Introduction

These notes summarize the last seven weeks of lectures (after Reading Week) in ECE203 (Discrete Mathematics). We (John Carter and Danny Heap) roughly follow the schedule (soon to be posted on CC-net):

```
week 8:
        Graphs: introduction, terminology, special graphs,
        representation and connectivity.
        Sections 8.1, 8.2, 8.3, 8.4

week 9:
        Graphs: Eulerian and Hamiltonian paths, shortest paths,
        Dijkstra and Floyd algorithms, planar graphs
        Sections 8.5, 8.6, 8.7

week 10:
        Graph colouring
        Trees: introduction, spanning trees, MCST, Kruskal and
        Prim's algorithm
        Sections 8.8, 9.1, 9.4, 9.5

week 11:
        Counting: permutations and combinations
        Sections 4.1, 4.2, 4.3

week 12:
        Counting: binomial coefficients,
        generalized permutations and combinations,
        inclusion/exclusion principle
        Sections 4.4, 4.5, 6.5

week 13:
        Discrete probability, conditional probability
        Sections 5.1, 5.2

week 14:
        Recurrence relations
        Sections 6.1, 6.2
```

Week 8
Graphs

Graphs are used in many fields to represent different problems:

- niches in an ecosystem
- planar circuit boards
- personal influence
- structure of the WWW
- efficient tours of city streets
- etcetera...

Since graphs consist (see below) of edges and nodes, basically any binary relation can be represented by edges and be embedded in a graph.

We emphasize simple and directed graphs (no self-loops or multiple edges) but a wide set of definitions is used by Rosen, and you may need to recognize them for some homework problems:

simple graph $G = (V, E)$, no self loops or multiple edges, edges are unordered pairs of vertices.

multigraph Multiple edges allowed beteen a given pair of vertices, no self-edges.

pseudograph Self-edges allowed.

directed graph $G = (V, E)$, edges are ordered pairs of vertices, self-loops are allowed.

directed multigraph Multiple edges with the same initial and terminal vertex.

Relatively small graphs can be represented using pictures — dots for vertices, lines/curves for edges. Draw examples of an influence graph, a six-team round-robin, niche overlaps in an ecosystem, precedence in computer statements.

adjacent: u and v are adjacent if there is an edge between them.

degree of a vertex: In an undirected graph, the number of edges incident to a vertex.

adjacent to (from): u is adjacent to v, and v is adjacent from u, if there is an edge from u to v.

in (out) degree: Number of edges that terminate ($\deg^-(v)$) at v; number of edges that originate ($\deg+(v)$).

Easy consequences for an undirected graph:

$$2|e| \sum_{v \in V} \deg(v).$$

A simple graph has an even number of vertices of odd degree.

Consequence for directed graphs:

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg+(v)$$

Special simple graphs

Here are some special graphs:

Complete graph, K_n: The graph with n nodes with exactly one edge between each distinct pair of nodes.

Cycle, C_n: For $n \geq 3$, cycles through each vertex and returns to start.

Wheel, W_n: Cycle with a central hub.

n-cube Q_n: One vertex for each of the the binary string of length n. Edges exist between edges that differ in exactly one bit.

Bipartite graphs, and $K_{m,n}$: Vertices can be partitioned into two disjoint sets, and all edges are between these two sets. $K_{m,n}$ is the bipartite graph where every vertex in one partition (of size m) has an edge to every vertex in the other partition (of size n).

A simple graph is bipartite if and only if you can colour it with (at most) two colours in such a way that neighbouring vertices never have the same colour. How would you go about automatically determining whether a graph is bipartite or not (devise an algorithm)?

Examples: C_8 is bipartite, K_5 is not.

Why, in a bipartite graph is $|E| \leq |V|^2/4$ (homework).

Applications of special graphs

Star and ring topologies are used in connecting local area networks. The wheel topology combines the two. Properties of C_n and W_n are useful in the latter two cases.

Message-passing between CPUs in parallel processing requires decisions about how many CPUs are mutually connected. All-to-all corresponds to K_n, which is simple but expensive. A linear or grid array is cheaper, but increases the number of hops. Hypercubes (Q_n) are somewhere in between. Each node has $\deg(n)$, which is also the maximum number of hops. There are 2^n nodes.

Joining and splitting graphs

A subgraph of $G = (V, E)$, is a graph $H = (W, F)$, where $W \subset W$, and $F \subset E$. In addition, the edges in F must correspond to vertices in W.

A union of simple graphs $G_1 = (V_1, E_1)$, and $G_2 = (V_2, E_2)$ is $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.

Graph representations

We've already represented some small graphs by drawing them. We can represent the same information in a few other ways.

adjacency list For each vertex, list its adjacent (neigbouring) edges). You can easily modify to list each the edges that are adjacent to a given vertex.

adjacency matrix If $|V| = n$, then an $n \times n$ matrix containing a 1 at entry i, j if there is an edge from v_i to v_j, and a zero otherwise represents a graph (this matrix will be symmetric for a simple graph). To represent multiple parallel edges, the non-zero entry can represent the number of edges from v_i to v_j.

incidence matrix If $|V| = n$ and $|E| = m$, then an $n \times m$ matrix containing a 1 at entry i, j if v_i touches (is incident with) e_j, and 0 otherwise, represents a simple graph.

Isomorphism

Sometimes two graphs two graphs may be drawn differently, or use different labels for their vertices, but they have the same structure. One way to picture this is whether you can re-draw one graph, without changing the adjacency of the vertices, to coincide with another. More formally. if $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, and there's a one-to-one, onto function (i.e. a bijection) $f : G_1 \to G_2$ with the property that u and v are adjacent in G_1 if and only if $f(u)$ and $f(v)$ are adjacent in G_2, then f is an *isomorphism* and G_1 and G_2 are *isomorphic*.

Isomorphism can be hard to establish. Non-isomorphism follows quickly if $|V_1| \neq |V_2|$, $|E_1| \neq |E_2|$, or some isomorphic property is not preserved (e.g. degrees of vertices).

Another way to demonstrate isomorphism is to re-order the adjacency matrix of the two graphs so that they coincide.

Graph isomorphism is NP-complete, although some good polynomial-time algorithms are known (e.g. NAUTY).

Connectivity

A path is a sequence of edges that begins at one vertex and terminates at another. For simple graphs, this corresponds to a list of vertices, where each pair in the list has an edge between them. For a directed graph, a list of vertices also describes a path, provided we interpret each edge as going from x_i to x_{i+1}.

Examples: paths in niche overlap graph, influence graph, computer network.

$x_1, \ldots, x_n$.

A **circuit** is a path that begins and ends on the same vertex, and has length greater than zero. A path **passes through** the vertices at the end points of the edges it **traverses**. A path is **simple** if it does not contain the same edge more than once.

An undirected graph is **connected** if there is a path between every pair of vertices (no need to verify "and back"). If a graph is connected, a simple path exists between every pair of vertices.

A path that isn't connected is a collection of connected subgraphs (in the worst case, the individual vertices). The **connected components** of a undirected graph are the maximal connected components.

Some vertices play a key role: remove them and a graph breaks into more connected components. Such a vertex is called a **cut vertex** (**articulation point**). Analogously, an edge whose removal produces more connected subgraphs is called a **cut edge** or **bridge**.

Things are a little more complicated for directed graphs, which are called **strongly connected** if there is a path from u to v, and from v to u whenever u and v are vertices in the graph. A directed graph is **weakly connected** if the underlying undirected graph is connected. Subgraphs of G that are strongly connected but not contained in any larger strongly connected components are G's **strongly connected components** or **strong components**.

Paths can be used to establish (non)isomorphisms between graphs, since paths and circuits are preserved by isomorphism. The existence of a simple circuit of a given length is a useful invariant to check.

Theorem: If A is the adjacency matrix of graph G, with respect to vertices $v_1, \ldots, v_n$, then the number of paths of length r from v_i to v_j is the (i, j)th entry of A^r.

(proof: induction on r).

Week 9
Euler paths and circuits

A simple path that traverses every edge of a graph is called an Euler path (after Swiss mathematician Leonard Euler). If, in addition, the path begins and ends at the same vertex, it is an Euler circuit. The bridges of Kallinin correspond to a non-Euler circuit (or path, for that matter).

Theorem: A connected multigraph has an Euler circuit if and only if each of its vertices has even degree.

Proof ("only if"): Suppose G has an Euler circuit that begins on node u. The first edge of the circuit contributes 1 to the degree of u. Every other node increases its degree by 2 when the circuit passes through it (possibly more than once), so they all have even degree. The starting node, u, has its degree increase by 1 when the circuit terminates there. So the degree of u is $1 + 1 + 2 \times$ (number of times u was passed through), and all nodes of G must have an even degree, as claimed.

Proof ("if"): Suppose G is a connected multigraph with each vertex having an even degree. Select any vertex, say u, to start off a circuit with. Select an unused edge to an adjoining vertex, and continue selecting unused edges until they are used up. This process terminates, since there are a finite number of edges. In addition, this process terminates at the original vertex u, since every other vertex (having even degree) has an out-going edge to match each in-coming edge. We now have a simple circuit beginning and ending at u, but we may not be done, since there may be some unused edges left over.

Form a new graph, G', from the unused edges and any vertices they are incident to. Since the original graph was connected, the connected components of G' must each share a vertex (let's call it u') with the edges in the circuit we already found. In addition, each vertex of G' has even degree, since an even number of edges were removed from each node to form the original circuit.

Repeat the process above to generate a new simple circuit that shares vertex u' with the first circuit. The two simple circuits can be spliced to form a single simple circuit.

Repeat this process, which must terminate since at each stage more edges are used, and produce an Euler circuit.

Some graphs have an Euler path but no Euler circuit — the simple path that uses all edges begins and ends on different vertices. By considering what the degree of the initial and terminal vertices must be, you get:

Theorem: A connected multigraph has an Euler path, and not an Euler circuit, if and only if it has exactly two vertices of odd degree, all other vertices have even degree.

Proof ("only if"): Suppose connected multigraph has an Euler path from u to v (distinct vertices). The first edge leading out of u contributes 1 to its degree, and the last edge leading into v contributes 1 to its degree. Every portion passes through vertices (including possible passes through u and v), contributing 2 to their degree each time. Thus u and v must have odd degree, and every other vertex has even degree, as claimed.

Proof ("if"): Suppose G is a connected multigraph with exactly two vertices (call them u and v) with odd degree (in other words, any other vertices have even degree. Create a larger graph, G' with an additional vertex w and edges (u, w) and (v, w). Every vertex of G' has even degree, so there is an Euler circuit beginning at w. If we remove w and the edges (u, w) and (v, w) from G', this Euler circuit becomes an Euler path.

A simple path that visits every vertex of a graph is called a **Hamilton path**. A simple circuit that visits every vertex of a graph is called a **Hamilton circuit**. Both are named after Irish mathematician Rowan Hamilton.

