Complexity in Computer Science

Discrete Math Background

Thomas Watson

February 2, 2026

Contents

Chapter A

Notation and Terminology

Math is like music. Musical notation is marks on a page that convey how to create certain experiences. The book *Complexity in Computer Science* contains "music" in the genre of computational complexity, and this chapter explains standard notation and terminology for conveying how to create these mental experiences.

A.1 Sets, tuples, and functions

A.1.1 Sets

A *set* is a collection of *elements* (also known as *members*), which may be any types of objects, even other sets. We write a set between braces. It doesn't matter what order we write the elements in or how many times we write an element—it only matters which elements are in the set and which aren't. For example,

$$\{1, \mathrm{a}, \{2, \mathrm{b}\}\} \;=\; \{\{\mathrm{b}, 2\}, \mathrm{a}, 1\} \;=\; \{\{2, 2, \mathrm{b}\}, 1, \mathrm{a}, \mathrm{a}\}$$

are three ways to represent the same set of three elements, and one these elements happens to be a set of two elements. Instead of explicitly listing all the elements, it's sometimes necessary or more convenient to specify a set using a description of its elements. For example,

$$\{x : x \text{ is an even integer and } 0 < x \leq 100\}$$

("the set of all x such that ...") is less cumbersome than writing all fifty elements, but it would also be fine to write $\{2, 4, 6, \ldots, 98, 100\}$ if the pattern is clear. The *size* (or *cardinality*) of a set S, denoted $|S|$, is the number of distinct elements it contains, which could be finite or infinite. The *empty set* $\emptyset = \{\}$ is the unique set of size zero. Here are some important sets of numbers:

$$\begin{aligned}
\text{the first } n \text{ positive integers } \quad [n] &= \{1, 2, 3, \ldots, n\} \\
\text{the naturals} \quad \mathbb{N} &= \{0, 1, 2, 3, \ldots\} \\
\text{the integers} \quad \mathbb{Z} &= \{\ldots, -2, -1, 0, 1, 2, \ldots\} \\
\text{the rationals} \quad \mathbb{Q} &= \{0, 1, \tfrac{1}{2}, \tfrac{3}{17}, -\tfrac{11}{6}, \ldots\} \\
\text{the reals} \quad \mathbb{R} &= \{0, 1, -\tfrac{11}{6}, \pi, \mathrm{e}, -\pi^2, \ldots\}
\end{aligned}$$

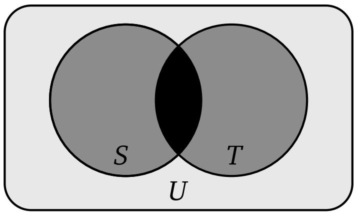
The notation $x \in S$ means x is an element of the set S. The notation $S \subseteq T$ means S is a *subset* of T (and T is a *superset* of S), which means all of S's elements are also elements of T.

(Each of the above sets of numbers is a subset of the next one.) For every set S, we have $S \subseteq S$ and $\emptyset \subseteq S$. Sets S and T are equal when they contain exactly the same elements as each other, which is equivalent to saying $S \subseteq T$ and $T \subseteq S$ both hold. The symbol $\notin$ means "is not an element of," and $\nsubseteq$ means "is not a subset of." If S and T are sets, the following operations produce new sets:

Operation	Notation	Definition
union	$S \cup T$	$= \{x : x \in S \text{ or } x \in T \text{ (or both)}\}$
intersection	$S \cap T$	$= \{x : x \in S \text{ and } x \in T\}$
difference	$S \smallsetminus T$	$= \{x : x \in S \text{ and } x \notin T\}$

If some "universe" set U is understood from context, and $S \subseteq U$, then the *complement* of S is denoted $\overline{S} = U \smallsetminus S$.

We imagine a set as a shape that encloses points representing the set's elements. This diagram illustrates that $S \cup T$ is the dark gray and black areas, $S \cap T$ is the black area, $S \smallsetminus T$ is the left dark gray area, and $\overline{S}$ is the light gray and right dark gray areas. The following handy facts say that if we pull a complement "through" a union or intersection, the operation flips to the other type:



- $\overline{S \cup T} = \overline{S} \cap \overline{T}$ (the light gray area).
  Intuitively, "x is in neither S nor T" is the same as "x is not in S and not in T."
- $\overline{S \cap T} = \overline{S} \cup \overline{T}$ (the light and dark gray areas).
  Intuitively, "x is not in both S and T" is the same as "x is either not in S or not in T."

Sets S and T are *disjoint* when they have no elements in common: $S \cap T = \emptyset$ (there's no black area). They *cover* the universe when all universe elements are in the union: $S \cup T = U$ (there's no light gray area). If S and T are disjoint and cover U, they form a *partition* of U, in which case every element is in exactly one of S or T (at most one since $S \cap T = \emptyset$, and at least one since $S \cup T = U$).

We can also partition a set U into more than two parts. When dealing with many subsets, say n of them, it's convenient to denote them $S_1, S_2, \ldots, S_n$ where the subscript i is the *index* of the set S_i. These subsets form a partition of U when both the following hold:

- They are pairwise disjoint: $S_i \cap S_j = \emptyset$ for all distinct indices $i \in [n]$ and $j \in [n]$.
- They cover the universe: $\bigcup_{i \in [n]} S_i = U$ (this notation means the union of all n subsets).

In this case, each element of U is in exactly one S_i subset (at most one by pairwise disjointness, and at least one by covering). A partition chops up U into pieces. Technically, to be considered a partition, each S_i should be nonempty, but this is just a quibble.

Perhaps confusingly, when talking about partitioning a computer's hard drive or memory, the word "partition" can refer to any one of the parts that the whole is divided into, but in math, "partition" refers to the overall decomposition, and the individual S_i subsets are called *parts*.

A.1.2 Tuples

A *tuple* is a finite sequence of elements where order and repetition matter. We write tuples between parentheses. For example:

$$(\mathrm{a},\mathrm{b},\mathrm{c}) \neq (\mathrm{c},\mathrm{a},\mathrm{b}) \neq (\mathrm{c},\mathrm{a},\mathrm{a},\mathrm{b})$$

A tuple is called an n-tuple when its *length* (number of components) is n. A 2-tuple is also called an *ordered pair*. The *cartesian product* of sets S and T is the set $S \times T = \big\{(x, y) : x \in S \text{ and } y \in T\big\}$ of all ordered pairs where the first component is an element of S and the second is an element of T. For example:

$$\begin{aligned}\{1,2\} \times \{\mathrm{a},\mathrm{b},\mathrm{c}\} \;=\; \big\{&(1,\mathrm{a}), (1,\mathrm{b}), (1,\mathrm{c}),\\ &(2,\mathrm{a}), (2,\mathrm{b}), (2,\mathrm{c})\big\}\end{aligned}$$

A standard deck of 52 playing cards is $[13] \times \{\spadesuit, \heartsuit, \clubsuit, \diamondsuit\}$ since each card has one of 13 possible "ranks" and one of 4 possible "suits." For example, ace of spades is $(1, \spadesuit)$, and king of diamonds is $(13, \diamondsuit)$.

A computer program can enumerate all elements of $S \times T$ using nested loops: The outer loop iterates over all elements $x \in S$, and the inner loop iterates over all elements $y \in T$, to enumerate each element $(x, y) \in S \times T$. There are $|S|$ options for the first component and $|T|$ options for the second component, so $|S \times T| = |S| \times |T|$ (assuming S and T are finite) where $\times$ on the left means cartesian product, and $\times$ on the right means multiplication of numbers. This is an example of overloading notation, like in programming languages where a single method name can refer to different methods depending on the types of the arguments.

The cartesian product of n sets $S_1 \times S_2 \times \cdots \times S_n$ is the set of all n-tuples where the i^{th} component comes from S_i, for each index $i \in [n]$. When taking cartesian products of a set with itself, we overload the "exponentiation" notation and write $S^2 = S \times S$ and $S^3 = S \times S \times S$, and so on. The set of all positive-length tuples whose components come from S is denoted S^+.

A *string* is a tuple written without the parentheses and commas, and where all components come from some finite *alphabet* set. For example, 0110100 is a string over the binary alphabet $\{0, 1\}$, and aacbc is a string over the alphabet $\{\mathrm{a},\mathrm{b},\mathrm{c}\}$ and over any superset such as $\{\mathrm{a},\mathrm{b},\mathrm{c},\mathrm{d}\}$—it doesn't matter that d wasn't used in this particular string. The length of a string x is denoted $|x|$ (overloading the "bars" notation for set size and for absolute value of a number). Over alphabet S, the set of all length-n strings is S^n, and the set of all positive-length strings is S^+, which we can partition into infinitely many parts according to length: $S^+ = S \cup S^2 \cup S^3 \cup \cdots$. The *empty string* (written "" in many programming languages) has length 0.

