induction (or iteration, if you prefer), T itself must be a MST. □

Some notes on Kruskal's Algorithm:

1. Computational issues. If you are going to implement Kruskal's Algorithm, it is best to first sort the edges in increasing order by weight. Probably the best way to check whether an edge can be added is to check that its endpoints are in different components. This means keeping track of which vertices lie in a common component, and updating that data whenever the algorithm successfully adds an edge.

2. Matroids. Note that the exchange rules (Proposition 2.10) hold in a more general context. Let
$E =$ ~~set of edges~~ set of vectors spanning a vector space S
$T, T' =$ ~~spanning trees~~ subsets of E that are bases for S
Then Kruskal's Algorithm can be used to find a minimum-weight basis. More generally, let

$E =$ any finite set,

$w : E \to \mathbb{R}$,

$\mathcal{B} =$ collection of subsets of E of the same size satisfying the exchange properties.

In this case $\mathcal{B}$ is called a **matroid basis system**, and Kruskal's Algorithm can be used to find an element of $\mathcal{B}$ of minimum weight. In fact, matroids can be characterized as set systems for which Kruskal's algorithm works. Matroids are of high importance both in combinatorial optimization and in algebraic combinatorics.

3. Prim's Algorithm is another way to efficiently compute an MST. It works like this:

```
Pick an arbitrary vertex v
X := {v}
T := ∅
while X ≠ V do
     Choose e = xy of minimum weight such that x ∈ X, y ∉ X
     T := T + e
     X := X + y
Output T
```

This method also produces an MST (proof omitted). It has the advantage of being somewhat easier to implement than Kruskal's algorithm, because it is easier to keep track of the single vertex set X than the component structure of a graph. On the other hand, Prim's algorithm is specifically about graphs and cannot be extended to matroids (the concept "cycle" has an analogue for matroids, but "vertex" doesn't).

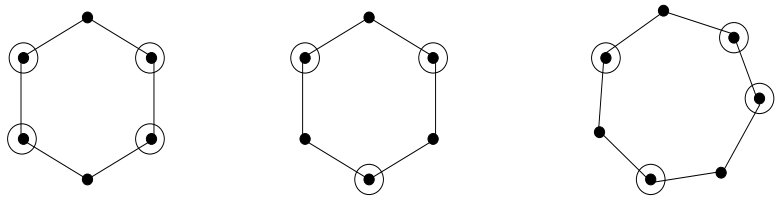
3. Matchings and Covers

Throughout this section, $G = (V, E)$ will be a connected simple graph. We will not generally distinguish between an edge set $A \subseteq E(G)$ and the corresponding spanning subgraph $(V(G), A)$. Doing so is usually more trouble than it's worth. Sometimes we'll need to say whether we mean to include all vertices, or just the set of vertices $V(A)$ incident to at least one edge of A. But the meaning of terms such as "acyclic" and "component" should be clear when they are applied to edge sets. Maybe this warning should be put in the first section of the notes next time.

3.1. Basic definitions.

Definition 3.1. A **vertex cover** of G is a set $Q \subseteq V(G)$ that contains at least one endpoint of every edge. An **edge cover** of G is a set $L \subseteq E(G)$ that contains at least one edge incident to every vertex.

Here are some pictures of vertex covers.

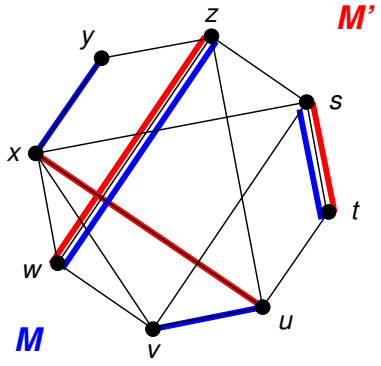


Warning: "Minimal" and "minimum" don't mean the same thing. In the first cover shown above ($G = C_6$, $Q = \{1, 2, 4, 5\}$), the cover is minimal because no proper subset is a cover, but it is not minimum because G has a cover of strictly smaller cardinality.

Of course, V itself is always a vertex cover, and E is always an edge cover (provided that G has no isolated vertices). The interesting problem is to try to find covers that are as small as possible.

Definition 3.2. A **matching** on G is an edge set $M \subset E$ that includes at most one edge incident to each vertex. A vertex is **matched** (or **saturated**) by M if it is incident to an edge in M. The set of matched vertices is denoted $V(M)$; note that $|V(M)| = 2|M|$. The matching M is **maximal** if it is not contained in any strictly larger matching; **maximum** if it has the largest possible size among all matchings on G; and **perfect** if $V(M) = V(G)$. More generally, a ***k*-factor** in G is defined to be a k-regular spanning subgraph; thus a perfect matching is a 1-factor.

Warning: Again, "maximum" is a stronger condition than "maximal." For example, in the figure below, the blue matching ***M*** is maximum; the red matching ***M′*** is maximal but not maximum.



The vertex analogue of a matching is a **coclique** (or *independent set* or *stable set*): a set of vertices such that no two are adjacent. In general we want to know how large a coclique or matching can be in a given graph.

So we have four related notions:

- **coclique:** a set of vertices touching each edge at most once
- **vertex cover:** a set of vertices touching each edge at least once
- **matching:** a set of edges touching each vertex at most once
- **edge cover:** a set of edges touching each vertex at least once

Define

$$\alpha = \text{maximum size of a coclique}, \qquad \beta = \text{minimum size of a vertex cover},$$
$$\alpha' = \text{maximum size of a matching}, \qquad \beta' = \text{minimum size of an edge cover}.$$

(This is West's notation, which may or may not be standard. The mnemonic is that invariants without primes involve sets of vertices; the primed versions involve edges. The symbol α, the first letter of the Greek alphabet, is actually fairly standard for the size of the largest coclique in G. The last letter, ω, is the size of the largest clique.)

The matching and edge cover problems are equivalent and can be solved in polynomial time. The coclique and vertex cover problems are equivalent and are NP-complete in general. However, all four problems are equivalent for bipartite graphs. Fortunately, many matching problems are naturally bipartite (e.g., matching job applicants with positions, students with advisors, or workers with tasks, or columns with rows).

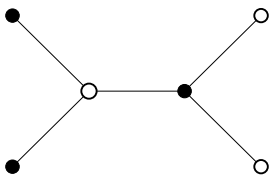
By the way, *counting* the maximum matchings of G is hard in general (although it can be done for, say, K_{2n} and $K_{n,n}$). It is unknown even for such nice graphs as Q_n.

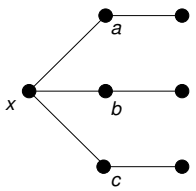
Example 3.3. For the cycle C_n, it should be clear that

$$\alpha = \alpha' = \left\lfloor \frac{n}{2} \right\rfloor \quad \text{and} \quad \beta = \beta' = \left\lceil \frac{n}{2} \right\rceil.$$

In particular $\alpha + \beta = \alpha' + \beta' = n$, and equality holds throughout if and only if the cycle is even. The first observation holds for all graphs, as we will see; the second one suggests that bipartiteness may be important.

Example 3.4. If G is bipartite, then each partite set is a vertex cover. On the other hand, a bipartite graph can have covers that are smaller than either partite set. (Example on the left below.) It seems plausible to try to build a minimum vertex cover by using vertices of large degree — but this doesn't always work either. (For example, in the graph on the right below, the unique vertex of largest degree is x, but the unique minimum vertex cover is $\{a, b, c\}$.)





3.2. **Equalities among matching and cover invariants.**

Proposition 3.5. $\alpha + \beta = n$.

Proof. A vertex set is a coclique iff its complement is a cover. In particular, the complement of a maximum coclique is a minimum cover. □

Proposition 3.6 (Gallai's identity)**.** $\alpha' + \beta' = n$.

Proof. First, we show that $\alpha' + \beta' \leqslant n$. Let M be a maximum matching, so that $|M| = \alpha'$. Let A be a collection of edges, each incident to one M-unmatched vertex. These edges are all distinct (otherwise M would not even be maximal), so $|A| = n - |V(M)| = n - 2\alpha'$. On the other hand, $A \cup M$ is an edge cover, so $|A \cup M| \geqslant \beta'$ and

$$\alpha' + \beta' \leqslant |M| + |A \cup M| = 2|M| + |A| = n. \tag{1}$$

Second, we show the opposite inequality. Let L be a minimum edge cover, so $|L| = \beta'$. For every edge $e = xy \in L$, we must have either $\deg_L(x) = 1$ or $\deg_L(y) = 1$, otherwise e could be removed from L to yield a smaller edge cover. This implies that every component of L is a star; in particular it is acyclic, so $|L| = n - c(L)$. Now construct a matching M by choosing one edge from every component of L. Then $|M| = c(L) \leqslant \alpha'$ and

$$\alpha' + \beta' \geqslant |M| + |L| = c(L) + n - c(L) = n. \tag{2}$$

Now combining (1) and (2) finishes the proof. □

Note that once the proof is complete, it follows that equality had to hold in both (1) and (2). That means that the proof gives us an easy way of constructing either a matching form an edge cover, or vice versa.

Proposition 3.7. $\alpha' \leqslant \beta$.

Proof. If M is a matching in G, then no vertex can cover more than one of the edges in M, so every vertex cover has to have at least $|M|$ vertices. □

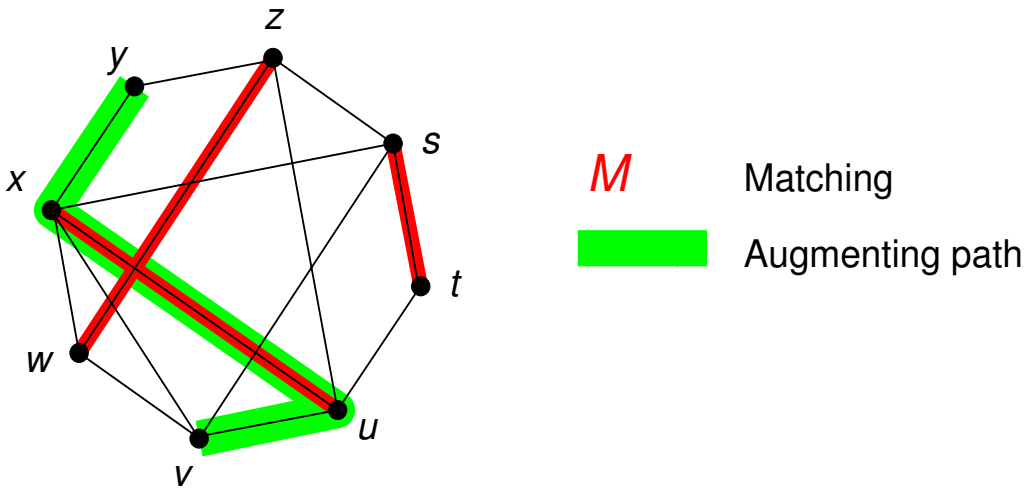
This kind of result is called a *weak duality*: every matching has size less than or equal to the size of every vertex cover. It implies that *if* we can find a matching and a vertex cover of the same size, then the matching must be maximum and the cover must be minimum — but does not guarantee that such a pair exists. In fact equality does not always hold, e.g., for an odd cycle. However, there is good news:

Theorem 3.8 (König-Egervàry Theorem)**.** *If G is bipartite then $\alpha' = \beta$.*

This is going to take some proof. We are actually going to construct an algorithm to enlarge a matching, one edge at a time. If the matching is maximum then the algorithm will actually certify that by producing a vertex cover of the same size as the matching.

3.3. **Augmenting paths.**

Definition 3.9. Let M be a matching in G and $P \subset G$ a path. A path P is ***M*-alternating** if its edges alternate between edges in M and edges not in M. It is ***M*-augmenting** if it is M-alternating and both endpoints are unmatched by M.



Note that every M-augmenting path has an even number of vertices (two unmatched endpoints and an even number of matched interior vertices), hence odd length.

If P is an M-augmenting p, q-path, then $M \triangle P$ is a matching with one more edge than M, where $\triangle$ means symmetric difference. (The operation of passing from M to $M \triangle P$ might be called "toggling P with respect to M": take the edges in $P \cap M$ out, and put the edges of $P \backslash M$ in.)

One fact about the symmetric difference operation: if $C = A \triangle B$ then $A = B \triangle C$ and $B = A \triangle C$. (Each of these equations says exactly the same thing: every element is contained in an even number of the sets A, B, C.)

The following lemma will be useful both immediately in proving Berge's theorem, and also later in proving Tutte's 1-factor theorem for nonbipartite matching.

Lemma 3.10. *Let M, N be matchings on G. Then every nontrivial component of $M \triangle N$ is either a path or an even cycle.*

Proof. Each vertex of G can have degree at most 2 in $M \triangle N$. Therefore, every nontrivial component H is either a path or a cycle. If H is a cycle, then its edges alternate between edges of M and edges of N, so H is even. □

Theorem 3.11. *(Berge, 1957) Let M be a matching on G. Then M is maximum if and only if G contains no M-augmenting path.*

Proof. ($\Longrightarrow$) If M has an augmenting path P then $M \triangle P$ is a bigger matching.

($\Longleftarrow$) Suppose that M is a non-maximum matching. Let N be a matching with $|N| > |M|$, and let $F = M \triangle N$. Then $|F \cap N| > |F \cap M|$, because

$$|F \cap N| = |N| - |M \cap N| \quad > \quad |F \cap M| = |M| - |M \cap N|.$$

In particular, some component H of F contains more edges of N than of M. By Lemma 3.10, H must be a path of odd length ℓ, say $H = v_0, v_1, \ldots, v_\ell$ with

$$v_0v_1 \in N, \quad v_1v_2 \in M, \quad v_2v_3 \in N, \quad \ldots, \quad v_{\ell-2}v_{\ell-1} \in M, \quad v_{\ell-1}v_\ell \in N.$$

Now, both F and N contain exactly one edge incident to v_1, namely v_1v_2. But $F = M \triangle N$, so $M = F \triangle N$, and so $M = F \triangle N$ does not contain any edge incident to v_1. By the same logic, $v_{2n} \notin V(M)$ as well. But this says precisely that H is an M-augmenting path. □

Berge's theorem reduces the problem of finding a maximum matching to the problem of determining whether a given matching has an augmenting path. This problem is easier when G is bipartite.

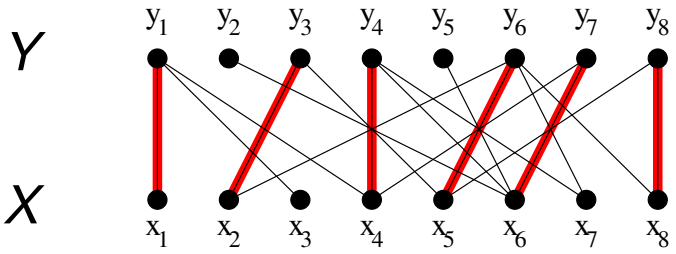
Notation: $N(x) = \{\text{neighbors of } x\}$, $N(X) = \bigcup_{x \in X} N(x)$

Here is the algorithm. It is really a form of breadth-first search.

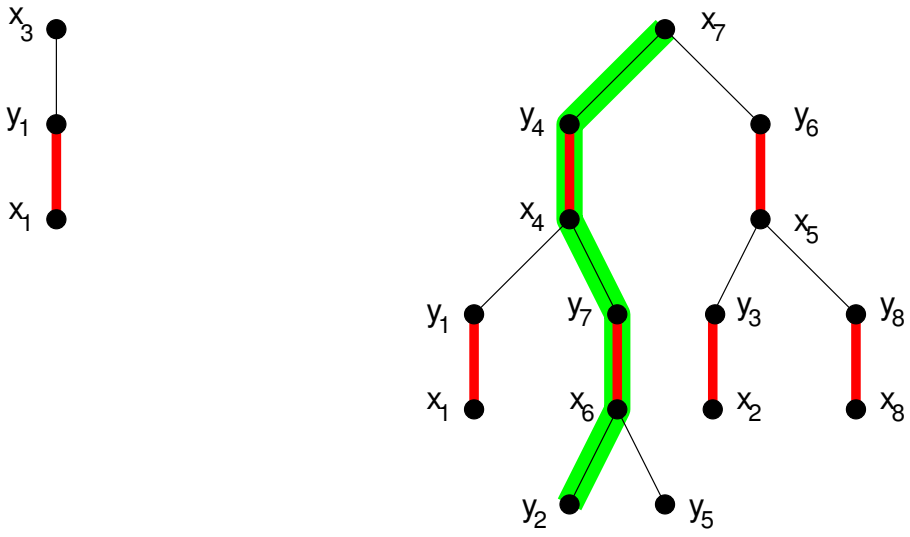
(1) Start at an unmatched vertex $x_0 \in X$. Let $S_0 = \{x_0\}$.
(2) Put in the edges x_0y for all $y \in N(x_0) \subset Y$.
(3) If any vertex added in step (2) is unmatched, then we have an augmenting path. Else, put in the edges that match every $y \in N(x_0)$ to its spouse (which must lie in x). Call this set of spouses S_1.
(4) Put in the vertices y and edges sy for all $s \in S_1$ and $y \in N(s)$.
(5) If any vertex added in step (4) is unmatched, then we have an augmenting path. Else, put in the edges that match every $y \in N(x_0)$ to its spouse (which must lie in x). Call this set of spouses S_2.
(∞) Iterate until either we find an augmenting path, or no new vertices are found in an even-numbered step.

If the algorithm does not find an augmenting path, repeat for every possible unmatched starting vertex $v_0 \in X \backslash V(M)$.

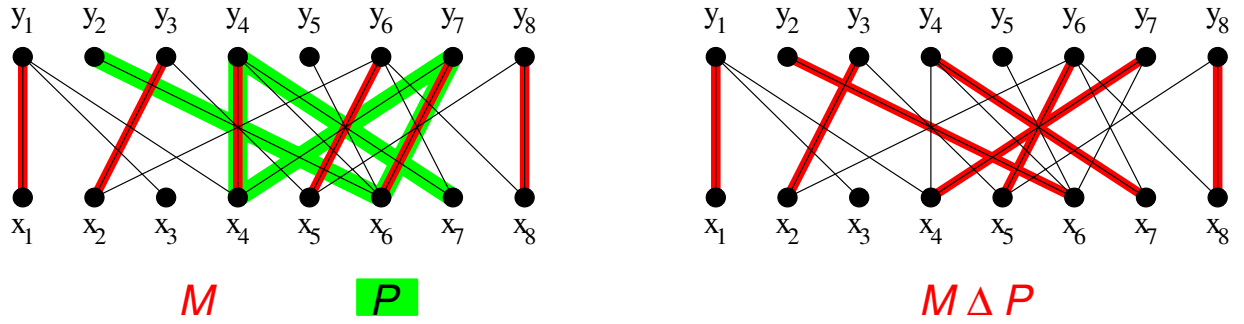
Example. Let us run the algorithm on the following graph and matching:



Here are the search trees we get by using x_3 and x_7, respectively, as the starting vertices:



If we start at x_3, the search peters out quickly, and no augmenting path is found. Starting at x_7, however, finds an augmenting path P Here is what P looks like in the original graph, and the larger matching $M' = M \triangle P$:



At this point, the search will find no augmenting path — we start at x_3, construct the two-edge search tree $x_3 - y_1 - x_1$, and terminate. Note that the search has told us that

$$N(\{x_1, x_3\}) \subseteq \{y_1\}.$$

But that means that $Q = \{y_1\} \cup (X \backslash \{x_1, x_3\})$ is a vertex cover, and $|Q| = 7 = |M'|$, which verifies (by weak duality) that M' is a maximum matching and Q is a minimum vertex cover.

Proposition 3.12. *Let G be a bipartite graph and M a matching with no augmenting path. Let $U = U_X \cup U_Y$ be the set of vertices visited in the (unsuccessful) call to the Augmenting Path Algorithm. Then*

$$Q = U_Y \cup (V(M) \backslash U_X)$$

is a vertex cover of cardinality equal to $|M|$.

Proof. First, we show that Q is a vertex cover. Note that $N(U_X) \subset U_Y$ by the construction of the APA. So if $e = xy$ with $x \in U_X$, then $y \in N(U_X) \subseteq U_Y \subseteq Q$, while if $x \notin U_X$ then $x \in Q$.

What about its cardinality? Every vertex in Q is matched. On the other hand, the vertices in U_Y are matched to vertices in U_X, which means that no edge of M meets Q more than once. Therefore, $|Q| \leqslant |M|$, and the reverse equality holds by weak duality. □

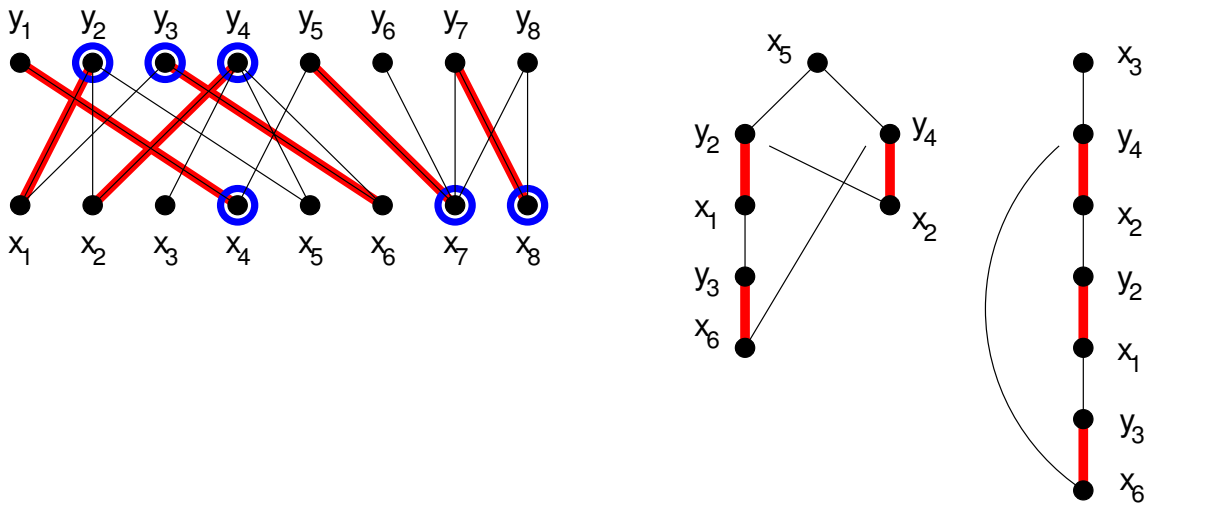
Here is another example. The matching on the left is maximum, with the unsuccessful search trees shown on the right. We have therefore

$$U_X = \{x_1,\ x_2,\ x_3,\ x_5,\ x_6\}, \qquad U_Y = \{y_2, y_3, y_4\}$$

which says that

$$Q = U_Y \cup (V(M) \backslash U_X) = \{y_2,\ y_3,\ y_4,\ \ x_4,\ x_7,\ x_8\}$$

is a vertex cover, as shown.



The punchline: Berge's Theorem together with Prop. 3.12 proves the König-Egerváry Theorem: $\beta = \alpha'$ for all bipartite graphs.

3.4. **Hall's Theorem and consequences.** Hall's Matching Theorem is a classical theorem about matchings with lots of proofs; the one I like uses the Augmenting Path Algorithm. (This is not Hall's original proof.)

Theorem 3.13 (Hall's Matching Theorem, 1935)**.** *Let G be an X,Y-bigraph. Then G has a matching saturating X if and only if $|N(S)| \geqslant |S|$ for every $S \subseteq X$.*

Proof. Necessity of Hall's condition ($\implies$): Let M be a matching saturating X. Then every $S \subseteq X$ is matched to a set of equal cardinality, which is a subset of $N(S)$.

Sufficiency of Hall's condition ($\impliedby$): Let M be a maximum matching, and suppose that $x \in X$ is unsaturated. Consider the unsuccessful search tree computed by the Augmenting Path Algorithm starting at x. Call its vertex set $S \cup T$, with $S \subseteq X$ and $T \subseteq Y$. Observe that:

- $|S| > |T|$, since every vertex in T is matched to a vertex in S, and S contains at least one unmatched vertex, namely x.
- $N(S) = T$, since that is precisely how the algorithm works.

Therefore, S violates Hall's condition. □

Hall's Theorem is not useful as an algorithm because actually computing $|N(S)|$ for every $S \subset X$ would require looking at all $2^{|X|}$ subsets. On the other hand, it is a great theoretical tool. Here are some consequences:

Corollary 3.14. *Every regular bipartite simple graph has a perfect matching.*

Proof. Let G be a k-regular X,Y-bigraph. By bipartite handshaking, $e(G) = k|X| = k|Y|$, so in particular $|X| = |Y|$.

Let $S \subseteq X$ and consider the induced subgraph $H = G|_{S \cup N(S)}$, which is bipartite with partite sets S and $N(S)$. Each vertex of S has degree *exactly* k in H, and each vertex of $N(S)$ has degree *at most* k in H. By the bipartite handshaking formula, $|S| \leqslant N(S)$. Since S was arbitrary, Hall's Theorem implies that G has a perfect matching. □

Corollary 3.15. *Every k-regular bipartite simple graph decomposes into the union of k perfect matchings. (Here "decomposes" refers to the edge set.)*

This corollary can be rephrased in terms of matrices. A simple bipartite graph can be recorded by its bipartite adjacency matrix, with a row for each vertex in X and a column for each vertex Y, with edges indicated by 1's and non-edges by 0's. The graph is k-regular iff every column and row sum is k (which requires the numbers of columns and rows to be the same). A matching corresponds to a **transversal**: a collection of 1's including exactly one entry in every row and column (this is essentially the same thing as a permutation matrix). The corollary then says that every $n \times n$ 0,1-matrix with all row and column sums equal to k can be written as the sum of k permutation matrices.

3.5. **Weighted bipartite matching and the Hungarian Algorithm.** Let G be a bipartite graph with partite sets X, Y, and let $w: E(G) \to \mathbb{R}_{\geqslant 0}$ be a weight function.

Problem: Find a matching M of maximum total weight, i.e., maximizing

$$w(M) = \sum_{e \in M} w(e).$$

WLOG, we may assume that $|X| = |Y| = n$ (adding isolated vertices if necessary), and that $G \cong K_{n,n}$ (adding edges of weight 0 if necessary). Then the maximum cardinality matchings are the $n!$ perfect matchings, and we may as well look for one of them.

Represent the pair (G, w) by an $n \times n$ matrix

$$W = (w_{ij})_{i,j \in [n]}$$

where $w_{ij} = w(x_i y_j)$.

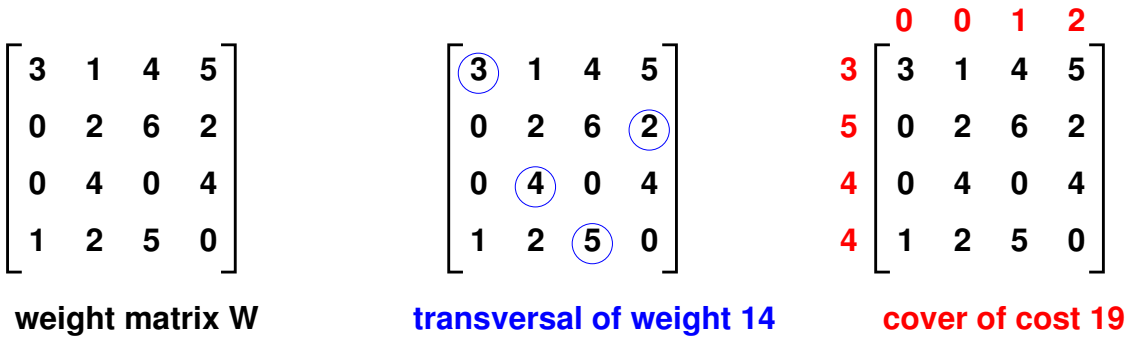
Definition 3.16. A **transversal** of W is a set of n matrix entries, one in each row and column. (Equivalent to a perfect matching on $K_{n,n}$; can be described by a permutation $\sigma: [n] \to [n]$.)

Definition 3.17. A **(weighted cover)** C of W is a list of row labels $a_1, \ldots, a_n$ and column labels $b_1, \ldots, b_n$ such that

$$\forall (i,j) \in [n] \times [n]: \quad a_i + b_j \geqslant w(x_i y_j). \tag{3}$$

The **cost** of the cover is $|C| = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i$.

Example:



Problem: Given a square nonnegative integer matrix, find a cover of minimum-cost.

Lemma 3.18. *The maximum weighted matching and minimum weighted cover problems are weakly dual. That is, for every matching M and cover C,*

$$w(M) \leqslant |C|.$$

Moreover, equality holds if and only if $w(x_i y_j) = a_i + b_j$ for every edge $x_i y_j \in M$. In that case, M and C are optimal.

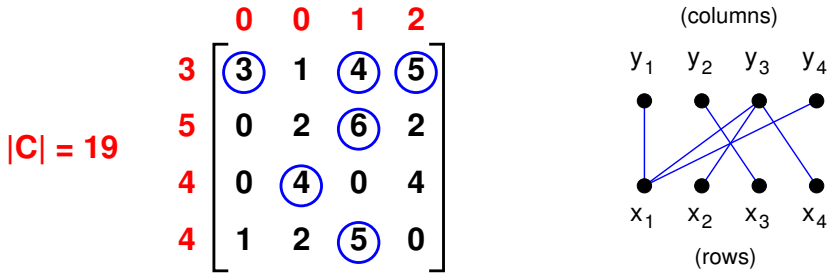
Proof. Represent M by a transversal σ of the weight matrix W. The cover condition (3) says that $w_{i,\sigma(i)} \leqslant a_i + b_{\sigma(i)}$ for all i, so

$$w(M) = \sum_{i=1}^{n} w_{i,\sigma(i)} \leqslant \sum_{i=1}^{n} (a_i + b_{\sigma(i)}) = |C|. \tag{4}$$

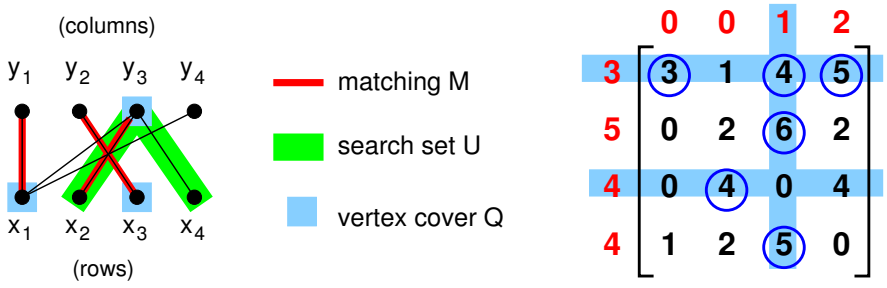
and equality holds in (4) if and only if it holds for each i, since the inequality is term-by-term. □

The cover shown above has cost 19. Can this be improved? If we could find a column or a row in which every entry was overcovered (i.e., for which the inequality (4) was strict), then we could decrease the label of that column. But there is not always such a column or row. The good news is that we can do something even more general in the spirit of the Augmenting Path Algorithm. The key is to increase the cover on some columns by some amount ϵ and decrease it on some rows by the same ϵ, amount, making sure that we decrease more labels than we increase.

To find this, circle the matrix entries that are covered exactly, i.e., those w_{ij} such that $w_{ij} = a_i + b_j$. The corresponding edges form a spanning subgraph $H \subseteq K_{n,n}$ called the *equality subgraph* $H = Eq(W, C)$.



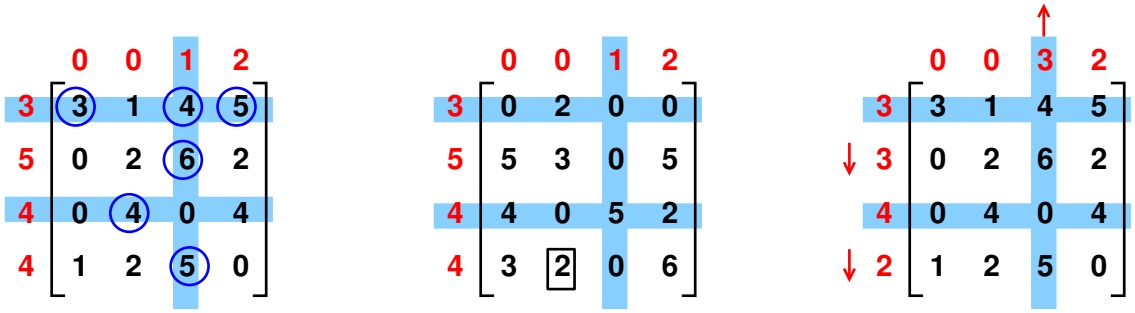
Now run the Augmenting Path Algorithm to find a maximum matching M on H, together with a minimum vertex cover Q. Recall that $|M| = |Q|$ by the König-Egerváry Theorem (which we proved using the APA), and that $Q = (X \backslash U_X) \cup U_Y$, where U is the set of vertices reached during the last (unsuccessful) search for an augmenting path. (Note that there is no guarantee of uniqueness for M and Q, because H may have several different maximum matchings and minimum vertex covers, but the APA will certainly produce one of each.) A possible output is shown below.



If $|M| = |Q| = n$, then $w(M) = |C|$ and we are done by weak duality (Lemma 3.18). Otherwise, we can use Q to find a less expensive cover, as follows. In terms of the matrix, $Q = Q_X \cup Q_Y$ corresponds to a collection of rows and columns (which we'll also call Q_X and Q_Y), of total cardinality $< n$, containing every circled matrix entry.

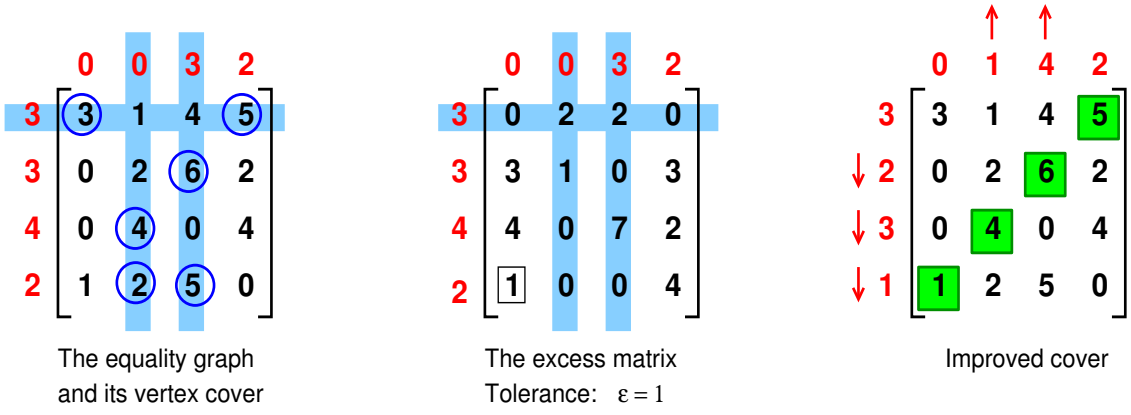
Construct the **excess matrix**, whose (i, j) entry is $w_{ij} - a_i - a_j$. (So the zeros in this matrix correspond precisely to edges of H.) Then paint blue the rows and columns corresponding to the vertices in Q. The numbers not painted blue in the excess matrix must all be positive; their minimum is the **tolerance**, denoted ε. Here $\varepsilon = 2$.