There is no nice algorithm for determining whether a graph has a Hamilton circuit or path, even though the question sounds very similar to the easily-answered one about Euler paths and circuits. Deciding whether G has a Hamilton path or circuit is NP-complete.

However, some properties are known:

- If G has a vertex with degree 1, it does not have a Hamilton circuit.
- If G has a vertex with degree 2, then both must be used in any Hamilton circuit.
- If G has a vertex with degree greater than 2, once a Hamilton circuit passes through this vertex all other edges incident with it are removed from consideration.
- A Hamilton circuit may not contain a smaller circuit.
- For $n > 2$ K_n has a Hamilton circuit: $v_1, v_2, \ldots, v_n, v_1$.

Dirac's Theorem: If G is a simple graph with n vertices ($n \geq 3$), such that degree of every vertex is at least $n/2$, then G has a Hamilton circuit.

(Proof omitted).

Ore's Thorem: If G is a simple graph with n vertices with $n \geq 3$ such that $\deg(u) + \deg(v) \geq n$ for every pair of nonadjacent vertices u and v, then G has a Hamilton circuit.

(Proof omitted).

The best-known algorithms for finding a Hamilton circuit, or even determining whether there is one, are exponential. An equivalent problem is the **travelling salesman problem:** Find the shortest route to visit a set of cities, and then return home.

A **Gray code** is simply a Hamilton circuit in Q_n.

Shortest paths

Graphs may have weights (real numbers) associated with their edges. These numbers might represent geographical distance, cost, time, or some other quantity. Such a graph is a **weighted graph**, and the definition of path length must be modified from the number of edges in a path to the sum of the weights of the edges in a path.

A reasonable question is what is the shortest path between two vertices? Note that this question may not have a well-defined answer if a graph contains negative circuits (such a circuit may allow us to drive path length down toward $-\infty$).

If we assume that a simple graph is connected and has nonnegative edge weights, then Dijkstra's algorithm will find the minimum path length from a single source (a distinguished vertex) to any other vertex.

Here's the algorithm for simple, connected graph $G = (V, E)$ with nonnegative edge weights and vertices $v_0, \ldots, v_{n-1}$ (the weight of edge (v_i, v_j) is $w(v_i, v_j)$). The source is v_0, and the path length to vertex v using only intermediate vertices in S is $L(v)$.

for $i = 1$ **to** n
    $L(v_i) = \infty$
$L(v_0) = 0$
$S = \emptyset$
while $S \neq V$
begin
    u = vertex not in S with minimal $L(u)$
    $S = S \cup \{u\}$
    for all vertices v not in S
        if $L(u) + w(u, v) < L(v)$ then $L(v) = L(u) + w(u, v)$
end

We'll call S after the ith iteration S_i (So $S_0 = \emptyset$), and the label on a vertex after the ith iteration $L_i(u)$. Then

Claim: After iteration i

1. The label of every vertex in S_i is the length of a shortest path from v_0 to this vertex, and
2. The label of every vertex not in S_i is the length of a shortest path from v_0 to this vertex provided every other vertex on the path is in S_i.

Proof (induction on i): When i is zero, the claim is true by inspection, since S_0 is empty and all paths from v_0 to any vertex that use intermediate vertices in S_0 are infinite, except the zero-length path to v_0.

For the inductive step we must show that if the claim holds for some $i \geq 1$, then it must also hold for $i + 1$. Assume that the claim holds for i. Choose some vertex u with minimal label $L_i(u)$ that is not in S_i, and assume this is the vertex to be added to S at step $i + 1$. If the claim holds for case i, then the vertices in S_i already have been labelled with the length of a minimum length path from v_0, so it only remains to show that the same is true of u to verify the first part of the claim for case $i + 1$. If this were not so, then there would be a path of length less than $L_i(u)$ from v_0 to u, and it must use some vertex that is not in S_i. Call the first vertex outside S_i in this allegedly shorter path v. This would mean that

$$L_i(v) + \text{length of path from } v \text{ to } u < L_i(u).$$

This would mean that $L_i(v) < L_i(u)$ since **every** path has nonnegative length. But this contradicts the choice of u, and is thus impossible. So the first part of the claim holds for $i + 1$.

To establish the second part of the claim, let v be any vertex not in S_{i+1}. There are two possibilities. A shortest path from v_0 to v using only vertices in S_{i+1} doesn't use u, so $L_{i+1}(v) = L_i(v)$. Otherwise, a shortest path from v_0 to v uses u, so u must be the last vertex visited in S_{i+1}, and

$L_{i+1}(v) = L_i(u) + w(u, v)$. The algorithm takes the minimum of these two quantities, so the second part of the claim holds for case $i + 1$.

The base case and induction step imply that the claim holds for $i = 0, \ldots n - 1$. After step $n - 1$ $S = V$, so for every vertex u, $L(u)$ is the length of a minimum path from v_0 to u.

Inspecting Dijkstra's algorithm you can see that the main loop makes n iterations, since there are n vertices to add to S. In each iteration we find the maximum over at most n values, hence $n-1$ comparisons. We then update the vertices not in S_k, using at most $n - 1$ operations. Hence we have $O(n(2(n - 1)))$ or $O(n^2)$ worst-case complexity.

Travelling salesperson problem

Consider a salesperson who must visit n cities, each pair having a road between them. If they want to minimize the distance travelled, they must calculate a Hamilton circuit of minimum length. The most straightforward way to do this is to consider **every** Hamilton circuit, and calculate its length. There are $(n - 1)!$ of those. A bit of economizing can be achieved by eliminating circuits that are identical to other circuits, except for the direction: $(n - 1)!/2$ of those. The number of circuits to consider still grows quickly: 25 cities yields $24!/2$, approximately 3.1×10^{23} circuits to consider, beyond the capabilities of any computer we can currently imagine.

Things don't get much better. The TSP is NP-complete and no algorithm with polytime complexity is known to exist. There are algorithms for an approximate solution (guaranteed to be no more than twice, three times, 10 times, ... as long as the optimal solution) that work for some cases. For example, if a graph satisfies the triangle inequality (as a geographical map on flat ground would) then there is a polytime algorithms that finds a solution no worse than 3/2 times as long as an optimal solution.

Iterated Dijkstra

If you're required to find the interpair minimum path length for every pair of vertices in a graph, you can iterate Dijkstra for each vertex of the graph (making each vertex the source, in turn). This yields $O(n^3)$ performance, but has the important drawback that the result may not be correct if the graph has any negative weights (but no negative cycles). A more robust algorithm is Floyd's Algorithm, an example of Dynamic Programming.

In Floyd's Algorithm, define $D(i, j, k)$ to be the minimum path length from vertex i to vertex j, provided you are only allowed to use the vertices $0, \ldots, k$ as intermediate steps on the path. Since you have to start somewhere, define $D(i, j, -1)$ to be the minimum path length from i to j where you use no intermediate vertices, in other words $w(i, j)$. Now for $k > -1$ you have:

$$D(i, j, k) = \min\{D(i, j, k - 1), D(i, k, k - 1) + D(k, j, k - 1).$$

... since this takes the minimum of paths from i to j that don't use k, and those that do. This allows us to fill in a table for $D(i, j, k)$ (increasing k at each step), until we find $D(i, j, n)$ (if n is the highest index) for every vertex, which is the minimum path length from i to j. The exercises to this chapter show an implementation of Floyd's algorithm that use only 3 parameters (k is implicit).

Planar graphs

A graph that can be drawn in the plane without any edges crossing is called **planar** (such a drawing is a **planar representation**. Since a planar graph may also be drawn with crossing edges, it may not be obvious that it is indeed planar, take K_4 and Q_3 as examples. Simply trying (and failing) to draw a graph without any crossing edges doesn't prove that it is nonplanar.

It takes some careful reasoning to show that $K_{3,3}$ is not planar. Suppose that $K_{3,3}$ is partitioned so that vertices v_1, v_2, and v_3 each have edges to v_4, v_5, and v_6. In a planar representation, the circuit v_1, v_5, v_2, v_4, v_1

divides the plane into two regions, call them R_1 and R_2. v_3 must go into one of these regions, WLOG R_1. Now the edges (v_3, v_5) and (v_3, v_4) divide this region into two subregions, say R_{11}, and R_{12}. Which region can v_6 go into? If it's placed in either R_{11} or R_{12}, then it must cross an edge to get to either v_1 or v_2. If it's placed in R_2, then it must cross an edge to get to v_3. So it's impossible to come up with a planar representation of $K_{3,3}$.

A planar representation of a graph divides the plane up into regions, one unbounded region and the others surrounded by edges.

Euler's Formula: If G is a connected, planar, simple graph, the number of regions is r, the number of edges is e, and the number of vertices is v, then

$$r = e - v + 2.$$

This is true for any planar graph. Variants of this formula hold if you try to draw a graph on a torus or other surface without any edges crossing.

Proof: Specify a planar representation of G. Construct a sequence of subgraphs $G_1, G_2, \ldots, G_e = G$ by successively adding an edge at each stage. Choose G_1 arbitrarily. Obtain G_n from G_{n-1} by adding an edge that is incident with a vertex in G_n, plus the vertex at the other end of this edge, if necessary. Let r_n, v_n and e_n be the values of r, v, and e for G_n. Proceed by induction on n.

Base case: Euler's formula holds for G_n when $n = 1$, since a single edge has 2 vertices and divides the plane into 2 regions, so $1 = 1 - 2 + 2$

Inductive step Assume Euler's formula holds G_n for some $n \geq 1$, and show this implies that it holds for G_{n+1}. Call the edge we add to G_n to form G_{n+1} (u_{n+1}, v_{n+1}). Assume that u_{n+1} is the vertex shared with G_n. Now there are 2 possibilities:

1. v_{n+1} is already part of G_n, so adding edge (u_{n+1}, v_{n+1}) completes a circuit and adds a new region. Since we assumed that $r_n = e_n - v_n + 2$, we now have one more region and one more edge, so
$$r_{n+1} = 1 + r_n = 1 + e_n - v_n + 2 = e_{n+1} - v_{n+1} + 2.$$
So the claim holds in this case.
2. v_{n+1} is not part of G_n, so the new edge doesn't create any new regions (both vertices are on the boundary of a common region, otherwise we'd need to cross an edge to join them), but the number of vertices and edges each increase by 1, and
$$r_{n+1} = r_n = e_n - v_n + 2 = e_{n+1} - v_{n+1} + 2.$$
So the claim holds in this case.

Since the truth of the claim for G_n implies it is true for G_{n+1}, this implies it true for every $n \leq e$. And G_e is our original graph, which thus satisfies Euler's Formula.