For a tuple or string x, the subscript notation x_i refers to the i^{th} component. For example, if $x = 110$ then $x_1 = x_2 = 1$ and $x_3 = 0$. Every $S \subseteq [n]$ has an associated *characteristic bit string* $x \in \{0,1\}^n$ where $x_i = 1$ if $i \in S$, and $x_i = 0$ if $i \notin S$ (so the locations of 1s indicate which elements are in S). For example, $\{2, 4, 5\} \subseteq [6]$ has characteristic bit string 010110.

If the alphabet has an ordering (as is naturally the case with numbers or letters), we can extend it to the *lexicographic order* of strings of any particular length: If $x \in S^n$ and $y \in S^n$ are distinct, then x precedes y when x_i precedes y_i (in the alphabet) for the lowest index i such that $x_i \neq y_i$. For example, babd comes before baca since the leftmost position where they disagree is the third, where b comes before c (the familiar alphabetical order). The lexicographic order of

$\{0,1\}^n$ is the numerical order if we interpret bit strings as binary (base 2) numbers 0 through $2^n - 1$, as we observe for $\{0,1\}^3$:

$$000,\ 001,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111$$

We sometimes deal with tuples of bit strings of a common length: $(\{0,1\}^m)^n$ is the set of all n-tuples whose components are bit strings of length m. An element's first subscript is the "outer index" (indicating which string), and the second subscript is the "inner index" (indicating which bit position within that string). For example, if $x = (110, 001, 100, 110)$ then $x_3 = 100$ (the 3rd string) and $x_{3,2} = 0$ (the 2nd bit of 100), while $x_2 = 001$ and $x_{2,3} = 1$. We sometimes write x_i as x^i and write $x_{i,j}$ as x^i_j. (The superscript is "outer", and the subscript is "inner.")

A.1.3 Functions

A *function* (or *mapping*) f lets us plug something in and get something out, by *evaluating* f. The notation $f\colon S \to T$ means $S \neq \emptyset$ is the *domain*—the set of things we can plug in—and $T \neq \emptyset$ is the *codomain*—the set of things allowed to come out. A function f associates to each element $x \in S$ (called a *preimage*) an element $f(x) \in T$ (called the *image*). Each element of S has exactly one image, but each element of T can have any number (zero or more) of preimages. We say f *maps* x to $f(x)$.

Example. This table defines a particular function $f\colon \{\mathrm{a},\mathrm{b},\mathrm{c}\} \to \{1,2,3\}$.

x	$f(x)$
a	3
b	1
c	3

The image of b is 1, and 3 has two preimages, and 2 has no preimages. ♦

(♦ marks the ends of examples.)

For each $y \in T$, we let $f^{-1}(y) = \{x \in S : f(x) = y\}$ denote the set of all preimages of y. These sets partition the domain S (ignoring codomain elements y with no preimages). An *injection* is a function where each codomain element has at most one preimage. A *surjection* is a function where each codomain element has at least one preimage. A *bijection* is a function that's both an injection and a surjection—each codomain element has exactly one preimage. The above example function is neither an injection nor a surjection.

By a programming analogy, we think of a preimage as an *argument*, and the image as the *(return) value*, and the domain and codomain as "types." But unlike in most programming languages, a function in math can't have "side effects"—we're guaranteed to get the same return value if we plug in the same argument again.

When the domain is a cartesian product of n sets, we may view a function as taking n arguments, rather than one argument that's an n-tuple: We write $f(x_1, x_2, \ldots, x_n)$ where x_i is the ith argument, rather than $f((x_1, x_2, \ldots, x_n))$ where the tuple $(x_1, x_2, \ldots, x_n)$ is the sole argument (which would be somewhat pedantic). Such a function is *n-ary*. We can represent a binary (2-ary) function with a 2-dimensional table where the rows are labeled with the possible first arguments, the columns are labeled with the possible second arguments, and each entry is the corresponding function value.

Example. A subtraction function $-\colon \{4,5,6\}\times\{1,2,3\}\to\{1,2,3,4,5\}$ (using $-$ as the function's name) would have this table:

$-$	1	2	3
4	3	2	1
5	4	3	2
6	5	4	3

We may think of binary functions as *operators* and write them with *infix* notation, such as $4-3$ instead of $-(4,3)$. ♦

We could represent n-ary functions with n-dimensional tables, but these aren't simple to visualize when $n>2$.

Composition is a way to combine functions. For any $f\colon S\to T$ and $g\colon R\to S$, the composed function $f\circ g\colon R\to T$ is defined by $(f\circ g)(x)=f(g(x))$, that is, applying g and then applying f to the result. For example, if $f\colon \mathbb{N}\to\mathbb{N}$ is $f(x)=x+1$, and $g\colon \mathbb{N}\to\mathbb{N}$ is $g(x)=2x$, then $(f\circ g)(x)=2x+1$ and $(g\circ f)(x)=2(x+1)$.

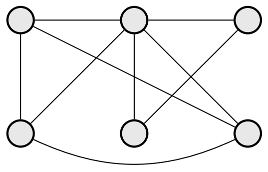
A *partial function* is allowed to be undefined for some arguments (so each domain element has at most one image, rather than exactly one). For example, the natural logarithm $\ln\colon \mathbb{R}\to\mathbb{R}$ is only defined on positive arguments. We call the original definition *total function* when we need to distinguish them. (A total function is a partial function that's defined on the whole domain.) Formally, for a partial function $f\colon S\to T$, there is a subset $S'\subseteq S$ and a total function $f'\colon S'\to T$ such that $f(x)=f'(x)$ if $x\in S'$, and $f(x)$ is undefined if $x\in S\setminus S'$. When we evaluate f, there's an implicit *promise* that the argument will come from S'.

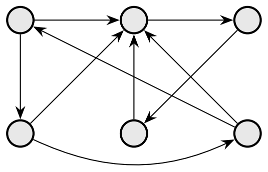
A.2 Graphs and trees

A.2.1 Graphs

Though "graph" commonly means a plot of a function $f\colon \mathbb{R}\to\mathbb{R}$, for us a *graph* is a discrete structure consisting of a finite set V of *nodes* (also known as *vertices*)—which we visualize as circles—and a finite set E of *edges* connecting pairs of nodes. There are two flavors: In an *undirected* graph, each edge is an unordered pair $\{u,v\}\subseteq V$, visualized as a line or arc joining two *endpoint* nodes u and v. In a *directed* graph, each edge is an ordered pair $(u,v)\in V^2$, visualized as an arrow from u to v.

Example.

Undirected graph: 

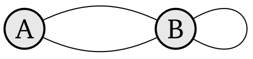
Directed graph:  ♦

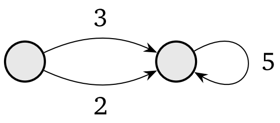
Graphs can model myriad situations. Here are a few examples.

- Edges may represent the ability to move data or physical objects from one place to another: In a computer network, nodes are hosts and routers, and edges are communication links (which may be directed or undirected depending on whether communication is one-way or bidirectional). In an airline network, nodes are airports, and directed edges are nonstop flight routes.
- Edges may represent relationships between entities: In a social network, nodes are people, and an undirected edge means two people are friends, and a directed edge means one is "following" the other. In the world wide web, nodes are webpages, and directed edges are hyperlinks.

By analogy with ice cream, we can add various "toppings," in any combination, to the definition of graphs. In the default versions, E is a set, so an edge $\{u,v\}$ or (u,v) is either present or not—there can't be multiple copies of it. If we want to allow multiple copies, we call them *multi-edges*. (For directed graphs, (u,v) and (v,u) are distinct edges, not copies of the same edge.) The default versions also forbid *self-loops*, which are edges that join a node to itself. (Formally, an undirected self-loop is $\{v\}$ and a directed self-loop is (v,v).) It's best to specify the exact combination of toppings when discussing graphs—we may wish to allow neither multi-edges nor self-loops, or one but not the other, or both (in which case a graph may have multi-self-loops). We might also let nodes and/or edges be annotated with labels or numbers.

Example. We illustrate multi-edges and self-loops in an undirected graph with labeled nodes (left) and in a directed graph with edge lengths (right).