Now decrease the labels on $X \backslash Q_X$ by ε, and increase the labels on all columns in Q_Y by ε. This is shown in the third matrix below, with the red arrows indicating which labels have been increased or decreased.



This operation maintains the cover conditions, since the only matrix entries that decrease are those with rows in $X \backslash Q$ and columns in $Y \backslash Q_Y$, but all those entries were already over-covered by at least ε. Moreover, the cost of the cover has dropped by $\varepsilon(n - |Q|)$.

Repeat this procedure until the equality subgraph has a perfect matching. In this case, it just takes one more step.



Now we're done — we have a perfect matching whose weight equals the cost of the cover.[2]

This procedure is called the *Hungarian Algorithm*. Here is a summary of the algorithm.

[2]Thanks to Lawrence Chen for catching a mistake in an earlier example of the procedure.

The Hungarian Algorithm

Input: weight function $w : E(K_{n,n}) \to \mathbb{R}$
Output: a maximum weighted matching M and minimum cover $C = (a_1, \ldots, a_n, b_1, \ldots, b_n)$

(0) `Initialize` $a_i = \max\{w_{i1}, \ldots, w_{in}\}$ `and` $b_j = 0$ `for all` $i, j \in [n]$
(1) H `=` $\{x_i y_j \mid a_i + b_j = w_{ij}\}$
(2) `Use the APA to compute a maximum matching` M `and a minimum cover` Q
(3) `while` $|Q| < n$ `do`
(4) `Let` $\varepsilon := \min\{w_{ij} - a_i - b_j : \ x_i, y_j \notin Q\}$
(5) `Set` $a_i := a_i - \varepsilon$ `for all` $x_i \in X \backslash Q_X$
(6) `Set` $b_j := b_j + \varepsilon$ `for all` $y_j \in Q_Y$
(7) `Recompute` H
(8) `Use the APA (starting with` M`) to recompute the pair` M, Q
(9) `Return` (M, C)

One notn-obvious fact is that after the cover is adjusted in steps 5 and 6, the new equality subgraph computed in step 7 will still have M as a matching. This is left as an exercise.

Observe that if all the weights were nonnegative integers to begin with, then the procedure will definitely terminate (in at most a number of steps equal to the original cost of the cover). We have proved:

Theorem 3.19. *For any bipartite graph G and weight function $w : E(G) \to \mathbb{N}_{\geqslant 0}$, the Hungarian Algorithm calculates a minimum cover $C = (a, b)$ and a maximum matching M, with $|C| = w(M)$.*

We don't need to assume positive weights, since adding the same constant to all n^2 edges does not change which matchings have maximum weight. Also, $\mathbb{N}$ can be replaced with $\mathbb{Q}$ — there are finitely many weights, so just multiply them all weights by some common denominator to convert them to integers. Again, this will not change which matchings have maximum weight.

What about real weights? The potential problem is that the sequence of cover values produced by the Hungarian Algorithm might be something like 2, 1.1, 1.01, 1.001, 1.0001, ..., and the algorithm might never terminate, even though the minimum cover value is actually 1. Fortunately, this doesn't happen, for purely combinatorial reasons, and moreover the number of steps is at worst quadratic in n. The proof is left as an exercise; carefully examining the following example should show you why.

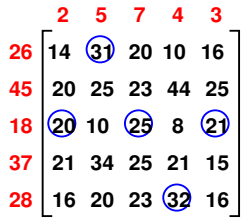
Example 3.20. Consider the weighted copy of $K_{5,5}$ with the following weight matrix and cover:

	2	**5**	**7**	**4**	**3**
26	14	31	20	10	16
45	20	25	23	44	25
18	20	10	25	8	21
37	21	34	25	21	15
28	16	20	23	32	16

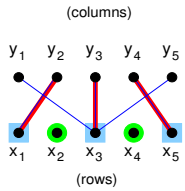
(The cover has been cooked up carefully to demonstrate how the algorithm works.) The following figures show the progress of the algorithm. Each iteration shows, left to right:

(1) The current cover, and the edges for which equality holds (circled in blue)
(2) The equality subgraph (shown as a graph), together with the output of the APA
(3) The excess matrix and the resulting tolerance value
(4) The improved cover

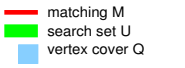
Iteration #1

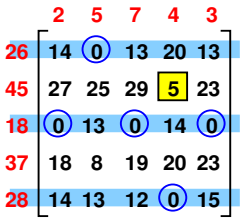


Cover and equality subgraph
Cover cost = 175

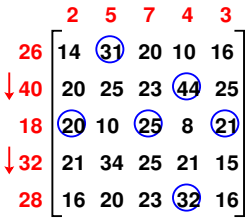


The Augmenting Path Algorithm



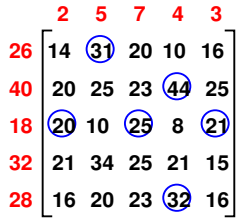


The excess matrix
Tolerance $\varepsilon = 5$

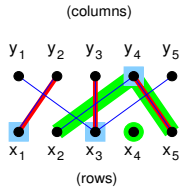


Improved cover

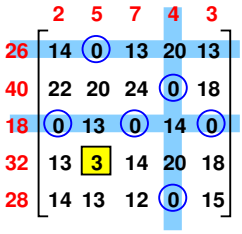
Iteration #2



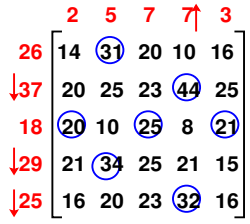
Cover and equality subgraph
Cover cost = 165



The Augmenting Path Algorithm

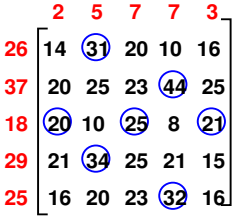


The excess matrix
Tolerance $\varepsilon = 3$

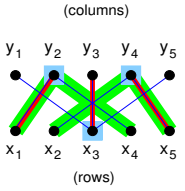


Improved cover

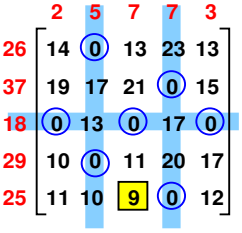
Iteration #3



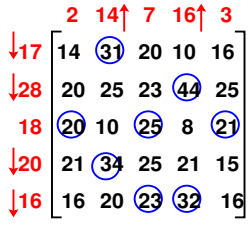
Cover and equality subgraph
Cover cost = 159



The Augmenting Path Algorithm

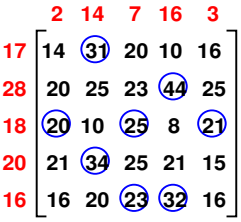


The excess matrix
Tolerance $\varepsilon = 9$

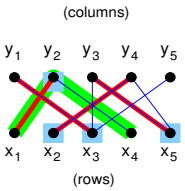


Improved cover

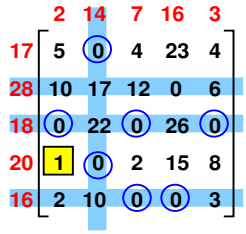
Iteration #4



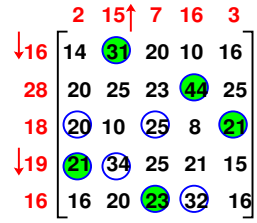
Cover and equality subgraph
Cover cost = 141



The Augmenting Path Algorithm
(aug path found; new matching)



The excess matrix
Tolerance $\varepsilon = 1$



Improved cover
and solution

3.6. **Stable matchings.** (Note: This material appears in the 3rd and subsequent editions of Diestel (2005); it is not in the 2nd edition (2000).)

You are the chair of the mathematics department, and you have to assign n students $X = \{x_1, \ldots, x_n\}$ to n advisors $Y = \{y_1, \ldots, y_n\}$. That is, you need to choose a perfect matching in $K_{n,n}$. (Let's suppose $n = 3$, so that the problem is a manageable size.) You'd like to do this in some way that reflects the preferences of each student and advisor, and you've asked each person to submit a form listing his or her three preferences in descending order. The data you have is as follows:

Student	Preference order
x_1	y_1, y_2, y_3
x_2	y_1, y_3, y_2
x_3	y_1, y_3, y_2

Advisor	Preference order
y_1	x_1, x_2, x_3
y_2	x_1, x_3, x_2
y_3	x_3, x_2, x_1

Unfortunately, it's not so clear what "optimal" means. Should we be trying to maximize overall happiness? Are the students' preferences more important than the advisors', or vice versa?

Since we know how to solve the weighted bipartite matching, we could try the following approach. Each student assigns two points to his top-choice advisor and one point to his second choice. Each advisor assigns two points to her top-choice student and one point to her second choice. We then get a weighted copy of $K_{3,3}$, with weight matrix as follows:

	y_1	y_2	y_3
x_1	$2 + 2 = \mathbf{4}$	$1 + 2 = \mathbf{3}$	$0 + 0 = \mathbf{0}$
x_2	$2 + 1 = \mathbf{3}$	$0 + 0 = \mathbf{0}$	$1 + 1 = \mathbf{2}$
x_3	$2 + 0 = \mathbf{2}$	$0 + 1 = \mathbf{1}$	$1 + 2 = \mathbf{3}$

We now select the matching of maximum total weight. Since n is so small, we can just do this by brute force instead of applying the Hungarian Algorithm. It's not hard to verify that the unique maximum-weight matching is $M = \{x_1y_2,\ x_2y_1,\ x_3y_3\}$.

There is a problem with this solution, though. x_1 and y_1 are each other's first choice, yet they are not paired in M. So what's to prevent them from deciding to work with each other, leaving x_2 and y_2 high and dry? Those two could pair up, it's true, but neither of them would be at all happy about it—each is the other's last choice. They'd certainly have cause to complain about the system. Meanwhile, if you use your power as chair to prevent x_1 and y_1 from working together, neither of them is going to thank you for it.

Maybe looking for a maximum-weight matching is not the way to go. What you'd really like is a system that will produce a matching in which no advisor-swapping will take place: that is, a *stable* matching. Not everybody will necessarily be happy, but at least no one (we hope) will have reason to complain about the system's fairness, and no one will have an incentive to ignore the rules of the game.

Definition 3.21. Let M be a perfect matching on $K_{n,n}$, and write $M(z)$ for the vertex matched to z. An **unstable pair** in a matching M is a pair (x, y) such that x prefers y over $M(x)$ and y prefers x over $M(y)$. A matching is **stable** if it has no unstable pair.

It is by no means obvious that a stable matching always exists for any given list of preferences—but it does. It can be found by the following algorithm, due to Gale and Shapley. We designate one partite set X as the *proposers* and the other, Y, as the *responders*. For clarity in pronouns, I will refer to the elements of X as male and the elements of Y as female, but this could just as easily be switched

The Gale-Shapley Stable Matching Algorithm — Deterministic Version

- Each proposer proposes to the top-choice responder who has not already rejected him.
- If the set of proposals forms a perfect matching, that matching is the output.
- Otherwise, each responder rejects all but her top choices of the proposals she receives.
- Repeat until the proposals form a perfect matching.

Theorem 3.22. *The Gale-Shapley algorithm terminates and produces a stable matching.*

Proof. First, we show that the algorithm does not get stuck — in another words, we never reach a situation in which some proposer has been rejected by *every* responder. Here is why this can't happen.

(1) If y ever issues a rejection, then she will receive at least one proposal at every subsequent stage of the algorithm. (Whoever survived y's previous cut will keep proposing to y until replaced by someone else.)
(2) Suppose that x has already been rejected by $n-1$ responders. In the iteration in which x does so, every responder other than y must receive at least one proposal (since they have each previously issued at least one rejection). But by the pigeonhole principle, that means that every responder receives *exactly* one proposal, and so the algorithm terminates with a perfect matching M.

By the way, each iteration that does not terminate removes at least one edge from the graph of potential matched pairs. Therefore, the algorithm terminates in at most $n^2 - n$ iterations.

Now, suppose that (x, y) is an unstable pair. That is,

- x prefers y to $M(x)$,
- y prefers x to $M(y)$,
- x and y aren't matched.

Since y prefers x to $M(y)$, it follows that x never proposed to y. But that must mean that x prefers $M(x)$ to y, which contradicts the definition of unstable pair. We conclude that M must be a stable matching. □

The algorithm would run equally well if the roles of advisors and students were switched. It turns out that the side doing the proposing is uniformly better off, and the side being proposed to is uniformly worse off. That is, if M is the matching produced by the Gale-Shapley algorithm and N is any other stable matching, then every $x \in X$ prefers $M(x)$ to $N(x)$ (unless $M(x) = N(x)$) and every y prefers $N(y)$ to $M(y)$ (unless $M(y) = N(y)$). The moral of the story is that it is better to propose than to be proposed to.

When I talked about this in class on 2/16/16, the Math 725 students asked several questions, which I list here with my attempts to answer them.

(1) *Does the algorithm still work if the partite sets do not have the same size?* (Brandon Caudell) Yes, I think so. If there are more proposers than responders, the algorithm terminates when every responder receives at least one proposal (and rejects all but her top choice). If there are more responders, the algorithm terminates when every responder receives at most one proposal. The first part of the proof can be modified to show that the termination state is indeed reached, and the second part (i.e., that the output matching is stable) makes no reference to cardinality and thus still goes through.
(2) *What if some proposer or responder has ties among his or her preferences?* (Joe Cummings) This is a subtle problem; see this Wikipedia article. Now you might have a unmatched pair x, y in which x prefers y over $M(x)$, but y is indifferent between x and $M(y)$; we might call such a pair *weakly unstable*. In this case, we'd have to do some more work to produce a matching without any pair that is even weakly unstable — it is not clear whether such a matching always exists.

(3) *Can there be a preference list where no proposer gets her first choice under Gale-Shapley? Can there be a preference list where no proposer or responder gets her first choice under Gale-Shapley?* Yes to both. Joseph Doolittle came up with an example in class for $n = 4$.
(4) *Does Gale-Shapley produce a universally optimal result for every proposer (i.e., if M is the Gale-Shapley output and M' is any stable matching, then each proposer is at least as happy under M than under M')? Likewise, does Gale-Shapley produce a universally pessimal*[3] *result for every responder?* Yes. See below.
(5) *Can there be a preference list where no proposer gets his first choice under any stable matching?* If you believe that the answers to (3) and (4) are yes, then the answer to this one is yes as well.
(6) *Can there be a preference list where no proposer or responder gets his/her first choice under any stable matching?* Probably, but I haven't constructed one.
(7) *Can there be a preference list where the algorithm takes the maximum number of iterations to finish, i.e., $n^2 - n$?* I am pretty sure the answer to this one is yes.
(8) *What about K_n instead of $K_{n,n}$?* In other words, suppose that each vertex in K_n has a preference order on the $n-1$ other vertices, and we still want to find a stable matching (let's assume n is even). This is called the *stable roommates problem*, and it does not always have a solution, but it is possible to check in polynomial time whether a solution exists and, if so, to find it.

The "rounds" in the Gale-Shapley algorithm are in fact unnecessary. Consider the following "non-deterministic" version of the algorithm:

The Gale-Shapley Stable Matching Algorithm — Nondeterministic Version

- Let x be some proposer who is not currently on a wait list.
- x proposes to the top-choice responder y who has not already rejected him.
- If y's wait list was empty, then she puts x on it. Otherwise, y chooses between x and the proposer currently on her wait list, and rejects the one she likes less.
- The algorithm terminates when all wait lists are full and we have a perfect matching.

It turns out that no matter how x is chosen in each iteration, the algorithm produces the same matching. The proof is an exercise, and is a key step in proving that Gale-Shapley is universally proposer-optimal and responder-pessimal.

A little more generally, one can also prove that no stable matching is universally better than any other, in the following sense:

Proposition 3.23. *Let M, M' be stable matchings for the same preference list. If at least one proposer x is happier in M than in M', than at least one responder y is happier in M' than in M.*

The proof is left as an exercise.

3.7. **Nonbipartite matching.** Matching is harder without the assumption of bipartiteness. The König–Egerváry Theorem ($\alpha' = \beta$) need not hold: for instance, the odd cycle C_{2n+1} has matching number $\alpha' = n$ and vertex cover number $\beta = n + 1$. (In fact the gap between the matching and cover numbers can be arbitrarily large.) In particular, the Augmenting Path and Hungarian Algorithms don't work, although there do exist polynomial-time algorithms to compute maximum-cardinality and maximum-weight matchings. One maximum-cardinality matching is Edmonds' Blossom Algorithm; this might make a good end-of-semester project. We are not going to look at algorithms, but we will prove two fundamental results in this area, namely Tutte's 1-Factor Theorem and the Berge-Tutte Formula. (Recall that "1-factor" is a synonym for "perfect matching" and more generally that "k-factor" means (the edge set of a) k-regular subgraph.)

[3] "Pessimal" is the antonym of "optimal."

For a simple graph G, not necessarily connected, define

$$o(G) = \text{number of odd(-order) components of } G.$$

Lemma 3.24. *Let H be a spanning subgraph of G. Then $o(H) = o(G) + 2k \equiv n \pmod 2$.*

Proof. Construct G from H by starting with n isolated vertices (hence n odd components) adding edges, one at a time. Each addition either doesn't change the component structure; makes an even component out of two even components; makes an odd component out of an even and an odd component; or makes an even component out of two odd components. The first three cases don't change the number of odd components; the last case decreases it by 2. □

Theorem 3.25 (Tutte's 1-Factor Theorem)**.** *A simple graph $G = (V, E)$ has a perfect matching if and only if it satisfies Tutte's condition:*

$$o(G - S) \leqslant |S| \quad \forall S \subseteq V. \tag{5}$$

Proof. ($\Longrightarrow$) Let M be a perfect matching of G, let $S \subseteq V$, and let $\bar{S} = V \backslash S$. Consider the graph H with vertex set $\bar{S}$ and edges $M|_{\bar{S}}$. This is a spanning subgraph of $G - S$, so

$$\begin{aligned} o(G - S) \leqslant o(H) &= \#\text{ isolated vertices in } H \\ &= \#\text{ vertices in } \bar{S} \text{ matched to a vertex in } S \\ &= \#\text{ vertices in } S \text{ matched to a vertex in } \bar{S} \qquad \text{(using } M \text{ as a bijection)} \\ &\leqslant |S|. \end{aligned}$$

($\Longleftarrow$) First, observe that by the argument of Lemma 3.24 adding one or more edges to G can only decrease the LHS of (5), hence preserves Tutte's condition. Accordingly, if the $\Longleftarrow$ direction is false, we can choose a maximal graph for which it fails — i.e., a graph G such that

- G satisfies Tutte's condition;
- G has no perfect matching; and
- adding any single missing edge to G produces a graph with a perfect matching.

We will show that these conditions imply a contradiction.

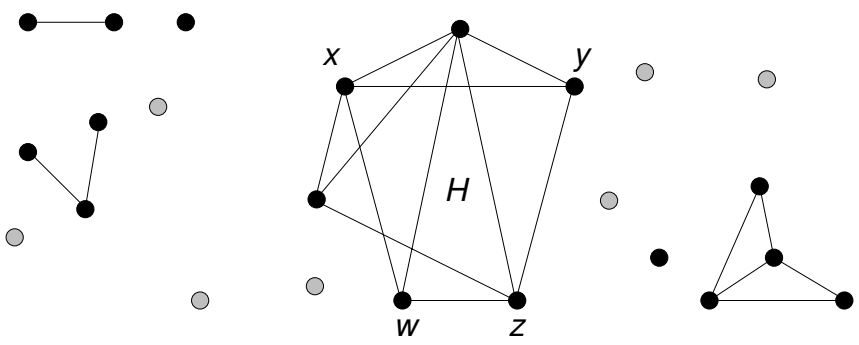
First, since G satisfies Tutte's condition, in particular $o(G - \varnothing) = o(G) \leqslant |S| = 0$, so $n(G)$ is even by Lemma 3.24.

Define $U = \{v \in V \mid vw \in E \ \forall w \in V \backslash \{v\}\}$. We will examine the graph $G - U$.

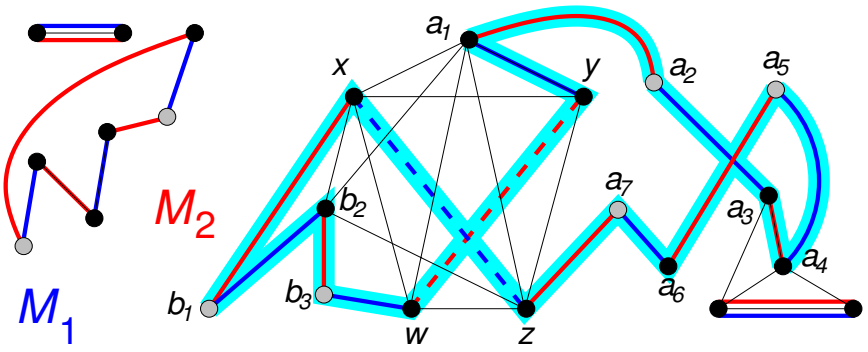
Case 1: $G - U$ is a disjoint union of cliques. We can construct a perfect matching on G as follows. Start with a maximal matching on $G - U$; the number of leftover vertices is $o(G - U)$. But by Tutte's condition, there are at least that many vertices in U, so all of those leftovers can be matched to vertices of U (in lots of ways). We have now matched all vertices outside U, and $G[U]$ is a clique, so the remaining vertices of U can be matched to each other (in lots of ways).

Case 2: $G - U$ has some component H that is not a clique. Then H must contain an induced P_3, i.e., $x, y, z \in V(H)$ and $xy, yz \in E(H)$, $xz \notin E(H)$. Also, since $y \notin U$, there is a vertex $w \in V(G - U)$ such that $wy \notin E(G)$. Note that w may or may not belong to H.

Here's the picture. The vertices of U are colored gray, and all edges with one or both endpoints in U are omitted (otherwise the figure would be unreadable; remember, every gray vertex is adjacent to every other vertex.)



By the choice of G, adding any single edge to G produces a graph with a perfect matching (which must contain that added edge). Accordingly, let M_1 and M_2 be perfect matchings of $G+xz$ and $G+wy$ respectively.



The dashed edges wy and xz do not belong to G; all other edges do. Let $F = M_1 \triangle M_2$; then $xz, wy \in F$.

Let C be the component of F containing xz (highlighted above). By Lemma 3.10, C is either a path or an even cycle. In fact, it's a cycle, because M_1 and M_2 are both perfect matchings.[4]

Case 2a: $yw \notin C$ (not the case of the example). Then

$$M_1 \triangle C \;=\; \big(M_2 \cap E(C)\big) \;\cup\; \big(M_1 - E(C)\big)$$

is a perfect matching that contains neither xz nor wy, so it is a perfect matching of G. That's a contradiction — we had assumed that G contained no perfect matching.

Case 2b: $yw \in C$. Label the vertices in cyclic order as

$$w,\ y,\ a_1,\ \ldots,\ a_p,\ z,\ x,\ b_1,\ b_q.$$

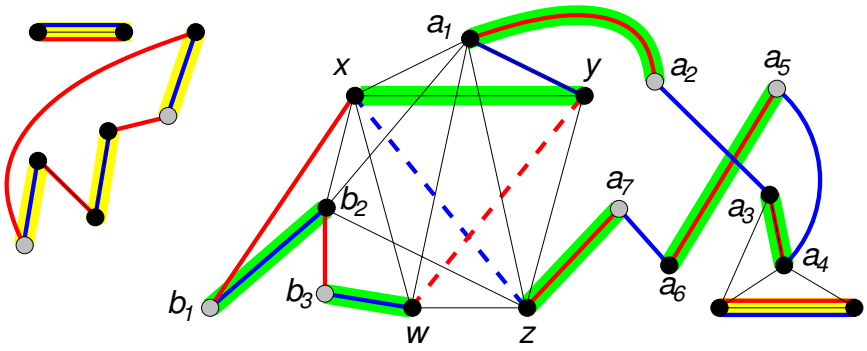
(It is possible that x and z are switched, but that case is equivalent because we have made no distinction between these vertices—they can be interchanged.) Note also that the numbers p and q are both odd (in the example, $p = 7$ and $q = 3$). This is because the path $y, a_1, \ldots, a_p, z$ has the same number of edges in M_1 and M_2, hence has an even number of edges and an odd number of vertices. Meanwhile, $|V(C)| = 4 + p + q$ is even, so p and q have the same parity.

Now, the edge set

$$M^* = \{a_1a_2,\ \ldots,\ a_{p-2}a_{p-1},\ a_pz,\ \ \ yx,\ \ \ b_1b_2,\ \ldots,\ b_{q-2}b_{q-1},\ b_qw\} \subseteq E(G)$$

[4] In more detail: If C were a path, then both of its endpoints would have to be in $V(M_1 \triangle M_2)$. OTOH, $V(M_1 \triangle M_2) \subseteq V(M_1) \triangle V(M_2)$ for any two matchings M_1, M_2 (this isn't hard to see). In this case, they're both perfect matchings, so $V(M_1) \triangle V(M_2) = \varnothing$.

(shown in green below) is a perfect matching on $V(C)$. Since $M_1 - E(C)$ (shown in yellow) is a perfect matching on $V(G) - V(C)$, it follows that $(M_1 - E(C)) \cup M^*$ is a perfect matching of G, as desired. □



Corollary 3.26 (Berge–Tutte Formula)**.** *Let $G = (V, E)$, $n = n(G)$, and for $S \subseteq V$ define*

$$u(S) = o(G - S) - |S|.$$

Let $m = \max\{u(S) \mid S \subseteq V(G)\}$. Then

$$\alpha'(G) \;=\; \frac{n - m}{2}. \tag{6}$$

The number $u(S)$ measures the extent to which S forms an obstruction to the existence of a perfect matching. Note that $u(\varnothing) = 0$, so the formula does say that $\alpha' \leqslant n/2$. Also, Tutte's condition says precisely that $\alpha' = n/2$ iff $u(S) \leqslant 0$ for all S, so formula (6) generalizes Tutte's theorem.

Proof. **Step 1: Prove that $\alpha' \leqslant (n - m)/2$.**

This is the easier step. Let M be a matching and $S \subseteq V(G)$. For each odd component H of $G - S$, either H has a vertex whose spouse is in S, or else H has an M-unmatched vertex. Therefore, there are at least $u(S)$ vertices which are not matched by M, and at most $n - u(S)$ matched vertices. So $|V(M)| \leqslant n - u(S)$. This is true for all S, so $|V(M)| \leqslant n - m$, and dividing by 2 gives the desired inequality.

Step 2: Prove that $\alpha' \geqslant (n - m)/2$. First note that

$$m \geqslant u(\varnothing) = o(G) \geqslant 0 \qquad \text{and} \qquad m \equiv n \pmod 2$$

by Lemma 3.24. Let $\tilde{G} = G \vee K_m$, where $\vee$ denotes the join. That is, $\tilde{G}$ is obtained from the disjoint union $G + K_m$ by adding an edge between *every* vertex of $\tilde{G}$ and *every* vertex of K_m. (If you like, $\tilde{G} = \overline{\overline{G} + \overline{K_m}}$.)

We claim that $\tilde{G}$ satisfies Tutte's Condition (5). (Details left to reader; it's routine using the definition of join.)

By Tutte's 1-Factor Theorem, $\tilde{G}$ has a perfect matching M.

- At most m edges of M have endpoints in K_m.
- Deleting all such edges yields a matching of G that matches at least $n(\tilde{G}) - 2m = n - m$ vertices.
- So $\alpha' \geqslant (n - m)/2$. □

Like Hall's Theorem 3.13 (which it implies), the Tutte-Berge Theorem does not yield an efficient algorithm for calculating α' — you would have to calculate $u(S)$ for every $S \subseteq V(G)$ — but it is a useful theoretical tool. Here is a famous corollary.

Theorem 3.27 (Petersen's theorem)**.** *Let G be a 3-regular simple graph with no bridge. Then G has a perfect matching.*

Proof. Fix $S \subseteq V(G)$ and let $k = o(G - S)$. If $k = 0$ then there is nothing to prove. Otherwise, let H be some odd component of G_S. Then $a = \sum_{x \in H} d_G(x) = 3|H|$ is odd, but on the other hand $b = \sum_{x \in H} d_H(x)$ is even. So $a - b$, which is the number of edges from H to S, is a positive odd number, but it cannot be 1 because then the edge it counts would be a bridge, so it is at least 3. This is true for every component H, so the number of edges from $G - S$ to S is at least $3o(G - S)$, but on the other hand it is also at most $3|S|$, again because G is 3-regular. It follows that $|S| \geqslant o(G - S)$, and we have proved that G satisfies Tutte's condition. □

Hall's Marriage Theorem can be proved from Tutte's 1-Factor Theorem; this is left as an exercise.

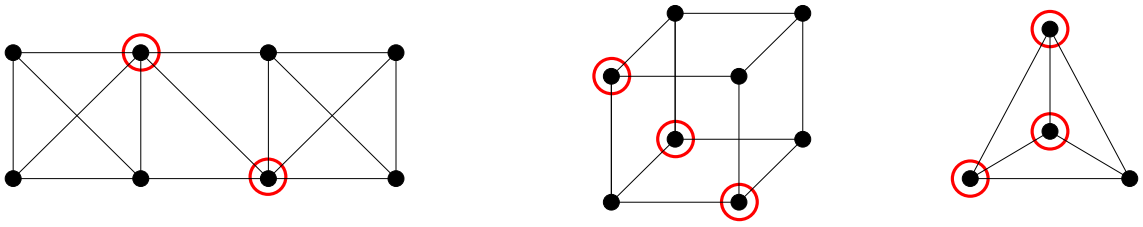
4. Connectivity, Cuts, and Flows

4.1. Vertex connectivity.

Let $G = (V, E)$ be a simple connected graph, and let $n = n(G) \geqslant 2$.

Definition 4.1. A **separator** (vertex cut, separating set) of $G = (V, E)$ is a vertex set $S \subsetneq V$ such that $G - S$ is disconnected or has only one vertex. Two vertices x, y are **separated** by S if they are in different components of $G - S$ (equivalently, every x, y-path has an internal vertex in S).

In the following figure, the vertex sets circled in red are separators.



For example, $N(v)$ is a separator for any $v \in V$, and a cut-vertex is just a separator of cardinality 1. If $G \cong C_n$, then any two nonadjacent vertices of G form a separator

Definition 4.2. Let $x \neq y \in V(G)$. The **local (vertex) connectivity** of the pair x, y is

$$\kappa(x, y) = \kappa_G(x, y) = \min\{|S| : \ S \text{ is an } x, y\text{-separator}\}$$

and the **(vertex) connectivity** of G is

$$\kappa = \kappa(G) = \min_{x,y} \kappa_G(x, y).$$

The graph G is ***k*-connected** iff $\kappa(G) \geqslant k$.

Some easy observations:

- $\kappa(G) = 0 \iff G$ is not connected, or $G = K_1$.
- $\kappa(G) = 1 \iff G$ is connected but has a cut-vertex.
- $\kappa(G) = n(G) - 1 \iff G \cong K_n$.
- If H is a spanning subgraph of G, then every separator of G is also a separator of H, so $\kappa(H) \leqslant \kappa(G)$.
- Loops and parallel edges do not affect connectivity.
- $\kappa(G) \leqslant \delta(G)$, because for every $v \in V(G)$, the set $N(v)$ is a separator. Note that v is an isolated vertex in $G - N(v)$. This even works if $G = K_n$.

As Diestel points out, it seems a little unnatural to define connectivity in this way. Since "connected" means that there is a path joining any two vertices, it seems as though "k-connected" ought to mean that there are k different paths between any two vertices. In fact "different" is not the right notion here: "disjoint" is.

Definition 4.3. Let $x, y \in V(G)$. A **disjoint path family**[5] for x, y (for short, "x, y-DPF") is a family of distinct x, y-paths $\mathcal{P} = \{P_1, \ldots, P_k\}$ such that no two of the P_i have a common vertex other than x and y. We will temporarily denote the maximum size of an x, y-DPF by $\lambda(x, y)$.

The left-hand graph in Figure 1(a) shows an x, y-DPF, and the three red vertices form an x, y-separator. In the right-hand graph (b), the path family shown is *not* a DPF, because some vertices belong to more than one path in the family. (It is true that no vertex other than x, y belongs to *all* of the paths, but that does not matter — in order to qualify as a DPF, no other vertex can belong to *even two* of the paths.)

[5]This terminology is not standard. West calls such a family "internally disjoint." Diesel just says "disjoint." Other sources use terms such as "independent."

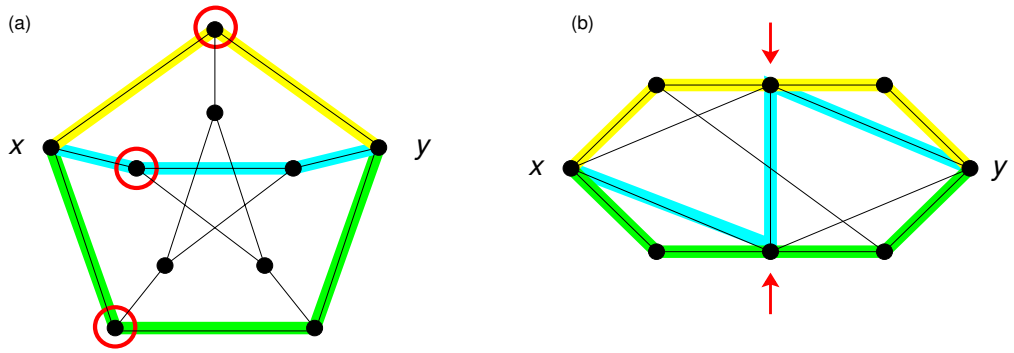


FIGURE 1. (a) A disjoint path family (DPF). (b) A path family that is not a DPF.

In fact, the graph in Figure 1(b) *does* have an x, y-DPF of size 3, shown in Figure 2(a). The red circled vertices form a x, y-separator; note that there is a bijection between vertices in the separator and paths in the DPF. If the dashed edge were removed, then the largest x, y-DPF would have size 2 and there would be a separator of size 2, as in Figure 2(b).

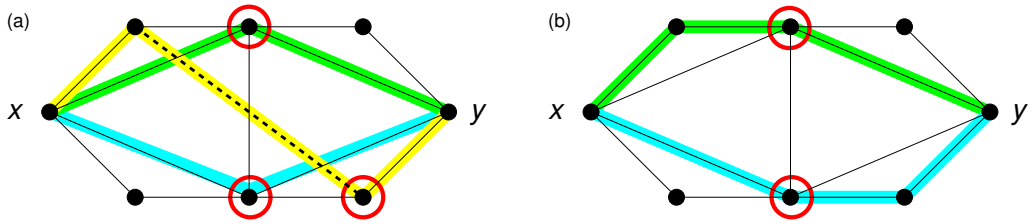


FIGURE 2. Maximum DPFs and minimum separators.