Some consequences of Euler's Formula:

Corollary 1: If G is a connected, planar, simple graph with e edges and v vertices and $v \geq 3$, then $e \leq 3v-6$.

Proof: Define the **degree** of a region as the number of edges on its boundary. Some edges are traversed twice on the boundary, and thus contribute 2 to the degree. Suppose a connected, planar, simple graph divides the plane into r regions. The degree of each region is at least 3, since we don't have self loops or multiple edges to allow degrees of 1 or 2. The unbounded region has degree at least 3, since we require at least 3 vertices in the hypothesis.

Notice that the sum of the degrees of the regions is exactly twice the number of edges (does that sound familiar?), and each of the degrees is at least 3 so:

$$2e = \sum_{\text{all regions } R} \deg(R) \geq 3r \Longrightarrow \frac{2}{3}e \geq r.$$

Plug this into Euler's formula:

$$\frac{2}{3}e \geq e - v + 2 \Longrightarrow 3v - 6 \geq e.$$

As claimed.

Corollary 2: If G is a connected, planar, simple graph, then G has a vertex of degree not exceeding five.

Proof: If G has fewer than 3 vertices, then all of them have degree at most 1, which is certainly no more than 5. Otherwise, Corollary 1 applies, and we know that $e \leq 3v - 6$, so $2e \leq 6v - 12$, which implies $2e < 6v$. If there were no vertex of degree 5 or less, then every vertex would have degree at least 6, and by the Handshaking Theorem $2e \geq 6v$. But that contradicts $2e < 6v$, which we've already established, so there must be a vertex with degree 6 or less.

You can use Corollary 1 to show that K_5 is not planar, since it has 10 edges and 5 vertices and:

$$e = 10 \not\leq 3 \times 5 - 6 = 9.$$

We've already used an *ad hoc* argument to show that $K_{3,3}$ is not planar, but this fact also follows from another consequence of Euler's Formula:

Corollary 3: If a connected, simple, planar graph has e edges, v vertices, $v \geq 3$, and no circuits of length three, then $e \leq 2v - 4$.

Proof: Suppose G is connected, simple, and planar with e edges, divides the plane into r regions, and has v vertices. Since there are no circuits of length 3, every region (including the unbounded one) has degree at least 4. Summing up the edges along the boundary of each region counts the edges twice, so

$$2e = \sum_{\text{all regions } R} \deg(R) \geq 4r \Longrightarrow \frac{1}{2}e \geq r.$$

Plug this into Euler's formula, and

$$e - v + 2 \leq \frac{e}{2} \Longrightarrow e \leq 2v - 4.$$

As claimed.

You can use Corollary 3 to show that $K_{3,3}$ is not planar. It has no circuits of length 3, it has 6 vertices and nine edges, so $v = 6$, and $e = 9$, and

$$9 \not\leq 2 \times 6 - 4 = 8.$$

So which graphs are planar or nonplanar? Define an **elementary subdivision** as the process of replacing an edge by two edges with a shared new vertex (the other two vertices are the original vertices of the original edge). Two graphs are **homeomorphic** if the can be generated from the same graph using only elementary subdivisions (0 or more of them). Then

Kuratowski's theorem: A graph is nonplanar if and only if it contains a subgraph homeomorphic to $K_{3,3}$ or K_5.

(proof omitted)

Week 10
Graph colouring

A **colouring** of a simple graph is an assignment of a colour to each vertex in such a way that adjacent vertices do not have the same colour. If you can colour a graph with n colours, this is analogous to partitioning the vertices into n sets where each set has no edges within it.

The **chromatic number** of a graph is the least number of colours needed to colour it. If G has chromatic number n then there is an n-colouring of G, but there is no $(n-1)$-colouring of G.

Examples: K_4, K_4-plus-a-node, and K_4-plus-a-node-plus-an-edge.

Four colour theorem: The chromatic number of a planar graph is no greater than four.

Proof: Takes (or took) more than 100 years (Appel and Haken). The solution examines hundreds of cases of possible counterexamples (using a computer program) and determines that no planar graphs of that type exist. Some mathematicians are unsatisfied with proof-by-program.

The question began with the observation that it always seemed possible to colour a map with four colours, although there are some maps that can be coloured with three colours. The conjecture was that all maps drawn on a plane could be coloured with a maximum of four colours.

The question got boiled down into graph theory as follows. Equate the regions of the map with vertices of a graph, and vertices have an edge between them if the corresponding regions have a boundary (not just a single point). The graphs formed in this way will be simple, connected, and planar. So the map-colouring problem gets re-stated as above.

The chromatic number of K_n is n, since the best you can do is colour each vertex a separate colour. The chromatic number of C_n is 2 if n is even, since alternating colours will work. If n is odd, a third colour is required and the chromatic number is 3.

The best-known algorithms for finding the chromatic number of a graph have exponential complexity. Even algorithms to find a colouring that is no worse than double the chromatic number have this complexity.

Scheduling problem

Designing a schedule without conflicts can be modelled as a graph colouring problem. Suppose you must find time slots for n exams in such a way that no student has two exams at the same time. This corresponds to a graph G with n vertices, and there is an edge between 2 vertices if there is at least 1 student who must write both exams. Finding the chromatic number of this graph will also tell you the smallest number of time slots that the exams can be fit into without ever having a student required to be in two places at once.

Trees

A special category of graphs, those that are connected, undirected, and with no simple circuits are called **trees**. These can be used to model geneology, parsing of mathematical expressions, searches for solutions to problems, and many other problems. The condition that a tree has no circuits means there are no self-loops or multiple edges, hence trees are simple graphs (but not every simple graph is a tree).

Any graph that contains no simple circuits contains a tree. In fact, every connected component of such a graph is a tree, and the entire collection of trees is called . . . a **forest**.

An equivalent definition of a tree is

Theorem: An undirected graph is a tree iff there is a unique simple path between any two of its vertices.

Proof omitted

Some terminology

Rooted tree: A tree where one vertex has been designated (distinguished) as the root, and every edge is directed away from the root. Generally we draw these with the root at the top (not very biological).

parent of v: is the unique vertex u such that there is a directed edge (u, v) (away from the root). The root has no parent.

child of u: if u is v's parent, then v is u's child.

siblings: vertices that share a parent.

ancestors of v: those vertices on a path from the root to v, not including v itself. The root is every vertex's ancestor, except itself.

descendents of v: those vertices that have v as an ancestor.

leaf: A vertex with no children.

internal vertex: A vertex with at least one child.

subtree rooted at u: subgraph of the tree consisting of u and all its descendents, and the edges incident to u's descendents.

m-ary tree: A tree where every vertex has no more than m children.

full m-ary tree: A tree where every internal vertex has exactly m children.

binary tree: An m-ary tree with $m = 2$.

ordered rooted tree: A rooted tree where children of internal nodes are ordered. This ordering is indicating in drawing the children from left to right.

ordered binary tree: A binary tree where the children of internal nodes are designated **right child** and **left child**. The subtrees rooted at these children at designated **left subtree** and **right subtree**.

Here are some properties of trees that aren't hard to prove:

Theorem: A tree with n vertices has $n - 1$ edges.

Proof: Homework question #20 on page 575 proves something very similar.

Theorem: A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices.

Proof: There are mi children in a full m-ary tree. This counts every vertex except the root, so there are $mi + 1$ vertices in total.

Once you know one of n (number of vertices), i (number of internal vertices), or l (number of leaves), you can use algebra to find the other two:

Theorem: A full$m$-ary tree with

1. n vertices has $i = (n - 1)/m$ internal vertices and $l = [(m - 1)n + 1]/m$ leaves.
2. i internal vertices has $n = mi + 1$ vertices and $l = (m - 1)i + 1$ leaves.
3. l leaves has $n = (ml - 1)/(m - 1)$ vertices and $i = (l - 1)/(m - 1)$ internal vertices.

Proof: Algebra.

Level of v in rooted tree: Length of the path from root to v. The root has level 0.

Height of a tree: Maximum level of its vertices, in other words the length of the longest path to one of the leaves.

Balanced tree: A tree of height h with all leaves at level h or $h - 1$.

Theorem: There are at most m^h leaves in an m-ary tree of height h.

Proof: When $m = 0$ we have a tree with a single leaf — the root, and height $h = 0$. This is the base case. Use complete induction: assume that the claim holds for every tree with height less than h, for some $h > 0$. Then the children of the root node are all m-ary trees with height at most $h - 1$, so they each have no more than m^{h-1} leaves. The sum of all the leaves is thus no more than $m \times m^{h-1}$ or m^h, and the induction step is proved.

Theorem: An m-ary tree of height h with l leaves must have $h \geq \lceil \log_m l \rceil$. If this m-ary tree is full and balanced, then $h = \lceil \log_m l \rceil$.

Proof: Omitted (it's in the text).

Spanning trees

Spanning tree: If G is a simple graph, then T is a spanning tree of G if T is a subgraph of G, is a tree, and contains every vertex of G.

Theorem: A simple graph is connected iff it has a spanning tree.

Proof (sketch): Suppose G has a spanning tree T. By the definition of a spanning tree, T is connected and contains every vertex in G. Thus every pair of vertices in G are connected: there is a path between them in T.

On the other hand, suppose G is a simple connected graph. If it has no simple circuits, then G is its own spanning tree. Otherwise, an edge can be removed from a simple circuit without making G disconnected. Remove edges from circuits until none are left, and the result is a spanning tree for G.

The proof just given assures us that we can always find a spanning tree for a connected simple graph G. However, checking at each step for simple circuits is an expensive operation. It would be easier to add edges from G to T in such a way that no circuits are created and T remains connected, until every vertex is included in T.

One approach is **Depth-first search** of a simple connected graph. The idea is to choose any vertex u arbitrarily as the starting point (and the root of the spanning tree). Now choose a neighbour of u that hasn't been visited yet (say v), add the edge (u, v) to T, and continue the process for neighbours of v. When this process terminates, return **(backtrack)** to unvisited neighbours of the parent of the last vertex visited, and then its parent, and finally back to u. Here's this procedure stated as an algorithm:

procedure *visit*(v: a vertex of G)
for each vertex w adjacent to v and not yet visited
    add w and edge (v, w) to T
    visit(w)
end for
procedure *DFS*(G: connected graph with vertices $v_1, \ldots, v_n$)
$T = v_1$ (and no edges)
visit(v_1)

For the procedure *visit* we need to keep track of the neighbours of v until we visit them. The appropriate algorithm for this is either recursive (as above) or uses a stack to keep track of neighbours that have not yet been explored.