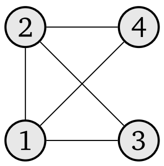


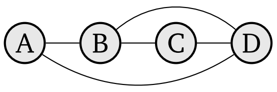


♦

Two graphs are *isomorphic* (to each other) when they have exactly the same structure, up to relabeling and rearranging the nodes. "Iso" means "same," and "morph" means "form."

Example. The following graphs are isomorphic, as shown by an *isomorphism* $f: \{1,2,3,4\} \to \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}\}$ specifying a relabeling.





v	$f(v)$
1	B
2	D
3	A
4	C

This f is not the only isomorphism between these graphs. ♦

If edge e has endpoints u and v, we say e is *incident* to u and to v, and we say u and v are *neighbors* and are *adjacent* to each other. If $e = (u,v)$ is a directed edge, we say u is an *inneighbor* of v, and v is an *outneighbor* of u.

In an undirected graph, the *degree* of a node v, denoted $\deg(v)$, is the number of edges incident to v (but a self-loop contributes 2 to the degree since looking "locally" around the node, we'd see two lines touching it). In a directed graph, the *indegree* indeg(v) is the number of incoming edges, and the *outdegree* outdeg(v) is the number of outgoing edges (so a self-loop contributes 1 to the indegree and 1 to the outdegree). In the example directed graph at the start of this section, the upper-middle node has indegree 4 and outdegree 1. A *source* is a node with indegree 0. A *sink* is a node with outdegree 0.

A *walk* in a graph goes from node to node to node (and so on) by following edges in the natural way. An undirected edge may be traversed in either direction, but a directed edge must be traversed in the direction the arrow points. Formally, a walk is a tuple of alternating nodes and edges $(v_0, e_1, v_1, e_2, \ldots, v_{k-1}, e_k, v_k)$ such that for each $i \in [k]$, $e_i = \{v_{i-1}, v_i\}$ in the undirected case, and $e_i = (v_{i-1}, v_i)$ in the directed case. A walk is a *trail* if no edge is reused: $e_i \neq e_j$ for all $i \neq j$. (If multi-edges are present, then each copy of an edge can be used once in a trail, since they are considered distinct edges.) A trail is a *path* if no node is reused: $v_i \neq v_j$ for all $i \neq j$. The *length* of a walk/trail/path is its number of edges.

Node v is *reachable* from node u when there exists a walk from u to v. In an undirected graph, if v is reachable from u then u is reachable from v by reversing the walk. In a directed graph, v might be reachable from u without u being reachable from v.

An undirected graph is *connected* when every two nodes are reachable from each other (so we can get from anywhere to anywhere else by following edges). A directed graph is *strongly connected* when every two nodes are reachable from each other. A directed graph might not be strongly connected even if the underlying undirected graph (obtained by erasing the arrowheads) is connected. The example directed graph at the start of this section is not strongly connected since, for example, there's no walk from the upper-right node to the lower-left node.

A disconnected undirected graph consists of multiple *connected components*. Node v's connected component contains all nodes reachable from v and all edges whose endpoints are reachable from v. This component is indeed connected since if u and w are both reachable from v, then they're reachable from each other: Concatenating a walk from u to v and a walk from v to w yields a walk from u to w (using only edges whose endpoints are reachable from v). Furthermore, the connected components partition the whole graph: If the connected components of nodes s and t aren't disjoint—say they have node v in common—then they're identical: Any node u that's reachable from s is also reachable from t (by walking from t to v to s to u) and vice versa. No edge goes between different connected components. Similarly, a directed graph's nodes are partitioned into strongly connected components, though edges may go between different strongly connected components.

A *closed walk*, *closed trail*, or *closed path* is like a walk, trail, or path (respectively) except it ends at the same node it started at. It need not have a designated "starting node," since it could be traversed starting from any of its nodes. A closed path is called a *cycle*, and returning to the starting node doesn't count as reusing that node.

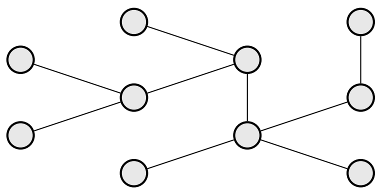
A directed graph with no cycles is a *directed acyclic graph*, or *dag* for short. Dags are useful for modeling dependencies:

- Nodes are jobs, and an edge means one must be finished before another begins.
- Nodes are processes in an operating system, and an edge means one holds a resource another is waiting for (so a cycle would indicate deadlock).
- Nodes are files in a build system (such as a makefile), and an edge means one must be

built before another.
- Nodes are commits in a version control repository, and an edge means one is an immediately preceding version of another (so indegree > 1 may represent a merge of branches).

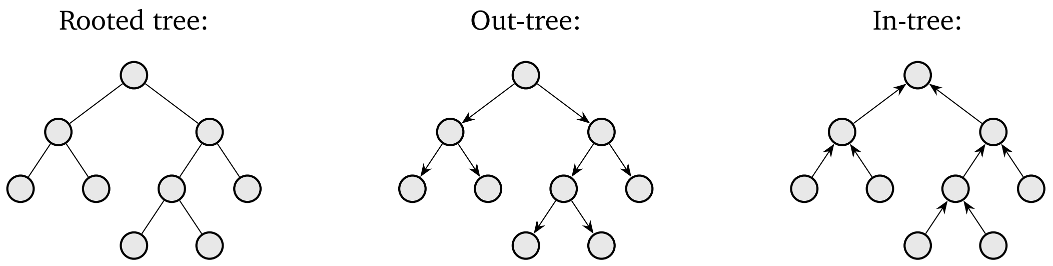
A.2.2 Trees

A connected undirected graph with no cycles is a *tree*. (In particular, a tree has no multi-edges or self-loops since those would be cycles of length 2 or 1.) An example appears to the right. Every tree is “barely connected”: Between any two nodes there is only one path, and deleting any edge would disconnect the graph. We prove these properties carefully in §B.6.1.



It’s common to designate one node as the *root*, in which case the tree is *rooted*. We typically draw the root at the top, with the rest of the tree dangling downward (unlike biological trees). We can turn a rooted tree into a directed acyclic graph by making all edges point away from the root (*out-tree*) or by making all edges point toward the root (*in-tree*).

Example.



♦

Nodes of degree 1 in a tree are *leaves*, and all other nodes in a tree are *internal* nodes. An exception is that when a rooted tree’s root has degree 1, we consider the root internal and not a leaf. In the degenerate case where a tree has only one node, it’s considered a leaf (and is also the root, if the tree is rooted).

In a rooted tree, an internal node’s *children* are its neighbors that are one step further from the root. Each non-root node has a unique *parent* neighbor that’s one step closer to the root (unlike real life, in which a person generally has two parents). We typically draw children below their parent. Two nodes are *siblings* if they have the same parent. A rooted tree is *binary* when every internal node has exactly two children. The previous example shows a binary tree.

Rooted trees can represent hierarchical relationships:

- If nodes are employees in a company, then parent could mean “boss” and child could mean “immediate subordinate,” with the root being the CEO.
- If nodes are classes in Java (or in any object-oriented language without multiple inheritance), then parent could mean “direct superclass” and child could mean “direct subclass,” with the root being the `Object` class.
- If nodes are directories in a simple file system, then parent/child relationships could indicate how directories are nested inside each other. (Unix-based operating systems allow

"hard links" to files, meaning the same file can be reached by multiple paths, in which case we wouldn't have a tree if we let files be nodes.)

If node u is on the unique path between node v and the root, then u is an *ancestor* of v, and v is a *descendant* of u. In other words, a node's ancestors are the nodes reachable by walking toward the root, and its descendants are the nodes reachable by walking away from the root. Each node is both an ancestor and a descendant of itself. The *subtree* rooted at node v consists of v's descendants and the edges connecting them: It's one of the two connected components formed by deleting the edge between v and its parent, assuming v is not the root.

A rooted tree is useful for organizing data because it describes a recursive way of partitioning the set of leaves. All leaves are in the root's subtree, and when we look at the root's children, the sets of leaves in their subtrees form a partition of all leaves. Then for any child of the root, its children's subtrees further partition its set of leaves, and so on. This process keeps *refining* the overall partition until the leaves are reached.

A.3 Logic

A.3.1 Propositional logic

A *proposition* is a statement that's unambiguously either true or false (though whether it's true or it's false might be unknown). *Boolean variables* such as P and Q represent arbitrary propositions with unspecified truth values (T for true, F for false). *Logical connectives* are functions—whose *truth tables* are shown below—for combining propositions to make new propositions.

	not / negation
P	$\neg P$
T	F
F	T

		and / conjunction	or / disjunction	xor / exclusive or	if then / implication	iff / if and only if
P	Q	$P \wedge Q$	$P \vee Q$	$P \oplus Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
T	T	T	T	F	T	T
T	F	F	T	T	F	F
F	T	F	T	T	T	F
F	F	F	F	F	T	T

For example, concerning a computer program running on a particular input, if P is "The program prints something (to the terminal)" and Q is "The program has a runtime error" then $P \vee Q$ means "The program prints something or has a runtime error, or both" and $P \Rightarrow Q$ means "If the program prints something, then it has a runtime error."