In general, every x, y-separator must have cardinality at least $\lambda(x, y)$. In other words, for each pair x, y there is a weak duality relation

$$\max\{|\mathcal{P}| : \mathcal{P} \text{ is an } x, y\text{-DPF}\} \leqslant \min\{|S| : S \text{ is an } x, y\text{-separator}\}. \tag{7}$$

Theorem 4.4 (Menger's Theorem). *Equality holds in* (7) *for all x, y; that is, $\kappa(x, y) = \lambda(x, y)$. In particular, G is k-connected if and only if every pair of vertices admits a DPF of size at least k.*

These two assertions are sometimes called the *local* and *global* formulations of Menger's theorem, respectively.

One direction is clear: if G has a separator S of size $k - 1$, then no pair x, y of vertices separated by S has a DPF of size k, since every x, y-path has to use a vertex of S. The other direction is the interesting one. There are three proofs in Diestel, none of which we will do — we are going to derive it as a corollary of the more powerful Max-Flow/Min-Cut Theorem.

Warning! Warning! Warning! Disjointness is a property of *families* of paths, not of individual paths. There is no such thing as a "disjoint path." If you ever find yourself saying, "Let $\mathcal{P}$ be a disjoint path family; add more disjoint paths to $\mathcal{P}$ until we have a total of $\lambda(u, v)$ paths," you have made a (very common) mistake.

In particular, you cannot necessarily construct a maximal DPF greedily — that is, not every maximal DPF is maximum. For example, if P_1 and P_2 are the yellow and green paths in Figure 1(b), then every other

x, y-path has an internal vertex in common with at least one of P_1 or P_2 in an internal vertex, so $\mathcal{P} = \{P_1, P_2\}$ is maximal, but it is not maximum because there is a larger x, y-DPF (Figure 2(a)).

4.2. **Edge connectivity.**

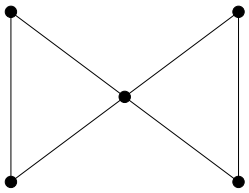
Definition 4.5. Let G be a connected simple graph. The **edge-connectivity** is

$$\kappa' = \kappa'(G) = \min\{|F| : \ F \subseteq E, \ G - F \text{ not connected}\}.$$

Such a set F is called a **disconnecting set**. (This terminology is in West but not Diestel.) The graph G is ***k*-edge-connected** iff $\kappa'(G) \geqslant k$.

Some observations:

- $\kappa'(G) > 0 \iff G$ is connected $\iff \kappa(G) > 0$.
- $\kappa'(G) \geqslant 2 \iff G$ is connected and has no bridge $\iff G$ is connected and is the union of cycles.
- $\kappa'(G)$ is not affected by loops, but can be affected by parallel edges. E.g., if H is formed from G by cloning every edge, then $\kappa'(H) = 2\kappa'(G)$.
- For every $v \in V$, deleting the set $E(v)$ of edges incident to v isolates v. Therefore, $\kappa'(G) \leqslant \delta(G)$.
- $\kappa(G)$ and $\kappa'(G)$ need not be equal. For example, the *bowtie* graph has $\kappa = 1$ and $\kappa' = 2$.



Theorem 4.6. *Every simple graph G satisfies $\kappa \leqslant \kappa' \leqslant \delta < n$. Moreover, given any integers $\kappa \leqslant \kappa' \leqslant \delta < n$, there exists a simple graph with those parameters.*

The proof is left as an exercise.

The following notation will be useful. IfX and Y are disjoint subsets of $V(G)$, then we can write

$$[X, Y] = [X, Y]_G = \{e \in V(G) \mid e \text{ has one endpoint in } X \text{ and its other endpoint in } Y\}.$$

Also, if A is an edge set and x is a vertex, we write $A(x)$ for the set of edges in A incident to x. E.g., $E(x) = [\{x\}, V\backslash\{x\}]_G$.

Definition 4.7. A **cut** or **edge cut** is a set of the form $[S, \bar{S}]$, where $\varnothing \neq S \subsetneq V$ and $\bar{S} = V\backslash S$. The sets S and $\bar{S}$ are called the **sides** of the cut.

Warning: Not every disconnecting set is a cut. For example, if G is any nontrivial connected graph then $E(G)$ is certainly a disconnecting set, but $E(G)$ can only be written in the form $[S, \bar{S}]$ if G is bipartite. On the other hand...

Proposition 4.8. *Let $F \subseteq E(G)$. Then $G - F$ is disconnected iff F contains a cut.*

Proof. If $F \supseteq [S, \bar{S}]$ then $G - F$ contains no path from S to $\bar{S}$. OTOH, if $G - F$ is disconnected then we can take S to be the vertex set of any component of $G - F$. □

This proposition seems trivial but is very useful. Given an edge set F that you want to show is a disconnecting set, it is typically easier and more natural to construct a vertex set S such that $F \supseteq [S, \bar{S}]$ than it is to show directly that $G - F'$ is not connected.

Remark 4.9. A cut can strictly contain another cut. For example, consider a 4-cycle with vertices labeled 1,2,3,4 in cyclic order. Let $S = \{1, 2\}$ and $T = \{1, 3\}$. Then $[T, \bar{T}]$ is the entire edge set and $[S, \bar{S}]$ is not. More generally, if G is x, y-bipartite, then $[X, Y] = E(G)$, which is certainly not a minimal cut unless $G = K_2$.

Definition 4.10. A **bond** is a minimal cut.

Proposition 4.11. *Let G be connected and let $F = [S, \bar{S}]$ be a cut. Then F is a bond if and only if $c(G - F) = 2$ (i.e., $G - F$ has exactly two components).*

Proof. If F is a bond then by definition $G - F + e$ is connected for every $e \in F$, so $c(G - F)$ must equal 2.

On the other hand, if $G-F$ has exactly two components, then they must be $G|_S$ and $G|_{\bar{S}}$ for some $\varnothing \neq S \subsetneq V$. So every $e \in F$ has one endpoint in each of S and $\bar{S}$, so $G - F + e$ is connected. Therefore no proper subset of F is a disconnecting set, hence F is a bond. □

Bonds behave like cycles. A spanning tree is an edge set that contains no cycle and meets every bond (since if it were disjoint from some bond then it would be disconnected). This is equivalent to saying that the *complement* of a spanning tree contains no bond and meets every cycle.

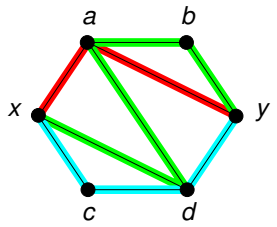
Proposition 4.12. *Let C, C' be cycles and let B, B' be bonds.*

(1) $C \triangle C'$ is the (edge-)disjoint union of cycles.
(2) If $C \neq C'$ are cycles in G and $e \in C \cap C'$, then $(C \cup C') - \{e\}$ contains a cycle.
(3) $B \triangle B'$ is a cut.
(4) If B, B' are bonds in G and $e \in B \cap B'$, then $(B \cup B') - \{e\}$ contains a bond.

The proofs are left to the reader. They are both closely related to the exchange rules for spanning trees (if T, T' are spanning trees and $e \in T - T'$, then there is some edge $e' \in T' - T$ so that $T - e + e'$ is a spanning tree).

Sneak preview: if G is planar, then there is a graph G^* called the *planar dual* of G, such that cycles of G correspond to bonds of G^*, and vice versa.

Definition 4.13. Let $x, y \in V(G)$. An **edge-disjoint path family** for x, y (for short, "x, y-EPF") is a family of distinct x, y-paths $\mathcal{P} = \{P_1, \ldots, P_k\}$ such that no two of the P_i have a common edge. We will temporarily denote the maximum size of an x, y-EPF by $\lambda'(x, y)$.



Every DPF is an EPF, but not vice versa. Therefore, $\lambda'(x, y) \geqslant \lambda(x, y)$ for all x, y.

Note that every bond separating x and y must have cardinality at least $\lambda'(x, y)$. In other words, we have a weak duality relation

$$\max\{|\mathcal{P}| : \mathcal{P} \text{ is an } x, y\text{-EPF}\} \leqslant \min\{|[S, \bar{S}]| : x \in S,\ y \notin S\}. \tag{8}$$

Theorem 4.14 (Menger's Theorem, edge version). *Equality holds in (8) for all x, y, that is, $\kappa'(x, y) = \lambda(x, y)$. In particular, G is k-edge-connected if and only if every pair of vertices admits a EPF of size at least k.*

Again, the $\Longleftarrow$ direction is easy, and the other direction is the hard one. We will soon show (in §4.5) that the edge version of Menger's theorem is a consequence of vertex version.

4.3. **The structure of 2-connected and 2-edge-connected graphs.** There are several useful equivalent conditions for 2-connectivity.

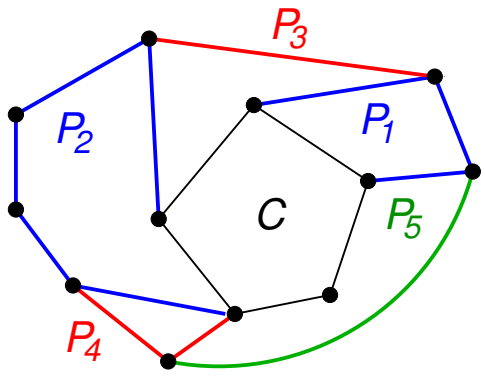
Definition 4.15. Let G be a connected graph. Let $v, w \in V(G)$, possibly equal. Construct a new graph G' from G by adding a new path

$$v = x_0,\ e_0,\ x_1,\ \ldots,\ x_{n-1},\ e_{n-1},\ x_n = w.$$

This path is called an **ear**. It is a **closed ear** if $v = w$ (in which case the trail is a cycle) and an **open ear** if $v \neq w$ (in which case the trail is a path).

Theorem 4.16 (Characterization of 2-Connectivity). *Let G be a simple connected graph with $n(G) \geqslant 3$. The following are equivalent:*

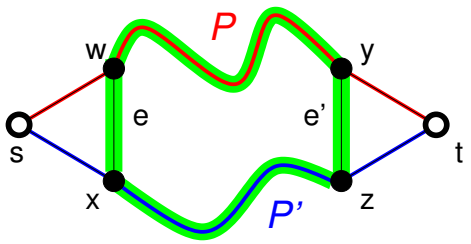
(A) G is 2-connected.
(B) Every two vertices of G lie on a common cycle.
(C) Every two edges lie on a common cycle.
*(D) G has an **open ear decomposition** $G = C \cup P_1 \cup \cdots \cup P_k$, where C is a cycle and each P_i is an open ear.*



Sketch of proof. (A $\Longleftrightarrow$ B): This is a special case of Menger's Theorem, which we will prove later.

(D $\Longrightarrow$ A): The original cycle is 2-connected, and so is any graph obtained by adjoining ears to a 2-connected graph.

(A/B $\Longrightarrow$ C): Let e, e' be edges. If e, e' share an endpoint, say $e = wx$ and $e' = yz$, then by 2-connectivity we can find an x, z-path P in $G - y$, and then $e + P + e'$ is a cycle. Otherwise, let $e = wx$ and $e' = yz$. Construct a new graph G' by adding vertices s, t and edges sw, sx, ty, tz. This is equivalent to adding two open ears, so G' is 2-connected. In particular G' has two disjoint s, t-paths P, P'. WLOG, $P = sw \cdots yt$ and $P' = sx \cdots zt$. Then $P + P' - sw - sx - ty - tz + e + e'$ is a cycle containing e and e'.

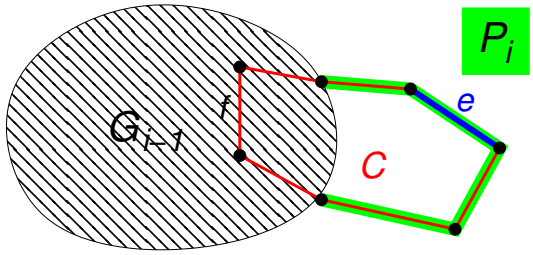


(C $\Longrightarrow$ D): If G is a graph satisfying (C), then the following algorithm produces an ear decomposition of G:

Let $G_0 \subseteq G$ be any cycle. (So G_0 is 2-connected.)
Initialize $i = 1$.
During the i^{th} step:

- Choose any edges $e \in E(G) - E(G_i)$ and $f \in E(G_i)$.
- Let C be a cycle containing e and f. (Such a cycle must exist by condition (C).)
- Let P_i be the smallest path in C that contains no edges of G_{i-1}.
- This is the new ear; set $G_i := G_{i-1} \cup P_i$.
- Increment i.

Repeat until $E(G_i) = E(G)$.



□

There is an analogous theorem for 2-edge-connected graphs. For the purpose of this statement, define a **circuit** to be a closed walk without repeated edges (but allowing repeated vertices).

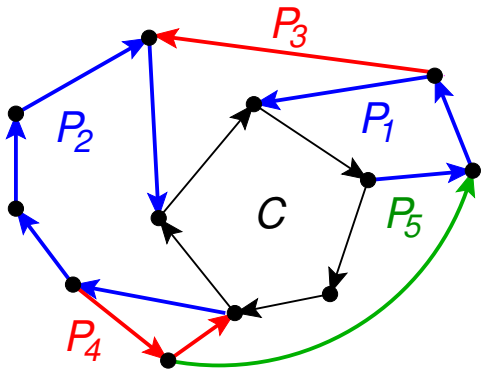
Theorem 4.17 (Characterization of 2-Edge-Connectivity). *Let G be a connected graph with $n(G) \geqslant 2$. The following are equivalent:*

(A) G is 2-edge-connected.
(B) Every two vertices $u, v \in V(G)$ lie on a common circuit.
*(C) G has an **ear decomposition** $G = C \cup P_1 \cup \cdots \cup P_s$, where C is a cycle and each P_i is an ear (either open or closed).*
*(D) G has a **strong orientation**, i.e., it is possible to orient all edges so that every edge belongs to a directed cycle.*

The figure on the left is an example of a closed-ear decomposition. Note that P_3 is a closed ear—it can be regarded as a closed path from v to v, where v is the indicated vertex of $C \cup P_1 \cup P_2$. Indeed, the graph is 2-edge-connected but not 2-connected.

The figure on the right shows how to translate a closed-ear decomposition into a strong orientation (an example of the implication (C) $\Longrightarrow$ (D) of the theorem): just orient the original cycle consistently (which

can be done in one of two ways), and whenever you add an ear, orient it consistently (which again can be done in one of two ways).



4.4. **Counting strong orientations.** Let G be a graph (with loops and multiple edges allowed). Let $S(G)$ be the set of strong orientations of G, and let $s(G) = |S(G)|$. What can we say about this number?

- $s(G) > 0$ iff G is 2-edge-connected, by Thm. 4.17.
- $s(G)$ is always even (unless $G = K_1$), because reversing the direction of every edge preserves strongness.
- Cycles have two strong orientations ("clockwise" and "counterclockwise").
- For consistency, this should still be true for the loop C_1. In fact, adding a loop to G should double the value of $s(G)$, since the loop itself can be oriented in one of two ways.
- The value of $s(K_n)$ is not so obvious. Starting with $n = 1$, the sequence begins 1, 0, 2, 24, 544, 22320, 1677488, ...

Theorem 4.18. *The invariant s satisfies a deletion/contraction recurrence: $s(G) = s(G/e) + s(G - e)$.*

Proof. Let D be a strong orientation of G, and suppose that there exists $e \in E(G)$ such that reversing e in D also gives a strong orientation. Then $D - e$ is a strong orientation of $G - e$. On the other hand, each strong orientation of $G - e$ certainly gives rise to a pair of strong orientations of G. So

$$s(G - e) = \frac{\#\{D \in S(G) \mid e \text{ is reversible}\}}{2}.$$

Now let D' be a strong orientation of G/e. Such a thing could come from a pair of strong orientations of G/e that are identical except for the orientation of e (so e is reversible), or from a single strong orientation in which e is not reversible. Therefore,

$$s(G/e) = \frac{\#\{D \in S(G) \mid e \text{ is reversible}\}}{2} + \#\{D \in S(G) \mid e \text{ is not reversible}\}.$$

Adding these two equations gives the desired recurrence. □

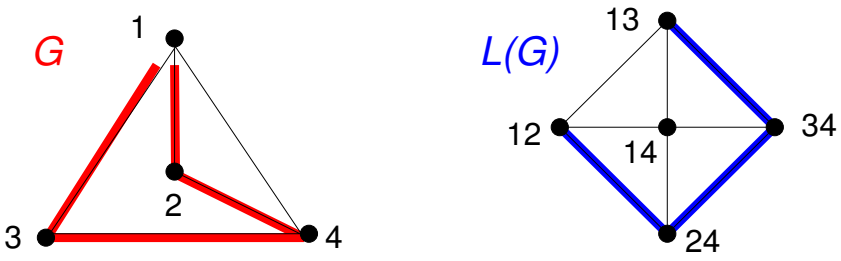
This should remind you of the deletion-contraction recurrence for $\tau(G)$. Stay tuned!

4.5. **Menger implies edge-Menger.**

Definition 4.19. Let G be a simple graph. The **line graph** $L(G)$ is defined by

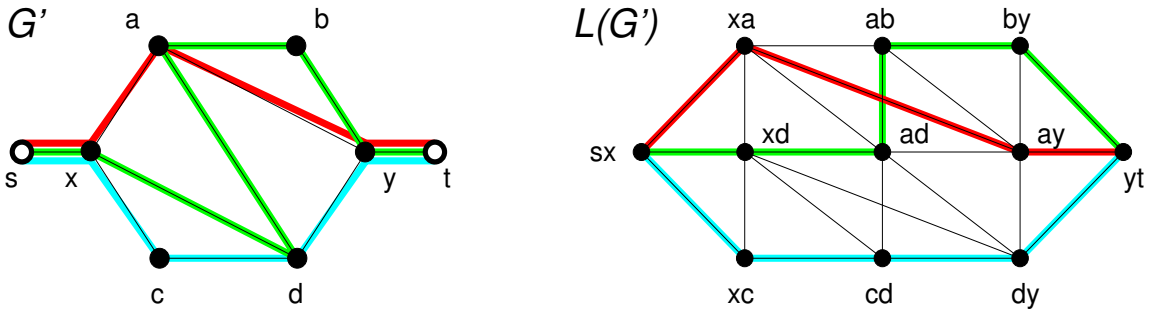
$$V(L(G)) = E(G),$$
$$E(L(G)) = \{ef : \ e, f \text{ have a common endpoint}\}.$$

Note that the edge set of a (closed) trail in G corresponds to the vertex set of a path in $L(G)$, as in the below figure.



Theorem 4.20. *The vertex version of Menger's Theorem implies the edge version.*

Proof. Assume that the vertex version of Menger's Theorem holds. Let G be a graph, $x, y \in V(G)$, $xy \notin E(G)$. Let G' be the graph formed from G by adding vertices s, t and edges sx, yt. Construct the line graph $L(G')$.



We are now going to apply vertex-Menger to $L(G')$ in order to deduce edge-Menger for G.

First, observe that a set $A \subseteq E(G)$ disconnects x from y in G if and only if the corresponding vertices separate sx from yt in $L(G')$. Therefore,

$$\kappa'_G(x, y) = \kappa_{L(G')}(sx, yt). \tag{9}$$

Second, the vertex version of Menger's theorem implies that

$$\kappa_{L(G')}(sx, yt) = \lambda_{L(G')}(sx, yt). \tag{10}$$

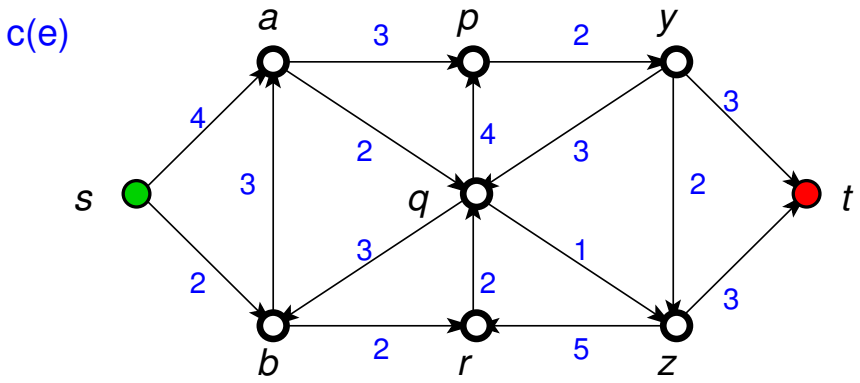
Third, observe that for each x, y-path P in G, there is a corresponding s, t-path $P' = sx + P + yt$ in G' and an sx, yt-path $\hat{P}$ in $L(G')$, and that P, Q are edge-disjoint if and only if $\hat{P}, \hat{Q}$ are internally disjoint. Therefore,

$$\lambda_{L(G')}(sx, yt) = \lambda'_G(x, y) \tag{11}$$

and chaining (9), (10), and (11) together completes the proof. □

4.6. **Network flows.**

Definition 4.21. A **network** or ***s, t*-network** $N = (G, s, t, c)$ consists of a simple digraph $G = (V, E)$ with two distinguished vertices s, t, called the **source** and **sink** respectively, and a **capacity function** $c : E \to \mathbb{N}_{>0}$. We may assume that G is a simple digraph: it has no loops, and that for every $x, y \in V$ there is at most one edge of the form $\overrightarrow{xy}$ and at most one edge of the form $\overrightarrow{yx}$.



(Note: Diestel does this a bit differently; he starts with an undirected graph in which each edge e can be thought of as a pair of anti-parallel edges $\overrightarrow{e}$ and $\overleftarrow{e}$, each with a different capacity. This is an equivalent model, but it requires more complex notation.)

We want to think of a network as modeling a situation where stuff (data, traffic, liquid, electrical current, etc.) is flowing from source s to sink t. The capacity of an edge is the amount of stuff that can flow through it (or perhaps the amount of stuff per unit time). This is a very general model that can be specialized to describe cuts, connectivity, matchings and other things in directed and undirected graphs.

A **flow** on N is a function $f : E \to \mathbb{R}$ that satisfies the constraints

(12) $\quad 0 \leqslant f(e) \leqslant c(e) \quad \forall e \in E \quad$ (the *capacity constraints*),

(13) $\quad f_{\text{in}}(v) = f_{\text{out}}(v) \quad \forall v \in V \backslash \{s, t\} \quad$ (the *conservation constraints*),

where for $v \in V$ we define

(14) $$f_{\text{in}}(v) = \sum_{e = \overrightarrow{uv}} f(e), \qquad f_{\text{out}}(v) = \sum_{e = \overrightarrow{vw}} f(e).$$

The function $f(e) = 0$ is of course a flow. Here is a nontrivial example. I will consistently use blue for capacities and red for flows.

Note that the conservation constraints say that flow cannot accumulate at any internal vertex.

The **value** $|f|$ of a flow f is the net flow into the sink:

(15) $$|f| \;:=\; f_{\text{in}}(t) - f_{\text{out}}(t) \;=\; f_{\text{out}}(s) - f_{\text{in}}(s).$$

To see the second equality, note that

$$\sum_{e \in E(G)} f(e) = \sum_{v \in E(G)} f_{\text{in}}(v) = \sum_{v \in V(G)} f_{\text{out}}(v)$$

and by the conservation constraints, most of the summands cancel, leaving only

$$f_{\text{in}}(s) + f_{\text{in}}(t) = f_{\text{out}}(s) + f_{\text{out}}(t)$$

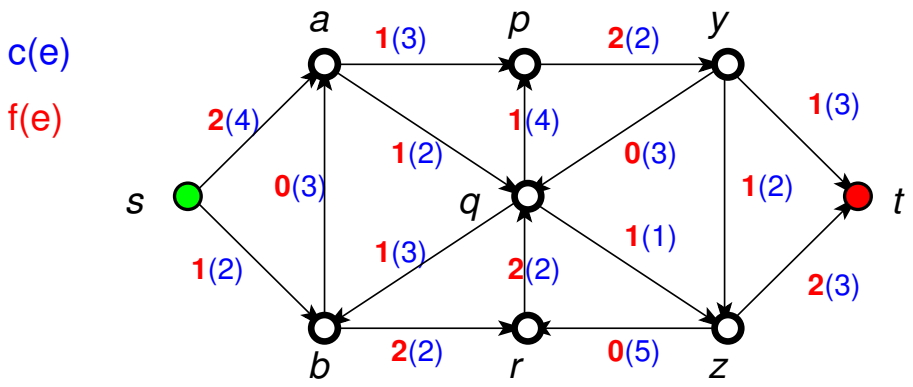


FIGURE 3. A capacity function and a compatible flow.

from which the second equality easily follows. Since we are concerned with maximizing $|f|$, we typically assume that s has no in-edges and t had no out-edges, so that (15) can be simplified to

$$|f| = f_{\text{in}}(t) = f_{\text{out}}(s). \tag{16}$$

The flow f shown in Figure 3 has $|f| = 3$.

Max-Flow Problem: Given a source-sink network (G, s, t, c), find a flow of maximum value.

We need a way of increasing the value of a given flow f, or showing that no such way exists. (This ought to remind you of the Augmenting Path Algorithm.) The naive way is to look for an "f-augmenting path"—an s,t-path $P \subseteq N$ in which no edge of P is being used to its full capacity, that is, such that $f(e) < c(e)$ for all $e \in P$. In this case, we can increase all flows along the path by some nonzero amount ε so as to preserve the conservation and capacity constraints, and increase the value of the flow by ε.

However, there can be nonmaximum flows where no such path P exists. Consider the network shown in Figure 4. Continuing the analogy with matchings and the APA, the flow f on the left is "maximal", in the sense that there does not exist any flow f' such that $|f'| > |f|$ and $f'(e) \geqslant f(e)$ for every $e \in E$. However, it is not maximum: $|f| = 1$, while the flow g on the left has $|g| = 2$.

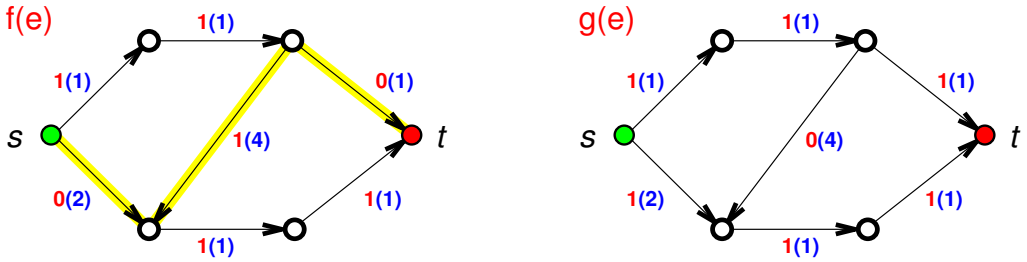


FIGURE 4. Two "maximal" flows, one with an augmenting path (highlighted).

There is a more general way to increase flow: Allow the augmenting path P to contain edges that point in the wrong direction, but along which the flow is nonzero. As far as the conservation constraints and the value of the flow is concerned, decreasing "backward" flow is equivalent to increasing "forward" flow. The "forward" edges a, c are not being used to full capacity, and the "backward" edge b contains flow "in the

wrong direction". So we can define a new flow $\tilde{f}$ by

$$\tilde{f}(a) = f(a) + 1, \qquad \tilde{f}(b) = f(b) - 1, \qquad \tilde{f}(c) = f(c) + 1,$$
$$\tilde{f}(e) = f(e) \text{ for all other edges } e.$$

Then $\tilde{f}$ satisfies the capacity and conservation constraints, and $|\tilde{f}| = |f| + 1$. (In fact $\tilde{f} = g$.)

Definition 4.22. Let f be a flow in an s,t-network $N = (G, s, t, c)$. Let P be an s,t-path in G, which may include backward edges $\overleftarrow{e}$ as well as forward edges $\overrightarrow{e} \in P$. The **tolerance** of an edge $e \in P$ is defined as

$$\varepsilon(e) = \begin{cases} c(e) - f(e) & \text{if } \overrightarrow{e} \in P, \\ f(e) & \text{if } \overleftarrow{e} \in P, \end{cases}$$

and the **tolerance** of the path P is

$$\varepsilon(P) = \min_{e \in P} \varepsilon(e).$$

The path P is **augmenting** for f if $\varepsilon(P) > 0$.

The proof of the following proposition is then completely routine.

Proposition 4.23. *If P is an f-augmenting path, then the function $\tilde{f}$ defined by*

$$\tilde{f}(e) = \begin{cases} f(e) + \varepsilon & \text{if } \overrightarrow{e} \in P, \\ f(e) - \varepsilon & \text{if } \overleftarrow{e} \in P, \\ f(e) & \text{otherwise,} \end{cases}$$

is a flow (i.e., it satisfies the capacity and conservation constraints), and $|f'| = \varepsilon + |f|$.

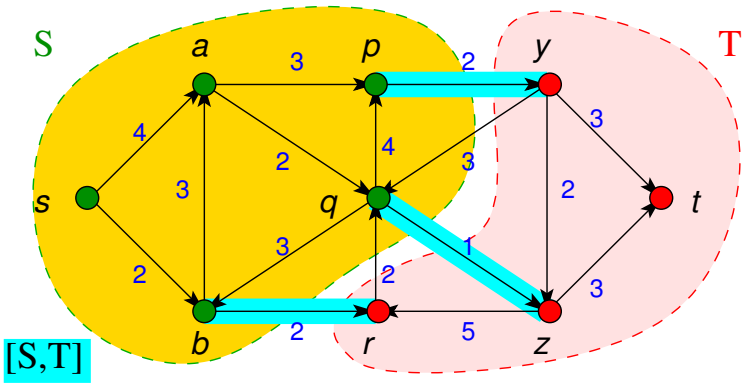
The dual problem to the Max-Flow problem is the Min-Cut problem.

Definition 4.24. Let $N = (G, s, t, c)$ be an s,t-network. A **source-sink cut** is a directed cut of the form

$$[S, T] = \{\overrightarrow{xy} \in E(G) : \ x \in S, \ y \in T\}$$

where $V(G) = S \cup T$, $s \in S$, and $t \in T$. (Note that this is a directed graph, so we only include edges from S to T, not from T to S.) The **capacity** of the cut is

$$c(S, T) = \sum_{e \in [S,T]} c(e).$$



In the figure above, $S = \{s, a, b, p, q\}$ (gold) and $T = \bar{S} = \{t, r, y, z\}$ (pink). The resulting source-sink cut is $[S, T] = \{\overrightarrow{br}, \overrightarrow{py}, \overrightarrow{qz}\}$ (highlighted in cyan), so $c(S, T) = 2 + 1 + 2 = 5$. Note that the T,S-edges $\overrightarrow{yq}$ and $\overrightarrow{rq}$ are not considered part of the cut.

Min-Cut Problem: Find a source-sink cut of minimum capacity.

A source-sink cut can be thought of as a bottleneck: a channel through which all flow must pass. Therefore, the capacity of any cut should be an upper bound for the maximum value of a flow — this is the "weak duality" inequality, analogous to the easy directions of results such as the König-Egerváry theorem and the various versions of Menger's theorem.

For a flow f and a vertex set $A \subseteq V$, define

$$f(A, \bar{A}) = \sum_{\vec{e} \in [A, \bar{A}]} f(e) \; - \sum_{e \in [\bar{A}, A]} f(e). \tag{17}$$

Proposition 4.25. *Let f be a flow, and let $A \subseteq V$. Then:*

$$f(A, \bar{A}) \; = \; \sum_{w \in A} (f_{\text{out}}(w) - f_{\text{in}}(w)). \tag{18}$$

In particular, if $[S, T]$ is a source-sink cut, then

$$f(S, T) \; = \; |f| \; \leqslant \; c(S, T). \tag{19}$$

That is, the Max-Flow and Min-Cut problems are weakly dual.

Proof. Add zero to $f(A, \bar{A})$ and be careful about the bookkeeping:

$$f(A, \bar{A}) = \left(\sum_{\vec{e} \in [A, A]} f(e) + \sum_{\vec{e} \in [A, \bar{A}]} f(e) \right) - \left(\sum_{\vec{e} \in [A, A]} f(e) + \sum_{\vec{e} \in [\bar{A}, A]} f(e) \right)$$

$$= \sum_{e = \overrightarrow{vw}:\; v \in A} f(e) - \sum_{e = \overrightarrow{vw}:\; w \in A} f(e)$$

$$= \sum_{w \in A} \left(\sum_{e:\; \text{head}(e) = w} f(e) \quad - \quad \sum_{e:\; \text{tail}(e) = w} f(e) \right)$$

$$= \sum_{w \in A} (f_{\text{out}}(w) - f_{\text{in}}(w)).$$

establishing (18).

In particular, if $[S, T]$ is a source-sink cut and f is any flow, then

$$f(S, T) = \sum_{w \in S} f_{\text{out}}(w) - f_{\text{in}}(w) = f_{\text{out}}(s) = |f|$$

but on the other hand $f(S, T) \leqslant c(S, T)$ by the capacity constraints (12). □

4.7. **The Ford-Fulkerson algorithm.** In fact, the Max-Flow and Min-Cut problems are *strongly* dual. They can be solved simultaneously in finite time by the following simple but very powerful algorithm, due to Ford and Fulkerson.

The Ford-Fulkerson Algorithm

Input: a network $N = (G, s, t, c)$
Output: a maximum flow f and minimum s,t-cut $[S, T]$

Initialize $f(e) = 0 \quad \forall e$.
Repeat:
    Let S be the set of all vertices reachable from s by an f-augmenting path
    If $t \in S$ ("breakthrough"), then increase flow along some augmenting path
until breakthrough does not occur.
Return the flow f and the cut $[S, \bar{S}]$.

Theorem 4.26 (The Max-Flow/Min-Cut Theorem — "MFMC"). *The Ford-Fulkerson algorithm finishes in finite time and computes a maximum flow and a minimum cut.*

Proof. Since everything in sight is an integer, each instance of breakthrough increases $|f|$ by at least 1. Therefore, the algorithm will terminate in a number of steps equal to or less than the minimum capacity of an s,t-cut.