Another approach is **Breadth-first search** of a simple connected graph. Again choose any vertex u arbitrarily as the root of the spanning tree. The neighbours of u become level 1 of the spanning tree, by adding all edges from u to them. The unvisited neighbours of the level 1 vertices become level 2 of the spanning tree, by adding all edges from level 1 vertices to them. And so on.

procedure *BFS* (G: connected graph with vertices $v_1, \ldots, v_n$)
$T = v_1$ (and no edges)
L = empty list (queue)
Put v_1 into L
while L is not empty
    remove the first vertex from L
    for each neighbour w of v
        if w is not in L or T then
            add w to the end of the list L
            add w and the edge $(v, 2)$ to T
        end if
    end for
end while

Minimum spanning trees (MST)

In many cases a connected graph has several spanning trees. It can be important to find the one that minimizes the sum of the edge weights — this may correspond to the lowest-cost solution to a problem, for example keeping a network of towns (or computer nodes) connected.

A **minimum spanning tree** in a connected, weighted graph is a spanning tree that has the smallest possible sum of edge weights. There are several ways to find one, and two are presented.

Prim's algorithm solves this problem by building up a spanning tree, edge-by-edge, always keeping the intermediate graph connected and without simple circuits, and choosing the lightest possible edge at each stage. To keep the developing solution connected and simple circuitless, you choose edges that are incident to one vertex already in the developing solution (keeping things connected), but with the other vertex not already in the developing solution (avoiding cycles).

procedure *prim*(G: weighted, connected, undirected graph with n vertices)
$T :=$ a minimum-weight edge (with its two vertices)
for $i := 1$ **to** $n - 2$
    $e :=$ a minimum-weight edge incident to some vertex in T that won't form a simple circuit if added to T
    $T := T$ with e and its end points added
end for T is a minimum spanning tree of G

A similar algorithm that builds an MST by selecting the lightest edge at each step (not necessarily connected to each other) so long as it doesn't create a cycle is **Kruskal's algorithm**.

procedure *kruskal*(G: weighted, connected, undirected graph with n vertices)
$T :=$ empty graph
for $i := 1$ **to** $n - 1$
    $e :=$ any edge in G with minimum weight that doesn't form a simple circuit when added to T
    $T := T$ with e and its end points added
end for T is a minimum spanning tree of G

In both Prim's and Kruskal's algorithm you may have more than one choice of a minimum edge to choose at any given step. The algorithms find minimum spanning trees whichever choice you make in such a situation, and there are sometimes more than one MSTs for a given graph. An extreme example is a graph with several spanning trees and with all edge weights equal (for example, weight 1).

Theorem: Prim's algorithm produces a minimum spanning tree of a connected, weighted, undirected graph.

Proof (by contradiction): Suppose that G is a connected, weighted, undirected graph with n vertices, and that Prim's algorithm produces $S = e_1, \ldots, e_{n-1}$ (together with vertices at their endpoints). Call the subgraph of S containing edges $e_1, \ldots, e_k$ S_k (so S_0 is an empty tree). Note that S_k is a tree for every $1 \leq k \leq n-1$, because of the way it is constructed. Suppose S is not a spanning tree. Then there is a maximum k such that S_k is contained in some spanning tree T. Since we are assuming that $S \neq T$, it must be the case that $k \neq n-1$. So T contains all the edges of S_k, but not e_{k+1}. Add E_{k+1} to T to form a new graph H. H has n edges (one more than T), so it must have a simple circuit (if it didn't, it would be a tree, and it has too many edges for that). The simple circuit contains e_{k+1}, since there was no simple circuit in T. There must also be an edge in this simple circuit that doesn't belong to S_{k+1}, since S_{k+1} has no circuits, by construction.

Start at an endpoint of e_{k+1} that is shared with S_k, and follow the circuit until you encounter the first edge (call it e) that is not in S_k. Delete this edge from H, and you obtain a tree (it has $n-1$ edges). Call it T'. T' contains S_{k+1}, and has weight $\leq T$, since (by Prim's algorithm) edge e_{k+1} was chosen instead of e, meaning that e_{k+1} had weight no more than e. But this contradicts the assumption that k is the maximum index (less than $n-1$) such that there is an MST containing S_k. Thus $k = n-1$ and S_{n-1} is a minimum spanning tree.

Here are the key steps in the proof:

1. Assume that S is not an MST, an assumption that we will try to use to lead to a contradiction.
2. Ask "where is the first edge in S where we went wrong?" Restate this as "what is the highest index k such that some MST contains $e_1, \ldots, e_k$?" Notice that k could be zero (**all** our edges are wrong), but k cannot be $n-1$, since then the only MST containing $e_1, \ldots, e_{n-1}$ would be S itself (which we've assumed is not an MST).
3. Let T be an MST that contains $e_1, \ldots, e_k$. Construct a new graph H by adding e_{k+1} to T.
4. Convince yourself that H contains a simple circuit (it has too many edges to be a tree). Convince yourself that it has a simple circuit containing e_{k+1} (no circuit existed before this edge was added).
5. Convince yourself that this simple circuit must contain an edge that is not in $e_1, \ldots, e_{k+1}$. Start at the endpoint shared by e_k and e_{k+1} and trace this circuit (away from e_{k+1} until you encounter the first edge that isn't in $e_1, \ldots, e_{k+1}$, call this edge e.
6. delete e from H (you're exchanging it for e_{k+1}), creating a new tree T'.
7. Show that T' has sum of edge weights $\leq$ than that of T, so it is an MST, yet it contains $e_1, \ldots, ek+1$, contradicting the assumption that the highest index is k.

You can prove Kruskal's algorithm with a similar approach, but in addition you have to show that you end up with a tree at the end.

By using the appropriate data structure for finding a minimum-weight edge at each step, Prim's algorithm can be implemented in $O(e \log v)$ complexity, whereas Kruskal can be implement with $O(e \log e)$ complexity. Thus Prim's algorithm is somewhat better, except in very sparse graphs (where $|E| \approx |V|$).

Week 11
Counting

Three fundamental counting techniques for finite sets can be manipulated in a surprising variety of ways:

1. $|A \times B| = |A| \times |B|$ (**product rule**).
2. If A and B are disjoint, $|A \cup B| = |A| + |B|$ (**sum rule**).
3. In general $|A \cup B| = |A| + |B| + |A \cap B|$ (**inclusion/exclusion**).

Most people are satisfied with finding these facts self-evident. For example, since $|A \times B|$ is the number of ordered pairs with the first component from A and the second from B, it seems clear that there are $|A| \times |B|$ of these. You could also demonstrate this with a bijection. If $|A| = n$ and $|B| = m$, then there is some bijection $f(A) \to \{1, \ldots, n\}$, or equivalently you can label the elements of $A = \{a_1, \ldots, a_n\}$. Similarly you can label the elements of $B = \{b_1, \ldots, b_m\}$. Now convince yourself that

$$g(a_i, b_j) = m(i-1) + j$$

... is a 1-to-1, onto function from $A \times B$ to $\{1, \ldots, mn\}$.

Similar bijections can be constructed for the sum rule and inclusion/exclusion rule, if desired.

Once you figure out how a problem can use one or more of these rules, you can solve many of them.

Dudley lock: A common combination lock has 60 possible positions, 0...59, and you must choose 3 positions in order. The total number of possible combinations is $|A \times A \times A|$, where $|A| = 60$, so there are 216,000 combinations. Or you can send mail to Dudley corporation.

Binary string of length n: Here $A = \{0, 1\}$ and we need to know $|A \times \cdots \times A| = 2 \times \cdots \times 2$ (n times), or 2^n.

Functions from A to B: If $|A| = n$, and $|B| = m$, then each function from A to B associates each element of A with an element of B. You can model the function as a set of ordered n-tuples, each element corresponding to the value of B that a particular value of A gets sent to. The number of such n-tuples is $|B \times \cdots \times B|$ (n times), or m^n.

One-to-one functions from A to B: You're out of luck if $|B| < |A|$ — two elements of A must be sent to some element of B. Assume that $|A| = n$ and $|B| = m$, and $m \geq n$. Count as before, but now we want the number of n-tuples where there are no repeat elements. We have our choice of m elements for the first position, $m - 1$ for the second, continuing down to $m - n + 1$ choices for the nth element. This is $m!/(m - n)!$.

Power set of A: How many subsets does A have (counting the empty set) if $|A| = n$? There's a one-to-one correspondence between subsets of A and a family of functions $f_i : A \to \{0, 1\}$, where $f_i(a_j)$ is 1 if a_j is in the subset corresponding to f_i, 0 otherwise. How many such functions can there be? The same as the number of binary strings of length n: 2^n. (You can also prove this fact using induction on the size of A, where the base case is A is empty).

C identifiers: An identifier in C is a string that can contain alphabetic characters, digits, or underscores. The first character must be alphabetic or an underscore. Only the first 8 characters matter in distinguishing identifiers. How many distinct identifiers are there? The first characters may take one of 53 values. Sum up the size of the dijoint sets of 1-, 2-, 3-, 4-, 5-, 6-, 7-, and 8- character identifiers:

$$53 + 53 \times 63 + 53 \times 63^2 + 53 \times 63^3 + 53 \times 63^4 + 53 \times 63^5 + 53 \times 63^6 + 53 \times 63^7 \approx 2.1 \times 10^{14}.$$

Binary strings beginning with 1 or ending with 11, of length 7: Count both sets separately: there are 2^6 of the first sort and 2^5 of the second. If we add these, we'll have counted strings that are in both sets twice, so we subtract the 2^4 strings that are in both:

$$2^6 + 2^5 - 2^4 = 80.$$

Pigeonhole Principle

Assume that the number of pigeons and holes for them to roost in is finite. If there are $k + 1$ to go into k holes, then there must be a hole that gets 2 or more pigeons. Pigeons come in integer increments, and we resist the idea of putting $(k + 1)/k$ pigeons in each hole.

Pigeonhole Principle: If there are at least $k + 1$ pigeons to be placed in k holes, then there is at least one hole with two or more pigeons.

The art of using the Pigeonhole Principle is deciding what are your pigeons and what are your holes. The same idea works to show that 3, 4, 5, ... or more pigeons must be in some hole:

Claim: Every integer n has a multiple consisting entirely of 1s and 0s.

Proof: Write down the numbers $1, 11, 111, \ldots, \underbrace{1111\cdots11}_{n+1 \text{ times}}$. There are $n + 1$ numbers in this list, and there are only n possible remainders after division by n: $\{0, 1, \ldots, n-1\}$. Our pigeons are numbers in the list, the holes are remainders after division by n, so two of the numbers must have the same remainder after division by n. Subtract these two numbers (yielding a number made up entirely of 1s and 0s), and you have a number divisible by n.

Generalized Pigeonhole Principle: If N pigeons are placed in k holes, then there is at least one hole containing $\lceil N/k \rceil$ pigeons or more.

Proof: Use the definition of ceiling to show that $\lceil N/k \rceil < (N/k) + 1$. Then if all the holes had less than $\lceil N/k \rceil$ pigeons, there would be a total of

$$k\left(\left\lceil \frac{n}{k} \right\rceil - 1\right) < k\left(\left(\frac{N}{k} + 1\right) - 1\right) = N.$$

This contradicts the fact that we've assumed we have N pigeons.