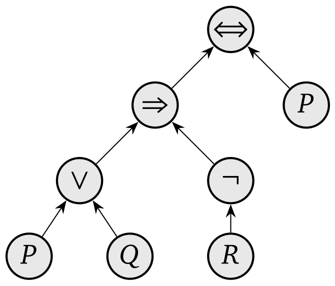
There are several ways to write $P \Rightarrow Q$ in English, including:

"if P then Q" "P implies Q" "P only if Q" "Q if P"

Also, "but" means "and" ($\wedge$) and "unless" means "or" ($\vee$).

We use these connectives to build boolean *formulas* such as $((P \vee Q) \Rightarrow \neg R) \Leftrightarrow P$. A formula corresponds to a rooted tree with structure determined by the parentheses, and with connectives labeling the internal nodes and variables labeling the leaves. A formula expresses a function $f\colon \{\mathrm{T},\mathrm{F}\}^n \to \{\mathrm{T},\mathrm{F}\}$ that takes a *truth assignment* (a tuple of truth values, one for each of the n variables in some fixed order) and returns the value obtained by plugging in that assignment.

Example. Here are the tree and truth table for the formula $((P \vee Q) \Rightarrow \neg R) \Leftrightarrow P$. The domain is in lexicographic order, where T comes before F.



P	Q	R	$((P \vee Q) \Rightarrow \neg R) \Leftrightarrow P$
T	T	T	F
T	T	F	T
T	F	T	F
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	F
F	F	F	F

♦

Two formulas are *equivalent* (denoted with $\equiv$) when they have the same truth table—that is, for every truth assignment, they yield the same truth value as each other. For example, $\neg(\neg P) \equiv P$. The connectives $\wedge$, $\vee$, $\oplus$, $\Leftrightarrow$ (all the binary ones defined above except $\Rightarrow$) are *commutative* (for example, $P \wedge Q \equiv Q \wedge P$) and *associative* (for example, $(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$), so we can write the variables in any order and omit the parentheses (for example, $R \wedge Q \wedge P$). We stress that $\equiv$ is not a logical connective—it's an assertion about two formulas.

A tedious way to check that two formulas are equivalent is to write their truth tables and check that all rows match. A potentially easier way is to transform one formula to the other by applying simple equivalences. Here are some common rules. Since $\neg$ has higher precedence than the binary connectives, $\neg P \vee Q$ means $(\neg P) \vee Q$.

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q \qquad P \Rightarrow Q \equiv \neg P \vee Q \qquad P \Leftrightarrow Q \equiv \neg(P \oplus Q)$$
$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q \qquad P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P \qquad P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)$$

Example. The top row rules show that our earlier formula $((P \vee Q) \Rightarrow \neg R) \Leftrightarrow P$ is equivalent to $\neg(((\neg P \wedge \neg Q) \vee \neg R) \oplus P)$. Using the top middle rule with $P \vee Q$ in place of the rule's P, and $\neg R$ in place of the rule's Q, creates a $\neg(P \vee Q)$ to which the top left rule applies. ♦

The left column is analogous to the rules $\overline{S \cup T} = \overline{S} \cap \overline{T}$ and $\overline{S \cap T} = \overline{S} \cup \overline{T}$ for sets. In fact, $\overline{P}$ is an alternative notation for $\neg P$, so we can write the rules as $\overline{P \vee Q} \equiv \overline{P} \wedge \overline{Q}$ and $\overline{P \wedge Q} \equiv \overline{P} \vee \overline{Q}$, and similarly when $\wedge$-ing or $\vee$-ing together more than two boolean variables.

We often use bits 1 and 0 in place of truth values T and F, and denote variables as $x_1, \ldots, x_n$. For example, plugging the assignment $x = 101$ into $(x_1 \wedge \overline{x_2} \wedge \overline{x_3}) \vee (x_1 \wedge x_2 \wedge x_3)$ yields 0. We extend the logical connectives to *bitwise operators* on bit strings (rather than individual bits) by operating on each coordinate separately. For example, $\overline{1010} = 0101$ and $1010 \wedge 0110 = 0010$.

A.3.2 Predicate logic

A *predicate* is like a proposition except it has one or more *free variables*, and the truth value may depend on the values of the free variables. For example, the predicate $P(x) =$ "$x^2 \geq 4$" is true for some values of x and false for others. A predicate is a function with codomain $\{\text{T}, \text{F}\}$. Even though the predicate "$x + 0 = x$" is true no matter what number x is, this is not considered a proposition, since it has a free variable.

One way to turn a predicate into a proposition is to plug in specific values for the free variables. Another way is to *bind* the free variables using *quantifiers*. The *universal* quantifier $\forall$ ("for all") asserts that every value of the variable makes the predicate true. The *existential* quantifier $\exists$ ("there exists") asserts that at least one value of the variable makes the predicate true.

- The *domain* matters. When we say $\forall x$ or $\exists x$, which possible values of x are we talking about? The values come from some domain set, which may be explicitly specified as in $\forall x \in \mathbb{R}$ (for all real numbers x) or implicitly understood from context. A quantified predicate's truth value may depend on the domain. For example, $(\forall x \in \mathbb{Z})\ (x \leq x^2)$ is true—every integer is at most its square—but $(\forall x \in \mathbb{R})\ (x \leq x^2)$ is false—the real value $x = 1/2$ is a *counterexample*.
- The *nesting* matters. Consider the predicate "$x < y$" with two free variables over domain $\mathbb{R}$. Binding y with $\exists$ creates the predicate "$\exists y\ (x < y)$" with one free variable, x. Then binding x with $\forall$ creates the proposition $\forall x\ \exists y\ (x < y)$ which is true—for every number there's a bigger number. We say $\exists y$ is nested inside $\forall x$. Binding in the other order, with $\forall x$ nested inside $\exists y$, creates the proposition $\exists y\ \forall x\ (x < y)$ which is false, because no individual number is bigger than all numbers. (The proposition even says y would need to be bigger than itself, since the value of y is not excluded from x's domain.)
- The *scope* matters. Using $\mathbb{N}$ as the domain, $\exists x\ ((x \text{ is even}) \wedge (x \text{ is odd}))$ is false—both occurrences of x in the predicate are bound to the same quantifier and so have the same value—but $(\exists x\ (x \text{ is even})) \wedge (\exists x\ (x \text{ is odd}))$ is true—each quantifier "declares" a new variable with a separate scope (section of the formula where the variable can be referred to), so this says some number is even and some (possibly different) number is odd. This is like programming languages allowing disjoint for-loops to declare loop counters with the same variable name, since the name goes out-of-scope after the first loop. We shouldn't write $\forall x\ (\cdots \exists x\ (\cdots x \cdots))$, as it would be ambiguous which variable the x in the inner scope refers to.

We can use quantifiers and logical connectives to build formulas out of placeholders such as $P(x)$ and $Q(y,z)$ representing arbitrary, unspecified predicates. Two such formulas are equivalent when they have the same truth value for every possible meaning of the placeholder predicates. For example, $\forall x\ \exists y\ P(x,y) \equiv \forall y\ \exists x\ P(y,x)$ since changing variable names doesn't affect the meaning.

Pulling a $\neg$ across a quantifier flips the quantifier's type:

$$\begin{aligned} \neg(\exists x\ P(x)) &\equiv \forall x\ (\neg P(x)) \\ \neg(\forall x\ P(x)) &\equiv \exists x\ (\neg P(x)) \end{aligned}$$

Intuitively, letting $P(x)$ be "person x voted":

$$\begin{aligned} \text{“nobody voted”} &= \text{“everybody didn't vote”} \\ \text{“not everybody voted”} &= \text{“somebody didn't vote”} \end{aligned}$$

This generalizes the rules $\overline{P \vee Q} \equiv \overline{P} \wedge \overline{Q}$ and $\overline{P \wedge Q} \equiv \overline{P} \vee \overline{Q}$, because $\forall x$ is just a big $\wedge$ over all values of x from the domain, and $\exists x$ is just a big $\vee$. For instance, $(\forall x \in \mathbb{N})\ P(x)$ means $P(0) \wedge P(1) \wedge P(2) \wedge \cdots$.