Let f and $[S, T]$ be the output of the FFA. The fact that breakthrough did not occur means that every forward edge of $[S, T]$ is being used to full capacity, and no backward edge has positive flow. That is,

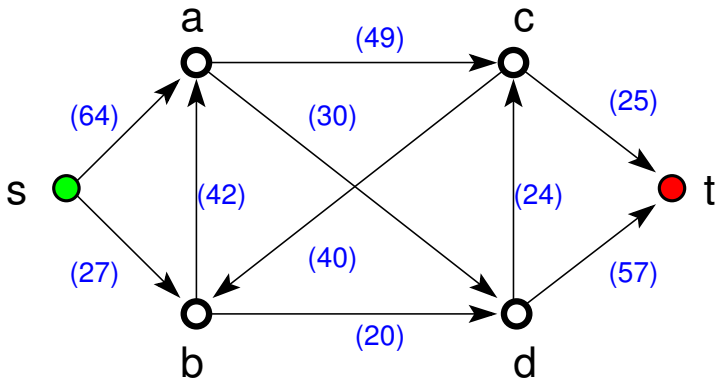
$$f(e) = c(e) \quad \forall\, \vec{e} \in [S, T] \qquad \text{and} \qquad f(e) = 0 \quad \forall\, \overleftarrow{e} \in [S, T].$$

But this says exactly that

$$|f| = f(S, T) = c(S, T)$$

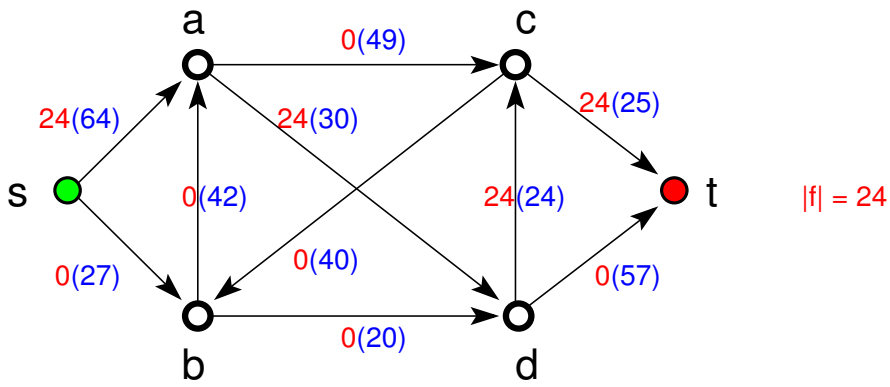
and so by weak duality, f is a maximum flow and $[S, T]$ is a minimum source-sink cut. □

Example 4.27. Let N be the network shown.

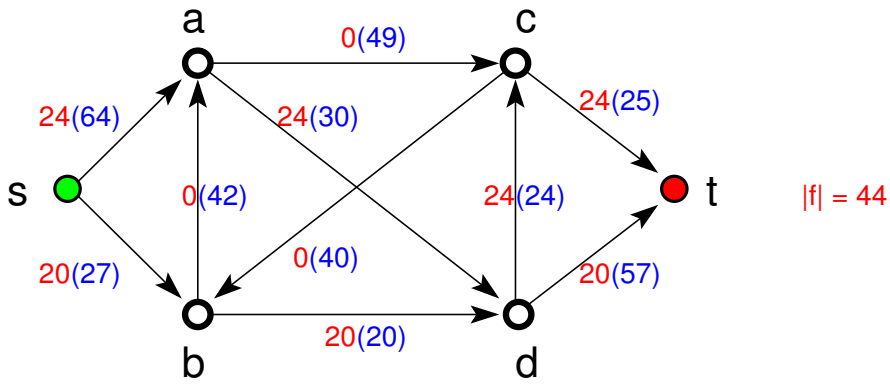


Initialize f to be the zero flow and work through the algorithm. Note that there may be several possible augmenting paths in each iteration, so in that sense the algorithm is not deterministic.

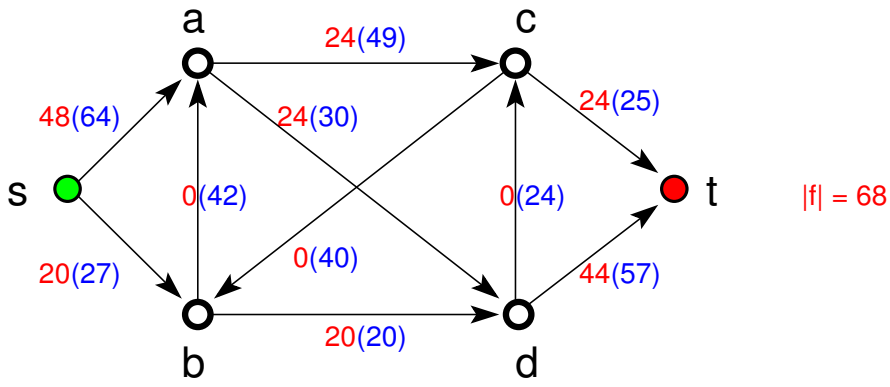
Step 1:
Augmenting path: $P = s, a, d, c, t$
Edge tolerances: $\varepsilon(sa) = 64$, $\varepsilon(ad) = 30$, $\varepsilon(dc) = 24$, $\varepsilon(ct) = 25$
Path tolerance: $\min\{64, 30, 24, 25\} = 24$



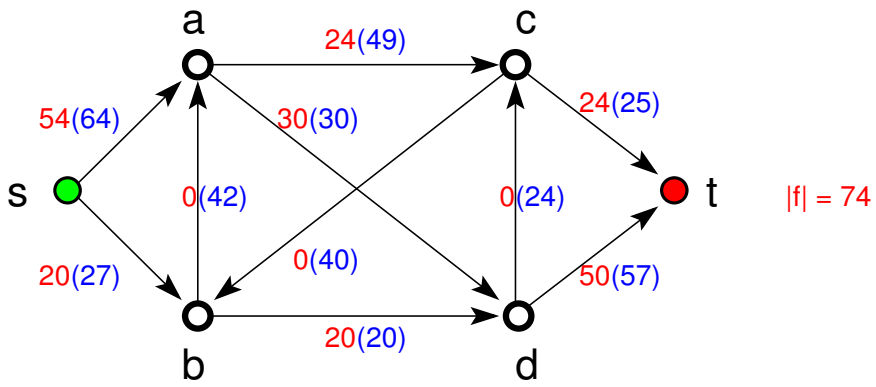
Step 2:
Augmenting path: $P = s, b, d, t$
Tolerance: $\varepsilon(P) = \min\{27, 20, 57\} = 20$



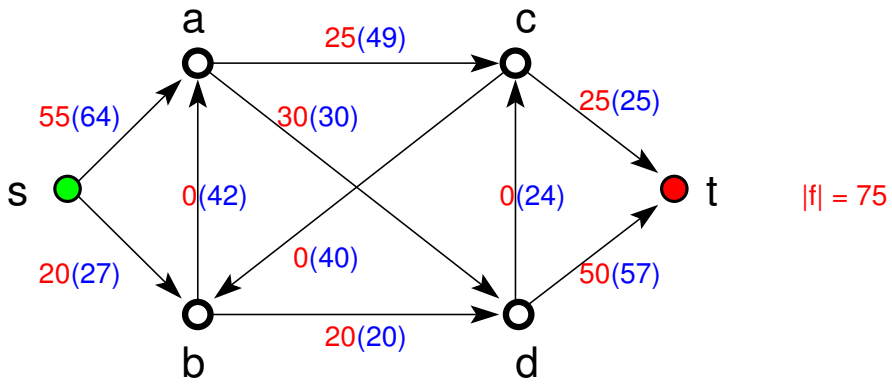
Step 3:
Augmenting path: $P = s, a, c, d, t$. Note that $\overrightarrow{dc} \in E$, so we have a backward edge.
Tolerance: $\varepsilon(P) = \min\{40, 49, 24, 37\} = 24$. Note that $\varepsilon(\overleftarrow{cd}) = f(\overrightarrow{dc}) = 24$.
New flow: Add 24 to $f(\overrightarrow{sa})$, $f(\overrightarrow{ac})$, $f(\overrightarrow{dt})$; subtract 24 from $f(\overrightarrow{dc})$.



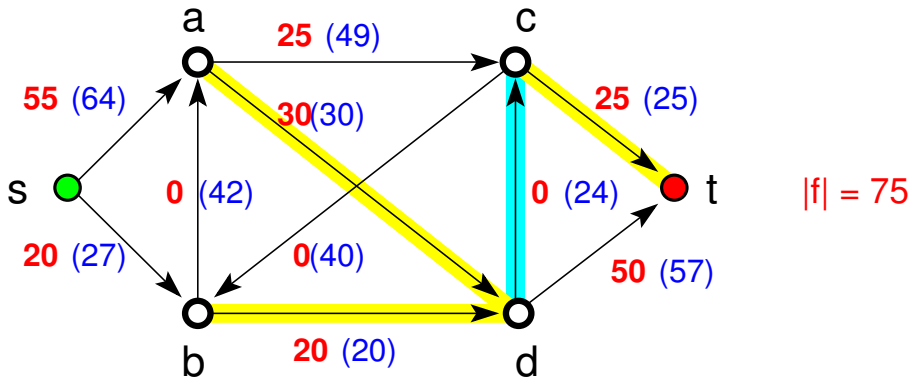
Step 4:
Augmenting path: $P = s, a, d, t$
Tolerance: $\varepsilon(P) = \min\{16, 6, 13\} = 6$



Step 5:
Augmenting path: $P = s, a, c, t$
Tolerance: $\varepsilon(P) = \min\{64 - 54, 49 - 24, 25 - 24\} = 1$



Step 6: At this point in the algorithm, breakthrough fails, since every edge in the cut $\big[S = \{s, a, b, c\},\ T = \{d, t\}\big]$ is either a forward edge being used at capacity (yellow), or a backward edge with flow 0 (blue).



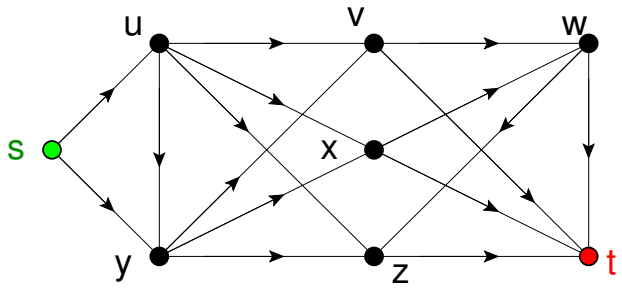
Moreover, $c(S,T) = c(\overrightarrow{ad}) + c(\overrightarrow{bd}) + c(\overrightarrow{ct}) = 30 + 20 + 25 = 75 = |f|$. So f is a maximum flow and $[S,T]$ is a minimum cut. The algorithm has succeeded!!

Note that the algorithm still works if the capacities are required only to be rational, rather than integers. It is really the same problem — since the network has finitely many edges, simply multiply every capacity by the greatest common denominator to convert the problem to an integer one. If the capacities are required to be

real, then the Max-Flow and Min-Cut problems are still strongly dual (in the sense of linear programming), but the Ford-Fulkerson algorithm may not terminate.

Example 4.28. This example is taken from V. Chvátal, *Linear Programming* (W.H. Freeman & Co., 1983), pp. 387–388. Let $N = (G, s, t, c)$ be the network shown below, with capacity function

$$\begin{aligned} c(\overrightarrow{uy}) = c(\overrightarrow{xw}) &= (-1+\sqrt{5})/2, \\ c(\overrightarrow{vw}) &= 1, \\ c(\overrightarrow{e}) &= 10^{100} \quad \text{for all other } e \in E(G). \end{aligned}$$



Here the success of the Ford-Fulkerson algorithm depends on the augmenting paths chosen. If the algorithm is unlucky, the sequence of augmenting paths used to increase flow might be

$P_1 = suyvwt$, $P_2 = syuvwzt$, $P_3 = suyxwt$, $P_4 = syuxwvt$, $P_5 = suyzwt$, $P_6 = syuzwxt$, $P_1, P_2, P_3, \ldots$

resulting in an infinite loop. (I haven't worked out the details.)

4.8. **Corollaries of the Max-Flow/Min-Cut Theorem.** Henceforth, we assume that all capacity functions and flows are integers.

Proposition 4.29 (Acyclic flows)**.** *Every network N has a maximum flow f that is* acyclic *in the sense that for every directed cycle $C \subseteq N$, there is at least one edge of C with $f(e) = 0$.*

Proof. More generally, any flow f can be "acyclized" as follows. If C is a directed cycle with $f(e) > 0$ for every $e \in C$, then $\varepsilon = \min\{f(e) : e \in C\} > 0$, and we can define a new flow $\tilde{f}$ by

$$\tilde{f}(e) = \begin{cases} f(e) - \varepsilon & e \in C, \\ f(e) & e \notin C. \end{cases}$$

Then $\tilde{f}$ satisfies the capacity and conservation constraints and $|\tilde{f}| = |f|$. Moreover, Moreover, $\tilde{f}$ has at least one more edge with zero flow than f did, so if we repeat this construction we will eventually produce an acyclic flow. □

Proposition 4.30 (Partitionability of flows)**.** *Every acyclic integer flow f can be "partitioned into paths." That is, there is a family of directed s,t-paths $\mathcal{P} = \{P_1, \ldots, P_k\}$ such that $k = |f|$ and*

$$f(e) = \#\{i \mid e \in P_i\}.$$

Warning: There is certainly no guarantee that $\mathcal{P}$ is a DPF or EPF as defined above in §4.1 and §4.2. In fact, typically it will be neither.

Proof. The following algorithm constructs such a family:

- Initialize $\mathcal{P} := \varnothing$.
- Start walking from s along edges of positive flow until you reach t.

- Put P into $\mathcal{P}$, and reduce flow by 1 along every edge of P.
- Repeat until $|f| = 0$.

Since f is acyclic, every walk P is a path. Also, the conservation constraints imply that we never get "stuck" when forming P — after entering an internal vertex of the network, it is always possible to leave. Finally, each time we put a new path into $\mathcal{P}$, the value $|f|$ decreases by 1. □

The warning above applies to networks in general. On the other hand, if we are clever about choosing the capacity function, we will automatically obtain additional constraints on the path family obtained by partitioning.

For example, let D be a digraph with $s, t \in V(D)$. Treat D as a **unit-capacity network**, i.e., $c(e) = 1$ for every $e \in E$. Then

- $\mathcal{P}$ is a (directed) s,t-EPF, and
- $c(S,T) = |[S,T]|$ for any source-sink cut $[S,T]$.

Therefore, applying the MFMC in this context, we obtain the directed edge version of Menger's theorem:

$$\max\{|\mathcal{P}| : \mathcal{P} \text{ is a directed } s,t\text{-EPF}\} = \min\{|[S,T]| : [S,T] \text{ an } s,t\text{-cut}\} = \kappa'(s,t).$$

This is a common technique to prove min/max theorems in graph theory: transform an arbitrary graph into a source/sink network in some clever way, then apply MFMC to the network to recover the desired result for your original graph.

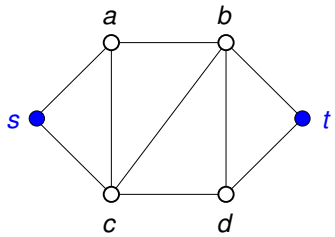
The undirected vertex version of Menger's theorem requires a slightly more elaborate transformation.

Theorem 4.31 (Menger's Theorem, finally!). *Let G be a simple graph, and let $s, t \in V(G)$ be nonadjacent. Then*

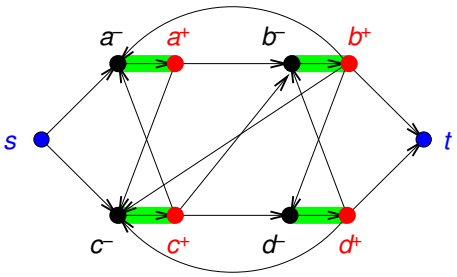
$$\max\{|\mathcal{P}| : \mathcal{P} \textit{ is an } x,y\textit{-DPF}\} \leqslant \min\{|S| : S \textit{ is an } x,y\textit{-separator}\}.$$

Proof. Construct a s,t-network N by splitting every internal vertex v into an in-vertex v^- and an out-vertex v^+. That is,

$$V(N) := \{s,t\} \cup \{v^-, v^+ \mid v \in V(G)\backslash\{s,t\}\},$$
$$E(N) := \{\overrightarrow{v^-v^+} : v \in V(G)\backslash\{s,t\}\} \quad \text{("private"; capacity = 1)}$$
$$\cup \{\overrightarrow{v^+w^-} : vw \in E(G)\} \quad \text{("public"; capacity = } \infty\text{)}.$$



G



N

For every $v \in V(N)\backslash\{s,t\}$, we have $f_{\text{out}}(v^-) = f_{\text{in}}(v^+) \in \{0,1\}$.

Let f be a feasible integer acyclic flow of N, partitioned into paths $P'_1, \ldots, P'_k$, where $k = |f|$. Each P'_i has the form

$$s,\ v_1^-,\ v_1^+,\ \ldots,\ v_\ell^-,\ v_\ell^+,\ t.$$

The path P'_i corresponds to the s,t-path P_i in G given by $s,\ v_1, \ldots,\ v_n,\ t$, and this correspondence is bijective. Moreover, each pair $\{v_-, v_+\}$ can occur in at most one P'_i (because $c(v^-v^+) = 1$), so each $v \in V(G)\backslash\{s,t\}$ belongs to at most one P_i — that is, the family $\mathcal{P} = \{P_1, \ldots, P_k\}$ is a DPF! So integer acyclic flows in N correspond exactly to s,t-DPF's in G, and

$$\max |f| = \lambda_G(s,t).$$

Meanwhile, the set of private edges is a s,t-cut of capacity $n(G) - 2$. On the other hand, any edge set containing at least one public edge has infinite[6] capacity. Therefore, every minimum source-sink cut consists only of private edges, and the corresponding vertices of G form an s,t-cut.

Therefore $\lambda_G(s,t) \ = \ \max |f| \ = \ \min c(S,T) \ = \ \kappa_G(s,t)$. □

Other applications of MFMC, many of which would make good projects, include the *Gale-Ryser Theorem* (characterizing degree sequences of a bipartite graph) and *score vectors of tournaments* (what in-/out- degree sequences can arise in an orientation of K_n?) Related problems and generalizations include networks with multiple sinks and sources; networks in which different vertices have different supplies and demands for flow; and cost networks (where each edge has a cost per unit flow, and the problem is to find a feasible flow of fixed value and minimum cost).

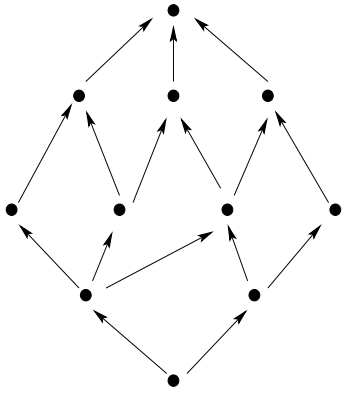
4.9. **Path covers and Dilworth's theorem.** A digraph D is called *transitive* if, whenever $\overrightarrow{uv}$ and $\overrightarrow{vw}$ are edges, then so is $\overrightarrow{uw}$. An acyclic transitive digraph is essentially the same thing as a *partially ordered set* (or *poset*): the edge $\overrightarrow{uv}$ is regarded as recording the relation $u < v$, and acyclicity and transitivity say that

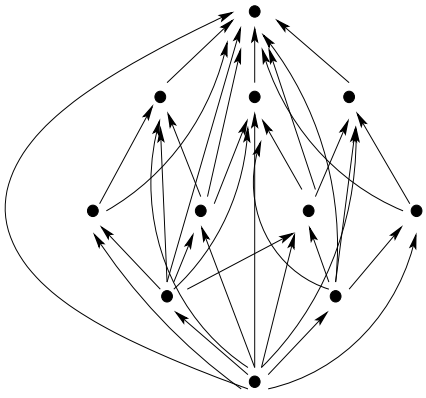
$$u < v \text{ and } v < w \quad \Longrightarrow \quad u < w$$

and

$$\text{it cannot be the case that both } u < v \text{ and } u > v.$$

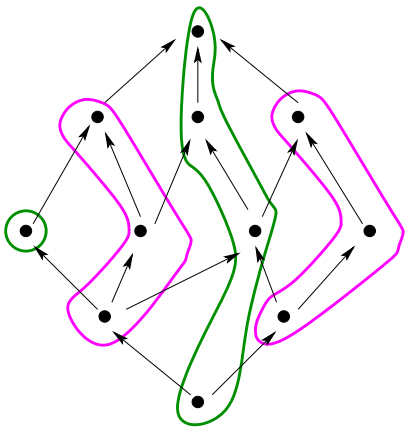
When drawing a transitive digraph, it's enough to specify a set of edges whose transitive closure contains all the other edges. For example, a transitive digraph whose edges include those shown on the left below must in fact contain all the edges shown on the right — but the right-hand picture is disgusting, so it's easier to show the left-hand picture and just remember that it's supposed to be transitive.

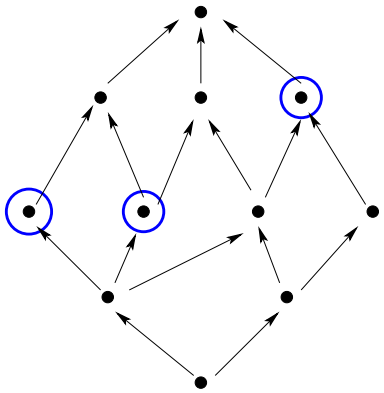




[6]If this bothers you, just define the capacity of the public edges large enough, say, oh, I don't know, $(73n + 86)!$, so that the sentence following the footnote is true.

A **path cover** in a transitive digraph is a collection $\mathcal{P}$ of directed paths that partition the vertex set (see figure on left, below — note that the bottom edge of the big green path is not shown, but it is in the transitive closure of the edges shown). An **independent set** is a collection I of vertices such that no two lie on any directed path (see figure on right, below).





In the language of posets, a path is a **chain** (a set of elements such that every two are comparable) and an independent set is a **antichain** (a set of elements such that no two are comparable).

Theorem 4.32 (Dilworth's Theorem)**.** *In every acyclic digraph D, the minimum size of a path cover equals the maximum size of an independent set.*

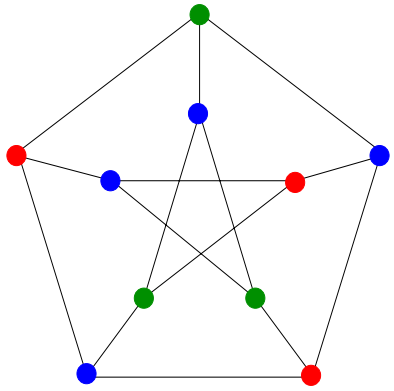
The proof is left as an exercise. As usual, the weak duality relation, namely $|\mathcal{P}| \geqslant |I|$ for every path cover $\mathcal{P}$ and independent set I, is the easy part. The strong duality relation is an application of the König-Egerváry Theorem (and thus ultimately of the Max-Flow/Min-Cut Theorem). There is an amazing generalization of Dilworth's Theorem due to Greene and Kleitman, but it is somewhat beyond the scope of graph theory — take algebraic combinatorics!

5. Coloring

5.1. **The basics.** Let $G = (V, E)$ be a simple graph and $n = n(G)$. Recall that the notation $[k]$ means the set of positive integers $\{1, 2, \ldots, k\}$.

Definition 5.1. A **[proper] coloring** of G is a function $c : V \to S$ such that $c(v) \neq c(w)$ whenever v, w are adjacent. Typically $S = [k] = \{1, 2, \ldots, k\}$ or $S = \mathbb{N}$. The elements of S are called **colors**. The **color class** corresponding to a color i is the sets $c^{-1}(i) \stackrel{\text{def}}{=} \{v \in V \mid c(v) = i\}$. Note that the color classes are cocliques and each vertex belongs to exactly one of them. The **chromatic number $\boldsymbol{\chi} = \boldsymbol{\chi}(\boldsymbol{G})$** is the smallest k such that there is a coloring $c : G \to [k]$. The graph G is **$\boldsymbol{k}$-colorable** if $\chi(G) \leqslant k$.

Example: Let G be the Petersen graph. It is impossible to color G properly using only two colors. On the other hand, three colors suffice, as in the following figure, so $\chi(G) = 3$. The statement "The Petersen graph is k-colorable" is true if and only if $k \geqslant 3$.



Some observations:

(1) A coloring of G is the same thing as a partition of $V(G)$ into cocliques, so $\chi(G)$ is the minimum number of cocliques needed to partition $V(G)$.
(2) $\chi(G) = 1$ iff $E(G) = \varnothing$.
(3) $\chi(G) = 2$ iff G is bipartite. (Note that the Petersen graph isn't.)
(4) If G contains parallel edges, then we can ignore them — a proper coloring of G is the same thing as a proper coloring of its underlying simple graph. OTOH, if G contains loops, then it is impossible to properly color it! So when studying coloring, it is typically OK to assume that any graph of interest is simple.
(5) $\chi(K_n) = n$, because a coloring of K_n must assign all vertices different colors. On the other hand, if G is a simple graph on n vertices that is not K_n, then $\chi(G) < n$.
(6) If G is **planar**—that is, it is possible to draw it in the plane so that no two edges cross—then $\chi(G) \leqslant 4$. This is the notorious *four-color theorem*. On the other hand, it is much easier to prove the *five-color theorem* that $\chi(G) \leqslant 5$ for all planar G; we will do this eventually.

Proposition 5.2. *Let α, χ, ω be the coclique, chromatic, and clique numbers. Then $\chi \geqslant \omega$ and $\chi \geqslant n/\alpha$.*

(Here ω is the **clique number** $\omega = \omega(G)$ is the number of vertices in the largest clique in G. Recall that α is the number of vertices in the largest coclique, so $\omega(G) = \alpha(\bar{G})$.)

Proof. If Q is a clique, then every proper coloring must assign different colors to each vertex in Q, hence must use at least $|Q|$ colors. If Q is a maximum clique, we have $\chi \geqslant |Q| = \omega$.

On the other hand, each color class in a proper coloring is a coclique, hence has size $\leqslant \alpha$. Therefore, if G is k-colorable, then $k\alpha \geqslant n$, which gives the second inequality. □

Equality need not hold. For example, if G is an odd cycle, then $\chi(G) = 3$ but $\omega(G) = 2$.

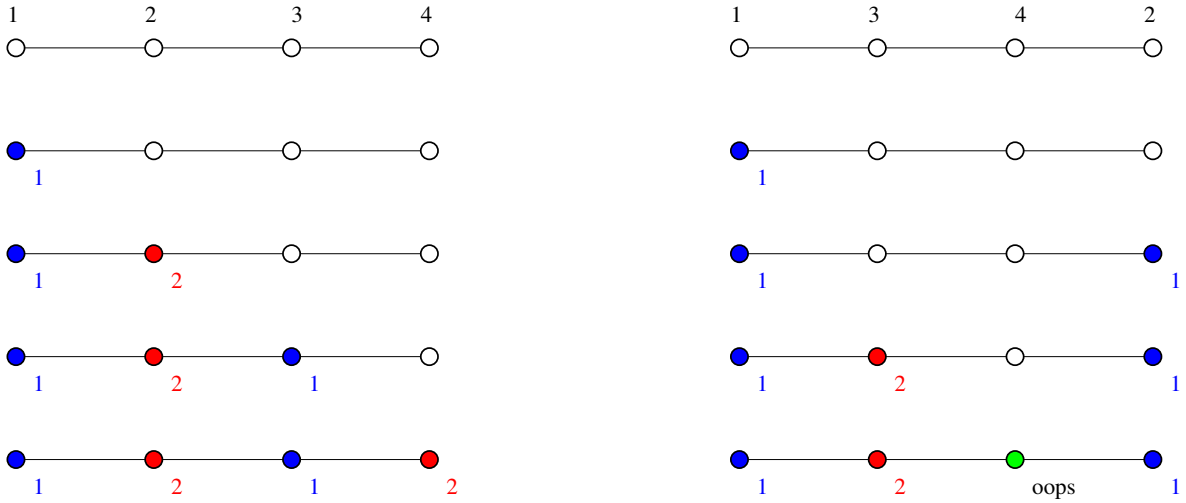
In general, computing $\chi(G)$ is hard — in fact, NP-complete. It can be reduced to the problem of computing the coclique number of the Cartesian product of G with a clique (Prop. 5.1.11).

5.2. **Greedy coloring.** Here is a simple way to produce a proper coloring $c : E(G) \to \mathbb{N}_{>0}$. Choose an ordering $v_1, \ldots, v_n$ of the vertices and assign colors to the vertices in that order, coloring each vertex with the smallest available color. That is,

$$c(v_i) := \min\left(\mathbb{N}_{>0} \backslash \{c(v_j) : \ 1 \leqslant j < i, \ v_i v_j \in E(G)\}\right).$$

That is, color the vertices one by one, assigning each vertex v the cheapest color available (i.e., the smallest number that has not already been assigned to a neighbor of v).

This algorithm *might* produce an optimal coloring. On the other hand, it might not. For instance, let $G = P_4$, so $\chi(G) = 2$ (remember, all trees are bipartite). Ordering the vertices left to right produces a 2-coloring, but coloring the two endpoints first doesn't:



Every graph has at least one ordering for which greedy coloring yields an optimal coloring (i.e., one that uses only χ colors and no more). Unfortunately, finding such an ordering is harder than finding the coloring itself. If $G = K_n$ then all orderings of course produce an optimal coloring; less trivially, the same property holds for the star $K_{1,n}$ and for odd cycles. I don't know if there is a characterization of all graphs that have this property. (Maybe threshold graphs?)

Proposition 5.3. $\chi \leqslant \Delta + 1$.

Proof. Choose any ordering $v_1, \ldots, v_n$ of V and use the greedy algorithm to construct a proper coloring f. Then f will be at worst a $(\Delta+1)$-coloring, because at each step in the algorithm, some color in $\{1, 2, \ldots, \Delta+1\}$ will always be available. □

This bound can be strengthened. Let $d_i = d_G(v_i)$. In the greedy coloring algorithm, the ith vertex has no more than $\min(i-1, \ d_i)$ earlier neighbors, so it is assigned a color no more than $1 + \min(i-1, \ d_i)$.

Therefore,

$$\chi \leqslant 1 + \max_{i \in [n]} \big\{ \min(i-1,\ d_i) \big\}. \tag{20}$$

This bound on χ is tightest if we order the vertices so that $d_1 \geqslant d_2 \geqslant \cdots \geqslant d_n$. In other words, greedy coloring is likely to work better if we assign colors to higher-degree vertices first. (Intuitively, it makes sense to color the hardest-to-color vertices when there are more colors available.)

Warning: Ordering by degree is not *guaranteed* to produce an optimal coloring; it's just more *more likely* to do so. In other words, it is a *heuristic*. (Exercise, possibly coming soon to a problem set near you: Construct a graph in which for *every* ordering of the vertices $v_1, \ldots, v_n$ such that the greedy algorithm yields an optimal coloring, there are some indices $i < j$ such that $d(v_i) > d(v_j)$.)

In fact, the easy bound $\chi \leqslant \Delta + 1$ can be improved. **Brooks' Theorem** says that If G is a connected simple graph other than a clique or an odd cycle, then $\chi(G) \leqslant \Delta$. We won't prove this (exercise?), but the idea is to cleverly construct an ordering of the vertices such that each vertex is preceded by at most $\Delta - 1$ of its neighbors, so that greedy coloring will use at most Δ colors.

5.3. **Alternative formulations of coloring.** An **edge-coloring** is a function $c : E(G) \to \mathbb{N}$ such that $c(e) \neq c(e')$ whenever e, e' have a common endpoint. In other words, it is a coloring of the line graph $L(G)$ (see Definition 4.19). The minimum number of colors needed for an edge-coloring is called the **chromatic index** of G, denoted $\chi'(G)$; thus $\chi'(G) = \chi(L(G))$. The chromatic index is much easier to pin down than the chromatic number: for instance, $\chi' = \Delta$ for bipartite graphs (a theorem of König's from 1916) and $\chi' \in \{\Delta, \Delta + 1\}$ for all graphs (Vizing's Theorem). I have chosen not to cover this material in depth, but it could make a good final project.

Suppose that each vertex $v \in G$ has an associated list S_v of colors that it is allowed to have. A **list-coloring** of G is then a coloring c such that $c(v) \in S_v$ for every $v \in V(G)$? Of course, this depends on the lists. A graph is ***k*-list-colorable**, or ***k*-choosable**, if for *every* family of lists $(S_v)_{v \in V(G)}$ there exists a compatible list-coloring. Analogously, an edge-list-coloring of G is a list-coloring of $L(G)$. Thus we can define

$$\mathrm{ch}(G) = \min\{k :\ G \text{ is } k\text{-choosable}\},$$
$$\mathrm{ch}'(G) = \min\{k :\ L(G) \text{ is } k\text{-choosable}\}.$$

It is easy to see that $\mathrm{ch}(G) \geqslant \chi(G)$ (just set all the lists equal). Equality does not hold: for example, $K_{3,3}$ is 2-colorable but not 2-choosable. The **List-Coloring Conjecture** says that $\mathrm{ch}'(G) = \chi;(G)$ for all graphs G. This is a major open problem, but one of the best-known theorems in this area is due to Fred Galvin, Emeritus Professor of Mathematics at the University of Kansas, who proved that the List-Coloring Conjecture is true if G is bipartite [*J. Combin. Theory Ser. B* **63** (1995), no. 1, 153–158]. Again, this would be an excellent final project!

5.4. **The chromatic polynomial (not in Diestel).**

Definition 5.4. The **chromatic polynomial** of G is

$$p_G(k) = \text{ number of } k\text{-colorings of } G.$$

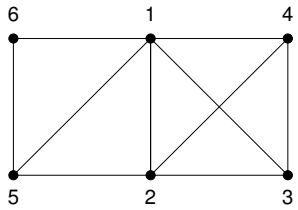
Some easy examples:

G	$p_G(k)$
$\overline{K_n}$	k^n
K_n	$k(k-1)(k-2)\dots(k-n+1)$
$G' + G''$	$p_{G'}(k) \cdot p_{G''}(k)$
has a vertex v of degree 1	$(k-1)p_{G-v}(k)$
Tree with n vertices	$(k-1)^{n-1}p_{K_1}(k) = k(k-1)^{n-1}$

The chromatic polynomial is a (much) stronger invariant than the chromatic number. I.e., $p_G(k)$ determines $\chi(G)$ — specifically, $\chi(G) = \min\{k : \ p_G(k) > 0\}$ — but not vice versa.

We have not yet proved that $p_G(k)$ is a polynomial function of k for all G, but we will do so shortly (justifying its name). But first some examples.

Example 5.5. If we are both clever and lucky about the order in which to color the vertices of G, then the number of colors available for each vertex will not depend on the previous choices. For example, let G be the graph below, with $V(G) = \{1, 2, 3, 4, 5, 6\}$.



Construct a proper coloring f by choosing colors $c(1), \dots, c(6)$ in that order. We have:

- k choices for $c(1)$;
- $k-1$ choices for $c(2)$ [can't be $c(1)$];
- $k-2$ choices for $c(3)$ [can't be $c(1)$ or $c(2)$ — note that those two must be different];
- $k-3$ choices for $c(4)$ [can't be $c(1)$, $c(2)$, or $c(3)$ — again, those must all be different];
- $k-2$ choices for $c(5)$ [can't be $c(1)$ or $c(2)$];
- $k-2$ choices for $c(6)$ [can't be $c(1)$ or $c(5)$].

Therefore, $p_G(k) = k(k-1)(k-2)(k-3)(k-2)(k-2) = k(k-1)(k-2)^3(k-3)$.

What made this calculation so easy was the following property:

> For every $v \in V$, the induced subgraph on $N(v) \cap \{1, 2, \dots, v-1\}$ is a clique.

An ordering of vertices with this property is called a **simplicial ordering**. (Alternative terms abound: simplicial elimination ordering, perfect elimination ordering, etc.)