Claim: Assume that no human has more than 300,185 hairs on their head (not including facial hair), and that there are 2,100,000 people in Toronto. Then there are 7 people in Toronto with exactly the same number of hairs on their head.

Question: How many cards must be selected from a deck in order to be sure there is three-of-a-kind?

Claim: Among $n + 1$ positive integers smaller than $2n$ there is at least one that divides another.

Proof: Decompose the $n+1$ integers into the highest power of 2 that divides them and an odd factor, so now we have a set $\{2^{k_1}q_1, \ldots, 2^{k_{n+1}}q_{n+1}\}$. The odd numbers $q_1, \ldots, q_{n+1}$ must be in the set $\{1, 3, 5, \ldots, 2n-1\}$, and there are n numbers from this set. Thus there are two numbers $2^{k_i}q_i$ and $2^{k_j}q_j$ where $q_i = q_j$. Choose the smaller of k_i and k_j, and the corresponding number divides the other.

Claim: In any group of 6 people there are either 3 mutual strangers or 3 mutual acquaintences (or possibly both).

Proof: Choose one of the six people. This person has either at least 3 acquaintences or 3 strangers among the other 5. WLOG they have 3 acquaintences. If any of the 3 acquaintences are acquainted with each other, then we have three mutual acquaintences. Otherwise we have 3 mutual strangers. This situation is sometimes expressed as $R(3,3) \leq 6$. $R(m,n)$ (the Ramsey number for (m,n)) is the minimum number of people such that either at least m are acquaintences or n are strangers. In fact, you can show that $R(3,3) = 6$ by constructing a situation where among 5 people the number of mutual acquaintences or strangers never exceeds 2. In general it's quite hard to find $R(m,n)$ where $3 \leq m \leq n$ (only 9 are known). For example $43 \leq R(5,5) \leq 49$, but no improvement on this bound has been shown.

Claim: Suppose 10 million Canadians earned less than 100,000 dollars last year, but more than nothing. Show that at least two of them earned exactly the same amount, to the penny.

Question: How many numbers do you need to select from the set $\{1, 3, 5, \ldots, 4n-3, 4n-1\}$ to ensure that you have at least one pair whose sum is $4n$?

Claim: If $|A| > |B|$, show that there are no 1-to-1 functions from A to B.

Claim: Suppose there are at least two people at a party. Then there are at least two people who know the same number of people.

Permutations and combinations

An ordered arrangement of r distinct elements from a set of n elements is denoted $P(n,r)$. In the extreme case, $P(n,n) = n!$. Calculating this number, using the product rule, gives n choices for the first element, $n-1$ choices for the second, ..., until we reach $n-r+1$ choices for the rth element, or

$$P(n,r) = n(n-1)\cdots(n-r+1) = \frac{n!}{(n-r)!}.$$

Question: How many circular arrangements of $ABCDEFGH$ are there? Two arrangements are considered the same if, once you find A, the sequences are the same reading clockwise .

Solution: Fix the position of A. Now there are 7 choices for the character counter-clockwise to its right, 6 choices for the character to the right of that, ..., or $7! = 5040$ circular arrangements in all.

Question: How many circular arrangements of $ABCDEFGH$ are there if you require that ADG occurs in the sequence (in counterclockwise order)?

Solution: Treat ADG as a single character, and solve as before: 5! or 120 circular arrangements.

Question: How many circular arrangements of $ABCDEFGH$ are there if you require that ADG **may not** occur in the sequence?

Solution: Subtract the number of prohibited arrangements from the total: $7! - 5! = 4920$.

An r combination of elements of a set is simply a subset (unordered, by definition) of size r. Denote the number of r-combinations from a set of size r as $C(n, r)$, sometimes denote $\binom{n}{r}$ — the binomial coefficient. The quantity can be calculated by noting that if we consider all the r-permuatations of the r-combinations, we generate all r-permutations of n elements or $P(n, r)$, so

$$P(r, r)C(n, r) = P(n, r) = \frac{n!}{(n-r)!} \Longrightarrow C(n, r) = \frac{P(n, r)}{r!} = \frac{n!}{r!(n-r)!}.$$

Notice the symmetry: $C(n, r) = C(n, n-r)$. This follows from the expression itself, by you can also reason independently: For each subset of size r that we select there is exactly one subset of size $n-r$ that we are deselecting. So the number of subsets of size r is the same as the number of subsets of size $n-r$. This proves that both expressions are counting the same objects in different ways, and thus must have the same value — a *combinatorial proof.*

Question: How many subsets of size two are there of the set $\{a, b, c, d, e\}$?

Solution: $C(5, 2) = C(5, 3) = 10$. For each subset of size 2 of $\{a, b, c, d, e\}$ there's a corresponding subset of size 1 that's been excluded.

Claim: $C(n, 0) + C(n, 1) + \cdots + C(n, n) = 2^n$.

Proof: This result could be solved using induction, with the base case $n = 0$ (the empty set has a single subset: itself). Another approach is to notice that every subset of a set of n elements corresponds to an ordered binary n-tuple that contains a 1 at position i if the ith element is in the subset, and a zero otherwise.

Question: How many ways are there for eight men and five women to stand in a row so that no two women are adjacent? How about eight indistinguishable men dressed in blue, and five indististinguishable women dressed in green?

Solution: In the first situation, first permute the men in 8! ways. For each of these permutations there are 9 inter-men "slots" to for 5 women to choose from, so $8! \times P(9, 5) = 609638400$ arrangements. In the second situation, there's only one way for the men to arrange themselves (we can't tell the arrangements apart). There are 9 inter-men "slots" for the women to distribute themselves among in $C(9, 5) = 126$ ways.

Question:] How many six-lowercase-letter strings (from our alphabet) contain:

1. The letter a?
2. The letters a and b?
3. a and b in consecutive positions, with all the letters distinct (a must precede b).
4. a occurs before b, all letters distinct.

Solution: Some solutions use permutations and combinations, some don't:

1. There are 26^6 strings of lowercase letters, 25^6 don't include a, so $26^6 - 25^6 = 64775151$ include an a.
2. There are 26^6 strings of lowercase letters, 25^6 don't contain a, 25^6 don't contain b, and 24^6 contain neither a nor b. So $26^6 - (2 \times 25^6 - 24^6) = 11737502$ strings contain both a and b
3. Glue a and b together. The four other letters can be arranged in $P(24, 4)$ ways, and then the remaining superletter can be inserted into 5 slots: $5 \times P(24, 4) = 1275120$ ways.
4. From six slots, select 2 for a and b in $C(6, 2)$ ways. Letters for the remaining 4 slots can be arranged in $P(24, 4)$ ways, for a total of $C(6, 2) \times P(24, 4) = 3825260$.

Week 12
Binomial theorem

The number $C(n, r)$ is also denoted $\binom{n}{r}$, for the binomial coefficient. The motivation for this notation is that the coefficients of the nth power of a binomial are related in the following way:

Binomial Theorem:

$$(x + y)^n = \sum_{r=0}^{n} \binom{n}{r} x^{n-r} y^r.$$

Proof: You could prove this by induction on n, however a combinatorial proof is probably more instructive. The product $(x + y)^n$ will have terms of the form $x^{n-r}y^r$, for r running from 0 to n. Each such term is formed by choosing r ys and $n - r$ xs from the factors of the form $(x + y)$. This can be done in $\binom{n}{r}$ ways.

This can be used in wide-differing applications:

Question: Expand $(x + y)^8$.

Claim: $\sum_{k=0}^{n} \binom{n}{k} = 2^n$.

Proof: This is (yet) another way of providing that the number of subsets of a set with n members is 2^n. Now use the binomial theorem with $x = y = 1$.

Claim: $(a + 1)^n = \sum_{r=0}^{n} \binom{n}{r} a^n$.

Proof: Binomial Theorem. Consider the two ways we had of calculating how many strings of length 6 had at least one a in them.

Claim: $\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0$.

There are many identities satisfied by binomial coefficients (and tons of them are collected in Pascal's Triangle). Here's one that helps us construct binomial coefficients by addition:

Pascal's Identity: Suppose n and k are positive integers, with $n \geq k$. Then

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}.$$

proof By definition $\binom{n+1}{k}$ is the number of subsets of size k from a set of size $n + 1$. Let x be a particular element of a subset with $n + 1$ elements. Then each subset of size k either includes x or it doesn't. Those that don't include x are chosen from among n elements, so there are $\binom{n}{k}$ of them. Those that do include x choose the other $k - 1$ elements from among n, so there are $\binom{n}{k-1}$ of them. The sum proves the claim.

You could also prove this by induction on the formula for $C(n, r)$, but that would be messy.

Vandermonde's Identity: Suppose m, n, and r are nonnegative integers with r no greater than either m or n. Then

$$\binom{m+n}{r} = \sum_{k=0}^{r} \binom{m}{r-k}\binom{n}{k}.$$

Proof: The left-hand side tells us how many ways there are to choose size r subsets of a set with $m+n$ elements. The right-hand side tells us the same thing (counting in a different way). For each subset of size $r-k$ we choose from a set of size m, we choose a subset of size k from a set of size n, and the product rule gives us the right-hand side.

Claim: $\binom{2n}{n} = \sum_{k=0}^{n} \binom{n}{k}^2$.

Proof: Vandermonde's Identity, with $m = n$.

Question: How many subsets of size 2 are there of the union of $\{a, b, c\}$ and $\{d, e, f\}$?

Solution: Use Vandermonde's Identity with $m = n = 3$ and $r = 2$.

More counting techniques

From a few simple ideas we've built up some powerful counting techniques. However some problems don't fit into these molds. First consider what happens to our count of combinations when we allow repetitions:

Question: How many different amounts of cash can you make by choosing coins from 1 penny, 1 nickle, 1 dime, and 1 loony?

Solution: Since each coin can be either in or out of the amount, the different amounts correspond to writing either a zero or a 1 on each coin. Hence 2^4 amounts

Question: If you can choose 3 coins from the denominations penny, nickle, dime, looney, how many different collections of coins can you have (there are at least 3 coins of each denomination available)?

Solution: You could begin listing 3 pennies, 2 pennies and a nickle, 2 pennies and a dime, 2 pennies and a looney, ... but that gets tiresome. Since there are 4 categories, you can model them as 3 bars (or walls) separating the 3 cs (for coins) into categories. From the $3+3$ possible positions (cs and bars), once you've chosen the position of the 3 c's, you have specified one choice. This means there are $C(6,3)$ ways to choose 3 coins from the 4 categories.

This exercise leads to the result:

Claim: There are $C(n+r-1, r)$ r-combinations from a set with n elements when repetition is allowed.