Example. With domain $\mathbb{R}$, $\neg\forall x\ \exists y\ (xy = 1)$ (not every number has a multiplicative inverse) is the same as $\exists x\ \forall y\ (xy \neq 1)$ (some number is such that every number fails to be a multiplicative inverse—witnessed by $x = 0$). To form the true proposition that every nonzero number has a multiplicative inverse, we could restrict x's domain to $\mathbb{R} \smallsetminus \{0\}$, or we could write $\forall x\ (x \neq 0 \Rightarrow \exists y\ (xy = 1))$. ♦

From our discussion of nesting, we can't always swap quantifiers of opposite types:

$$\forall x\ \exists y\ P(x,y) \not\equiv \exists y\ \forall x\ P(x,y)$$

We can always swap adjacent quantifiers of the same type having the same scope:

$$\forall x\ \forall y\ P(x,y) \equiv \forall y\ \forall x\ P(x,y) \qquad \text{and} \qquad \exists x\ \exists y\ P(x,y) \equiv \exists y\ \exists x\ P(x,y)$$

We can merge such quantifiers into a single one that introduces the pair (x,y) simultaneously, like it's a single variable with a cartesian product domain: $\forall(x,y)\ P(x,y)$ and $\exists(x,y)\ P(x,y)$.

We can't always swap alternating quantifiers, but we can sometimes swap them:

Example. If S and T are any nonempty finite sets of numbers, then $(\forall x \in S)\ (\exists y \in T)\ (x \leq y)$ has the same truth value as $(\exists y \in T)\ (\forall x \in S)\ (x \leq y)$ since both say $\max(S) \leq \max(T)$, that is, T's maximum value is at least as large as S's. Similarly, we can swap the quantifiers in $(\exists x \in S \cap T)\ (\forall y \in S \cup T)\ (x \leq y)$ since both ways say $\min(S) = \min(T)$. ♦

Logic with quantifiers provides an alternative way to express relationships among sets:

Example. If R, S, T are three subsets of some universe U:

- $R = S \cap T$ can be expressed as $(\forall x \in U)\ (x \in R \iff (x \in S\ \wedge\ x \in T))$.
- $R \not\subseteq S \smallsetminus T$ can be expressed as $(\exists x \in R)\ (x \notin S\ \vee\ x \in T)$. ♦

Supposing $P(n)$ is a predicate over natural numbers, here are some ways of expressing propositions concerning "how many values of n make $P(n)$ true":

- $P(n)$ holds for a unique (exactly one) value of n: $\exists n\ (P(n)\ \wedge\ (\forall m \neq n)\ \neg P(m))$ says that P is true for some value of the variable and false for all other values. In this expression, m's domain is $\mathbb{N} \smallsetminus \{n\}$, which depends on the value of n. Or, the expression can be an abbreviation for $\exists n\ (P(n)\ \wedge\ \forall m\ (m \neq n\ \Rightarrow\ \neg P(m)))$.
- $P(n)$ holds for infinitely many values of n: $\forall m\ (\exists n \geq m)\ P(n)$ says there's no biggest value for which P holds, since there's always a bigger value that makes P true, no matter how high we go. This can be an abbreviation for $\forall m\ \exists n\ (n \geq m\ \wedge\ P(n))$.
- $P(n)$ holds for all "large enough" values of n: $\exists m\ (\forall n \geq m)\ P(n)$ says that P is true for all values that are at least some threshold. This expression is equivalent to $\neg\forall m\ (\exists n \geq m)\ \neg P(n)$, meaning it is not the case that $P(n)$ is false for infinitely many values of n. In other words, P is true for all but finitely many values.

Throughout this section, we assumed formulas are *well-formed* ("make sense"). A non-well-formed formula contains a "syntax error" such as: improperly nested parentheses, a variable appearing outside the scope of the quantifier that declared it, a type mismatch (such as $P(\forall x\, Q(x))$ where $\forall x\, Q(x)$ has a boolean value but P might have a non-boolean domain), using a quantifier to bind something that's not a variable, and so on.

A.4 Growth of functions

A.4.1 Comparing growth rates

Given two functions on natural numbers, such as $f(n) = 100n^2$ and $g(n) = n^3$, which is "bigger"? This isn't the right question, since f is bigger for some values of n, and g is bigger for others. Instead we should ask which function "grows faster." For this example, g has another factor of n where f has a factor of 100, so g is bigger when $n > 100$. The ratio $g(n)/f(n) = n/100$ increases without bound as n goes to infinity, so g grows faster even though it takes a while to outstrip f.

The most common way to compare growth rates of functions is with *big-O notation*. If $f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$ are two functions with codomain the set of nonnegative reals, then $f(n) = O(g(n))$ (read "f is big-oh of g" or "f is order g") means:

- Informally: f grows at most as fast as g, or f is *asymptotically* upper bounded by g.
- Formally: $f(n) \leq c \cdot g(n)$ for all large enough n, where c is a constant (independent of n).
- In logic notation: $(\exists c > 0)\ (\exists n_0 \in \mathbb{N})\ (\forall n \geq n_0)\ (f(n) \leq c \cdot g(n))$.

It's common to use n_0 ("n nought") as the name of the threshold where the asymptotics "kick in." (Here, n and n_0 are separate variables despite the related names.) The $=$ sign in $f = O(g)$ is perhaps misleading since it doesn't mean the two sides are "the same" in any sense. Sometimes we write $f \leq O(g)$ to emphasize the $\leq$ in the definition, but it means the same thing. Some authors write $f \in O(g)$, letting $O(g)$ denote a set of functions. The $=$ notation has prevailed historically, for better or for worse.

Returning to the example with $f(n) = 100n^2$ and $g(n) = n^3$, we can say $f = O(g)$ using $c = 1$ and $n_0 = 100$ (or $c = 100$ and $n_0 = 0$, or other possibilities). But $g \neq O(f)$ since for any c and any n_0, picking an integer $n \geq n_0$ such that $n > 100c$, we have $g(n) = n^3 > 100cn^2 = c \cdot f(n)$. Thus g intuitively grows faster than f.

There's a stronger sense in which f grows slower than g, expressed with *little-o notation*. In general, $f(n) = o(g(n))$ (read "f is little-oh of g") is defined just like big-O but with $\forall$ instead of $\exists$ for c's quantifier: $(\forall c > 0)\ (\exists n_0 \in \mathbb{N})\ (\forall n \geq n_0)\ (f(n) \leq c \cdot g(n))$. In other words, no matter how tiny c is, $f(n) \leq c \cdot g(n)$ for all large enough n (where the threshold for "large enough" depends on c). Thus, as we increase $n = 0, 1, 2, \ldots$, eventually f becomes and stays $\leq g/100$, and if we wait longer then eventually f becomes and stays $\leq g/1000$, and so on; f just can't "keep pace" with g.

Just as O is analogous to $\leq$, and o is analogous to $<$, we have analogues of $\geq$, $>$, and $=$:

- $f = \Omega(g)$ (big Greek letter omega) means $g = O(f)$.
- $f = \omega(g)$ (little Greek letter omega) means $g = o(f)$.
- $f = \Theta(g)$ (big Greek letter theta) means $f = O(g)$ and $g = O(f)$ both hold.

In the following summary, we abbreviate each definition by omitting *(i)* that c's domain is $\mathbb{R}_{>0}$, *(ii)* the "for all large enough n" part between c's quantifier and the predicate, and *(iii)* the argument n to f and g in the predicate. This compressed form highlights the essential characteristics.

Notation	$f = O(g)$	$f = o(g)$	$f = \Omega(g)$	$f = \omega(g)$	$f = \Theta(g)$
Analogy	$\leq$	$<$	$\geq$	$>$	$=$
Meaning	$\exists c\ (f \leq c \cdot g)$	$\forall c\ (f \leq c \cdot g)$	$\exists c\ (f \geq c \cdot g)$	$\forall c\ (f \geq c \cdot g)$	$\exists c\ (\frac{1}{c} \cdot g \leq f \leq c \cdot g)$

We shouldn't take the analogies too seriously. For example, although "<" is equivalent to "not ≥", the asymptotic analogues aren't equivalent. It's true that $f = o(g)$ implies $f \neq \Omega(g)$, but not always vice versa: Exercise A.13 shows that there exist f and g such that $f \neq \Omega(g)$ and yet $f \neq o(g)$. The fundamental distinction is that $f = o(g)$ requires $f(n) \leq c \cdot g(n)$ for all large enough n, while $f \neq \Omega(g)$ requires this only for infinitely many n. However, functions for which this distinction matters are typically contrived; most "ordinary" functions we encounter behave as expected—either $f = o(g)$ or $f = \Theta(g)$ or $f = \omega(g)$.