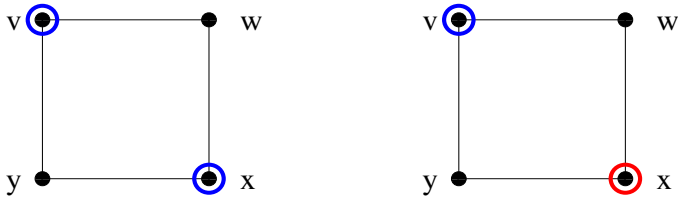
Theorem 5.6. *Let G be a simple graph. The following conditions are equivalent:*

(1) G has a simplicial elimination ordering.
(2) G has no induced cycle of length $\geqslant 4$. Equivalently, every cycle in G of length 4 or more has a chord *(an edge between two vertices that are not adjacent in the cycle)*
(3) Either G is a clique, or $G = G_1 \cup G_2$, where G_1 and G_2 are chordal induced subgraphs and $G_1 \cap G_2$ is a clique.

Graphs that satisfy these conditions are called **chordal graphs** for the reason of condition (2), and are a very interesting and important family. The implication (1) $\implies$ (2) is not too had but (2) $\implies$ (1) is trickier. The equivalence of (2) and (3) is known as *Dirac's theorem*. The chromatic polynomials of chordal graphs always split into linear factors — using greedy coloring with the reverse of a simplicial ordering, as in Example 5.5

Example 5.7. If we don't have an SEO, then it's harder to calculate the chromatic polynomial, because the number of colors available for each vertex will not depend on the previous choices.

Let $G = C_4$. We will try to determine $p_G(k)$. Start by choosing the colors of two opposite vertices v, x. The problem is that since $vx \notin E$, we don't know whether or not $c(v) = c(x)$, and so we have two different possibilities for the number of colors available for the other vertices.



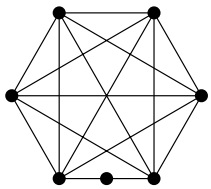
- If $c(v) = c(x)$ (**left**), then there are k choices for $c(v) = c(x)$, and $k-1$ independent choices for each of $c(u), c(w)$.
- If $c(v) \neq c(x)$ (**right**), then there are $k(k-1)$ independent choices for $c(v)$ and $c(x)$, and $k-2$ independent choices for each of $c(u), c(w)$.

Therefore, the chromatic polynomial of C_4 is $k(k-1)^2 + k(k-1)(k-2)^2 = k(k-1)(k^2-3k+3)$.

Alternately, we could have colored v, then w, then x, but then we'd still have to worry about whether or not $c(v) = c(x)$.

This is a relatively simple case; for bigger graphs, these calculations can get much uglier, with cases, subcases and subsubcases galore. (Similar tedium arises if G is chordal but the ordering of vertices is not an SEO.)

Since you asked, there do exist non-chordal graphs whose chromatic polynomials split. The smallest example is the graph obtained by taking K_6 and subdividing one edge, as shown below.



5.5. **The chromatic recurrence.** A more systematic (though still exponentially difficult) way to calculate $p_G(k)$ is by the following recurrence.

Theorem 5.8 (Chromatic Recurrence)**.** *For every $e \in E(G)$,*

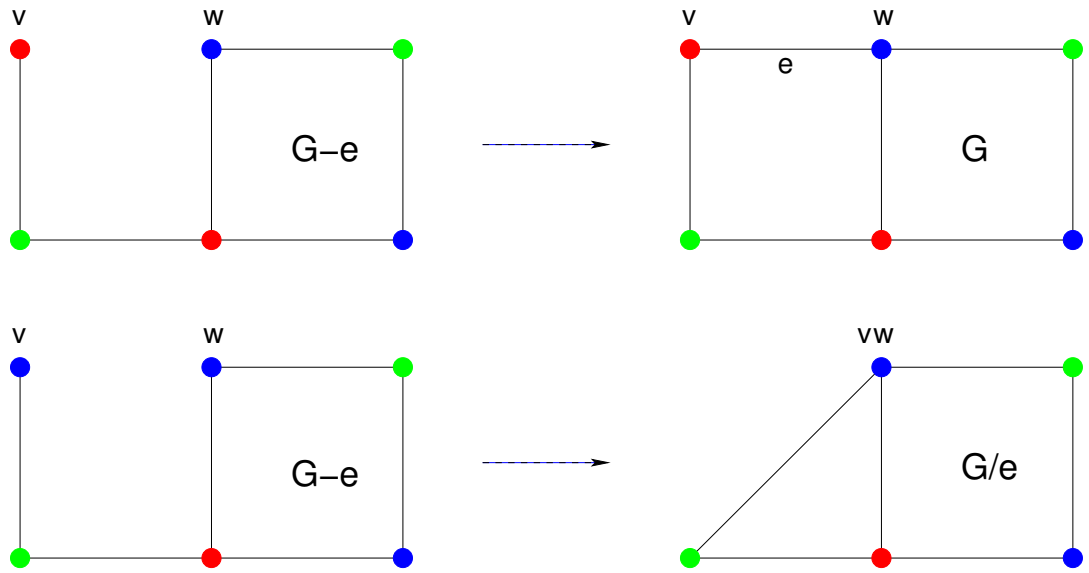
$$p_G(k) \;=\; p_{G-e}(k) \;-\; p_{G/e}(k).$$

Proof. Let v, w be the endpoints of e. First, every proper coloring f of G is certainly a proper coloring of $G - e$ (as deleting an edge will not turn a proper coloring improper).

OTOH, a proper coloring f of $G-e$ is a proper coloring of G if and only if $c(v) \neq c(w)$. Therefore,

$$p_{G-e}(k) \;-\; p_G(k) \;=\; \#\{\text{proper } k\text{-colorings } f \text{ of } G-e \text{ with } c(v)=c(w)\}.$$

But if $c(v)=c(w)$, then f corresponds to a proper coloring of G/e — just color the merged vertex $x=vw$ by the color $c(v)=c(w)$. Conversely, a proper coloring of G/e can be "expanded" to a proper coloring of G with $c(v)=c(w)=c(x)$. Therefore, the right-hand side of the previous equation is just $p_{G/e}(k)$, and we're done. □



Corollary 5.9. *$p_G(k)$ is a polynomial in k.*

This is immediate from the chromatic recurrence, by induction on the number of edges and the base case $p_{\overline{K_n}}(k) = k^n$.

Example 5.10. Let $G = C_4$. For each edge e, $G-e \cong P_4$ (a tree with 3 edges) and $G/e \cong K_3$, so

$$\begin{aligned} p_{C_4} &= p_{P_4} \;-\; p_{K_3} \\ &= k(k-1)^3 \;-\; k(k-1)(k-2) \\ &= k(k-1)\big((k-1)^2-(k-2)\big) \;=\; k(k-1)(k^2-3k+3) \end{aligned}$$

confirming the earlier calculation. More generally, there is a reasonably nice formula for p_{C_n} in terms of n (exercise!).

- One very nice feature of the chromatic recurrence is that the edge e can be chosen arbitrarily. This is counterintuitive, but is actually characteristic of all deletion-contraction recurrences.

- If a contraction produces parallel edges, then we can remove all but one member of each parallel class; this doesn't affect the chromatic polynomial.

- If G has lots of edges, it may be more convenient to run the algorithm backwards. That is, if v, w are nonadjacent vertices of G, then the chromatic recurrence gives

$$p_G(k) \;=\; p_{G+vw}(k) \;+\; p_{G/vw}(k)$$

where $G+vw$ is formed by adding an edge between v and w and G/vw is obtained by fusing v and w into a single vertex.

Example 5.11. Let G be the "near-complete graph" consisting of K_n with a single edge $e = xy$ removed. Then

$$\begin{aligned} p_G = p_G(k) &= p_{G+e} \ + \ p_{G/e} \\ &= p_{K_n} \ + \ p_{K_{n-1}} \\ &= k(k-1)(k-2)\cdots(k-n+1) \ + \ k(k-1)(k-2)\cdots(k-n+2) \\ &= k(k-1)(k-2)\cdots(k-n+3)(k-n+2)^2. \end{aligned}$$

Another consequence is the following fact about the chromatic polynomial of any graph:

Theorem 5.12. *Let G be a simple graph with $n = n(G)$ and $r = e(G)$.*
(1) The coefficients of $p_G(k)$ alternate in sign.
(2) $p_G(k) = k^n - rk^{n-1}+$ lower-order terms.

I.e., G is monic of degree n, and the second-leading coefficient tells you the number of edges. So these two (very basic) invariants of a graph are determined by its chromatic polynomial.

Proof. By induction on r.

Base case: If $r = 0$, then $p_G(k) = k^n$ and the claims hold trivially.

Inductive step: Suppose $r > 0$. Pick an edge $e \in E(G)$. By induction, the theorem is true for both $G - e$ and G/e. That is,

$$p_{G-e}(k) = \sum_{i=0}^{n} (-1)^i a_i k^{n-i},$$
$$p_{G/e}(k) = \sum_{i=0}^{n-1} (-1)^i b_i k^{n-1-i},$$

where the a_i and b_i are nonnegative integers with

$$a_0 = b_0 = 1, \qquad a_1 = b_1 = r - 1. \tag{21}$$

By the chromatic recurrence,

$$\begin{aligned} p_G(k) &= p_{G-e}(k) \ - \ p_{G/e}(k) \\ &= \left(\sum_{i=0}^{n} (-1)^i a_i k^{n-i} \right) \ - \ \left(\sum_{i=0}^{n-1} (-1)^i b_i k^{n-(i+1)} \right) \\ &= \left(\sum_{j=0}^{n} (-1)^j a_j k^{n-j} \right) \ - \ \left(\sum_{j=1}^{n} (-1)^{j-1} b_{j-1} k^{n-j} \right) \\ &= k^n + \sum_{j=1}^{n} (-1)^j (a_j + b_{j-1}) k^{n-j}. \end{aligned}$$

This polynomial is evidently monic of degree n, and alternates in sign. Also, the next-to-leading coefficient (i.e., on k^{n-1}) is $a_1 + b_0 = r = e(G)$ by (21). □

5.6. **Colorings and acyclic orientations.** An orientation of a graph G is called **acyclic** if it contains no directed cycle. Let $a(G)$ be the number of acyclic orientations.

For example:
- If G is a forest with e edges, then all 2^e orientations of G are acyclic, so $a(G) = 2^e$..
- If $G = C_n$, then all but two of its orientations are acyclic, so $a(C_n) = 2^n - 2$.

• If $G = K_n$, then it can be proven that every acyclic orientation corresponds to a total ordering of the vertices in which each edge points from the smaller to the greater vertex. In particular, $a(K_n) = n!$.

Theorem 5.13 (Stanley, 1973)**.** $a(G) = (-1)^{n(G)} p_G(-1)$.

Example: If G is a forest with n vertices and c components, then

$$(-1)^n p_G(-1) \;=\; (-1)^n(-1)^c(-1-1)^{n-c} = 2^{n-c} = 2^r.$$

Example: If $G = K_n$ then $(-1)^n p_{K_n}(-1) = (-1)^n \prod_{i=0}^{n-1}(-1-i) = \prod_{i=0}^{n-1}(i+1) = n!$.

Sketch of proof. Let $\tilde{a}(G) = (-1)^n p_G(-1)$. If $e(G) = 0$, then $p_G = k^n$, so $\tilde{a}(G) = 1$ is indeed the number of acyclic orientations.

Otherwise, induct on $e(G)$. The chromatic recurrence implies that $\tilde{a}(G) = \tilde{a}(G-e)+\tilde{a}(G/e)$ for every edge e. In fact, $a(G)$ satisfies the same recurrence. To prove this, classify the acyclic orientations of G according to whether or not the orientation of e can be reversed, and relate them to acyclic orientations of $G - e$ and of G/e. The details are left to the reader — compare the proof of Theorem 4.18. □

Actually, Stanley proved something more general, namely a combinatorial interpretation for the numbers $p_G(-k)$ for all $k \in \mathbb{N}$ (not just $k = 1$).

We now have seen four ostensibly unrelated graph invariants that satisfy deletion-contraction recurrences: the number of spanning trees, the number of strong orientations, the chromatic polynomial, and the number of acyclic orientations. What's behind all this?

The connection between colorings and acyclic orientations is actually quite deep. Given a coloring $c : V(G) \to \mathbb{N}$, there is a natural way to define an acyclic orientation: simply orient each edge in the direction of increasing color. (Do you see why this is acyclic?) Conversely, given an acyclic orientation, one can study the class of all colorings that give rise to it.

This connection can be explained geometrically. Given a simple graph G on vertex set $[n]$, we can associate each edge $ij \in G$ with a hyperplane in $\mathbb{R}^n$, namely

$$H_{ij} = \{\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n \mid x_i = x_j\}.$$

The set $\mathcal{A}_G = \{H_{ij} \mid ij \in E(G)\}$ is called the **graphic hyperplane arrangement** of G. It partitions $\mathbb{R}^n$ into a number of regions.

What is a coloring $c : V(G) \to \mathbb{N}$? It is just a point in

$$(\mathbb{R}^n \backslash \mathcal{A}_G) \cap \mathbb{N}^n.$$

Indeed, any integer point $\mathbf{x}$ in $\mathbb{R}^n$ can be regarded as a function $c : V(G) \to \mathbb{N}$ sending i to x_i, and the condition that $\mathbf{x} \notin \mathcal{A}_G$ says precisely that the function is a coloring.

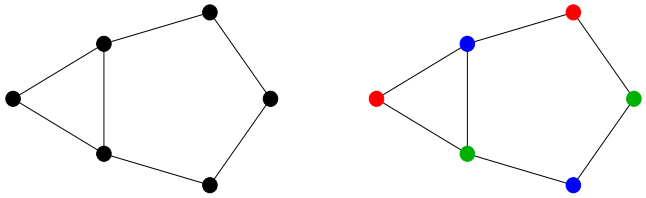
Meanwhile, the acyclic orientations correspond precisely to the regions of $\mathcal{A}_G$. If $e = ij$ is any edge, then which side of the hyperplane H_{ij} a point $\mathbf{x}$ is on tells you which one of x_i or x_j is bigger, hence gives an acyclic orientation of e. Moreover, if D is any orientation, then the system of linear inequalities

$$\{x_i < x_j \mid \vec{ij} \in D$$

has its solution space equal to a region of $\mathcal{A}_G$ if D is acyclic, and empty otherwise.

This is just the tip of the iceberg of the theory of hyperplane arrangements. For much more on this connection, see, e.g., various works by Matthias Beck.

5.7. **Perfect graphs.** We have seen that $\chi(G) \leqslant \omega(G)$ in general, and that equality need not hold; for example, if $G = C_n$ with $n \geqslant 5$ odd, then $\chi(G) = 3$ and $\omega(G) = 2$. When does equality hold? In a sense this is not the right question to ask. Consider the graph obtained by identifying copies of C_3 and C_5 along an edge.



This graph has $\chi = \omega = 3$ (the 3-coloring shown on the right is optimal), but it still contains an induced 5-cycle. So while we've fixed the problem with C_5 that $\chi > \omega$, we've sort of cheated. This leads

Definition 5.14. A graph G is **perfect** if $\chi(H) = \omega(H)$ for every induced subgraph $H \subseteq G$.

This is a very well-studied class of graphs. It is not hard to prove that chordal graphs (see Theorem 5.6 above) are perfect. The first major theorem about perfect graphs, first proved by L. Lovász [*J. Comb. Theory Ser. B* **13** (1972), 95–98] is as follows:

Theorem 5.15 (Perfect Graph Theorem)**.** *G is perfect if and only if $\bar{G}$ is perfect.*

Lovász's proof is widely admired, and would make an excellent final project. A major recent advance was the characterization of perfect graphs in terms of excluded induced subgraphs, conjectured by Berge in 1963 and proven by M. Chudnovsky, N. Robertson, P. Seymour and R. Thomas [*Ann. Math.* (2) **164** (2006), no. 1, 51–229]:

Theorem 5.16 (Strong Perfect Graph Theorem)**.** *G is perfect iff it contains no induced subgraph isomorphic to C_n or $\overline{C_n}$ for any $n \geqslant 5$.*

The Strong Perfect Graph Theorem implies the Perfect Graph Theorem, since the condition of the former is clearly preserved under complementation. (This would decidedly *not* make a good final project — note the length of the paper. But there is probably a more manageable summary of the proof in some other source.)

Instead of giving Lovász's original proof of the PGT, I will follow Diestel in giving a really slick linear algebra proof, due to G. Gasparian [*Combinatorica* **16** (1996), 209–212], of a result stronger than the Perfect Graph Theorem.

Theorem 5.17. *G is perfect if and only if $n(H) \leqslant \alpha(H)\omega(H)$ for all induced subgraphs $H \subseteq G$.*

Note that this condition is self-complementary (because complementation swaps α and ω), so this result implies Theorem 5.15.

Proof. As usual, one direction (in this case $\Longrightarrow$) is easy and one ($\Longleftarrow$) is hard.

Suppose that G is perfect. Let $H \subseteq G$ and let c be an optimal coloring of H. Then the number of color classes is $\chi(H) = \omega(H)$, and each color class is a coclique in H, hence has size at most $\alpha(H)$. It follows that $n(H) \leqslant \alpha(H)\omega(H)$.

For the converse, let G be a graph of smallest possible order such that G is not perfect but

$$n(H) \leqslant \alpha(H)\omega(H) \tag{22}$$

for all induced $H \subseteq G$. Let $\alpha = \alpha(G)$ and $\omega = \omega(G)$.

First, we claim that for every nonempty coclique $A \subseteq V$ we have

$$\chi(G - A) = \omega(G - A) = \omega. \tag{23}$$

The first equality follows because $G - A \subsetneq G$ is perfect. For the second equality, clearly $\omega(G - A) \leqslant \omega(G)$, and if $\chi(G - A) < \omega(G)$ then $\chi(G) \leqslant \omega(G)$ (since any $(k-1)$-coloring of $G - A$ can be extended to a k-coloring of G by assigning color k to all vertices in the coclique A). But that would imply that G is perfect, which we have assumed it isn't.

Now, we cook up a big batch of cliques and cocliques. Define

$$\begin{aligned}
A_0 = \{u_1, \ldots, u_\alpha\} &= \text{some maximum coclique in } G, \\
A_1, \ldots, A_\omega &= \text{color classes of some } \omega\text{-coloring } c_1 \text{ of } G - u_1 \text{ (note that these are all cocliques)}, \\
A_{\omega+1}, \ldots, A_{2\omega} &= \text{color classes of some } \omega\text{-coloring } c_2 \text{ of } G - u_2, \\
&\ldots \\
A_{(\alpha-1)\omega+1}, \ldots, A_{\alpha\omega} &= \text{color classes of some } \omega\text{-coloring } c_\alpha \text{ of } G - u_\alpha.
\end{aligned}$$

Every A_i is a coclique, so by (23) there is a clique K_i of size ω in $G - A_i$.

Claim 1: For every ω-clique $K \subseteq V(G)$, we have $K \cap A_i = \varnothing$ for exactly one $i \in \{0, \ldots, \alpha\omega\}$.

To see this, first note that if $K \cap A_0 = \varnothing$, then for every i, K is a clique in $G - u_i$, hence meets every color class of c_i. OTOH, if $K \cap A_0 = \{u_i\}$ for some i (which must be unique), then by the same argument K meets every color class of c_j for $j \neq i$, and meets all but one of the color classes of c_i.

Now, we define some matrices. Let $\theta = \alpha\omega + 1$.

Matrix	Size	Description
J	$\theta \times \theta$	0 on main diagonal; 1 elsewhere
A	$\theta \times n$	Rows are incidence vectors of $A_0, \ldots, A_{\alpha\omega}$
B	$n \times \theta$	Columns are incidence vectors of $K_0, \ldots, K_{\alpha\omega}$

Claim 2: $AB = J$.

To see this, note that the dot product of the i^{th} row of A with the j^{th} column of B is just the cardinality of $A_i \cap K_j$. For $i = j$, that intersection is empty by the construction of K_j, but then Claim 1 implies that $K_j \cap A_i \neq \varnothing$ for every $i \neq j$, and the intersection can't have size more than 1 because A_i is a coclique and K_j is a clique, so it is exactly 1.

On the other hand, J is nonsingular (details left to the reader[7]). In other words,

$$\operatorname{rank} J = \theta \leqslant \min(\operatorname{rank} A, \operatorname{rank} B) \leqslant \min(\theta, n)$$

which implies that

$$\theta = \alpha\omega + 1 \leqslant n.$$

But this contradicts (22) for $H = G$. So all this is impossible, and we are done. □

[7]Hint: J is diagonalizable, and it has the same eigenspaces as the Laplacian of K_θ.

6. Planarity and Topological Graph Theory

Here's a classical brain teaser. *Three houses are to be linked to three utilities (water, gas and electricity, let's say). No two links can cross. How can this be done?* In graph theory terms, this is the problem of drawing $K_{3,3}$ in the plane without any two edges crossing. In fact, this is impossible, as we will show. Here are some related questions we will try to get at.

- Which graphs are *planar*, i.e., can be drawn in the plane without any edge crossings?
- What properties do planar graphs have? (The Four-Color Theorem is one of the most famous.)
- What can we say about drawings of graphs on surfaces other than the plane (e.g., the torus)?

6.1. Plane graphs, planar graphs, and Euler's formula.

Up until now, we've defined graphs *purely combinatorially*. That is, vertices and edges have simply been elements of abstract sets, not points and curves in space. We have insisted that the structure of a graph is independent of the way it is drawn. Now we are going to try to understand graphs by studying how they *can* be drawn.

Definition 6.1. A **plane curve** is the image of a continuous function $\phi : [0,1] \to \mathbb{R}^2$. (It doesn't hurt to impose stronger conditions than continuity: e.g., ϕ is differentiable, or of class C^∞, or piecewise linear.) The curve is **closed** if $\phi(0) = \phi(1)$. It is **simple** if ϕ is injective (except possibly $\phi(0) = \phi(1)$).

Definition 6.2. A **plane graph** is a pair $\Gamma = (V, E)$, where V consists of a finite set of points in $\mathbb{R}^2$, and E consists of a finite set of simple curves such that

(1) each curve has two endpoints in V (which can be the same)[8];
(2) no curve meets any point in V other than its endpoints;
(3) two curves can only meet at points in V.

I will consistently use Γ for geometric plane graphs and G for combinatorial graphs.

A plane graph Γ defines a graph G in an obvious way, so we can treat Γ as a graph and speak of its cycles, bridges, loops, colorings, etc. We say that Γ is a **(plane) drawing** of G. A **combinatorial invariant** of a plane graph Γ is one that depends only on G; that is, it is the same for all other drawings of G. For instance, the number of vertices, edges and components are clearly combinatorial invariants.

Definition 6.3. A graph G is **planar** if it has a plane drawing.

Proposition 6.4. *Every subgraph of a planar graph is planar. In addition, planarity is unchanged by adding or removing loops, or edges parallel to another edge.*

Proof. For the first assertion, if $H \subseteq G$ then every drawing of G gives rise to a drawing of H by simply erasing the vertices and edges not in H. For the second assertion, we can add a loop to a drawing by inserting a very small closed curve at a vertex, and we can add a parallel an edge by cloning the corresponding curve and wiggling it slightly. (Technically these arguments require some $\epsilon - \delta$-type analysis, but making them precise in this way is not worth the trouble.) □

One fact from topology we will need is the Jordan Curve Theorem (more properly the Jordan-Schönflies Theorem), which asserts that every simple closed plane curve partitions $\mathbb{R}^2$ into two pieces.[9] More precisely, if C is any simple closed curve, then $\mathbb{R}^2 \backslash C$ has two path-connected components.[10]

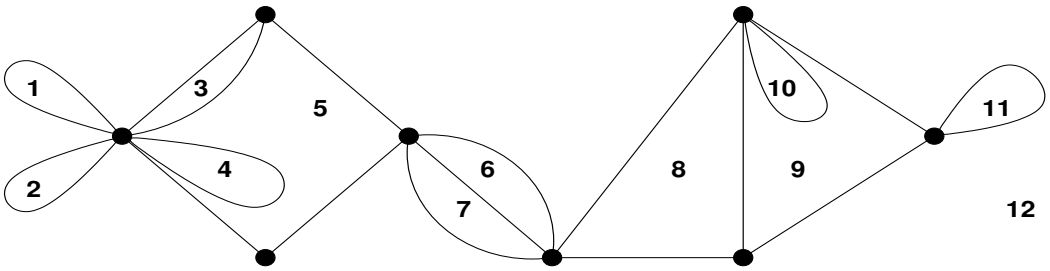
[8] Diestel requires implicitly that the two endpoints of each curve must be different, and explicitly that different curves have different pairs of endpoints. In other words, he wants his graphs to be simple. I don't think we need those restrictions.

[9] This seemingly obvious fact is surprisingly hard to prove in general (although it is not so bad for, say, piecewise-C^∞ curves).

[10] Two what? For any space X, you can define a relation on its points as follows: $x \sim y$ if x, y are the endpoints of some curve $\phi : [0,1] \to X$. This is an equivalence relation (check this!), and its equivalence classes are called the path-connected components of X.

Definition 6.5. Let $\Gamma = (V, E)$ be a plane graph. The path-connected components of $\mathbb{R}^2 \backslash \Gamma$ are called the **faces** of Γ. (Technically this is an abuse of notation — we should write $\mathbb{R}^2 \backslash \bigcup_{e \in E}$ rather than $\mathbb{R}^2 \backslash \Gamma$ — but the latter is much simpler and its meaning is clear.) We write F or $F(\Gamma)$ for the set of faces, and f or $f(\Gamma)$ for the number of faces.

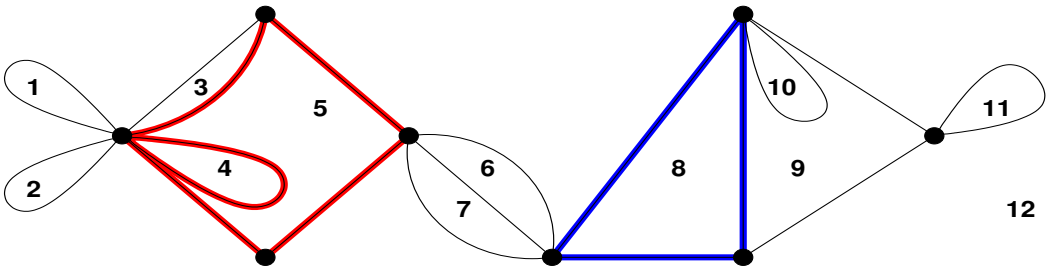
For instance, the following plane drawing has 12 faces. Faces $1, \ldots, 11$ are bounded, while face 12 is the **unbounded face**.



As we will see soon, the number of faces does not depend on the particular drawing; it is actually a combinatorial invariant.

Moreover, there is nothing special about the unbounded face as opposed to the bounded ones. Here is why. When you add a point at infinity to $\mathbb{R}^2$, you get a topological sphere. (This may be easier to visualize in reverse. Start with a sphere and poke a hole in it by deleting one point. Then you can flatten out the punctured sphere into a plane, without any more cutting, tearing or pasting.) So we can regard any plane graph as living on a sphere, where all the faces are bounded. If we now pick any face f, poke a hole in its interior, and flatten things out, we obtain a drawing of the same graph in which f is now the unbounded face.

Each face f has a **boundary** ∂f (Diestel uses the term "frontier"), which can be represented as a closed walk in G. The **length** of f is $\ell(f) = |\partial f|$. The following diagram shows ∂f_5 (red) and ∂f_8 (blue); notice that $\ell(f_5) = 5$ and $\ell(f_8) = 3$.

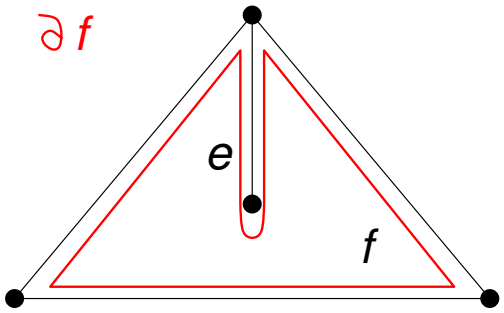


Each edge lies in the boundaries of exactly two faces (a consequence of the Jordan Curve Theorem), so we immediately obtain a useful analogue of the Handshaking Theorem.

Proposition 6.6. *For any plane graph Γ, we have*

$$\sum_{f \in F(\Gamma)} \ell(f) = 2e(\Gamma).$$

Remark 6.7. Bridges have to be treated specially, since the same face lies on both sides of a bridge (in fact, this property characterizes bridges). Accordingly, if f is a face lying on both sides of a bridge e, we regard ∂f as containing two copies of e. For example, the bounded face f shown below has length $\ell(f) = 5$.



Note that this convention is consistent with Prop. 6.6. (Recall that for the original version of handshaking, we decided that a loop incident to a vertex should contribute 2 to its degree. This is the dual statement.)

Theorem 6.8. *Let Γ be a planar graph. Then*

$$f(\Gamma) = e(\Gamma) - n(\Gamma) + c(\Gamma) + 1.$$

In particular, the number of faces is a combinatorial invariant.

Proof. First, if Γ is acyclic, then $f = 1$ and the desired equation reduces to $e - n + c = 0$, which is equivalent to Proposition 1.24.

Let H be a maximal acyclic subgraph of Γ (so that $|H| = n - c$), and let $e \notin H$. Then e is not a bridge; moreover, recall that there is a unique cycle C_e in $H \cup \{e\}$ (formed by e together with the unique path in H between the endpoints of e; if e is a loop then that cycle is just $C_e = \{e\}$).

If we delete e, then the two faces bordering e merge into a single face, so the total number of faces will drop by one. If we keep deleting edges, we will eventually get an acyclic graph, so

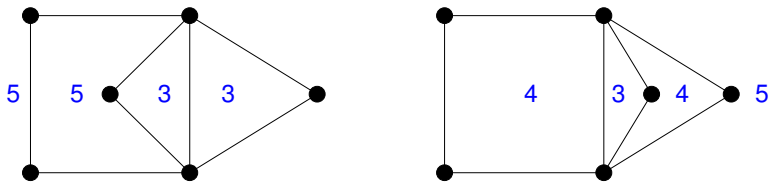
$$f(\Gamma) - |E(\Gamma) \backslash H| = 1$$

and since $|H| = n - c$, so $f - (e - (n - c)) = 1$, or $f = e - n + c + 1$ as desired. (Essentially, this argument is induction on $e - n + c$.)

Alternately, we could contract the edges in H one by one. Each contraction leaves the number of regions unchanged. We eventually get a graph with $E(\Gamma) \backslash H$ edges, all of which are loops, hence $e - |H| + 1$ regions. Since $|H| = n - c$, we get the same result. □

Corollary 6.9 (Euler's formula)**.** *For every connected plane graph (hence every connected planar graph), we have $f = e - n + 2$. More generally, $f \geqslant e - n + 2$ for all planar graphs, with equality if and only if G is connected.*

Remark 6.10. The two plane graphs shown below are combinatorially isomorphic, and in both cases the sum of the face lengths is 16 (twice the number of edges), but the faces themselves have different lengths. So while f is a combinatorial invariant, the multiset of face lengths is not.



6.2. **Applications of Euler's formula.** Combining Euler's formula with the handshaking and length-sum formulas gives very useful upper bounds for the number of edges in a planar graph, and lets us immediately show that certain graphs cannot be planar (in particular solving the houses-and-utilities brainteaser).

Recall that the **girth** of a graph G is the length of the smallest cycle in it (or ∞ if G is acyclic). For example, a graph is simple if and only if its girth is at least 3.

Theorem 6.11. *Let G be a simple, non-acyclic planar graph with girth $g \geqslant 3$. Then*

$$e \leqslant \left(\frac{g}{g-2}\right)(n-2). \tag{24}$$

Proof. Let Γ be a drawing of G. Each face of Γ has length $\geqslant g$, so the length-sum formula (Prop. 6.6) gives

$$2e = \sum_{s \in F(\Gamma)} \ell(s) \geqslant gf \geqslant g(e-n+2)$$

where the second inequality comes from Corollary 6.9. Now solving for e gives the desired inequality. □

Corollary 6.12. *(1) If G is simple and planar, then $e \leqslant 3n-6$.*
(2) If G is simple, bipartite, and planar, then $e \leqslant 2n-4$.
(3) K_5 and $K_{3,3}$ are nonplanar.

Proof. (1) Substitute $g = 3$ into (24).
(2) Substitute $g = 4$ into (24).
(3) K_5 has $(n, e) = (5, 10)$, hence cannot be planar by (1). $K_{3,3}$ is bipartite and has $(n, e) = (6, 9)$, hence cannot be planar by (2).

□

By Proposition 6.4, no graph that contains K_5 or $K_{3,3}$ as a subgraph can be planar. The converse of this statement is false: the Petersen graph has no K_5- or $K_{3,3}$-subgraph, but it has $(n, e, g) = (10, 15, 5)$, which does not satisfy (24), so it is not planar. (More generally, there exist nonplanar graphs of every girth $g < \infty$.)

On the other hand, these bounds can be used to characterize graphs that are *maximally planar*, i.e., adding any single edge produces a non-planar graph. For example, $K_5 - e$ is clearly of this form.

Proposition 6.13. *A graph G on $n \geqslant 3$ vertices is maximally planar iff it is planar and $e = 3n-6$.*

Proof. Clearly, maximally planar implies connected. In addition, G is maximally planar if and only if every face has length 3 (since adding an edge to a plane drawing must join two vertices in a common face of length $\geqslant 4$). Now,

$$\begin{aligned} e = 3n-6 = 3(n-2) &\iff e = 3(e-f) && \text{(by Euler's formula)} \\ &\iff 2e = 3f \\ &\iff \ell(s) = 3 \quad \forall s \in F && \text{(by the length-sum formula).} \end{aligned}$$

□

Shortly, we will show that K_5 and $K_{3,3}$ are, in a precise sense, the only minimal obstructions to planarity. The first step is to define precisely what "minimal obstruction" means.

6.3. **Minors and topological minors.** We have already worked with contraction of edges. For the time being, we are going to redefine G/e by removing any loops or parallel edges[11] created by the contraction, so that the operation keeps us in the world of simple graphs.

[11] I.e., remove all but one element of each parallel class of edges.

Definition 6.14. Let $V_1, \cdots, V_k$ be a partition of $V(G)$ into nonempty, pairwise-disjoint subsets, each of which induces a connected subgraph. The corresponding **contraction** is the graph $X = G/\mathcal{V}$ with vertices $V(X) = [k]$, and an edge from i to j whenever $[V_i, V_j] \subset E(G)$ is nonempty. (This is the same as successively contracting the edges of a maximal forest in $G_{V_1} + \cdots + G_{V_k}$.)

This version of contraction always produces a simple graph X. Note that contracting a single edge $e = xy$ corresponds to contracting with respect to the partition in which $\{x, y\}$ is the only non-singleton block. For example, consider this picture, taken from Diestel, 2nd ed., p.19.