Proof: Model each r-combination of a set with n elements as a list of r cs and $n-1$ bars. From these $n+r-1$ positions, each r-combination corresponds to choosing r positions for the cs.

Question: How many solutions does

$$x_1 + x_2 + x_3 + x_4 = 15$$

have if x_1, x_2, x_3, x_4 are all nonnegative integers?

Solution: This problem corresponds to distributing 15 1s into 4 boxes labelled x_1, x_2, x_3, x_4. This can be done in $C(15+3, 15)$ ways: $C(18, 15) = C(18, 3) = 18(17)(16)/6 = 816$.

Now consider what happens to our count of permutations of n elements when some repeat.

Question: How many different ways are there to arrange the letters in $MISSISSIPPI$?

Solution: There are 4 Ss, 4 Is, 2 Ps, and 1 M, for 11 characters in all. From the 11 available positions, you can choose 4 for the Ss in $C(11,4)$ ways. From the remaining 7 slots, you can choose 4 for the Is in $C(7,4)$ ways. From the remaining 3 positions, you can choose 2 for the Ps in $C(3,2)$ ways. And from the last available position, you can choose a position for the M in $C(1,1)$ way, yielding:

$$C(11,4) \times C(7,4) \times C(3,2) \times C(1,1) = \frac{11!}{4!7!} \times \frac{7!}{4!3!} \times \frac{3!}{2!1!} \times \frac{1!}{1!0!} = \frac{11!}{4!4!2!1!}.$$

Claim: The number of different permutations of n objects when there are n_1 indistinguishable objects of type 1, n_2 indistinguishable objects of type 2, ..., n_k indistinguishable objects of type k is

$$C(n,n_1)C(n-n_1,n2),\cdots C(n-n_1-n_2\cdots-n_{k-1},n_k) = \frac{n!}{n_1!n2!\cdots n_k!}.$$

Proof: The positions for the objects of type 1 can be chosen in $C(n,n_1)$ ways. For each of these, the positions for the objects of type 2 can be chosen in $C(n-n_1,n_2)$ ways. The product rule yields the result, and cancellation in the expression for the ratio of factorials yields the expression on the right.

Question: How many ways are there to distribute 5 cards to each of 4 players from a standard deck with 52 cards?

Solution: Line up the 52 cards in some order from left to right. Now create another deck with 5 cards marked "hand 1," 5 cards marked "hand 2," 5 cards marked "hand 3," 5 cards marked "hand 4," and 32 cards marked "rest of deck." If you permute this second deck, and line up the cards with the original deck, it corresponds to a way of distributing 4 hands. The number of ways you can permute your specially made-up deck is

$$\frac{52!}{5!5!5!5!32!},$$

... according to the claim above

Claim: In how many ways can you distribute n distinguishable objects into k distinguishable boxes so that n_i objects go into box i, where $i = 1, 2, \ldots, k$?

$$\frac{n!}{n_1!n_2!\cdots n_k!}.$$

Proof: Line up the n objects in some order. Now create another collection of n items with n_i of type i. Permute this second collection in $n!/(n_1!n_2!\cdots n_k!)$ ways, each permutation corresponds to a distribution of the original n items into boxes.

Inclusion/exclusion

We've already seen that:

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

This result can be extended to three sets:

$$\begin{aligned} |A \cup B \cup C| &= |A \cup B| + |C| - |(A \cup B) \cap C| \\ &= |A| + |B| + |C| - |A \cap B| - |(A \cap C) \cup (B \cap C)| \\ &= |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|. \end{aligned}$$

There are two patterns to notice here. For the union of two sets our formula has $2^2 - 1$ terms, and for the union of three sets our formula has $2^3 - 1$ terms. Another pattern is the alternating sum: add intersections of 1 set, subtract intersections of two sets, add intersections of three sets... Can this be generalized?

Claim: Suppose $A_1, \ldots, A_n$ are finite sets.

$$|A_1 \cup \cdots \cup A_n| = \sum_{1 \le i \le n} |A_i| - \sum_{1 \le i < j \le n} |A_i \cap A_j| + \cdots + (-1)^{n+1} |A_1 \cap \cdots \cap A_n|.$$

Proof: The left hand side counts every element of the union of the n sets. Suppose x is an element of the union $A_1 \cup \cdots \cup A_n$. We'll check that the right-hand side counts x exactly once (no more, no less). x must belong to $1 \le r \le n$ of the sets $A_1, \ldots, A_n$. The first summation counts x $C(r,1)$ times, the second summation subtracts counts of x $C(r,2)$ times, the third summation counts x $C(r,3)$ times, yielding:

$$C(r,1) - C(r,2) + \cdots + (-1)^{r+1} C(r,r)$$

...times. Since the alternating sum $\sum_{k=0}^{r} (-1)^k C(r,k)$ is zero (proved previously), the sum in question equals $C(r,0) = 1$. So x is counted exactly once, and the right-hand side counts every element of the union exactly once. Also notice that the expression in the claim has $\sum_{k=1}^{n} C(n,k) = 2^n - 1$ terms, fitting the pattern we already noticed.

Question: How many solutions of $x_1 + x_2 + x_3 = 11$ have $x_1 \le 3$, $x_2 \le 4$, and $x_3 \le 6$, and all the x_i are nonnegative integers?

Solution: We already know (walls and balls) how to find the number of solutions when the only restriction is that the x_i are nonnegative integers: $C(13,2) = 78$. Suppose A is the set of solutions where $x_1 > 3$, B is the set of solutions where $x_2 > 4$, and C is the set of solutions where $x_3 > 6$. If we subtract $|A \cup B \cup C|$ from 78, we'll have our answer.

Using our previous technique, we can see that $|A| = C(9,2) = 36$, $|B| = C(8,2) = 28$, and $|C| = C(6,2) = 15$. Similarly, $|A \cap B| = C(4,2) = 6$, $|A \cap C| = C(2,2) = 1$, and all the other intersections are empty, so

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C| = 36 + 28 + 15 - 6 - 1 = 72$$

$78 - 72 = 6$, so there are 6 solutions satisfying the constraints.

Counting summarized

You've seen a number of counting techniques, and you need to decide how to cpply them to a particular problem. You need to ask:

- Arrangements versus selections (permutations versus combinations)?
- Repetitions allowed or not?
- Is it easier to count something or its complement?
- Does this problem resemble something I've seen before?

Sketch solutions to these:

1. Find the number of 8-letter strings that can be formed using all 8 letters of $MONTREAL$
   (a) no restrictions
   Solution: eight distinct characters can be permuted in 8! ways.
   (b) first letter must be M
   Solution: Once the first letter has been fixed at the beginning, there are 7! permutations of the other 7 characters.
   (c) vowels together
   Solution: Glue OEA into a super-letter, and permute the resulting six letters in 6! ways. However, for each of these arrangements there are 3! arrangements of the three vowels, so 6!3! in all.

(d) MN separate

Solution: Remove M and N and permute the remaining 6 letters in 6! ways. There are 7 gaps in which to place M and N (before, between, and after the other six letters) and you can select one for the M, and N in $P(7,2)$ ways. Altogether, $P(7,2) \times 6!$.

Alternatively, count the number of arrangements where MN are together, and subtract this number from 8!.

(e) not all vowels together

Solution: Subtract the solution for vowels together from 8!.

(f) vowels all apart

Solution: Permute the non-vowels in 5! ways. There are 6 gaps before, between, and after these letters, and you can assign three of these two the three vowels in $P(6,3)$ ways. So, altogether $P(6,3) \times 5!$ arrangements.

Week 13
More counting examples

1. Find the number of arrangements of 4 women and 6 men in a row

   (a) no restriction

   Solution: Permute ten distinct objects in 10! ways.

   (b) women can never be adjacent

   Solution: Permute the six men in 6! ways. There are now seven gaps before, between, and after the men, and these can be assigned to the four women in $P(7,4)$ ways, for a total of $6! \times P(7,4)$ arrangements.

   (c) men indistinguishable, women indistinguishable

   Solution: From ten available slots, choose four for the women in $C(10,4)$ ways. Equivalently, choose six for the men in $C(10,6)$ ways.

   (d) as previous, but women may not be adjacent

   Solution: Place the men with a space between each. There are seven spaces, and you can choose the four that will be occupied by women in $C(7,4)$ ways.

2. Find the number of committees of size 4 that can be formed from 6 women and 4 men

   (a) no restriction

   (b) equal number of men and women

   (c) one or more women

   (d) Gordon and Nina won't serve together

3. 20 coloured light of which 4 are defective. How many ways are there to select 8 lights so that at least 6 work?

4. Find the number of 5-card poker hands that contain

   (a) one pair (at least)

   (b) 2 pairs

   (c) a flush (all same suit)

   (d) a full house (a pair plus three-of-a-kind)

5. Number of four-letter "words" from $TORONTONIAN$.

6. From the digits $0, 1, 2, 3, 4, 5, 6$ how many nonnegative numbers less than 5000 can be formed?

(a) If repetitions are allowed? (cases)

(b) No digit may be repeated?

(c) Odd, with no digits repeated? (leave with them)

Probability

Terminology:

Experiment: A process with a finite set of outcomes: tossing a coin, rolling a die, having kids.

Sample space: Set of all possible outcomes of an experiment.

Event: Set of outcomes: e.g. (from one roll of a die) rolling an even number, rolling more than 9, rolling a 5, rolling over 10 (empty set)

Probability: A number associated with an event E that satisfies $0 \leq P(E) \leq 1$ and the sum of the probabilities of all outcomes is 1. Probability reflects the strength of our belief that an event will occur.

Equally likely events: $P(E_1) = P(E_2)$.

Impossible event: $P(E) = 0$.

Certain Event: $P(E) = 1$.

Probability with equally likely outcomes: If we believe all outcomes are equally likely, then $P(E) = |E|/|S|$.

- tossing a coin
- tossing a coin 3 times
- tossing a pair of dice (contrast outcomes and events)

If we assume all outcomes are equally likely, then we simply calculate the size of the sample space, $|S|$, and the size of the favourable event, $|E|$, and $P(E) = |E|/|S|$. Often the sample space can be listed, or drawn as a tree.

Sums of probabilities: $P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2)$. Example: E_1 is the set of all three-coin tosses that contain exactly two heads, and E_2 is the set of all three-coin tosses that begin with a head.

Complementary Event: $P(E_1) = 1 - P(E_2)$.

Question: What is the probability that three coin tosses result in TTH (in that order)? What is the probability that three coin tosses result in two tails and a head (in any order)?

Solution: In the first case there are 8 equally likely outcomes, and our event $E = \{TTH\}$ is one of them, so the probability is $P(E) = 1/8$. In the second case our sample space is still 8 possible outcomes, but now our event is a set of 3 of them: $E = \{TTH, THT, HTT\}$, so $P(E) = 3/8$.