A.4.2 Properties of asymptotic notation

The *transitive* property of $\leq$ for numbers (if $a \leq b$ and $b \leq c$ then $a \leq c$) has an analogue for growth of functions:

- If $f = O(g)$ and $g = O(h)$ then $f = O(h)$.

  To see this, suppose c_1, n_1 are the constants in the definition of $f = O(g)$ and c_2, n_2 are the constants for $g = O(h)$. Then for all $n \geq \max(n_1, n_2)$ we have $f(n) \leq c_1 \cdot g(n) \leq c_1 \cdot (c_2 \cdot h(n)) = (c_1 c_2) \cdot h(n)$, which shows that $f = O(h)$. A similar thing holds for o.

For nonnegative numbers, if $a_1 \leq b_1$ and $a_2 \leq b_2$ then $a_1 a_2 \leq b_1 b_2$. This property has a big-O analogue, where $(f_1 f_2)(n)$ is defined as $f_1(n) \cdot f_2(n)$:

- If $f_1 = O(g_1)$ and $f_2 = O(g_2)$ then $f_1 f_2 = O(g_1 g_2)$.

  To see this, suppose c_1, n_1 are the constants in the definition of $f_1 = O(g_1)$ and c_2, n_2 are the constants for $f_2 = O(g_2)$. Then for all $n \geq \max(n_1, n_2)$ we have $f_1(n) \cdot f_2(n) \leq (c_1 \cdot g_1(n)) \cdot (c_2 \cdot g_2(n)) = (c_1 c_2) \cdot (g_1(n) \cdot g_2(n))$, which shows that $f_1 f_2 = O(g_1 g_2)$. This property naturally extends to multiplying more than two functions. A similar thing holds for o.

For adding together several functions, the property looks strange at first glance: Addition is equivalent to maximum, where $(\max(f_1, f_2, \ldots, f_k))(n)$ is defined as $\max(f_1(n), f_2(n), \ldots, f_k(n))$:

- $f_1 + f_2 + \cdots + f_k = \Theta(\max(f_1, f_2, \ldots, f_k))$.

  This is because for any k nonnegative numbers, "max ≤ sum ≤ $k \cdot$max", so $\max_{i \in [k]} f_i(n) \leq \sum_{i \in [k]} f_i(n) \leq k \cdot \max_{i \in [k]} f_i(n)$ for all n. The constant factor k is "absorbed" into the c that's "hidden inside" the Θ. For different values of n, different functions f_i may attain the maximum, but for most functions we encounter, one f_i individually will attain the maximum for all large enough n.

Here's a "cheat sheet" for growth rates of common functions:

- *Polynomials*: Higher powers of n grow faster: $n^a = o(n^b)$ when $a < b$. A polynomial is a sum of monomials, each of which is a coefficient times a power of n. The degree is the highest exponent of a monomial with nonzero coefficient, and this power of n dominates asymptotically. Coefficients don't matter since they're absorbed into the hidden constant factor. Lower-order terms don't matter since a sum is big-Θ of the maximum term. For example, if $f(n) = 7n^4 + 10n^3 + 2n + 100$ then $f = \Theta(n^4)$ since f has degree 4. We can also say $f = O(n^{4.5})$ and $f = O(n^5)$ and so on, since these are valid upper bounds. The upshot is that we don't care about coefficients or low-order terms—they're just details that don't affect the overall growth rate.
- *Logarithms*: We have $\log_a = \Theta(\log_b)$ for all bases $a, b > 1$ since $\log_a(n) = \log_b(n)/\log_b(a)$; the factor $1/\log_b(a)$ gets absorbed into the hidden constant. Thus, we often write $O(\log n)$ without specifying the base, since it doesn't matter. Also, we can ignore constant powers inside the log since $\log_a(n^b) = b \cdot \log_a(n) = \Theta(\log n)$. Logarithms grow slower than every polynomial, even when the log is raised to a large power: $\log^a n = o(n^b)$ for arbitrarily large a and arbitrarily small $b > 0$. For example, $n \log^5 n = o(n^{1.1})$. (Technically, $\log(0)$ is undefined, so the log function can't have domain $\mathbb{N}$, which includes 0. This detail doesn't matter, due to the threshold n_0 for specifying "large enough" n.) Logs are base 2 when the base is unspecified. We may write $\log n$, omitting the parentheses in $\log(n)$, when there's no ambiguity.
- *Exponentials*: Exponentials with a larger base grow faster: $a^n = o(b^n)$ when $a < b$. Note that $2^{O(n)}$ is not the same as $O(2^n)$: The former has a hidden constant c in the exponent, and if $c > 1$ then $2^{c \cdot n} = (2^c)^n$ grows faster than 2^n since it has a bigger base. Exponentials grow faster than every polynomial, even when the exponential has n raised to a small power: $n^a = o(2^{n^b})$ for arbitrarily large a and arbitrarily small $b > 0$. For example, $n^5 = o(2^{n^{0.1}})$.

If we view "1" as a function that ignores n and always returns 1, then $o(1)$ represents an arbitrary function that converges to 0 as n goes to infinity. Similarly, $O(1)$ represents an unspecified positive constant. Thus $n^{O(1)}$ represents a polynomial of unspecified degree, as does $2^{O(\log n)} = 2^{O(1) \cdot \log n} = (2^{\log n})^{O(1)} = n^{O(1)}$.

A.4.3 Multivariate functions

If $f(m,n) = 10\sqrt{mn}$ and $g(m,n) = m + n$, we'd like to say $f = O(g)$ since when $m \le n$ we have $\sqrt{mn} \le \sqrt{n^2} = n \le m + n$, and similarly when $n \le m$. Defining big-O for functions of multiple variables is more subtle than the one-variable case, because "for all large enough (m,n)" is ambiguous. There are two common definitions, which we denote $O_\wedge$ and $O_\vee$. We focus on the two-variable case for concreteness. For $O_\wedge$, "large enough" means $m \ge m_0$ and $n \ge n_0$ for some thresholds $m_0, n_0 \in \mathbb{N}$, while for $O_\vee$ it means $m \ge m_0$ or $n \ge n_0$. If $f = O(g)$ under either definition, then this still holds when we use the same number $\max(m_0, n_0)$ for both thresholds, so we might as well just have one shared threshold, which we call n_0. This leads to our official definitions, which are identical except the $\wedge$ versus $\vee$:

- $O_\wedge$: $(\exists c > 0)\,(\exists n_0 \in \mathbb{N})\,(\forall (m,n) \in \mathbb{N}^2)\,\big((m \geq n_0 \ \wedge\ n \geq n_0) \ \Rightarrow\ f(m,n) \leq c \cdot g(m,n)\big)$
- $O_\vee$: $(\exists c > 0)\,(\exists n_0 \in \mathbb{N})\,(\forall (m,n) \in \mathbb{N}^2)\,\big((m \geq n_0 \ \vee\ n \geq n_0) \ \Rightarrow\ f(m,n) \leq c \cdot g(m,n)\big)$

Which definition is "right" depends on the circumstances. Sometimes the bound holds for all (m,n), so it doesn't matter (like our example $10\sqrt{mn} = O(m+n)$). In general, we prefer $O_\vee$ since it makes a stronger statement: $f = O_\vee(g)$ implies $f = O_\wedge(g)$. By default, O without a subscript means $O_\vee$ unless specified otherwise.

If $f = O_\vee(g)$ then for every single value $m = 0$, $m = 1$, $\ldots$, when we fix ("hardcode") that constant for m, the big-O holds for the resulting single-variable function of n (and similarly if we fix any constant value of n). For some functions, $O_\vee$ is too much to ask for: If $f(m,n) = m + n$ and $g(m,n) = mn$ then $f = O_\wedge(g)$ using $c = 2$ and $n_0 = 1$, but $f \neq O_\vee(g)$ since $g(0,n) = 0$ and $f(0,n) = n \neq O(0)$. We could alleviate this annoyance by changing the domain from $\mathbb{N}$ to $\mathbb{N} \setminus \{0\}$, but other annoying examples would persist—no quick fix erases the fundamental difference between $O_\wedge$ and $O_\vee$.