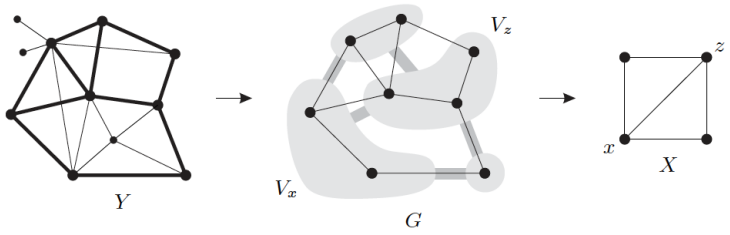
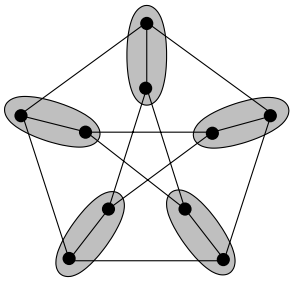


Fig. 1.7.2. $Y \supseteq G = MX$, so X is a minor of Y

Here G is the graph in the middle, and the gray blobs are the blocks V_i, which are called the **branch sets**. If we don't want to specify the partition $\mathcal{V}$, we can just say that "G is an MX" or "$G = MX$". Strictly speaking, it would be most correct to regard MX as the set of all graphs from which X can be obtained by contraction, and write $G \in MX$, but the notation $G = MX$ seems common. (Mnemonically, you might think of the symbol M as standing for "magnification," as suggested by Andrei Elliott.)

Here's another example, which shows that the Petersen graph is an MK_5.



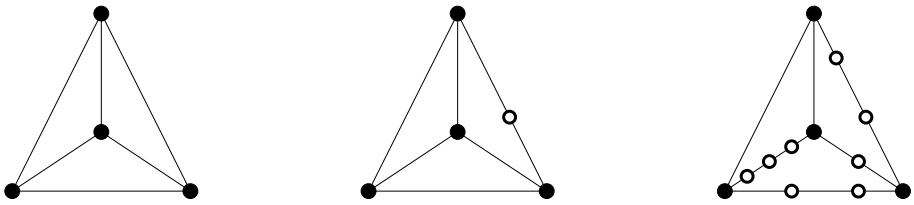
Definition 6.15. A graph X is a **minor** of a graph Y if Y has a subgraph that is an MX. Equivalently, X can be obtained from Y by some sequence of vertex deletions, edge deletions, and edge contractions. (Note that we can perform these operations in any order — we don't have to save all the contractions for the end.)

Proposition 6.16. *If G is planar, then every minor of G is planar. Equivalently, if X is not planar then no MX is planar.*

Deletion of edges and vertices clearly preserves planarity. The harder case is contraction, which boils down to a topological statement: if B is a closed, connected, simply connected, bounded region in $\mathbb{R}^2$ (such as a simple curve), then the space obtained by formally squashing B down to a single point is also topologically equivalent to $\mathbb{R}^2$ (the technical word for this equivalence is *homeomorphism*). Again, I omit the details but they might be fun for the analysis-minded to work out an explicit homeomorphism, at least for the case that B is a line segment.

We next describe a more restricted notion of the minor relation. **Subdividing** an edge e with endpoints x, y means inventing a new vertex z and replacing e with two new edges xz, yz. A **subdivision** of a graph X is

any graph G that can be obtained by subdividing edges in sequence. In this case, every vertex v of X is also a vertex of G, with $d_G(v) = d_X(v)$. These are called the **branch vertices** of the subdivision. The other vertices all have degree 2, and are called **subdividing vertices** (the hollow vertices in the figure below).

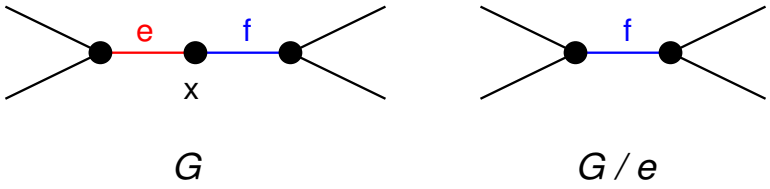


If G is a subdivision of X, then we say that "G is a TX" or "$G = TX$." So all three graphs above are TK_4's. Mnemonically, the symbol T stands for "topological subdivision." (In fact subdivision is a homeomorphism.) Subdividing an edge or a set of edges does not affect whether or not a graph is planar (this should be easy to see). In particular, every TK_5 and every $TK_{3,3}$ is nonplanar, and any graph that contains such a subdivision is nonplanar.

Definition 6.17. A graph X is a **topological minor** of a graph Y if Y has a subgraph that is a TX.

As a simple example, if $n \leqslant m$, then C_n is a minor of C_m (because contracting any $m - n$ edges in C_m produces C_n) as well as a topological minor (because subdividing any sequence of $m-n$ edges in C_n produces C_m). The graph K_5 is a minor of the Petersen graph, but not a topological minor, because there is no way to subdivide edges of K_5 to produce a subgraph of the Petersen graph.

Every subdivision can be undone by a contraction, so every TX is also an MX. The converse is not true in general — for example, contracting an edge in K_n produces K_{n-1}, but subdividing an edge in K_{n-1} does not produce K_n. On the other hand, if $x \in V(G)$ has degree 2, then contracting one of the edges incident to x is a reversible process — subdivide the other edge incident to x.



Proposition 6.18. *Let G, H be graphs.*

(1) If H is a topological minor of G, then H is a minor of G.
(2) If H is a minor of G and $\Delta(H) \leqslant 3$, then H is a topological minor of G.

Proof. (1) follows from the statement that every TX is an MX.

For (2), suppose that H is a minor of G, so that it is obtained from a subgraph of G by a sequence of edge contractions $A \to A/e$, where $e = xy$. Contracting an edge incident to a vertex of degree 1 is the same thing as deleting that vertex, so we can assume that every time we contract an edge $e = xy$, its endpoints each have degree at least 2. Also, we can assume that x, y have no common neighbor z, because in that case $A/xy = (A - xz)/xy$ (since contracting xy causes xz and yz to become parallel). Therefore, $d_{A/e}(xy) = d_A(x) + d_A(y) - 2$ (since its neighbors are $(N(x)\backslash\{y\}) \cup (N(y)\backslash\{x\})$), and since $\Delta(H) \leqslant 3$, at least one of $d(x), d(y)$ must be exactly 2. By the preceding discussion, all these contractions are reversible by subdivision, so H is a topological minor of G. □

Remark 6.19. Here is another fact about topological minors and trees. Consider taking a tree Y and repeatedly contracting every edge incident to a vertex of degree 2. These are all reversible, so $Y = TX$ for every tree produced in this way. Each contraction deletes a single 2 from the degree sequence. So ultimately, Y is a TX for some tree X with the same number of leaves as Y, and no vertices of degree 2. For each fixed number ℓ of leaves, there are only finitely many possibilities — for instance, you can convince yourself using handshaking that the number of non-leaf vertices of X is at most $\ell - 2$.

6.4. **Kuratowski's Theorem.** The main theorem about planar graphs is the following:

Theorem 6.20 (Kuratowski's Theorem, 1930)**.** *A graph is planar if and only if it does not contain K_5 or $K_{3,3}$ as a topological minor.*

A closely related result is the following:

Theorem 6.21 (Wagner's Theorem, 1937)**.** *A graph is planar if and only if it does not contain K_5 or $K_{3,3}$ as a minor.*

The $\implies$ directions of both theorems easily follow from what has come before. The $\impliedby$ direction is the hard part. Since every topological minor is a minor, Kuratowski's Theorem is a stronger result on its face than Wagner's Theorem (convince yourself that the implication goes the right way). The first step is to show that actually the two theorems are equivalent.

Lemma 6.22 (Diestel, Lemma 4.4.2)**.** *A graph G contains K_5 or $K_{3,3}$ as a minor if and only if it contains one of them as a topological minor.*

Proof. The $\impliedby$ direction follows from Prop. 6.18 (1). For the $\implies$ direction, if G has a $K_{3,3}$ minor then it is a topological minor by Proposition 6.18 (2), so we are reduced to proving the following statement:

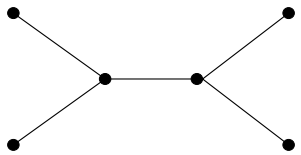
> If G has a K_5 minor then it has either a K_5 topological minor or a $K_{3,3}$ minor.

Let K be a minimal subgraph of G that is an MK_5, and let $V_1, \ldots, V_5$ be the branch sets of K. Minimality has a number of consequences:

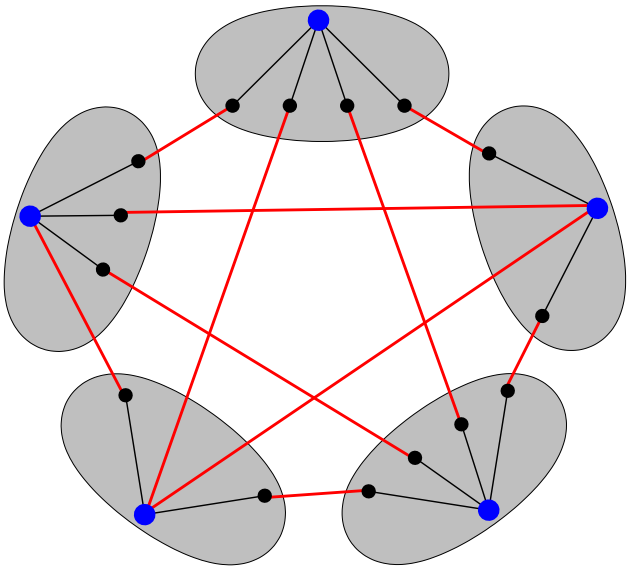
(1) Each subgraph $G_i := K|_{V_i}$ must be a tree — it is connected by the definition of contraction, and it has to be acyclic, otherwise we could remove a non-bridge and obtain a smaller MK_5.
(2) The only possible leaves of G_i are the vertices with neighbors in other branch sets (since again, any other leaf could be removed)
(3) K has exactly one edge e_{ij} between each two branch sets V_i, V_j.

Thus, for each i, the tree $T_i := G_i + \{e_{ij} : j \neq i\}$ has exactly four leaves, one in each V_j for $j \neq i$.

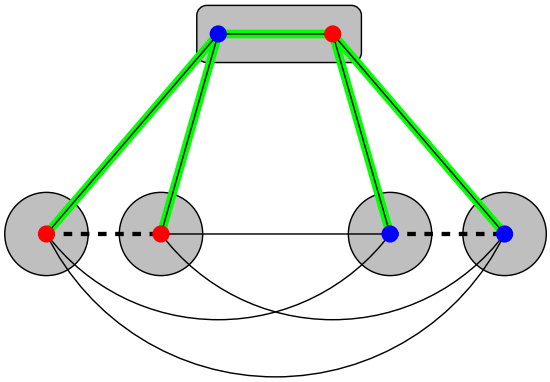
By Remark 6.19, T_i is a TY for some tree Y with four leaves and no degree-2 vertices. There are only two possibilities for Y, namely $K_{1,4}$ and the tree H shown below.



If T_i is a $TK_{1,4}$ for every i, then K is a TK_5 — contract each T_i down to a $K_{1,4}$ and we get a TK_5: specifically, we get a K_5 in which each edge has been subdivided at most twice. In this figure, the degree-4 vertices of the $K_{1,4}$'s are colored blue, and each T_i consists of a blue vertex and its four neighbors. (In particular, two T_i's can overlap.)



Otherwise, if $T_i = TH$ for some i, then contracting V_i onto the two non-leaves of H and contracting all other vertices to a single point gives a $K_{3,3}$, as shown below. So in this case K contains an $MK_{3,3}$. (The coloring indicates the partite sets of $K_{3,3}$, and the edges of H are highlighted. The dashed edges are part of K but not part of the $M_K 3, 3$ it contains.)



□

Of course, we have not proved Kuratowski's Theorem at this point — we have just shown that it is equivalent to Wagner's Theorem. For short, let us call either a TK_5 or a $TK_{3,3}$ a **TKK**.

The strategy at this point is going to be to as follows:

(1) Show that Wagner's Theorem holds for 3-connected graphs, i.e., that every 3-connected graph with no K_5 or $K_{3,3}$ minor is planar. (Actually we'll show something stronger, due to Tutte: every such a graph has a drawing in which every face is bounded by a convex polygon.) This is Lemma 6.23 and Prop. 6.24.
(2) Show that every minimal counterexample to Wagner's theorem must in fact be 3-connected, and therefore by part (1) cannot exist.

Lemma 6.23 (Diestel, Lemma 3.2.1)**.** *Suppose that G is 3-connected and $|G| > 4$. Then G has some edge e such that G/e is 3-connected.*

Proof. Suppose not. Then for every $xy \in E(G)$, the graph G/xy has a separator S of size 2. This separator must consist of the contracted vertex v_{xy} and one other vertex, say z. (Otherwise, it would be a separator in G, but $\kappa(G) \geqslant 3$.) It follows that $\{x, y, z\}$ is a separator in G. This separator is clearly minimal, so each of x, y, z has a neighbor in every component C of $G - \{x, y, z\}$ (because deleting the other two vertices of T leaves G connected).

Choose x, y, z, C so that C is as small as possible, and let $v \in N_C(z)$. Note for later use that this implies that

$$N_G(v) \subseteq C \cup \{x, y, z\}. \tag{25}$$

By the same logic as before, G/zv is not 3-connected, so there is a vertex w such that $\{z, v, w\}$ is a separator, and again each of z, v, w has a neighbor in every component of $G - \{z, v, w\}$.

Let D be a component of $G - \{z, v, w\}$ that contains neither x nor y. (Such a component must exist because x and y are adjacent.) Then $N_G(v) \cap D \subseteq C$ by (25). Also, D is a component of $G - \{x, y, z, v, w\}$, hence contained in some component of $G - \{x, y, z\}$, which must be C. On the other hand, $v \in C \backslash D$, so $D \subsetneq C$, and this contradicts the choice of C. □

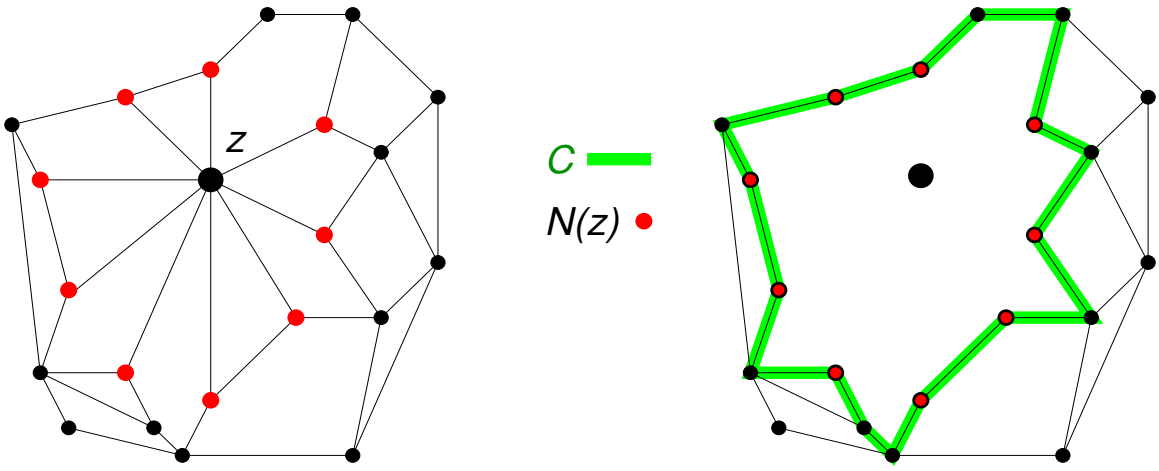
Proposition 6.24. *Every 3-connected graph G with no K_5 or $K_{3,3}$ minor is planar.*

Proof. We will actually show something stronger (due to Tutte): every such graph has a **good embedding** — a drawing in which every face is a convex polygon and no three vertices are collinear.

We induct on $n = n(G)$, which must be at least 4. If $n = 4$, then $G = K_4$, which has a good embedding.

Now suppose that $n > 4$. By Lemma 6.23, G has an edge, say $e = xy$, such that $H = G/e$ is 3-connected. Every minor of H is a minor of G, so H has no K_5 or $K_{3,3}$ minor. By induction, H has a good embedding. Let z be the fused vertex, and let $X = N_G(x) \backslash \{y\}$ and $Y = N_G(y) \backslash \{x\}$, so that $N_H(z) = X \cup Y$.

The plane graph $H - z$ has a face containing the point z (namely, the face formed by merging all faces of H whose boundary contains z). Let C be the boundary of this face; then $N_H(z) \subseteq C$.

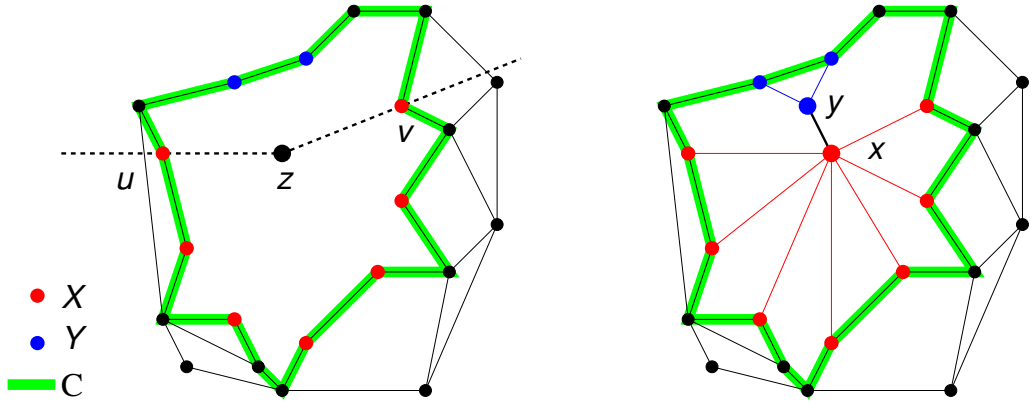


(By the way, if I have accidentally made three vertices collinear in any of these figures, I don't want to hear about it.)

Let $u_1, \ldots, u_\ell$ be the vertices of C, listed in cyclic order. If that list looks like

$$\underbrace{u = u_1}_{\in X}, \quad \underbrace{u_2, \ldots, u_{k-1}}_{\in X \setminus Y}, \quad \underbrace{v = u_k}_{\in X}, \quad \underbrace{u_{k+1}, \ldots, u_\ell}_{\in Y \setminus X}$$

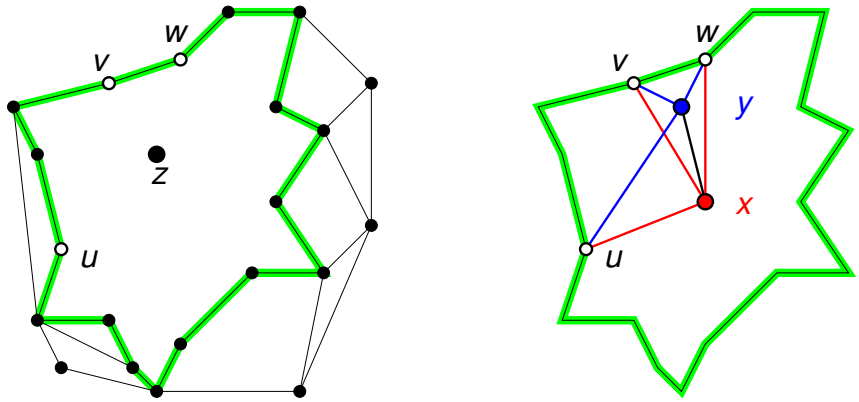
with $1 < k < \ell$, then we can construct a good embedding of G by putting x where z is, and pushing y a little bit into the wedge formed by u_1 and u_k (bounded by the dotted lines in the example).



What if C does not have this form? There are two possibilities.

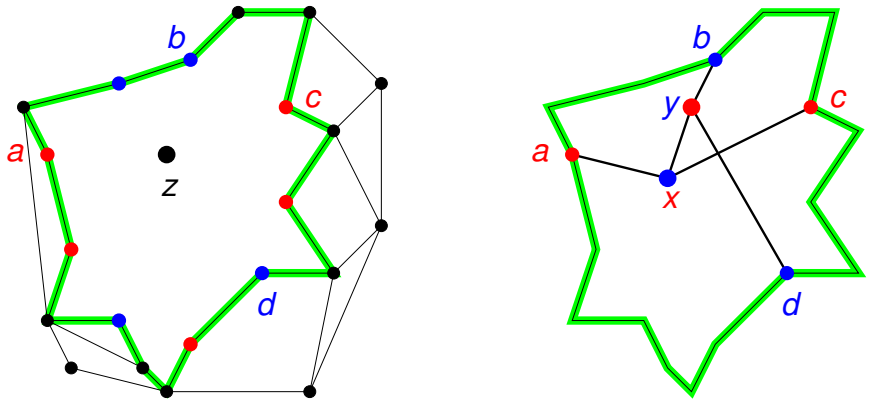
Case 1: There are three vertices $u, v, w \in N(x) \cap N(y)$.

That is, G contains edges xu, xv, xw; yu, yv, yw; and xy. Meanwhile, C can be partitioned into three paths between each two of u, v, w. All this together forms a TK_5 with branch vertices u, v, w, x, y, a contradiction.



Case 2: a, b, c, d are vertices of C, listed in cyclic order, with $a, c \in N(x)$ and $b, d \in N(y)$.

Then G contains edges ax, cx, yb, yd, and yx, and C can be partitioned into ab-, cb-, cd-, and ad-paths. All this together forms a $TK_{3,3}$.



Again this is a contradiction. So Cases 1 and 2 are both impossible, completing the proof. □

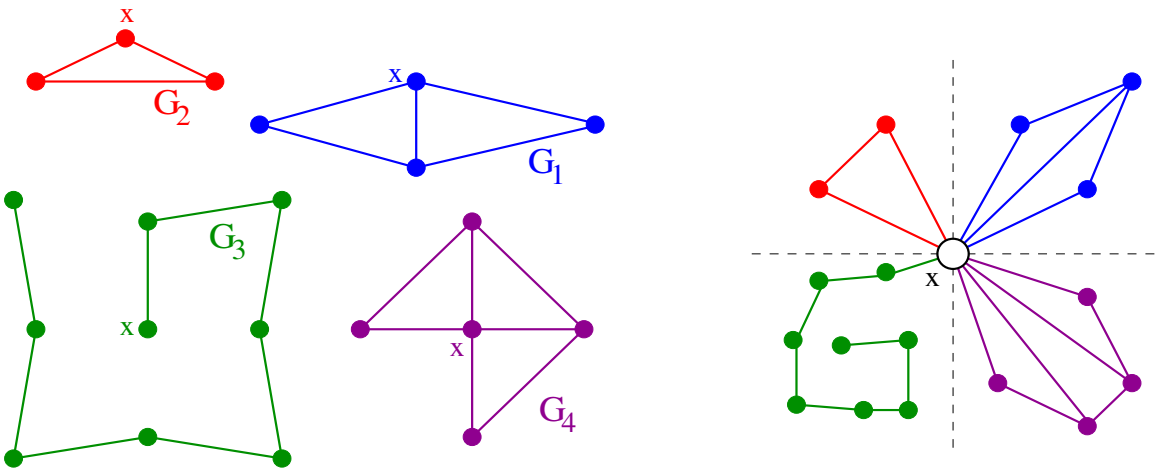
Now we still reduce the general case to the 3-connected case. For this I will follow West's presentation (pp.247–248).

Definition 6.25. Let $S \subset V(G)$. An ***S*-lobe** is a subgraph of the form $G[V(H) \cup S]$, where H is a component of $G - S$.

Lemma 6.26. *Let G be a minimal nonplanar graph. Then G is 2-connected.*

Proof. Certainly G is connected (otherwise it would have some nonplanar component, which would contradict minimality). Suppose that it has a cut-vertex x, with lobes $G_1, \ldots, G_k$ (remember that a lobe is an induced subgraph of the form $G[V(H) + x]$, where H is a component of $G - x$). If every lobe is planar, then we can draw G_i with x on the outer face (say at the origin), squeeze it so that it fits in a "pie slice" of angle $< 2\pi/k$, and attach all the lobes together to get a plane drawing of G. □

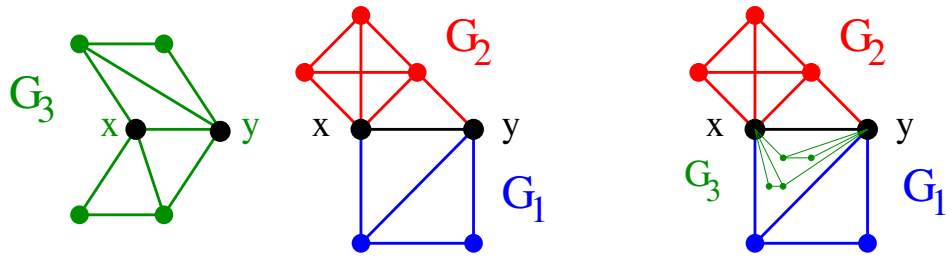
For example:



Lemma 6.27. *Suppose that G is nonplanar and that $S = \{x, y\}$ is a vertex cut of G. Then adding the edge xy to some S-lobe produces a nonplanar graph.*

Proof. Let $G_1, \ldots, G_k$ be the S-lobes of G, and let $H_i = G_i + xy$. Suppose that every H_i is planar: then we can show by induction on k that G is planar. If $k = 1$ this is trivial (because $H_i \supset G$). Otherwise, given a

plane drawing of $H' = H_1 \cup \cdots \cup H_{i-1}$, we can take a drawing of H_i with xy on the outer face and shrink it to fit in one of the faces of H' bounded by xy. For example:

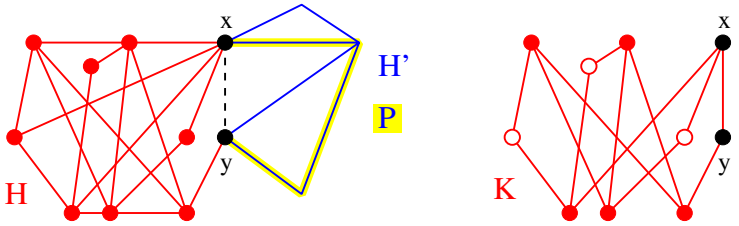


Therefore, if G is nonplanar, one of the H_i must be nonplanar. □

Proposition 6.28. *Suppose that there is a nonplanar graph with no TKK as a subgraph. Let G be such a graph with the fewest edges among all such graphs. Then G is 3-connected.*

Proof. Deleting an edge certainly cannot create a TKK, so the minimality assumption implies that $G - e$ is planar for every $e \in E(G)$. Therefore, by Lemma 6.2.5, G is 2-connected.

Now suppose that G is not 3-connected. Let $S = \{x, y\}$ be a separator. Then $e(H + xy) < e(G)$ for every S-lobe H. (This is clear if $xy \in E(G)$; otherwise, every other S-lobe has at least two edges.) By Lemma 6.2.6, there is some S-lobe H such that $H + xy$ is nonplanar, and the minimality assumption implies that $H + xy$ has a TKK, say K. Let P be a path in G from x to y which avoids every other vertex of H (for instance, we can choose P to be a path through a different S-lobe H'). Then $H + P$ is a TKK in G, which is a contradiction. □



Fact: Whether or not a graph is planar can be determined efficiently—in fact, in $O(n)$ time. Algorithms for planarity testing would be a good subject for an end-of-semester project.

6.5. **The Five-Color Theorem.**

Theorem 6.29 (The Five-Color Theorem)**.** *If G is planar, then $\chi(G) \leqslant 5$.*

It is easy to prove that every planar graph is 6-colorable. The bound $n \leqslant 3n - 6$ for planar graphs says in particular that $\delta(G) \leqslant 5$ (since the sum of all degrees is at most $6n - 12 \leqslant 6n$). Choose an ordering in which the last vertex v_n has degree $\leqslant 5$, delete it, and now do the same thing for $G - v_n$ (which is also planar), and so on recursively. Greedy coloring using this order will not use more than 6 colors. (Equivalently, apply (20).) Meanwhile, it is true, but very difficult to prove, that every planar graph is 4-colorable — this is arguably the most famous theorem in graph theory and was open for over a century, with a long, colorful[12]

[12]Sorry, couldn't help it.

story of failed attempts to prove it. The five-color theorem, on the other hand, is not trivial, but it is doable by us mortals.

First, we prove a key lemma.

Lemma 6.30. *Let G be a plane graph and $x \in V(G)$ with $\deg(x) \geqslant 4$. Let a, b, c, d be neighbors of G, listed in cyclic order around G (there can be other neighbors between them). Then every a, c-path in $G - x$ crosses every b, d-path.*

Proof. Let C be the cycle bounding the union of the regions containing x. Thus C is the disjoint union of an a, b-path, a b, c-path, a c, d-path, and a d, a-path. If P, P' are a, c- and b, d-paths that do not cross, then neither of these paths cross C (since they can be taken to stay outside its interior), so $P + P' + C$ forms a TK_4. Inverting the picture so that x is on the unbounded face, we obtain an outerplanar drawing of a TK_4, which cannot exist by a homework problem. (Note: This statement can also be proved from scratch, as Diestel does, by showing that $P \cup \{x\}$ is a cycle that separates b from d.) □

Proof of the Five-Color Theorem. Every plane graph G has $e(G) \leqslant 3n(G) - 6$, so the sum of vertex degrees is at most $6n - 12$, and it follows that G has at least one vertex of degree < 5.

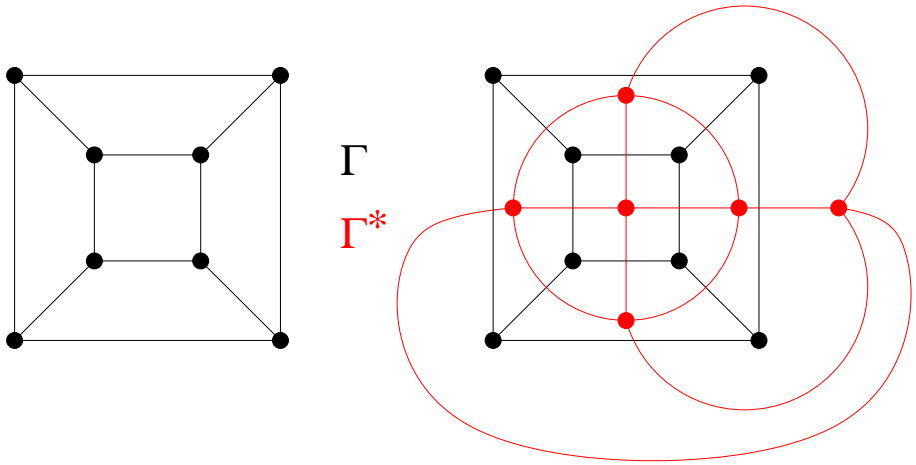
Let G be a minimal plane graph that is not 5-colorable. Then $\delta(G) \geqslant 5$, for if $\deg(x) = 4$ then $G - x$ is 5-colorable by minimality, and we can construct a 5-coloring of G by assigning x some color not assigned to any of its neighbors. Combined with the previous observation, we see that G has some vertex of degree exactly 5. Call this vertex x and let its neighbors be a, b, c, d, e in cyclic order in the plane drawing, colored red, orange, yellow, green, blue respectively.

I claim that there does not exist both (i) an a, c-path consisting entirely of red and yellow vertices, and (ii) a b, d-path consisting entirely of orange and green vertices. If both existed, then they would have to cross, but since G is planar that crossing point would have to be a vertex, and the two paths have no vertices in common.

WLOG, suppose that a path of form (i) does not exist. Another way of saying this is that a and c are in different components of the induced subgraph of $G - x$ on the red and yellow vertices. If we toggle red and yellow in the component containing a, then we will still have a proper 5-coloring, but a will now be yellow and c will remain hello. So we can then color x red and obtain a proper 5-coloring of G. □

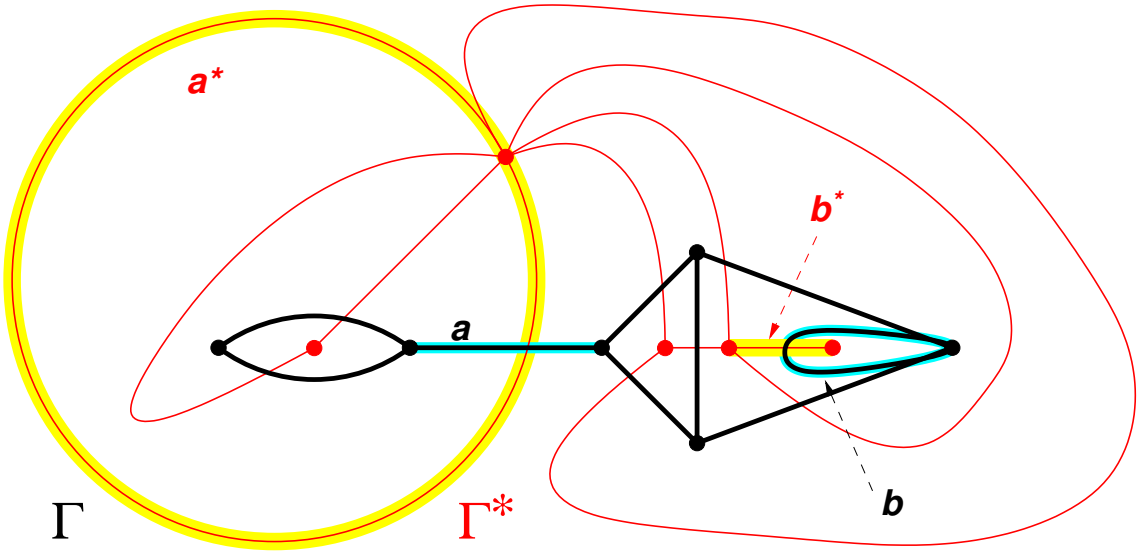
6.6. **Planar duality.**

Definition 6.31. Let Γ be a plane graph. The **plane dual** of Γ is the plane graph Γ^* with vertices $V(\Gamma^*) = F(\Gamma)$, with an edge of Γ^* drawn across each edge of Γ. That is, for each edge e that separates two (possibly equal) faces $f, f' \in F(\Gamma)$, there is a dual edge $e^* = ff' \in E(\Gamma^*)$.

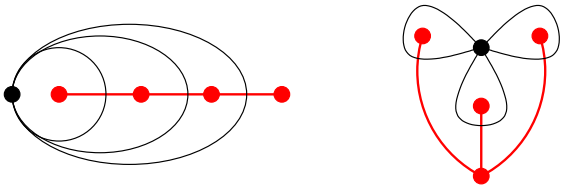


By definition, $n(G^*) = f(G)$ and $e(G^*) = e(G)$. It follows from Euler's formula that $f(G^*) = n(G)$. This can also be seen directly: every face of G^* encloses exactly one vertex of G.

Notice that if edge a is a loop, then a^* is a bridge, and if b is a loop, then b^* is a bridge:



Warning: The plane dual is not a combinatorial invariant without further assumptions on G. For example, if G has one vertex and three loops, then the dual of a "flower" drawing of G is $K_{3,1}$, while the dual of a "Hawaiian earring" drawing of G is P_4.