Question: What is the probability that a random binary string of length 10 contains exactly 2 ones?

Solution: There are $C(10, 2) = 45)$ ways to choose the positions of the two ones, and there are $2^{10} = 1024$ binary strings of length 10, so $P(E) = 45/1024$.

Events are collections of outcomes and may overlap or be disjoint. In addition, events may be correlated in the sense that the knowledge that E_1 has occurred may alter (increase or decrease) the probability that E_2 has occurred. Draw Venn diagrams to scale to make this tangible.

If the knowledge that E_1 has occurred has no effect on the probability that E_2 occurs, we say that E_1 and E_2 are **independent**. This can be expressed as saying that $P(E_2)$ is the same whether we consider the original sample space, S, or the restricted sample space E_1. In symbols, E_1 and E_2 are independent means:

$$\frac{|E_2|}{|S|} = P(E_2) = \frac{P(E_1 \cap E_2)}{P(E_1)} \Rightarrow P(E_1)P(E_2) = P(E_1 \cap E_2).$$

Notice that two events that have an empty intersection (mutually exclusive) are strongly **dependent**, unless one of them has probability zero.

Question: Are the events that three coin tosses result in an even number of heads, and three coin tosses result in an even number of tails dependent or independent?

Solution: Three coin tosses with an even number of heads is $E_1 = \{TTT, THH, HTH, HHT\}$, and three coin tosses with an even number of tails is $E_2 = \{HHH, HTT, THT, TTH\}$. The intersection of E_1 and E_2 is empty, so $P(E_1 \cap E_2) = 0$ which is not equal to $P(E_1)P(E_2) = 1/4$. Knowledge that an even number of tails occurred makes it unlikely (impossible, in fact) that an even number of heads occurred.

Question: Are the events that three coin tosses result in an even number of heads and three coin tosses start with a tail independent?

Solution: Three coin tosses beginning with a tail is $E_1 = \{TTT, TTH, THT, THH\}$, and three coin tosses with an even number of heads is $E_2 = \{TTT, THH, HTH, HHT\}$. $E_1 \cap E_2 = \{TTT, THH\}$, so $P(E_1)P(E_2) = 1/4 = P(E_1 \cap E_2)$, and these events are independent. Knowledge that an even number of heads occurred doesn't change the probability that the three tosses begin with a tail.

Question: Are the events that a family with three children has at most one boy, and that a family with three children has at least one boy and at least one girl dependent or independent?

Solution: Assume that the eight possible outcomes, BBB, BBG, BGB, BGG, GBB, GBG, GGB, and GGG are equally likely. If E_1 is the set of all three-child family outcomes with at most one boy, then $P(E_1) = 4/8$. If E_2 is the set of all three-child family outcomes with at least one boy and at least one girl, then $P(E_2) = 6/8$. $E_1 \cap E_2$ are those family outcomes where there is exactly one boy, so $P(E_1 \cap E_2) = 3/8$. Since $(6/8)(4/8) = 3/8$, E_1 and E_2 are independent.

If you change the problem to two-child families, then $P(E_1)$ becomes $3/4$, $P(E_2)$ becomes $2/4$, and $P(E_1 \cap E_2)$ becomes $2/4$, which is different from $3/8$ ($P(E_1)P(E_2)$), so the events aren't independent.

Knowledge that one event occurs can alter the probability of another event occurring. Since events are sets of outcomes, knowing that an outcome is contained in E_2 may affect the likelihood that it is contained in E_1, since now our sample space changes from all possible outcomes to just E_2. We call this **conditional probability** the probability of E_1 given E_2, written $P(E_1|E_2)$, and it can be computed (assuming that $P(E_2) > 0$:

$$P(E_1|E_2) = \frac{|E_1 \cap E_2|}{|E_2|} = \frac{(1/|S|)|E_1 \cap E_2|}{(1/|S|)|E_2|} = \frac{P(E_1 \cap E_2)}{P(E_2)}.$$

Notice that E_1 and E_2 are independent is equivalent to $P(E_1) = P(E_1|E_2)$.

Question: What is the conditional probability that a two-child family has two boys, given that they have at least one boy?

Solution: There are four outcomes in the sample space, and only one of them has two boys, so $P(E_1) = 1/4 = P(E_1 \cap E_2)$ (every family that has two boys has at least one boy). Three of the outcomes have at least one boy, so $P(E_2) = 3/4$. So $P(E_1|E_2) = (1/4)/(3/4) = 1/3$. So the knowledge that there is at least one boy increases the probibility that there are 2 boys.

Monty Hall problem: With equal probability there's a prize behind one of three doors, the other two doors have something worthless. You tentatively pick a door (let's call it door 1). Monty Hall (cheesey game show host) reveals something worthless behind door 2 (he always reveals something behind one of the doors you didn't pick). Should you switch to door 3 or not?

Solution: A hurried analysis makes it looks as though the chances that the prize are behind door 1 or 3 are 50:50. However, the fact that Monty Hall always reveals something worthless behind a door you **didn't** pick changes the problem substantially, and your chances of find the prize are 2/3 if you switch to door 3.

One way of thinking of this is that rather then tentatively picking door 1, you are deciding to pick one of doors 2 or 3 (if you decide to switch). Initially the probability that the prize is behind one of those doors is 2/3, and when Monty reveals the junk, if the prizes was behind one of those two doors, you've got it by picking the other.

To make the calculation precise, call the event that the prize is behind door 1 D_1, similarly the event that the prize is behind door 2 is D_2, and the event that the prize is behind door 3 is D_3 — all of these have probability 1/3. The event that Monty shows door 1 (once you've tentatively chosen door 1) is S_1, the event that Monty shows door 2 is S_2, and the event that Monty shows door 3 is S_3.

Notice that $D_1 = \{D_1 \cap S_2, D_1 \cap S_3\}$, and $D_3 = \{D_3 \cap S_2\}$. Turning things around, $S_2 = \{D_1 \cap S_2, D_3 \cap S_2\}$. So, $P(S_2) = 1/3 + 1/6 = 1/2$. $P(D_1 \cap S_2) = 1/6$, so $P(D_1|S_2) = 1/3$. However, $P(D_3 \cap S_2) = P(D_3) = 1/3$, so $P(D_3|S_2) = 2/3$, and you should choose door 3!

Week 14
Recurrences

Some counting problems don't yield to the techniques we've looked at so far. Some of these unyielding problems end up having a structure where the nth element of a sequence is related to elements $0, \ldots, n-1$. For example

Rabbit breeding: Suppose you want to predict the number of breeding pairs of rabbits will be produced by an initial pair of rabbits. To simplify your calculations, you make the following assumptions

1. Once a pair of rabbits exists, they never die (they are immortal).
2. Each breeding pair of rabbits produces exactly one more breeding pair weekly, except that a breeding pair needs two week (to mature) before they produce their very first pair of offspring.

You take the following approach. You start allowing the rabbits to breed on a Friday, and you tally the results on subsequent Fridays. So, on Friday 0 you have $F_0 = 0$ breeding pairs, on Friday 1 you have $F_1 = 1$ breeding pair. On Friday 2 you still have the breeding pair from F_1 (these rabbits never die), but no new breeding pairs are born (since on F_0 there were none), so $F_2 = 1$. On Friday 3, you still have the old breeding pair from F_2, but in addition the breeding pair that existed on F_1 produces a pair of offspring, so $F_3 = 2$. You write down a general formula:

$$F_n = \begin{cases} 0 & n == 0 \\ 1 & n == 1 \\ F_{n-2} + F_{n-1} & \text{otherwise} \end{cases}$$

Notice that the formula for F_n uses values from $F_0, \ldots, F_{n-1}$, except in the initial cases where $n = 0$ or $n = 1$. This is called a **recurrence relation**, and (in this case) you can calculate F_n for any n with $O(n)$ steps.

Tower of Hanoi: A puzzle from the 1800s asked you to move a stack of n rings (stacked in decreasing order of width) one-at-a-time from one of three rings to another. You were never allowed to place a wider ring on a narrower one.

One (of many) approaches reasoned that before moving the largest of the n rings to their destination, you would need to move the smaller $n-1$ rings to an intermediate location, then move the largest ring to its destination, and then move the smaller $n-1$ rings from their intermediate location to the destination. In addition, it was clear that it took 1 move to get a stack of 1 rings from its initial location to its destination. So, if you call the number of moves required to move n rings H_n, this formula says you'll need

$$H_n = \begin{cases} 1 & n == 1 \\ 2H_{n-1} + 1 & \text{otherwise} \end{cases}$$

. . . to move a stack of n rings.

Counting binary strings: How many binary strings of length n are there that have never have two adjacent zeros? At first this sounds like something from Chapter 4, except that it asks for the number for a general n (not some particular n like 3 or 6). It doesn't seem to yield to the counting techniques of combinations or permutations, but it does have a recursive structure.

If you consider the empty string to be one string, then there is 1 string of length zero ("") with no repeated zeros. There are two strings of length 1 without repeated zeros: 1 and 0. Now call the number of binary strings without adjacent zeros of length n $BSWAZ_n$, and consider $n > 1$. These can be broken into two classes: those that end with a 1 and those that end with a 0. Those that end with a 1 can be formed by simply adding a 1 to every valid binary string of length $n-1$ that doesn't have adjacent 0s. Those that end with a zero must end with 10 in order to qualify, so wwe can form them by adding 10 to all the valid binary strings of length $n-2$. Counting these up we get $BSWAZ_n = BSWAZ_{n-2} + BSWAZn - 1$:

$$BSWAZ_n = \begin{cases} 1 & n = 0 \\ 2 & n = 1 \\ BSWAZ_{n-2} + BSWAZ_{n-1} & \text{otherwise} \end{cases}$$

This should look familiar.

How many parenthesizations?: Consider matrix multiplication, which is associative but not commutative. So (if they have the right dimensions) the matrix product $M_0M_1M_2$ can be multiplied in the following ways: $M_0(M_1M_2)$ or $(M_1M_1)M_2$. If there were four matrices, we'd have to consider 5 possible groupings. If we call the number of ways to group $n+1$ matrices G_n, then $G_0 = G_1 = 1$, and for larger n, we make the "top-level" division into two products between two matrices, then

$$G_n = G_0G_{n-1} + G_1G_{n-2} + \cdots + G_{n-1}G_0$$

These numbers grow quickly with n and the sequence $\{G_n\}$ is called the **Catalan numbers**, (which solves a number of counting problems).

Notice that our four examples give us a procedure for finding the nth element of a sequence, but not a closed form (a function or expression that does it in one step). We can reduce H_n to a closed form by repeatedly unwinding it until we see something that we recognize ($H_n = 2^n - 1$). The Catalan numbers yield to a high-powered tool called generating functions before they take on a (nearly familiar) closed form. The rabbits and binary strings yield to an approach that solves an entire class of problems.