We sometimes encounter functions with a real number variable $0 < \varepsilon \leq 1$ (epsilon) that goes to 0, rather than a natural number that goes to infinity. We modify the definitions of O to say "for all small enough ε" (that is, $\varepsilon \leq \varepsilon_0$ or $\varepsilon \leq 1/n_0$). For example, if $f(n,\varepsilon) = 5n/\varepsilon^2 + 10n^2/\varepsilon$ and $g(n,\varepsilon) = (n/\varepsilon)^2$, then $f = O_\vee(g)$.

We won't need little-o notation for multivariate functions.

A.5 Counting

Given a description of a finite set, how can we determine the set's size? Listing all its elements to count them one by one is usually tedious and error-prone, even for moderately sized sets. Even worse, if the set has an unspecified parameter n (for example, $\{0,1\}^n$), then brute force doesn't even make sense. We need techniques to compute the size easily and express it as a function of any parameters.

A.5.1 Basic principles

Addition principle:	If $S_1, \ldots, S_n$ partition S, then $\lvert S\rvert = \sum_{i=1}^{n} \lvert S_i\rvert$.
Subtraction principle:	If $S \subseteq T$ then $\lvert S\rvert = \lvert T\rvert - \lvert T \setminus S\rvert$.
Multiplication principle:	If $S = S_1 \times \cdots \times S_n$ then $\lvert S\rvert = \prod_{i=1}^{n} \lvert S_i\rvert$.
Division principle:	If $S_1, \ldots, S_n$ partition S and have the same size s, then $n = \lvert S\rvert/s$.

We now illustrate these principles with an extended example. For starters, by the multiplication principle, $|\{0,1\}^n| = \prod_{i=1}^{n} |\{0,1\}| = 2^n$ (there are 2 options for the first bit, 2 options for the second bit, and so on). Now, how many functions $f\colon \{0,1\}^n \to \{0,1\}$ are there? For each $x \in \{0,1\}^n$, there are 2 options for the value of $f(x)$, so we get 2^n many factors of 2. In other words, if we imagine f as a table with 2^n rows, then the output column is a bit string of length 2^n, so by the multiplication principle, there are $|\{0,1\}^{2^n}| = 2^{2^n}$ many possibilities for f.

How many partial functions $f\colon \{0,1\}^n \to \{0,1\}$ are there? The answer is 3^{2^n} since for each x, there are 3 options for $f(x)$, namely 0, 1, or undefined. This count includes total functions, which are a special case of partial functions. What about non-total partial functions? The subtraction principle says there are $3^{2^n} - 2^{2^n}$ many. The next paragraph shows another way to count the non-total partial functions. This serves two purposes: to demonstrate a useful style of reasoning, and to increase our appreciation of the subtraction principle's power.

Letting $N = 2^n$, we view a non-total partial function as a tuple in $\{0,1,\text{undefined}\}^N$ with at least one "undefined" component. Letting S be the set of all such tuples, we partition S according to the location of the first "undefined":

$$S_i \;=\; \big\{f \in \{0,1,\text{undefined}\}^N \;:\; f_j \in \{0,1\} \text{ for all } j < i, \text{ and } f_i = \text{undefined}\big\}$$

These sets indeed form a partition since each $f \in S$ is in S_i for exactly one $i \in [N]$. By the addition principle, $|S| = \sum_{i=1}^{N} |S_i|$. We have $|S_i| = 2^{i-1} \cdot 1 \cdot 3^{N-i}$ since there are 2 options (0 or 1) for each of the first $i-1$ positions, 1 option (undefined) for the i^{th} position, and all 3 options for each remaining position. Therefore, $|S| = \sum_{i=1}^{N} 2^{i-1} \cdot 3^{N-i}$. This is an exact answer, but it's not very satisfying since it has a summation. To get a closed-form expression, we first massage the summands into a more convenient form. To make the exponents of 2 and 3 match, we rewrite $3^{N-i} = 3^{(N-1)-(i-1)}$ and factor 3^{N-1} out of the sum, leaving $\sum_{i=1}^{N} 2^{i-1} \cdot 3^{-(i-1)} = \sum_{i=1}^{N} (2/3)^{i-1}$. This is a *geometric sum*—the terms are consecutive powers of a common base—which we can evaluate using the "telescoping" trick: With some foresight, we ask what would happen if we multiply the sum by $1 - 2/3$. We'd be taking one copy of the sum and subtracting off another copy with every power increased by 1, so the only terms that wouldn't cancel are the lowest power in the first copy and the highest power in the subtracted copy. Letting $a = 2/3$ for brevity:

$$(1-a)\sum_{i=1}^{N} a^{i-1} \;=\; \big(a^0 + a^1 + a^2 + \cdots + a^{N-1}\big) - \big(a^1 + a^2 + a^3 + \cdots + a^N\big) \;=\; a^0 - a^N$$

Dividing by $1-a$ gives $\sum_{i=1}^{N} a^{i-1} = (a^0 - a^N)/(1-a) = 3 \cdot (1 - (2/3)^N)$. Recalling the 3^{N-1} we factored out earlier, the final answer is $3^N \cdot (1 - (2/3)^N) = 3^N - 2^N$, which agrees with the answer the subtraction principle gave us.

Let's see the division principle at work. Letting $\overline{x}$ denote the bitwise negation of x, how many total functions $f\colon \{0,1\}^n \to \{0,1\}$ are such that $f(x) \neq f(\overline{x})$ for all x? Since $\overline{\overline{x}} = x$, we can partition $\{0,1\}^n$ into parts, each consisting of a pair of strings that are bitwise negations of each other. Every part has size 2, so the division principle says the number of parts is $2^n/2$. To describe a function with the desired property, we could specify the value of $f(x)$ for one of the two inputs x in each part of the partition, and $f(\overline{x})$ would automatically get the opposite value. Since there are 2 options for the values in each of the $2^n/2$ parts, there are $2^{2^n/2}$ such functions.

A.5.2 Permutations and combinations

A *permutation* is an ordering of n distinct elements. Assuming the elements are the numbers $[n] = \{1,\ldots,n\}$, this means an n-tuple where each element of $[n]$ appears exactly once, such as $(4,2,5,1,3)$ for $n = 5$. The number of permutations is the factorial $n! = n\cdot(n-1)\cdot(n-2)\cdots 3\cdot 2\cdot 1$ (with $0! = 1$ as a special case) because there are n options for the first component, and then $n-1$ options for the second component (namely, any element of $[n]$ except the one used for the first component), then $n-2$ options for the third component, and so on. This illustrates a more general form of the multiplication principle: The set of all permutations of $[n]$ is not a cartesian product since the options for each component depend on the other components, but the same reasoning applies as in the basic multiplication principle.

As a warm-up for counting combinations, let's calculate the maximum possible number of edges in an n-node undirected graph with no multi-edges or self-loops. An edge is specified by distinct nodes u and v, so there are n options for u and then $n-1$ options for $v \neq u$. But the answer isn't $n(n-1)$ since this double counts each edge—either endpoint could be called u, and the other endpoint v, since an undirected edge is an unordered pair. We partition the set of all $n(n-1)$ ordered pairs of distinct nodes, so each part consists of two orders (u,v) and (v,u), corresponding to one undirected edge $\{u,v\}$. Each part has size 2, so the division principle gives the answer $\frac{n(n-1)}{2}$.

Now, we generalize this to counting *combinations* (subsets) of size k of a set of n elements. (Undirected graph edges correspond to $k=2$.) In terms of characteristic bit strings, this means counting strings in $\{0,1\}^n$ of *weight* (number of 1s) k. If we wanted to count *ordered* subsets (k-tuples of distinct elements) of $[n]$, there would be n options for the first component, then $n-1$ options for the second, and so on, and $n-(k-1)$ options for the k^{th} component, for a total of $n(n-1)\cdots(n-k+1) = n!/(n-k)!$ such tuples. Partitioning the set of all such tuples so each part contains all $k!$ different orderings of some size-k subset, the division principle says the number of parts—and hence the number of size-k subsets—is $n!/(k!(n-k)!)$. This quantity is so common it has a special notation $\binom{n}{k}$ ("n choose k"). Without the "subset counting" interpretation, it's not even obvious that the formula for $\binom{n}{k}$ has an integer value. From the formula, it's apparent that $\binom{n}{k} = \binom{n}{n-k}$, which makes sense since choosing k elements to keep is equivalent to choosing $n-k$ elements to discard. Also $\binom{n}{0} = \binom{n}{n} = 1$, which makes sense since there's exactly one way to choose nothing and one way to choose everything.