On the other hand, if G satisfies certain additional hypotheses (for example, if G is 3-connected), then all of its embeddings have isomorphic duals. In this case it is legitimate to speak of **the** planar dual of the graph G.

In addition, if Γ is a connected plane graph then $(\Gamma^*)^* \cong \Gamma$. In the following figure, the double duals are shown in blue.

Theorem 6.32. *An edge set $A \subseteq E(\Gamma)$ is a cycle if and only if the dual edge set A^* is a bond in $E(\Gamma^*)$.*

Proof. Suppose that A contains a cycle C. Then $A^* \supseteq [S, \bar{S}]$, where S (resp. $\bar{S}$) is the set of faces inside C (resp. outside C). Meanwhile, the minimal sets A containing cycles are just cycles themselves. So A is a cycle if and only if it is a minimal disconnecting set—that is, a bond.

On the other hand, if A is acyclic, then it encloses no region, so $\Gamma^* - A^*$ is connected and A^* contains no cut. □

Corollary 6.33. *e is a bridge in Γ if and only if e^* is a loop in Γ^*.*

(This is one justification for the use of the word "coloop" as a synonym for "bridge".)

Theorem 6.34. *Let Γ be a plane graph. TFAE:*

(1) *Γ is bipartite.*
(2) *Every cycle in Γ has even length.*
(3) *Every face of Γ has even length.*
(4) *Every vertex of Γ^* has even degree.*
(5) *Γ^* is Eulerian.*

Proof. The implications (1) $\iff$ (2) $\implies$ (3) $\iff$ (4) $\iff$ (5) are clear from the definitions, or from things we have already proved. So we need only show (3) $\implies$ (2).

Suppose that (2) fails. Let C be an odd cycle. Let $q_1, \ldots, q_r$ be the faces lying inside C. Let

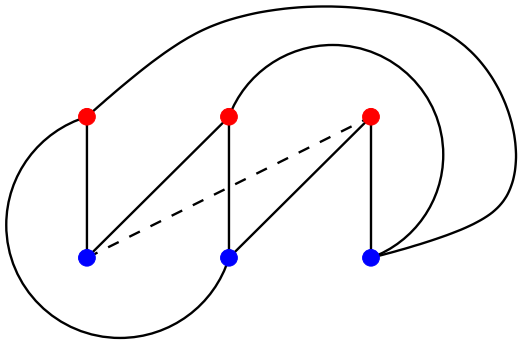
$$E_1 = E(C),$$
$$E_2 = \{e \in E(\Gamma) \mid e \text{ lies inside } C\},$$
$$E_3 = \{e \in E(\Gamma) \mid e \text{ lies outside } C\}.$$

Then $E(\Gamma) = E_1 \cup E_2 \cup E_3$. Each edge of E_1 (resp. E_2, E_3) borders one (resp. two, zero) of the faces q_i. Therefore,

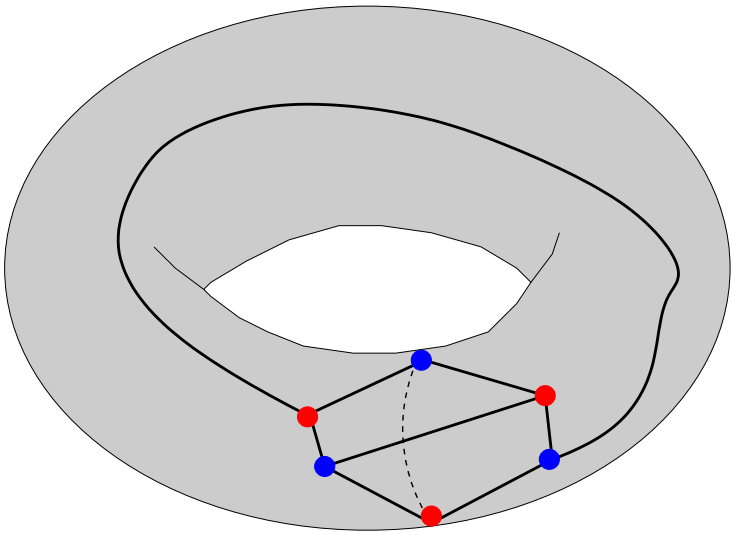
$$\sum_{i=1}^{n} \ell(q_i) = |E_1| + 2|E_2|$$

is odd. Therefore $\ell(q_i)$ is odd for at least one face q_i. □

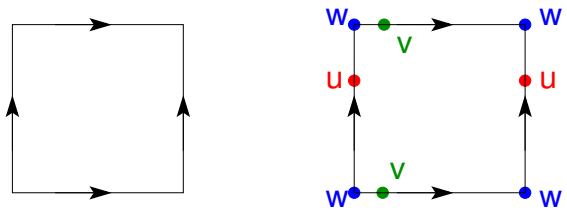
6.7. **The genus of a graph.** We can't embed $K_{3,3}$ in the plane (or, equivalently, the sphere), so what if we build a bridge to avoid crossings?



Essentially, we are adding a "handle" to the sphere to get a torus. This enables us to embed more graphs than can be embedded in the plane: for example, $K_{3,3}$.

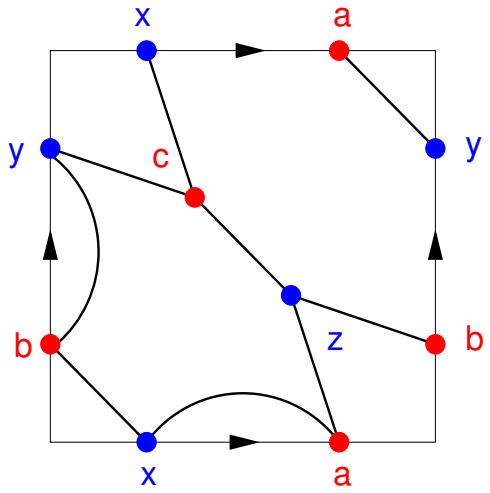


This picture is awkward, but there's a nicer way to draw pictures on the torus. To construct a torus, we could take a sheet of paper, glue the top and bottom edges together, and glue the left and right sides together. The topological diagram for this gluing is as shown on the left:

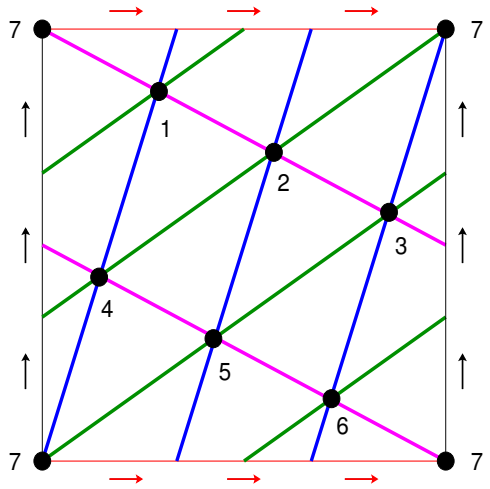


The arrows indicate the orientations of the edges when they are glued together. (Reversing one of the arrows would produce a Klein bottle instead of a torus.) Thus the two red dots represent the same point v. So do the two green dots (v) and the four blue ones (w).

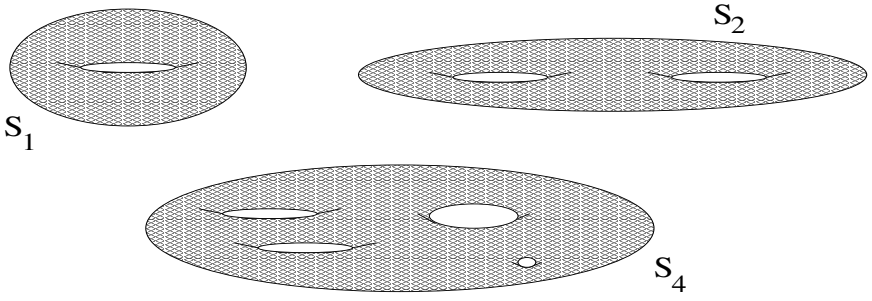
That means that we can represent toroidal embeddings of graphs by drawing them on a square. For example, here's an embedding of $K_{3,3}$:



And here's a toroidal K_7:



Definition 6.35. The ***g*-holed torus** S_1 is the surface obtained by adding g handles to the sphere.



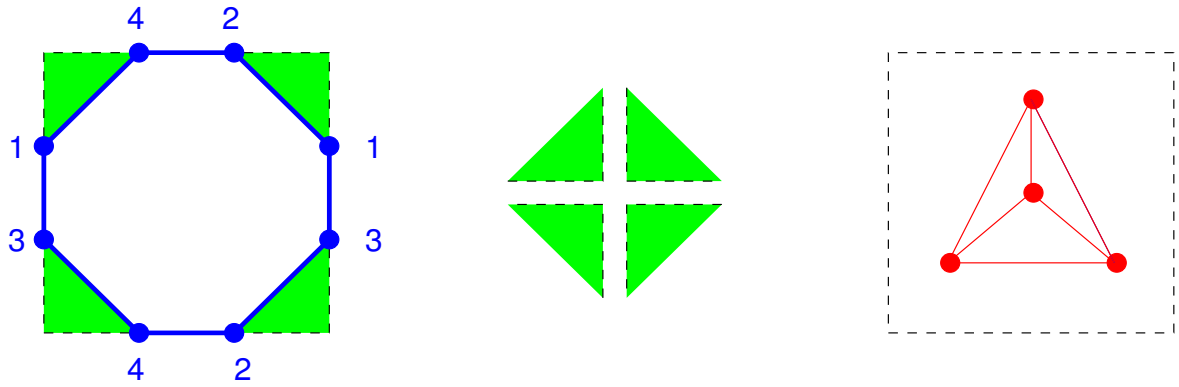
Definition 6.36. The **genus** of a graph G is

$$\gamma(G) = \min\{g \mid G \text{ embeds on } S_g\}.$$

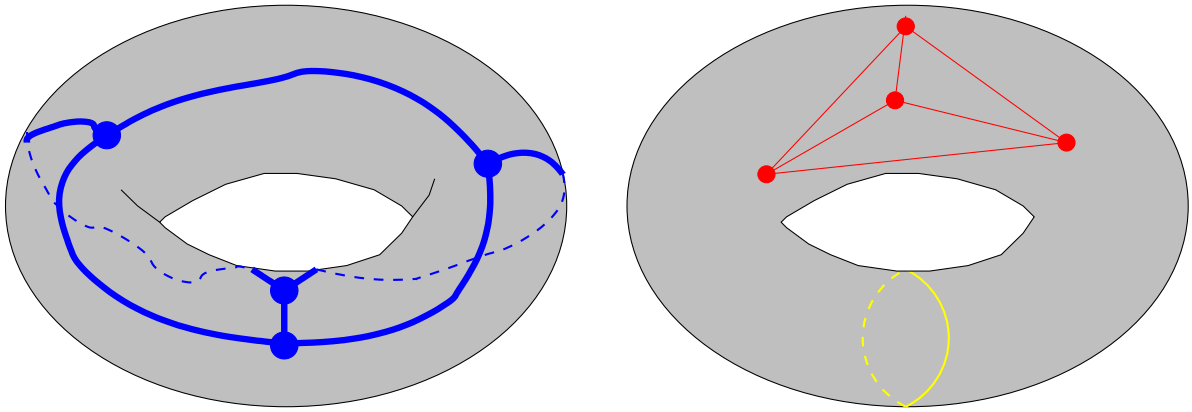
Notice that $\gamma(G) \leqslant \nu(G)$ (the minimum number of crossings in any plane drawing of G), because we can eliminate a crossing by adding a bridge. However, the inequality can be quite sharp. In fact, not only does K_5 have genus 1, but so does K_7 (whose crossing number is 9)! (See p. 267 for the figure.)

There's an analogue of Euler's formula for graphs embedded on surfaces of higher genus. However, we have to be careful to talk only about embeddings in which the faces are **2-cells**, i.e., homeomorphic (topologically equivalent) to $\mathbb{R}^2$.

For example, consider the following two embeddings of K_4 on S_1.



The embedding on the left is a 2-cell embedding, because gluing the sides of the dashed square together joins the four green shaded triangles into a diamond, as shown. The embedding on the right is *not* a 2-cell embedding. The "outside" face (shown in yellow) is not **simply connected**; i.e., there are closed curves that cannot be contracted to a point, such as the circle formed by the top face of the dashed square (shown in yellow below).



Proposition 6.37 (Euler's formula for tori)**.** *If G has a 2-cell embedding on a surface of genus g, then*

$$n - e + f \;=\; 2 - 2g.$$

Notice that g doesn't have to *equal* the genus of G for there to be such an embedding and for this to work, merely be *at least* the genus.

For instance, this says that a toroidal embedding of K_4 ought to have $f = 2-2g-n+e = 2-2-4+6 = 2$ faces.

Corollary 6.38. *Any graph G of genus g satisfies $e \leqslant 3(n-2+2g)$.*

6.8. **Heawood's Theorem.** If we can find a number c such that every genus-γ graph has a vertex of degree $< c$, then it will follow that every genus-γ graph has $\chi(G) \leqslant c$. It suffices to find c such that every genus-γ graph has *average* degree c.

Theorem 6.39 (Heawood 1890)**.** *If G is embeddable on a surface of genus $\gamma > 0$, then*

$$\chi(G) \leqslant \lfloor c \rfloor$$

where

$$c = \frac{7+\sqrt{1+48\gamma}}{2}.$$

Proof. It suffices to prove that G has a vertex of degree $\leqslant c-1$; the desired result will then follow from induction on n. There is no problem if $n \leqslant c$, so we assume that $n > c$.

The quantity c is a positive root of the polynomial $c^2 - 7c + (12-12g) = 0$, which is equivalent to $c - 1 = 6 - \frac{12-12g}{c}$. Therefore,

$$\begin{aligned} \frac{2e}{n} &\leqslant \frac{6(n-2+2g)}{n} && \text{(by Corollary 6.38)} \\ &= 6 - \frac{12-12g}{n} \\ &< 6 - \frac{12-12g}{c} \\ &= c-1. \end{aligned}$$

So the average degree of G is less than c, which means that at least one vertex must have degree $\leqslant c-1$ as desired. □

Note that $c = 4$ for $\gamma = 0$. On the other hand, the argument isn't valid unless $\gamma > 0$. It evaluates to 7 for $\gamma = 1$; that is, every toroidal graph is 7-colorable.

In fact, Heawood's bound is sharp for $\gamma > 0$; this is quite nontrivial but can be proven more easily than the Four-Color Theorem. So, strangely, the problem of determining the maximum chromatic number of genus-γ graphs is most difficult when $\gamma = 0$.

Wagner's Theorem can be generalized for the torus, in the following sense:

Theorem 6.40. *For every $n \geqslant 0$, there is some* **finite** *set Φ_n of (isomorphism types of) graphs such that*

$$\gamma(G) \leqslant n \quad \Longleftrightarrow \quad G \text{ has no minor in } \Phi_n.$$

For $n = 0$, we have $\Phi_0 = \{K_5, K_{3,3}\}$. Lots and lots of elements of Φ_1 are known, but not the complete list. How do we even know that the set is finite? By the following amazing result, due to Robertson and Seymour:

Graph Minor Theorem: In every infinite list of graphs, some graph is a minor of another.

It follows from the GMT that every list of minimal obstructions must be finite, since no two elements of it are comparable.

7. The Tutte Polynomial

We have seen lots of invariants of graphs that satisfy a deletion-contraction recurrence, including:

$$\begin{aligned} \tau(G) &= \tau(G-e) + \tau(G/e) && \text{(number of spanning trees)} \\ a(G) &= a(G-e) + a(G/e) && \text{(number of acyclic orientations)} \\ p_G(k) &= p_{G-e}(k) + p_{G/e}(k) && \text{(chromatic polynomial)} \end{aligned}$$

We'd like to put all these invariants under one roof.

7.1. Definitions and examples.

Definition 7.1. Let G be a graph. The **Tutte polynomial** $T(G) = T_G(x,y) = T(G;x,y)$ is defined by the recurrence

(26)

$$T(G;x,y) = \begin{cases} 1 & \text{if } E(G) = \varnothing, & \text{(a)} \\ x \cdot T(G/e) & \text{if } e \text{ is a bridge,} & \text{(b)} \\ y \cdot T(G-e) & \text{if } e \text{ is a loop,} & \text{(c)} \\ T(G-e) + T(G/e) & \text{if } e \text{ is an ordinary edge (neither bridge nor loop).} & \text{(d)} \end{cases}$$

(Note that we are no longer working with simple graphs — we do want to keep track of loops and parallel edges.)

There is a big problem with this definition: it is not clear that the polynomial $T(G;x,y)$ is independent of the choice of edge e. We'll do so soon, but first, some examples.

Example 7.2. Let $F = F_r$ be a forest with r edges (any forest). Then $\boldsymbol{T(F_r) = x^r}$. For $r = 0$, this is case (a) of the recurrence (26). Otherwise, every edge of F is a bridge, and F/e is a forest with $r-1$ edges. By induction, $T(F/e) = x^{r-1}$, so $T(F) = x^r$ by case (b).

Example 7.3. Let L_r be the graph with one vertex and r loops. A similar argument, using case (c) of (26), gives $\boldsymbol{T(L_r) = y^r}$. More generally, any graph with r edges, all of which are loops, has Tutte polynomial y^r.

Example 7.4. How about K_3? Let e be any edge (it doesn't matter which one). Then case (d) of (26) gives

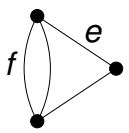
$$\begin{aligned} T(K_3) &= T(K_3 - e) + T(K_3/e) \\ &= T(P_3) + T(C_2). \end{aligned}$$

Now P_3 is a tree with two edges, so its Tutte polynomial is x^2. For the digon C_2, let f be either edge. Then

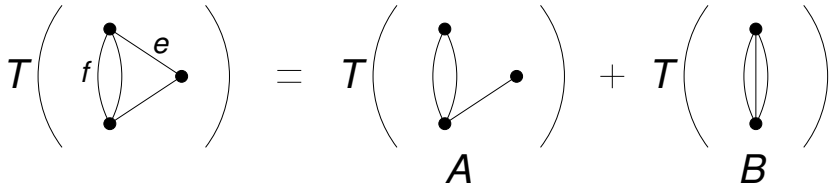
$$\begin{aligned} T(C_2) &= T(C_2 - f) + T(C_2/f) \\ &= T(K_2) + T(L_1) \\ &= x + y \end{aligned}$$

so $\boldsymbol{T(K_3) = x^2 + x + y}$.

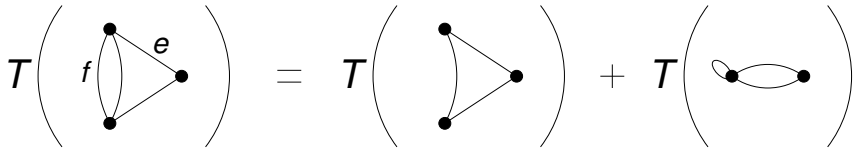
Example 7.5. We will compute the Tutte polynomial of the following graph G:



Applying the recurrence with the edge e gives



while applying the recurrence with f gives



There seems to be no particular reason why these two calculations ought to yield the same answer. But they do. First,

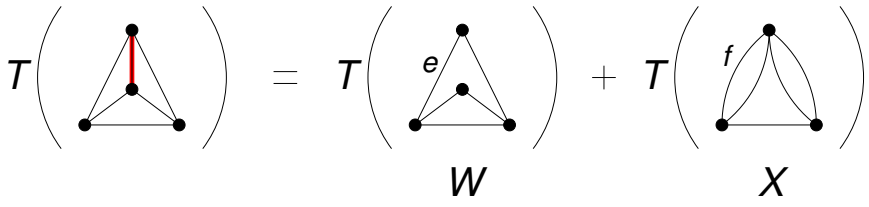
$$T(A) = x \cdot T(C_2) \;=\; x(x+y),$$
$$T(B) = T(C_2) + T(L_2) \;=\; (x+y) + (y^2),$$

so using e gives $T(G) = x^2 + xy + y^2 + x + y$. On the other hand,

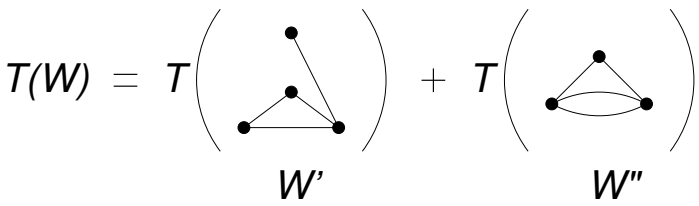
$$T(Y) = T(K_3) \;=\; x^2 + x + y,$$
$$T(Z) = y \cdot T(C_2) \;=\; y(x+y),$$

so we get the same answer for $T(G)$ using f.

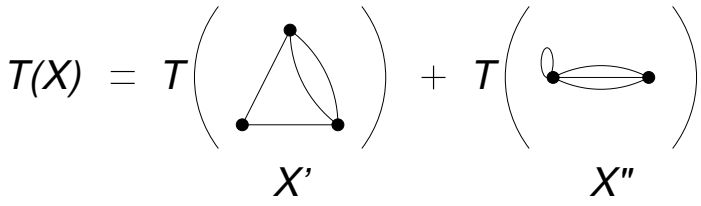
Example 7.6. One more calculation: $T(K_4)$. Of course, it doesn't matter which edge we start with; in the following diagram, we use the one indicated in red.



So we now have to calculate the Tutte polynomials of W and X. Apply the recurrence (26) with the labeled edge $e \in E(W)$:



Now $T(W') = x \cdot T(K_3)$, and W'' is the graph G of Example 4. On the other hand, applying the recurrence to $f \in E(X)$ gives



So $X' \cong G$, and $T(X'') = y \cdot T(B)$, with B as in Example 4. Putting it all together:

$$\begin{aligned} T(K_4) &= T(W) + T(X) \\ &= T(W') + T(W'') + T(X') + T(X'') \\ &= x(x^2 + x + y) + 2(x^2 + xy + y^2 + x + y) + y(x + y + y^2) \\ &= (x^3 + y^3) + (3x^2 + 4xy + 3y^2) + (2x + 2y). \end{aligned}$$

There are some interesting things about this polynomial. First, it is symmetric in x and y—that is, if we swap x and y, the polynomial is unchanged. Second, if we plug in various values of x and y, the numbers that come out are rather suggestive:

$$\begin{aligned} &T(K_4;\, 0,0) = 0, \\ &T(K_4;\, 0,1) = 6, && T(K_4;\, 1,1) = 16, \\ &T(K_4;\, 0,2) = 24, && T(K_4;\, 1,2) = 38, && T(K_4;\, 2,2) = 64. \end{aligned}$$

It's not entirely clear what all this means, but $24 = 4!$ is the number of acyclic orientations of K_4 and 16 is the number of spanning trees, among other things. (These are not coincidences!)

In order to prove that the Tutte polynomial is well-defined by the recurrence (26), we will give an explicit formula for $T(G;\, x, y)$.

Definition 7.7. The **rank** $r_G(A) = r(A)$ of an edge set $A \subseteq E(G)$ is defined to be

$$r(A) \;=\; \max\{|X| :\; X \subseteq A \text{ is acyclic}\}. \tag{27}$$

The rank of G itself, denoted $r(G)$, is just the rank of its edge set. This is the size of a maximal spanning forest in G, namely $n(G) - c(G)$. In addition, we define the **corank** and **nullity** of A by

$$\begin{aligned} \operatorname{cork}(A) = \operatorname{cork}_G(A) &= r(G) - r(A), \\ \operatorname{null}(A) = \operatorname{null}_G(A) &= |A| - r(A). \end{aligned}$$

Equivalently, the corank is the minimum number of edges that need to be added to A in order to increase its rank to $r(G)$ (that is, to make it contain a maximal forest), and the nullity is the minimum number of edges that need to be deleted from A in order to make it acyclic.

We can now state the fundamental theorem about the Tutte polynomial.

Theorem 7.8. *For every graph $G = (V, E)$, we have*

$$T(G;\, x, y) \;=\; \sum_{A \subseteq E} (x-1)^{\operatorname{cork}(A)} (y-1)^{\operatorname{null}(A)}.$$

For obvious reasons, this formula is referred to as the *corank-nullity* form of the Tutte polynomial.

Before giving the proof, we verify this formula for the graph $G = K_3$. Here $r(G) = 2$. Note that $r(A) = \min(|A|, 2)$ for all $A \subseteq E$. Theorem 7.8 lets us calculate the Tutte polynomial as follows:

$\lvert A\rvert$	# of sets $A \subseteq E$	$r(A)$	Contribution to $T(G; x, y)$					
0	1	0	$(x-1)^2(y-1)^0$	$=$	x^2	$-2x$		$+1$
1	3	1	$3(x-1)^1(y-1)^0$	$=$		$3x$		-3
2	3	2	$3(x-1)^0(y-1)^0$	$=$				3
3	1	2	$(x-1)^0(y-1)^1$	$=$			$+y$	-1
					x^2	$+x$	$+y$	

which agrees with the calculation in Example 3.

Proof of Theorem 7.8. Define

$$\tilde{T}(G) \;=\; \sum_{A \subseteq E} (x-1)^{r(G)-r(A)} (y-1)^{|A|-r(A)}.$$

We need to show that $\tilde{T}(G) = T(G)$ for all graphs $G = (V, E)$. We will do this by induction on $|E|$.

In the base case $E = \varnothing$, we have $r(\varnothing) = |\varnothing| = 0$, so the theorem says that $\tilde{T}(G) = (x-1)^0(y-1)^0 = 1 = T(G)$ (by case (a) of (26)).

For the inductive step, suppose that $T(G') = \tilde{T}(G')$ for all graphs G' with $e(G') < e(G)$. In particular, if we choose $e \in E$ arbitrarily, then Theorem 1 holds for $G - e$ and (provided that e is not a loop) for G/e.

First, suppose that e is a loop. Let $G' = G - e$. Then $r(G') = r(G)$, and for $A \subseteq E$,

$$r_G(A) = r_G(A - e) = r_{G'}(A - e).$$

Therefore,

$$\begin{aligned}
\tilde{T}(G) \;&=\; \sum_{A \subseteq E} (x-1)^{\mathrm{cork}(A)} (y-1)^{\mathrm{null}(A)} \\
&=\; \sum_{\substack{A \subseteq E \\ e \notin A}} (x-1)^{\mathrm{cork}(A)} (y-1)^{\mathrm{null}(A)} \;+\; \sum_{\substack{A \subseteq E \\ e \in A}} (x-1)^{\mathrm{cork}(A)} (y-1)^{\mathrm{null}(A)} \\
&=\; \sum_{A \subseteq E \setminus \{e\}} (x-1)^{\mathrm{cork}_{G'}(A)} (y-1)^{\mathrm{null}_{G'}(A)} \;+\; \sum_{A' \subseteq E \setminus \{e\}} (x-1)^{\mathrm{cork}_{G'}(A)} (y-1)^{\mathrm{null}_{G'}(A)+1}
\end{aligned}$$

(think of A' as $A \setminus \{e\}$, and observe that $\mathrm{null}(A') = \mathrm{null}(A) - 1$)

$$\begin{aligned}
&=\; \sum_{A \subseteq E \setminus \{e\}} (x-1)^{\mathrm{cork}_{G'}(A)} (y-1)^{\mathrm{null}_{G'}(A)} \;+\; (y-1) \sum_{A' \subseteq E \setminus \{e\}} (x-1)^{\mathrm{cork}_{G'}(A)} (y-1)^{\mathrm{null}_{G'}(A)} \\
&=\; (1 + y - 1) \sum_{A \subseteq E \setminus \{e\}} (x-1)^{\mathrm{cork}_{G'}(A)} (y-1)^{\mathrm{null}_{G'}(A)} \\
&=\; y \cdot \tilde{T}(G') \;=\; y \cdot T(G - e)
\end{aligned}$$

(by the definition of $\tilde{T}$, and then by induction).

Second, if e is a bridge, then $\tilde{T}(G) \;=\; x \cdot T(G/e)$. The proof is similar (now it's the corank that changes instead of the nullity) and is left as an exercise.

Third, suppose that e is neither a bridge nor a loop. Then $r(G) = r(G - e) = r(G/e) + 1$, and for $A \subseteq E$,

$$r_G(A) = \begin{cases} r_{G-e}(A) & \text{if } e \notin A, \\ r_{G/e}(A - e) + 1 & \text{if } e \in A. \end{cases}$$

So we can calculate $T(G)$ as

$$\sum_{\substack{A\subseteq E\\ e\notin A}} (x-1)^{r(G)-r(A)}(y-1)^{|A|-r(A)} + \sum_{\substack{A\subseteq E\\ e\in A}} (x-1)^{r(G)-r(A)}(y-1)^{|A|-r(A)}$$

$$= \sum_{A:\ e\notin A} (x-1)^{r(G-e)-r_{G-e}(A)}(y-1)^{|A|-r_{G-e}(A)} + \sum_{A:\ e\in A} (x-1)^{r(G/e)-r_{G/e}(A-e)}(y-1)^{|A-e|-r_{G/e}(A-e)}$$

$$= \sum_{A\subseteq E-\{e\}} (x-1)^{r(G-e)-r_{G-e}(A)}(y-1)^{|A|-r_{G-e}(A)} + \sum_{A\subseteq E-\{e\}} (x-1)^{r(G/e)-r_{G/e}(A-e)}(y-1)^{|A-e|-r_{G/e}(A-e)}$$

$$= \tilde{T}(G-e) + \tilde{T}(G/e)$$

which agrees with case (d) of (26). □

Corollary 7.9. *The choice of edge does not matter when computing the Tutte polynomial by deletion-contraction. Moreover, $T(G;\, x, y)$ has nonnegative integer coefficients (for short, $T(G;\, x, y) \in \mathbb{N}[x, y]$).*

Now that we have it, what do we do with it? Well, all the other deletion-contraction invariants that we know about can be obtained as *specializations* of the Tutte polynomial—that is, by setting the parameters x and y to other values.

Theorem 7.10. $\tau(G) = T(G;\, 1, 1)$.

For example, let G be as in Example 4. Here G is connected, so "maximal spanning forest" is the same thing as "spanning tree". We calculated $T(G;\, x, y) = x^2 + xy + y^2 + x + y$, so $T(G;\, 1, 1) = 5$. Indeed, $\tau(G) = 5$.

First proof of Theorem 7.10. Plugging $x = y = 1$ into (26), we find that

$$T(G;\, 1, 1) \;=\; \begin{cases} 1 & \text{if } E(G) = \varnothing, \\ T(G/e;\, 1, 1) & \text{if } e \text{ is a bridge}, \\ T(G-e;\, 1, 1) & \text{if } e \text{ is a loop}, \\ T(G-e;\, 1, 1) + T(G/e;\, 1, 1) & \text{otherwise.} \end{cases}$$

This is precisely the recurrence defining $\tau(G)$. □

Second proof of Theorem 7.10. Plug $x = y = 1$ into the explicit formula of Theorem 7.8. It looks as though this will kill every term, but actually some of the terms—namely, those with both $r(G) - r(A) = 0$ and $|A| - r(A) = 0$—are identically 1, and will be unaffected by the substitution $x = y = 1$. Every other term will indeed be killed. Therefore

$$T(G;\, 1, 1) \;=\; \#\{A \subseteq E \mid r(G) = r(A),\ |A| = r(A)\}. \tag{28}$$

But $r(A) = r(G)$ if and only if A is connected (as a spanning subgraph of G) and $r(A) = |A|$ if and only A is acyclic. Therefore, the edge sets A counted in (28) are precisely the spanning forests of G. □

Some more specializations that come from the corank-nullity expansion:

$T(G;\, 2, 1)$ = number of acyclic subsets of $E(G)$,

$T(G;\, 1, 2)$ = number of maximum-rank subsets of $E(G)$ (= spanning subgraphs with $c = c(G)$),

$T(G;\, 2, 2) = 2^{e(G)}$.

Another very useful fact is the following.

Proposition 7.11. *(1) If $G_1, \ldots, G_n$ are the connected components of G, then $T(G) = T(G_1) \cdots T(G_n)$.*
(2) If G_1 and G_2 are graphs on disjoint vertex sets and G is obtained by identifying a vertex of G_1 with a vertex of G_2, then $T(G) = T(G_1)T(G_2)$.

These results are immediate from the corank/nullity generating function. Another way to phrase them is that $T(G)$ is the product of the Tutte polynomials on its blocks. (A block of G is either a maximal 2-connected subgraph or a cut-edge; if every graph has a unique decomposition into blocks and each edge belongs to exactly one block.)

7.2. **The chromatic polynomial from the Tutte polynomial.** Every invariant defined by a linear deletion-contraction recurrence can be obtained from the Tutte polynomial. In particular, we've seen that the number $s(G)$ of strong orientations, and the number $a(G)$ of acyclic orientations satisfy such recurrences. In fact

$$s(G) = T(G;\, 0, 2) \quad \text{and} \quad a(G) = T(G;\, 2, 0).$$

The proofs of these facts are similar to the first proof of Theorem 2. In fact, there's a universal recipe for obtaining any deletion-contraction invariant as a Tutte polynomial evaluation (see Bollobas). Sometimes the evaluation requires a correction factor for the number of vertices or components of G (data that the Tutte polynomial does not keep track of).

A basic example is the chromatic polynomial $p_G(k)$. In order to obtain $p_G(k)$ from $T(G; x, y)$, we could proceed purely algebraically, by figuring out what to plug into the Tutte recurrence to recover how the chromatic recurrence (Theorem 5.8). However, I think the following approach is more enlightening.

For the purpose of the following discussion, the term "k-coloring" is going to mean *any* function $f : V(G) \to [k]$. A coloring f is *proper* at an edge $xy \in G$ if $f(x) \neq f(y)$, and it is *proper* on G if it is proper at every edge.