Solving recurrence relations (closed form)

The recurrence relation for the number of moves in the Tower of Hanoi puzzle, $H_n = 2H_{n-1} + 1$ yields to an informal "unwinding" argument, so that $H_n = 2^n - 1$, for $n = 1, 2, 3, \ldots$. If the "hand-waving" portion

of the unwinding seems unconventional, use a straight-forward proof by induction with the base case $n = 1$ being just a move of a single ring, and the induction step requiring you to show that if $H_{n-1} = 2^{n-1} + 1$, then $H_n = 2^n + 1$. This is a **closed form**, basically an expression for H_n that takes a fixed number of steps using some familiar function.

Another recurrence relation we looked at for the number of ways to group (parenthesize) $n + 1$ matrices yielded the recurrence relation

$$G_n = G_0G_{n-1} + G_1G_{n-2} + \cdots + G_{n-1}G_0.$$

The sequence $\{G_n\}$, with $G_0 = G_1 = 1$ is called the **Catalan numbers**, and it yields to an approach using generating functions. The closed-form solution, $G_n = C(2n, n)/(n + 1)$ suggests some sort of combinatorial argument, but it's not obvious how to get it using combinatorial methods.

The third pattern is suggested by the Fibonacci sequence, $\{F_n\}$ (rabbits), and the bit strings without adjacent zeros, $\{B_n\}$. Although B_n looks as though it may have a combinatoric solution, it doesn't seem to save time compared to adding up terms from B_0 to B_n. Both B_n and F_n have the form:

$$a_n = a_{n-1} + a_{n-2}$$

... plus some initial conditions to establish a_0 and a_1. For different choices of initial conditions you get different sequences, and (a short induction proof will establish this rigorously) once you've specified a_0 and a_1 you have uniquely specified the entire sequence $\{a_n\}$. This is called a linear homogeneous recurrence relation of degree 2: linear because a_n is a sum of multiples of a_{n-1} and a_{n-2} each with exponent 1, homogeneous because only multiples of the a_i are on the right-hand side, and constant coefficients because the a_i are multiplied by constants that don't depend on n, degree 2 since the a_n depends on previous terms down to a_{n-2}. There is a general technique for solving (finding closed forms) for this sort of recurrence, which may remind you of solutions to differential equations.

First notice that if you have two solutions to a recurrence relation, linear combinations of those two solutions are also a solution. This means that if $\{s_n\}$ satisfies $s_n = s_{n-1} + s_{n-2}$ and $\{t_n\}$ satisfies $t_n = t_{n-1} + t_{n-2}$, then

$$f_n = as_n + bt_n = as_{n-1} + bt_{n-1} + as_{n-2} + bs_{n-2} = f_{n-1} + f_{n-2},$$

... is also a solution. Once some solutions, you can generate a whole family of solutions in this way. Each particular solution differs by its initial conditions, the value of a_0 and a_1.

After a bit of experimentation with F_n and B_n it seems plausible that the terms of each sequence behave like powers of some base that lies between 1 and 2. In the case of B_n, the first few terms are $1, 2, 3, 5, 8, 13, \ldots$ and it seems as though $1.5a_{n-1} \leq a_n \leq 2a_{n-1}$. Manipulating this guess a bit you can see that if the solution to $a_n = a_{n-1} + a_{n-2}$ were of the form $r^n = r^{n-1} + r^{n-2}$, a base r_0 that solved this would have to be a solution to $r^n - r^{n-1} - r^{n-2} = 0$, and dividing this by r^{n-2} you get a quadratic equation: $r^2 - r - 1 = 0$. This equation has two roots:

$$\frac{1 \pm \sqrt{5}}{2}.$$

Call these two roots r_1 and r_2. You can verify that each of them satisfy the recurrence relation $a_n = a_{n-1} + a_{n-2}$ if you set $a_n = r_i^n$, for $i = 1, 2$. Any linear combination $ar_1 + br_2$ is also a solution, so now we want to find the right values for a and b that satisfy initial conditions for, say, $\{F_n\}$ — $F_0 = 0$ and $F_1 = 1$, and satisfying

$$F_n = a\frac{1 + \sqrt{5}}{2} + b\frac{1 - \sqrt{5}}{2}.$$

This means that $F_0 = a + b = 0$, so $b = -a$, and $F_1 = 1 = ar_1 - ar_2 = 1$, or $a = 1/(r_1 - r_2) = 1/\sqrt{5}$, and $b = -1/\sqrt{5}$. This gives a closed form for F_n:

$$F_n = \frac{1}{\sqrt{5}}\left(\frac{1 + \sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1 - \sqrt{5}}{2}\right)^n.$$

This approach works so long as $r_1 \neq r_2$, so that you can divide by their difference. Once you've settled what a and b are, there is only one sequence that satisfies both the initial conditions and the general recurrence relation, so this must be the expression for F_n.

The same approach works for B_n, only now the initial conditions are different so you need to solve:

$$\begin{aligned} ar_1^0 + br_2^0 &= 1 \Longrightarrow b = 1 - a \\ ar_1 + br_2 &= 2 \Longrightarrow a = \frac{2 - r_2}{r_1 - r_2} = \frac{3 + \sqrt{5}}{2\sqrt{5}} \end{aligned}$$

$$b = \frac{\sqrt{5} - 3}{2\sqrt{5}},$$

$$B_n = \frac{3 + \sqrt{5}}{2\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^n - \frac{3 - \sqrt{5}}{2\sqrt{5}} \left(\frac{1 - \sqrt{5}}{2} \right)^n.$$

A completely different approach to calculating B_n yields

$$B_n = C(n+1, 0) + C(n, 1) + C(n-1, 2) + \cdots + C(n/2, n/2).$$

... so now you have a relationship between binomial coefficients and the fibonacci sequence.

General solution

The approach for solving F_n and B_n work for an arbitrary recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ with $a_0 = k_0$ and $a_1 = k_1$. If the characteristic equation:

$$r^2 - c_1 r - c_2 = 0$$

... has two distinct roots r_1 and r_2, then these are each solutions to the recurrence $a_n = c_1 a_{n-1} + c_2 a_{n-2}$, as is any linear combination $\alpha r_1 + \beta r_2$. If you can find α and β that satisfy the initial conditions k_0 and k_1, then this is the unique solution to the recurrence relation.

$$\begin{aligned} k_0 &= a_0 = \alpha r_1^0 + \beta r_2^0 \Longrightarrow \beta = k_0 - \alpha \\ k_1 &= a_1 = \alpha r_1 + (k_0 - \alpha) r^2 \Longrightarrow \alpha = \frac{k_1 - k_0 r_2}{r_1 - r_2} \\ & \beta = \frac{k_0 r_1 - k_0 r_2 - k_1 + k_0 r_2}{r_1 - r_2} = \frac{k_0 r_1 - k_1}{r_1 - r_2}. \end{aligned}$$

So long as $r_1 \neq r_2$ this solution exists and gives the unique solution to the recurrence relation. At no point in the argument do we depend on r_1, r_2, α, β being real numbers, so the solution is valid even if some of these are complex numbers.

Example: Suppose you have a $2 \times n$ grid to completely cover with tiles. You've got two types of tiles: 1×2 tiles and 2×2 tiles. For a given n in how many ways can you tile a $2 \times n$ grid?

Solution: Call the number of ways you can tile a $2 \times n$ grid T_n. $T_0 = 1$ (use exactly zero tiles...), and $T_1 = 1$ (use a single 1×2 tile). For $n > 1$ the nth column is filled by either one 1×2 tile, the tips of two 1×2 tiles, or the end of one 2×2 tiles. This suggests the recurrence relation

$$T_n = \begin{cases} 1 & n = 0 \\ 1 & n = 1 \\ T_{n-1} + 2T_{n-2} & \text{otherwise} \end{cases}$$

The characteristic equation is $r^2 - r - 2$, which has roots $r_1 = 2$, and $r_2 = -1$, so the family of solutions is $\alpha r_1^n + \beta r_2^n$. Solving for initial conditions $T_0 = 1$ and $T_1 = 1$ gives $\alpha = 2/3$ and $\beta = 1/3$, so the general solution is $a_n = (2/3)2^n + (1/3)(-1)^n = (1/3)(2^{n+1} + (-1)^n$.

This solution depends on dividing by $r_1 - r_2$, so the characteristic equation must have distinct roots. What happens if the characteristic equation has a single root of multiplicity 2? In other words, you've got a recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$, and the corresponding characteristic equation $r^2 - c_1 r - c_2 = 0$ has a single root r_0. Certainly multiples of r_0 will also be solutions to the recurrence of the form $\{ar_0^n\}$, using the same argument as before. However these won't generate enough solutions to be able to solve for every possible pair of initial conditions. It turns out (educated guess time again...) that a second solution is provided by $\{nr_0^n\}$.

To see that this is true, re-write the characteristic equation as $0 = r^2 - c_1 r - c_2 = (r - r_0)^2$, so that $r^2 - 2r_0 r + r_0^2 = r^2 - c_1 r - c_2$. By comparing coefficients, you can see that $c_1 = 2r_0$ and $c_2 = -r_0^2$. You want to verify that $nr_0^n = c_1(n-1)r_0^{n-1} + c_2(n-2)r^{n-2}$:

$$\begin{aligned} c_1(n-1)r_0^{n-1} + c_2(n-2)r_0^{n-2} &= (2r_0)(n-1)r_0^{n-1} - r_0^2(n-2)r_0^{n-2} \\ &= 2(n-1)r_0^n - (n-2)r_0^n \\ &= nr_0^n. \end{aligned}$$

By the same argument as before $a_n = \alpha r_0^n + \beta n r_0^n$ provides a family of solutions. For a given pair of initial conditions $a_0 = k_0$, and $a_1 = k_1$, if you can solve for α and β, you can exhibit the unique solution:

$$\begin{aligned} a_0 &= k_0 = \alpha \\ a_1 &= k_1 = k_0 r_0 + \beta r_0 \Longrightarrow \beta = \frac{k_1 - k_0 r_0}{r_0}. \end{aligned}$$

This solution assumes that $r_0 \neq 0$, which is certainly true so long as $c_2 \neq 0$ (true, since our recurrence has degree 2).

Example: Solve the recurrence $a_n = 4(a_{n-1} - a_{n-2})$ given initial conditions $a_0 = 1$ and $a_1 = 3$.

Solution: The corresponding characteristic equation is $r^2 - 4r + 4 = 0$. This has a single root, 2, so the general solution is $a_n = \alpha 2^n + \beta n 2^n$ for some α and β. With the initial conditions $a_0 = 1$ and $a_1 = 3$, we solve for $\alpha = 1$ and $\beta = 1/2$, so the formula is $a_n = 2^n + n2^{n-1}$.