Proposition 7.12 (Whitney's formula)**.** *Let G be a graph with n vertices and rank function r. Then*

$$p_G(k) = \sum_{A \subseteq E(G)} (-1)^{|A|} k^{n - r(A)}.$$

Proof. This is an application of inclusion-exclusion. If we want to count proper k-colorings, we start by counting all colorings of G. Then, for each edge, subtract the number of colorings that are improper on that edge. We have undercounted the colorings that are improper on two edges; this accounts for the summands with $|A| = 2$. But then we have overcounted all the colorings that are improper on three edges... and so on. This means that

$$p_G(k) = \sum_{A \subseteq E(G)} (-1)^{|A|} \#\{\text{colorings that are improper on (at least) } A\}.$$

Being improper on A is equivalent to being constant on each component of the subgraph of G induced by A. The number of components is $n - r(A)$, so there are $k^{n-r(A)}$ such colorings. □

Whitney's formula has the same general form as the corank/nullity form of the Tutte polynomial — it is a generating function of edge sets of G, with summands that involve rank and cardinality. To connect them, we do a little algebra:

$$\begin{aligned}
p_G(k) &= k^n \sum_{A\subseteq E(G)} (-1)^{|A|} k^{-r(A)} \\
&= k^n \sum_{A\subseteq E(G)} (-1)^{|A|-r(A)} (-k)^{-r(A)} \\
&= (-k)^{-r(E)} k^n \sum_{A\subseteq E(G)} (-1)^{|A|-r(A)} (-k)^{r(E)-r(A)} \\
&= (-1)^{-r(E)} k^{n-r(E)} \sum_{A\subseteq E(G)} (-1)^{|A|-r(A)} (-k)^{r(E)-r(A)} \\
&= (-1)^{n(G)-c(G)} k^{c(G)} \sum_{A\subseteq E(G)} (y-1)^{|A|-r(A)} (x-1)^{r(E)-r(A)} \Big|_{x-1=-k,\ y-1=-1} \\
&= (-1)^{n(G)-c(G)} k^{c(G)} T(G;\, 1-k, 0).
\end{aligned}$$

I'll state that as a theorem for ease of reference:

Theorem 7.13. *Let G be a graph with n vertices and c components. Then*

$$p_G(k) = (-1)^{n-c} k^c T(G;\, 1-k, 0).$$

Corollary 7.14. *The number of acyclic orientations of any graph G is $a(G) = T(G;\, 2, 0)$.*

Proof. Recall that $a(G) = |p_G(-1)|$. Plug $k = -1$ into both sides of Theorem **??**. (Taking absolute values is unnecessary in this formula because $T(G;\, 2, 0)$ is always nonnegative.) □

These results should make you wonder what the dual of a coloring is — i.e., what combinatorial object has a generating function equal to $T(G; 0, 1-k)$ (plus appropriate correction factors)?

Definition 7.15. Let $G = (V, E)$ be a graph with oriented edge set D, and let A be a finite abelian group, written additively and with identity element 0. An **nowhere-zero *A*-flow** is a function $\phi : D \to A\backslash\{0\}$ such that the net flow into every vertex equals the net flow out; that is, for every $v \in V(G)$

$$\forall v \in V: \qquad \sum_{e=\overrightarrow{uv}\in D} \phi(e) = \sum_{e=\overrightarrow{vw}\in D} \phi(e).$$

The choice of orientation does not affect the number of nowhere-zero A-flows (reversing an edge e corresponds to replacing $\phi(e)$ with $-\phi(e)$), so this number depends only on G and A. Observe that if A has a cut-edge then there are no nowhere-zero A-flows (do you see why?) Dually, if A has a loop e then $\phi(e)$ can take on any of $|A| - 1$ values with no constraints.

Theorem 7.16. *Let $k = |A|$. Then the number of nowhere-zero A-flows on G is*

$$p^*_G(k) = (-1)^{e-n+c} T(G;\, 0, 1-k).$$

We omit the proof (exercise?) In particular, this quantity is independent of the group structure of A and depends only on its cardinality, which is certainly not obvious from the definition. Accordingly, we can speak of k-flows rather than A-flows.

The function $p^*_G(k)$ is called the **flow polynomial** of G. Theorems 7.13 and 7.16 together say that if G is connected and planar, then

$$p_{G^*}(k) = k p^*_G(k).$$

A famous unsolved problem in graph theory is Tutte's 5-Flow Conjecture, which asserts that every 2-edge-connected graph G has at least one nowhere-zero 5-flow, i.e., $p_G^*(5) > 0$. Jaeger proved the statement for 8-flows in 1976 and Seymour proved it for 6-flows in 1981; this is still the state of the art.

7.3. **Edge activities.** The coefficients of $T_G(x, y)$ are all nonnegative integers. What do they count? In other words, there should be some set $\mathcal{M}$ ("mysteries") and some combinatorially defined functions $i : \mathcal{M} \to \mathbb{N}$ and $e : \mathcal{M} \to \mathbb{N}$ such that

$$T_G(x, y) = \sum_{m \in \mathcal{M}} x^{i(M)} y^{e(M)} = \sum_{j,k \in \mathbb{N}} \#\{M \in \mathcal{M} \mid i(M) = j,\ e(M) = k\}\, x^j y^k.$$

If we plug $x = y = 1$ into this formula, we get $T_G(1, 1) = |\mathcal{M}|$, which suggests that the M's should be the maximal forests in G. What are the functions i and e? Well...

Assume WLOG that G is connected. Order the edges $E = E(G)$ as $e_1,\ e_2,\ \ldots,\ e_s$, and let T be a spanning tree.

Definition 7.17. An edge $e_i \in T$ is **internally active** with respect to T if it is the smallest edge of the cut between the two components of $T - e_i$. Equivalently,

$$e_j \notin T,\ T - e_i + e_j \text{ a tree} \quad \Longrightarrow \quad j \geqslant i.$$

Meanwhile, an edge $e_j \in E - T$ is **externally active** with respect to T if it is the smallest edge of the unique cycle in $T + e_j$. Equivalently,

$$e_i \in T,\ T - e_i + e_j \text{ a tree} \quad \Longrightarrow \quad i \geqslant j.$$

Let $a(T)$ be the number of internally active edges of T (its "internal activity") and let $b(T)$ be the number of externally active edges (its "external activity"). Call the ordered pair $(a(T), b(T))$ the **biactivity** of T.

Theorem 7.18. *For all G and all orderings of $E(G)$, we have*

$$T(G;\, x, y) = \sum_T x^{a(T)} y^{b(T)}.$$

That is, the coefficient t_{ij} of $x^i y^j$ in $T(G;\, x, y)$ is the number of spanning trees of G with biactivity (i, j)

The proof, again, involves verifying the recurrence.

For a particular spanning tree T, the numbers $a(T), b(T)$ of course depend on the choice of ordering. So the collection of trees with given biactivity is not an isomorphism invariant. However, the *number* t_{ij} of such trees does not depend on the ordering.

We will see the Tutte polynomial again shortly, in the context of network reliability. Other applications of the Tutte polynomial outside pure graph theory include invariants in knot theory (Kauffman bracket, Jones polynomial) and statistical mechanics (the Potts model). The Tutte polynomial is actually an invariant not just of graphs, but of *matroids*, where its applications include weight enumerators of linear codes (Greene); counting chambers in a hyperplane arrangement (Zaslavsky); and much more.

8. Probabilistic Methods in Graph Theory

8.1. A very brief taste of Ramsey theory.

Theorem 8.1 (Ramsey's Theorem)**.** *Let p, q be positive integers. Then there is an integer n such that every graph G of order at least n has either a K_p or a $\overline{K_q}$ as a subgraph.*

Traditionally, the result is phrased a little differently: for every p, q, there is some N such that whenever $n \geqslant N$ and the edges of K_n are 2-colored red and blue, the result contains either a red K_p or a blue K_q. (Of course, the graph G referred to in the theorem is just the spanning subgraph with the red edges.)

Proof. We proceed by double induction on the pair (p, q). If $p = 1$, then we can take $n = q$. Likewise if $q = 1$ then we can take $n = p$.

Now, suppose the claim is true for all pairs (p', q') such that $p' \leqslant p$, $q' \leqslant q$, and $(p', q') \neq (p, q)$, and let $R(p', q')$ be the smallest number such that all graphs of order $\geqslant R(p', q')$ have either a K_p or a $\overline{K_q}$. Let $n \geqslant R(p, q-1) + R(p-1, q)$. Color the edges of K_n red and blue. Fix a vertex $x \in G$ and let

$$A = \{y \in V(G) \mid xy \text{ is colored red}\},$$
$$B = \{y \in V(G) \mid xy \text{ is colored blue}\}.$$

Then $|A| + |B| = n - 1$, so either $|A| \geqslant R(p-1, q)$ or $|B| \geqslant R(p, q-1)$ (both inequalities can't be false). WLOG, suppose the former. If A contains a blue K_q then there is nothing to prove, while if $A' \subseteq A$ is a red K_{p-1} then $A' \cup \{x\}$ is a red K_p. □

The theorem shows that $R(p, q)$ (called a **Ramsey number**) is well-defined for all positive integers p, q, and gives the recursive upper bound $R(p, q) \leqslant R(p-1, q) + R(p, q-1)$. This bound, however, is far from sharp, and the exact value of (most) Ramsey numbers is unknown.

Ramsey's theorem admits numerous generalizations. For example, we can use more than two colors:

Theorem 8.2. *Let $p_1, \ldots, p_\ell$ be positive integers. Then, for any sufficiently large n, whenever the edges of K_n are colored with ℓ colors, there is some monochromatic clique of size i and color i.*

The proof is analogous to that of Theorem 8.1. There are also natural generalizations to hypergraphs and even to infinite graphs!

What about trying to find a *lower* bound for Ramsey numbers? Hold that thought.

8.2. The reliability polynomial.

Suppose that we have a connected graph $G = (V, E)$ in which the edges might disappear. (Think of a power grid in which there is a chance that some lines might go down, or a city during a snowstorm which might block access to some roads.) We would like the network to remain connected. How can we measure the reliability of the network, i.e., the probability that it remains connected?

In order to obtain a model we can work with, we will assume that every edge has probability p of staying alive ($0 \leqslant p \leqslant 1$), and that the edges are all independent. (In the language of probability, the edges correspond to a family of IID random variables; IID stands for "independent and identically distributed".) Therefore, the probability that any given $A \subseteq E$ is exactly the set of surviving edges is $p^{|A|}(1-p)^{|E|-|A|}$, and the probability that the graph remains connected is

$$R_G(p) = \sum_{\substack{A \subseteq E \\ A \text{ connected}}} p^{|A|}(1-p)^{|E|-|A|} = (1-p)^{|E|} \sum_A \left(\frac{p}{1-p}\right)^{|A|}.$$

In fact, we can recover this quantity from the Tutte polynomial. Being connected is equivalent to having corank 0, and we can kill off all the positive-corank edge sets by setting $x = 1$ in Theorem 7.8:

$$\begin{aligned} T(G;\,1,y) &= \sum_{A \text{ connected}} (y-1)^{|A|-r(A)} \\ &= (y-1)^{-r(E)} \sum_{A} (y-1)^{|A|} \end{aligned}$$

since $\operatorname{cork} A = 0$ also means $r(A) = r(E)$. Set $y - 1 = p/(1-p)$, i.e., $y = p/(1-p) + 1 = 1/(1-p)$, which gives

$$\begin{aligned} T(G;\,1,1/(1-p)) &= \left(\frac{p}{1-p}\right)^{-r(E)} \sum_{A} \left(\frac{p}{1-p}\right)^{|A|} \\ &= \left(\frac{p}{1-p}\right)^{-r(E)} (1-p)^{-|E|} R_G(p) \end{aligned}$$

or equivalently

$$R_G(p) = \left(\frac{p}{1-p}\right)^{r(E)} (1-p)^{|E|} T(G;\,1,1/(1-p)) = p^{r(G)}(1-p)^{\operatorname{null} G} T(G;\,1,1/(1-p)). \tag{29}$$

In particular, $R_G(p)$ is a polynomial function of p, called the **reliability polynomial**.

This is great for theory, but it is not practical for large graphs because computing the reliability polynomial is exponentially hard. Also, we might want to measure reliability for graphs for which we have messy or incomplete data. Fortunately, we can use the random graph model described above in another way.

8.3. **Random graphs.** Suppose that G is a graph with a billion vertices and a trillion edges. What is the probability that $G\ldots$

- is connected?
- has an isolated vertex?
- is 4-colorable?
- is 2-connected?

Applications include studying large networks (e.g., the Internet, Facebook) in which it is impossible or infeasible to obtain a complete list of vertices, but which we may be able to model locally.

For example, suppose we want to solve the six-degrees-of-separation problem. Construct a graph G in which the vertices are the n people in the world and edges indicate that you have met at least once (however you choose to define "met") . What is the probability that any two people have met? We'll make the assumption that the probabilities are independent for any two pairs of people (not really true, but not totally silly, and probably necessary if you want to be able to calculate anything at all). We'll also assume that the number of acquaintances of any single person is about constant, and doesn't depend on n. (The number of people I meet each day is affected very little, or not at all, by, say birth rates in a country I have never visited.) Therefore,

$$\Pr[xy \in E(G)] \sim \frac{1}{n}.$$

Now we can model this as a graph problem. Let G be a graph on n vertices in which each of the $\binom{n}{2}$ possible edges has probability $\frac{k}{n}$ of occurring, where k is a constant. Can we then calculate

$$\lim_{n\to\infty} \Pr[G \text{ has diameter} \leqslant 6]?$$

Of course, we could replace "diameter $\leqslant 6$" with many other graph properties of interest. We could also ask what happens if we change the assumption about how p behaves as $n \to \infty$. In general, p will tend to 0,

but how fast it does so — like $1/n$? like $1/n^2$? like $1/\ln n$? — can change the answers to these questions dramatically.

Let's get precise.

Definition 8.3. A **discrete probability space** or **probability model** $(S, \Pr)$ is a finite set S together with a map $\Pr : S \to [0,1]$ such that $\sum_{s\in S} \Pr(s) = 1$. A subset of S is called an **event**. The **probability** of an event A is

$$\Pr(A) = \sum_{s\in S} \Pr(s).$$

Two events A, B are *independent* if $\Pr(A \cap B) = \Pr(A)\Pr(B)$.

We are studying the probability model whose underlying set S is the set of simple graphs on n labeled vertices. In particular, $|S| = 2^{\binom{n}{2}}$. By saying that each edge occurs with independent probability p, we are saying that for $G \in S$,

$$\Pr(G) = p^{e(G)}(1-p)^{\binom{n}{2}-e(G)}.$$

This setup is called the **Erdős–Rényi random graph model**, denoted $\mathcal{G}(n,p)$. There are other random graph models that are interesting, but this is the most classical and best-studied one and the one we will concentrate on exclusively. Be warned that it is common to talk about "the graph $\mathcal{G}(n,p)$" as though it were a specific graph, rather than a probability model. For example, "the probability that $\mathcal{G}(n,p)$ is connected" really means "the value $\Pr[A]$, where A is the set of connected graphs on n vertices under the probability model $\mathcal{G}(n,p)$".

Proposition 8.4. *If $p > 0$ is constant, then $\mathcal{G}(n,p)$ is connected almost always.*

The first step is to define what "almost all" means. Specifically, let q_n be the probability that $\mathcal{G}(n,p)$ is connected. Then to say that almost all graphs are connected is to say that

$$\lim_{n\to\infty} q_n = 1.$$

So, let's calculate q_n. The game plan is going to start with combinatorics, then continue with analysis and probability.

Proof. Recall that a graph $G = (V,E)$ is connected if and only if $E \cap [A, \bar{A}] \neq \varnothing$ for every $\varnothing \subsetneq A \subsetneq V$. We can calculate the probability of that happening exactly, since $|[A,\bar{A}]| = |A||\bar{A}|$. Therefore,

$$\begin{aligned} 1 - q_n &= \Pr\left(\bigcup_A G \cap [A,\bar{A}] = \varnothing\right) \\ &\leqslant \sum_A \Pr\left(G \cap [A,\bar{A}] = \varnothing\right) \\ &= \frac{1}{2}\sum_{k=1}^{n-1} \binom{n}{k}(1-p)^{k(n-k)} \end{aligned}$$

(where $k = |A|$, and the $1/2$ factor arises because $[A,\bar{A}] = [\bar{A}, A]$)

$$< \sum_{k=1}^{\lfloor \frac{n-1}{2} \rfloor} n^k (1-p)^{k(n-k)}$$

(since the sum is a palindrome and $\binom{n}{k} < n^k$)

$$< \sum_{k=1}^{\lfloor \frac{n-1}{2} \rfloor} n^k (1-p)^{kn/2} \;<\; \sum_{k=1}^{\infty} (n(1-p)^{n/2})^k \;=\; \frac{x}{1-x},$$

where $x = n(1-p)^{n/2}$. On the other hand, $\lim_{n\to\infty} x = 0$ because exponential decay kills polynomial growth (check it with L'Hôpital's Rule if you wish). It follows that $\lim_{n\to\infty} 1 - q_n = 0$, i.e., $\lim_{n\to\infty} q_n = 1$. □

Notice that we used some inequalities that are not close to sharp, like $\binom{n}{k} < n^k$. That's fine if all we care about is to show that $q_n \to 1$, but if you want more precise information on how fast in converges to 1 then you would want to tighten up those inequalities.

8.4. **Back to Ramsey theory.** Consider the Erdős-Rényi random graph $\mathcal{G}(n, 1/2)$. For each set A of k vertices, let E_A be the event that A is either a clique or a coclique. In particular, $\Pr[E_A] < 1$ if and only if there exists a graph on n vertices with no k-clique or k-coclique — i.e., if $R(k, k) > n$.

This is a simple observation, but a very powerful one, because it means that we can prove combinatorial theorems using probabilistic methods. This was one of Erdős's main contributions to combinatorial theory. Check this out. Let E be the event that $\mathcal{G}(n, 1/2)$ contains either a k-clique or a k-coclique. Then $E = \bigcup_A E_A$ (with the sum over all $A \subseteq [n]$ of size k) and $\Pr[E_A] = 2^{1-\binom{k}{2}}$ for any A. Therefore,

$$\Pr[E] \;=\; \Pr\left[\bigcup_A E_A\right] \;\leqslant\; \sum_A \Pr[E_A] \;=\; \binom{n}{k} 2^{1-\binom{k}{2}}.$$

That is, if $\binom{n}{k} 2^{1-\binom{k}{2}} < 1$ then $R(k, k) > n$. This implies the following result:

Theorem 8.5. *(Erdős 1947)* $R(k, k) > 2^{k/2}$.

Proof. For $k = 3$, we have $R(3, 3) = 6 > 2^{3/2}$, so suppose $k \geqslant 4$. Then $k! > 2^k$, so

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k-1)}{k!} < \frac{n^k}{2^k}.$$

(This is a tighter inequality than the crude $\binom{n}{k} < n^k$ we used earlier, although it is still pretty crude.) Therefore, if $n < 2^{k/2}$ then

$$\binom{n}{k} 2^{1-\binom{k}{2}} \;<\; \frac{n^k}{2^k} 2^{1-\binom{k}{2}} \;<\; 2^{k^2/2-k+1-\binom{k}{2}} = 2^{-k/2+1} < 1$$

and we are done by the previous discussion. □

Using Stirling's approximation

$$\sqrt{2\pi n}(n/e)^n \leqslant n! \leqslant e^{n/12}\sqrt{2\pi n}(n/e)^n$$

to approximate the binomial coefficients results in a tighter bound, which we state without proof:

Proposition 8.6. $R(k, k) \geqslant (e\sqrt{2})^{-1} k 2^{k/2}$ *for all* $k \geqslant 3$.

8.5. **Random variables, expectation and Markov's inequality.**

Definition 8.7. A **[discrete] random variable** is a function $X : S \to \mathbb{R}$, where $(S, \Pr)$ is a finite probability space.

Any graph invariant $(n, e, \alpha, \chi, \dots)$ can be regarded as a random variable on $\mathcal{G}(n, p)$. So can graph properties; remember that a property is the same thing as a set of graphs. The **indicator variable** of a graph property $P \subseteq \mathcal{G}(n, p)$ is

$$X_P(G) = \begin{cases} 1 & \text{if } G \in P, \\ 0 & \text{if } G \notin P. \end{cases}$$

The **expectation** or **expected value** or **mean** X is

$$\mu = \mathbb{E}(X) = \sum_{G \in S} \Pr(G) X(G).$$

The expectation can be regarded as the average value of X over the probability space.

As an example, let X_k be the number of k-cycles can we expect to find in $\mathcal{G}(n,p)$. Let X_C be the indicator variable for a particular k-cycle C, so that $X_k = \sum_C X_C$. Then

$$\mathbb{E}(X_k) = \frac{k!}{2k}\binom{n}{k}p^k = \frac{n(n-1)\cdots(n-k+1)}{2k}p^k. \tag{30}$$

since there are $\binom{n}{k}$ choices for the vertices in the cycle and $k!/2k$ ways to order them cyclically, and each cycle occurs with probability p^k. Note that even though the variables X_C are not pairwise independent, the expectation of their sum is nevertheless the sum of their expectations — the principle of **linearity of expectation**. If Y_k was instead the number of *induced* k-cycles, then similarly

$$\mathbb{E}(Y_k) = \frac{n(n-1)\cdots(n-k+1)}{2k}p^k(1-p)^{\binom{k}{2}-k}.$$

It is typically easier to compute expectations than probabilities, so a common technique is to try to replace probabilities by expressions involving expectations, using inequalities such as the following.

Proposition 8.8 (Markov's Inequality)**.** *If $X : S \to \mathbb{N}_{\geqslant 0}$ is a random variable, then $\Pr(X \geqslant n) \leqslant \mathbb{E}(X)/n$.*

Proof.

$$\mathbb{E}(X) \;=\; \sum_{k\geqslant 0} k\cdot\Pr(X=k) \;\geqslant\; \sum_{k\geqslant n} k\cdot\Pr(X=k) \;\geqslant\; \sum_{k\geqslant n} n\cdot\Pr(X=k) \;=\; n\cdot\Pr(X\geqslant k). \qquad \square$$

The proposition can easily be extended to real-valued discrete random variables (and to random variables on infinite probability spaces, though we won't need that here). A useful consequence is the following: if $\{X_1, X_2, \dots\}$ is a sequence of random variables, then

$$\lim_{n\to\infty}\mathbb{E}(X_n) = 0 \quad\Longrightarrow\quad \lim_{n\to\infty}\Pr(X_n = 0) = 1. \tag{31}$$

8.6. **Graphs with high girth and chromatic number.** Here is a famous application of random graph theory:

Theorem 8.9 (Erdős 1959)**.** *For any $k \in \mathbb{N}$, there exists a graph with girth $g > k$ and chromatic number $\chi > k$.*

In general, graphs with high girth have few edges, while graphs with high chromatic number have lots of edges. Erdős's proof begins with the hope that the probability p in the Erdős-Rényi model can be chosen carefully so that most graphs with m edges have high chromatic number, yet small enough that most graphs with m edges have high girth.

Say that a cycle is **short** if its length is $\leqslant k$ and **long** otherwise, so that a graph has girth $> k$ if and only if it has no short cycles. Also, call a coclique **big** if it has at least $n(G)/k$ vertices. Note that if G has chromatic number $\leqslant k$, then at least one color class in an optimal coloring is big. Therefore, if G has no big cocliques then its chromatic number is $> k$. Thus, we are looking for a graph with no short cycles and no big cocliques.

In order to make p small enough to rule out the possibility of short cycles, it turns out that we have to make $p < 1/n$ (more precisely, we have to require that $pn \to 0$ as $n \to \infty$). Unfortunately, this constraint on p rules out the probability of any cycles at all (exercise) — which means that G will almost surely be acyclic, hence bipartite.

So instead we set $p = n^{\varepsilon-1}$ for some small positive ε. This will make big cocliques unlikely, and while it will produce short cycles in G, if ε is small enough then it will not produce very many of them. We can then delete a vertex in each short cycle to produce a graph with no short cycles and no big cocliques — i.e., large girth *and* large chromatic number.

First observe that for any integer r, we have

$$\Pr(\alpha \geqslant r) \leqslant \sum_{U \subseteq [n]:\ |U|=r} \Pr(U \text{ is a coclique}) = \binom{n}{r}(1-p)^{\binom{r}{2}}. \tag{32}$$

Proposition 8.10. *Let $k \geqslant 0$ be an integer and let $p = p(n)$ depend on n such that $p \geqslant (6k \ln n)/n$ for $n \gg 0$. Then*

$$\lim_{n\to\infty} \Pr\left(\alpha \geqslant \frac{n}{2k}\right) \;=\; 0.$$

In other words, if p remains above a certain threshold, then almost always $\mathcal{G}(n,p)$ will not contain any small cocliques. (Why that particular bound for p? Erdős certainly didn't pull this out of thin air; he did the calculation first and then reverse-engineered the bound to work.)

Proof. Let $r = \lceil n/2k \rceil$. By (32) we have

$$\begin{aligned}
\Pr[\alpha \geqslant r] \;&\leqslant\; \binom{n}{r}(1-p)^{\binom{r}{2}} \\
&\leqslant\; n^r(1-p)^{\binom{r}{2}} && \text{(crude but valid)} \\
&=\; (n(1-p)^{(r-1)/2})^r \\
&\leqslant\; (ne^{-pr/2+p/2})^r && \text{(since } 1-p \leqslant e^{-p} \text{ for all } p\text{)} \\
&\leqslant\; (ne^{-3\ln n/2+p/2})^r && \text{(since } -pr \leqslant -(6k\ln n/n)(n/2k) = -3\ln n\text{)} \\
&=\; (n\, n^{-3/2}\, e^{p/2})^r \\
&=\; (n^{-1/2}\, e^{p/2})^r
\end{aligned}$$

which vanishes as $n \to \infty$. □

Proof of Theorem 8.9. The graph K_3 has $g = \chi = 3$, so suppose $k \geqslant 3$. Fix ε with $0 < \varepsilon < 1/k$ and let $p = n^{\varepsilon-1}$. Let X denote the number of short cycles in $\mathcal{G}(n,p)$; recall that "short" means "of length at most k". Then $X = X_3 + \cdots + X_k$ in the notation of the previous section, so by (30) we have

$$\mathbb{E}(X) \;=\; \sum_{r=3}^{k} \frac{n(n-1)\cdots(n-r+1)}{2r} p^r \;=\; \frac{1}{2}\sum_{r=3}^{k} n^r p^r \;=\; \frac{1}{2}\sum_{r=3}^{k} n^{r\varepsilon} \;\leqslant\; \frac{k-2}{2} n^{k\varepsilon}$$

and now Markov's inequality says that

$$\Pr(X \geqslant n/2) \;\leqslant\; \mathbb{E}(X)/(n/2) \;\leqslant\; (k-2)n^{k\varepsilon-1}.$$

But $\varepsilon < 1/k$ implies $k\varepsilon - 1 < 0$, so we conclude that

$$\lim_{n\to\infty} \Pr(X \geqslant n/2) = 0.$$

So for n sufficiently large,

$$\Pr[X \geqslant n/2] < 1/2. \tag{33}$$

Moreover, since $p = n^{\varepsilon-1}$, it is indeed the case that $p \geqslant (6k\ln n)/n$ for n sufficiently large. By Proposition 8.10, we can choose n large enough so that

$$\Pr[\alpha \geqslant n/2k] < 1/2. \tag{34}$$

Therefore, by (33) and (34) together,

$$\Pr[X \geqslant n/2 \text{ or } \alpha \geqslant n/k] < 1/2 + 1/2 = 1.$$

which implies that there must exist a graph G with $X(G) < n/2$ and $\alpha(G) < n/2k$. Delete a vertex from each of the (fewer than $n/2$) short cycles of G to get a graph H. Then H has girth $> k$, at least $n/2$ vertices, and $\alpha(H) \leqslant \alpha(G)$, so

$$\chi(H) \geqslant \frac{|H|}{\alpha(H)} \geqslant \frac{n/2}{\alpha(G)} > k.$$

□

This is the revision frontier — everything hereafter comes with no guarantee of legibility, let alone correctness.

8.7. **Threshold Functions.** Suppose we have some graph property Q that is *monotone*, i.e., is preserved by adding more edges. The simplest example is the property of having an edge! So are connectedness, k-connectivity (for any constant k), having minimum degree d (for any constant d), and non-planarity. Examples of non-monotone graph properties include regularity and chordality.

If p is a function of n, then how does

$$Q(G,p) := \lim_{n\to\infty} \Pr[\mathcal{G}(n,p) \text{ has property } Q]$$

depend on $p(n)$?

A theme of random graph theory is the appearance of **thresholds** in problems like this. For many monotone properties Q, there is some function $t = t(n)$ such that

- if $p \to 0$ faster than $t \to 0$, then $Q(G,p) = 0$;
- if $p \to 0$ slower than $t \to 0$, then $Q(G,p) = 1$.

More precisely,

$$Q(G,p) = \begin{cases} 0 & \text{if } \lim_{n\to\infty} p/t = 0, \\ 1 & \text{if } \lim_{n\to\infty} p/t = \infty. \end{cases}$$

Theorem 8.11. *Let Q be the property of having no isolated vertices (i.e., having $\delta \geqslant 1$). Then*

$$t(n) = \ln n/n$$

is a threshold function for Q.

In other words, if p is any function that decays even a little bit faster than t (say $p = 1/n$), then for large enough n, the probability that G will have an isolated vertex tends to 1. If p decays more slowly than t (say $p = \ln n/n^{0.99}$), then for large enough n, the probability that G will have an isolated vertex tends to 0.

First part of the proof. Let X_i be the event that vertex i is isolated and let X = number of isolated vertices. So

$$\mathbb{E}(X) = \sum_{i=1}^{n} \mathbb{E}(X_i) = \sum_{i=1}^{n} (1-p)^{n-1} = n(1-p)^{n-1}. \tag{35}$$

Now we start messing around with this sum. First,

$$(1-p)^{n-1} \sim (1-p)^n = e^{n\ln(1-p)} = e^{n(-p-p^2/2-p^3/3-\cdots)} < e^{-np} \tag{36}$$

(where $\sim$ means "has the same limit as $n \to \infty$"; note that $\lim_{n\to\infty}(1-p) = 1$). So

$$\mathbb{E}(X) < ne^{-np} \sim ne^{-np}. \tag{37}$$

Set $p = c \ln n/n$; then we get

$$\mathbb{E}(X) < ne^{-c \ln n} = n^{1-c}. \tag{38}$$

If $p(n)$ is above the threshold $t(n)$, then $c \to \infty$ and $n^{1-c} \to 0$.

We've now proved that if $p(n)/t(n) \to \infty$, then

$$\lim_{n\to\infty} \mathbb{E}(\text{number of isolated vertices}) = 0.$$

It follows (from Markov's Inequality) that $\lim_{n\to\infty} \Pr[X = 0] = 1$, i.e., G almost surely has no isolated vertices, i.e., $\delta(G) \geqslant 1$ almost surely.

The second part is to show that if $p(n)/t(n) \to 0$, then G almost surely *does* have an isolated vertex. Hold that thought; we'll finish the proof later. □

Lots of properties exhibit this threshold behavior, so we can talk about the "evolution" of a random graph as $p(n)$ increases.

- If $n^2 p \to 0$, then almost surely G **has no edges**. (This is very easy to prove.)
- If $np \to 0$ (or, if you prefer, $p = o(1/n)$) then $G_{n,p}$ is almost surely **acyclic**.
- If $np \to c$ for some $c \in (0, 1)$, then $G_{n,p}$ is almost surely **a bunch of components, each of which has at most one cycle**.

$np \to 1$ is called the *phase transition*.

- If $np \to c$ for some $c > 1$, then $G_{n,p}$ almost surely consists of a "giant component" containing lots of the vertices and a whole mess of small components, each containing at most one cycle.

(To be more specific, the giant component has more than $(1 - 1/c)n$ vertices, while the second largest component has only $O(\log n)$.)

This explains the term "phase transition": imagine the graph as a potful of water molecules being cooled. The cooler the water gets, the more likely any two water molecules are to bond with each other (i.e., p is increasing). In the narrow range around 32°F, the graph becomes slushy and then solidifies suddenly into a big block of ice with some bits of slush lying around.

- If $np \to \infty$, but $np - \log n \to -\infty$, then $G_{n,p}$ almost surely consists of a "**giant component**" containing almost all the vertices and a few acyclic components.

(The giant ice block is starting to swallow the bigger bits of slush, leaving only a few small drops of water here and there.)

- If $np - \log n \to \infty$, then $G_{n,p}$ is almost surely **connected**.

8.8. **Using the variance for lower thresholds.** In order to show that $t(n)$ is an upper threshold for some graph property — i.e., that the property almost surely doesn't occur when $p/t \to 0$ — we need to bound probability from above, and the standard tool for this is Markov's inequality. On the other side of the threshold, though, we need a way of bounding probabilities from below.

Definition 8.12. Let X be a random variable with mean μ. The **variance** of X is

$$\sigma^2 = \mathrm{Var}(X) = \mathbb{E}\big((X - \mu)^2\big).$$

This is a measure of how much X deviates from its expected value. A useful formula for the variance is

$$\begin{aligned}\mathrm{Var}(X) &= \mathbb{E}\big(X^2 - 2X\mu + \mu^2\big) = \mathbb{E}(X^2) - 2\mu\mathbb{E}(X) + \mu^2 = \mathbb{E}(X^2) - \mu^2 \\ &= \mathbb{E}(X^2) - \mathbb{E}(X)^2. \qquad (39)\end{aligned}$$

Lemma 8.13 (Chebyshev's inequality)**.** *Let X be a random variable with mean μ. Then*

$$\Pr[|X - \mu| \geqslant \lambda] \;\leqslant\; \mathrm{Var}(X)/\lambda^2.$$

Proof. By Markov's inequality and the definition of variance, we have

$$\Pr[|X - \mu| \geqslant \lambda] \;=\; \Pr[(X-\mu)^2 \geqslant \lambda^2] \;\leqslant\; \mathbb{E}\big((X-\mu)^2\big)/\lambda^2 \;=\; \mathrm{Var}(X)/\lambda^2. \qquad \square$$

Proposition 8.14. *Let X be a $\mathbb{N}$-valued random variable on $\mathcal{G}(n,p)$ with mean μ. If $\mu > 0$ for n sufficiently large and $\mathrm{Var}(X)/\mu^2 \to 0$, then $X(G) \geqslant 1$ for almost all G.*

Proof. If $X(G) = 0$ then $|X(G) - \mu| = \mu$. So

$$\Pr[X(G) = 0] \;\leqslant\; \Pr[|X(G) - \mu| = \mu] \;\leqslant\; \Pr[|X(G) - \mu| \geqslant \mu] \;\leqslant\; \sigma^2/\mu^2 \;\xrightarrow[n\to\infty]{}\; 0. \qquad \square$$

Corollary 8.15 (Second Moment[13] Method)**.** *If $\mathbb{E}(X^2)/\mathbb{E}(X)^2 \to 1$ then $\Pr(X = 0) \to 0$.*

Proof. The hypothesis and (39) imply that

$$\frac{\mathrm{Var}(X)}{\mu^2} = \frac{\mathbb{E}(X^2) - \mu^2}{\mu^2} = \frac{\mathbb{E}(X^2)}{\mu^2} - 1 \quad\to\quad 1 - 1 = 0. \qquad \square$$

As an application, we complete the proof of Theorem 8.11, which states that

Second part of proof. Let $t = \ln n/n$. We wish to show that if $p/t \to \infty$ then G almost surely has no isolated vertex. As before, let X_i be the indicator variable for the event that i is isolated and let $X = \sum X_i$. We have already calculated $\mathbb{E}(X) = n(1-p)^{n-1}$, so $\mathbb{E}(X)^2 = n^2(1-p)^{2n-2}$. Meanwhile,

$$\mathbb{E}(X^2) = \mathbb{E}\left(\sum_{i=1}^n X_i^2 + 2\sum_{i<j} X_iX_j\right) = \sum_{i=1}^n \mathbb{E}(X_i) + 2\sum_{i<j}\mathbb{E}(X_iX_j) = n(1-p)^{n-1} + 2\binom{n}{2}(1-p)^{2n-3}$$

$\square$

[13]The k^{th} **moment** of a random variable X is $\mathbb{E}(X^k)$.