Partitioning $\{0,1\}^n$ according to weight, the addition principle says $2^n = \sum_{k=0}^{n}\binom{n}{k}$. Another way to see this is by plugging $x=1$ and $y=1$ into the *binomial formula* $(x+y)^n = \sum_{k=0}^{n}\binom{n}{k}x^k y^{n-k}$. This formula holds since to completely expand $(x+y)^n$ into a sum of 2^n many terms, we obtain each term by selecting x or y from each of the n factors, and there are $\binom{n}{k}$ ways to choose a term with k factors of x and $n-k$ factors of y. Due to its role in the binomial formula, $\binom{n}{k}$ is called a *binomial coefficient*.

As a final example, let's count the ways to partition a set of n elements, say $[n]$, into parts $A_1,\dots,A_m$ such that $|A_i| = k_i$ for each $i \in [m]$ (where n and m are fixed, each k_i is fixed, and $k_1+\cdots+k_m = n$). We have to choose k_1 elements from $[n]$ for A_1, then k_2 elements from $[n] \setminus A_1$ for A_2, then k_3 elements from $[n] \setminus (A_1 \cup A_2)$ for A_3, and so on, giving the answer:

$$\binom{n}{k_1}\binom{n-k_1}{k_2}\binom{n-k_1-k_2}{k_3}\cdots\binom{n-k_1-\cdots-k_{m-1}}{k_m}$$

This looks messy, but it simplifies since the $(n-k_1-\cdots-k_i)!$ in the i^{th} factor's denominator cancels the $(i+1)^{\text{st}}$ factor's numerator, leaving just $n!/(k_1!\,k_2!\cdots k_m!)$ after the smoke clears. The latter expression suggests the answer might be derivable directly from the division principle. Indeed, if we consider one of the $n!$ permutations of $[n]$ and declare that A_1 is the first k_1 elements and A_2 is the next k_2 elements, and so on, then this overcounts by a factor $k_1!\,k_2!\cdots k_m!$ since there are $k_1!$ ways to order the first k_1 elements, $k_2!$ ways to order the next k_2 elements, and so on, which all correspond to the same partition of $[n]$.

A.6 Probability

A.6.1 Discrete probability spaces

Probability theory models situations that have unpredictability or uncertainty about what will happen. A *discrete probability space* consists of two things:

- A *sample space* is a finite set S whose elements are called *outcomes*.
- A *distribution* assigns a probability value $\mathbf{Pr}[s] \geq 0$ to each outcome $s \in S$, such that $\sum_{s\in S} \mathbf{Pr}[s] = 1$.

This specifies a *random experiment* in which exactly one outcome occurs, and each outcome's probability value is its chance of occurring. The distribution's *support* is the set of all outcomes having nonzero probability. Some classic discrete probability spaces are:

- Fair coin toss: $S = \{\mathrm{H}, \mathrm{T}\}$ representing heads and tails, and $\mathbf{Pr}[\mathrm{H}] = \mathbf{Pr}[\mathrm{T}] = 1/2$.
- Die throw: $S = [6]$ and $\mathbf{Pr}[s] = 1/6$ for each $s \in [6]$.
- Shuffle of a standard deck of cards: S is all $52!$ permutations of the deck set $[13] \times \{\spadesuit, \heartsuit, \clubsuit, \diamondsuit\}$ (a card is a (rank, suit) pair), and $\mathbf{Pr}[s] = 1/52!$ for each permutation s.

Each of these example probability spaces happens to be *uniform*: All outcomes have the same probability $1/|S|$. In particular, a uniform distribution has full support S.

An *event* is a subset of outcomes $A \subseteq S$, and its probability is $\mathbf{Pr}[A] = \sum_{s\in A} \mathbf{Pr}[s]$. In particular, $\mathbf{Pr}[\emptyset] = 0$ and $\mathbf{Pr}[S] = 1$.

Example. To find the probability of getting an odd number in a die throw, we consider the event $A = \{1, 3, 5\} \subseteq [6]$ and see that $\mathbf{Pr}[A] = \mathbf{Pr}[1] + \mathbf{Pr}[3] + \mathbf{Pr}[5] = 1/6 + 1/6 + 1/6 = 1/2$. ♦

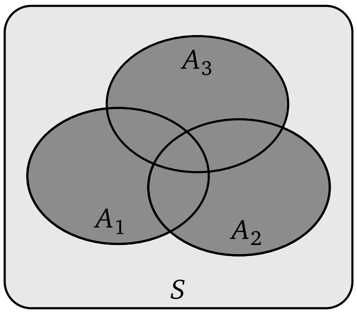
For a uniform probability space, $\mathbf{Pr}[A] = |A|/|S|$ and so computing probabilities is just a matter of counting.

Example. To find the probability that the first five cards in a shuffled deck have the same suit as each other (a "flush," including the possibility of a "straight flush"): There are 4 options for the suit, and $13!/(13-5)!$ options for the ranks of the first five cards, and $(52-5)!$ ways to order the rest of the deck, so the answer is $4 \cdot (13!/8!) \cdot 47! \,/\, 52!$. ♦

We represent a distribution as a function $D \colon S \to \mathbb{R}_{\geq 0}$ where $D(s) = \mathbf{Pr}[s]$. When discussing several distributions, the subscript in $\mathbf{Pr}_D$ says which distribution we're talking about. We may also write $\mathbf{Pr}_{s\sim D}$ to say outcome s is *sampled* from distribution D, and we may write an event as a predicate with s as the free variable. When the variable name s is enough to identify which probability space we're talking about, we may just write $\mathbf{Pr}_s$.

Example. Suppose $D \colon [6] \to \mathbb{R}_{\geq 0}$ is a "loaded" die where $\mathbf{Pr}[s]$ is proportional to s, so $D(s) = s/21$ since $1+2+3+4+5+6 = 21$. Suppose $U \colon [6] \to \mathbb{R}_{\geq 0}$ is the uniform distribution (a fair die). For the event $A = \{1, 3, 5\}$, we have $\mathbf{Pr}_D[A] = 1/21 + 3/21 + 5/21 = 9/21 = 3/7$ and $\mathbf{Pr}_U[A] = 1/2$. We may also write $\mathbf{Pr}_{s\sim D}[s \text{ is odd}] = 3/7$ and $\mathbf{Pr}_{s\sim U}[s \text{ is odd}] = 1/2$. ♦

We visualize a probability space as an oval, or other shape, with area 1. An event corresponds to a sub-oval enclosing points that represent outcomes in the event. An event's probability is the sub-oval's area. This makes the handy *union bound* visually clear:



$$\mathbf{Pr}[A_1 \cup A_2 \cup \cdots \cup A_n] \;\le\; \mathbf{Pr}[A_1] + \mathbf{Pr}[A_2] + \cdots + \mathbf{Pr}[A_n]$$

The left side is the area of the union, and the right side is the sum of the areas, which overcounts the outcomes that are in more than one A_i event.

If $f : S \to T$ is a function and D is a distribution over S, then sampling $s \sim D$ and applying f yields a sample $f(s)$ from some distribution over T. The latter distribution is denoted $f(D)$ and defined by $\mathbf{Pr}_{f(D)}[t] = \mathbf{Pr}_{s\sim D}\big[f(s) = t\big] = \sum_{s\,:\,f(s)=t} \mathbf{Pr}_D[s] = \mathbf{Pr}_D\big[f^{-1}(t)\big]$. (Note that $f(D)$ doesn't mean the function composition $f \circ D$, which would be a "syntax error.") An image's probability is the sum of its preimages' probabilities. This coalesces the event $f^{-1}(t) \subseteq S$ into a single outcome $t \in T$, giving a "coarser granularity" view. This is useful for purging extraneous detail from a probability space.

Example. Suppose S is the set of all permutations of a 52-card deck, and T is the set of all "hands" (sets of 5 cards). If $f(s)$ is the set of the first 5 cards in s, and D is uniform over S, then $f(D)$ is uniform over T since the number of permutations with a particular first hand doesn't depend on the hand. Revisiting our flush example, we calculate the probability as $4 \cdot \binom{13}{5} / \binom{52}{5}$ since there are 4 suits and $\binom{13}{5}$ ways to choose 5 cards of a given suit, out of $\binom{52}{5}$ possible hands. ♦

A *joint* distribution is a distribution D over a cartesian product sample space $S = S_1 \times S_2$. Suppose we sample $(s_1, s_2) \sim D$ but ignore s_2 and just view s_1 as the outcome. This is a sample from a distribution D_1 over S_1, known as D's *marginal* distribution on the first coordinate. We define $D_1 = f_1(D)$ where $f_1 : S_1 \times S_2 \to S_1$ is the *projection* function that simply returns its first argument. In other words, $D_1(s_1) = \sum_{s_2 \in S_2} D(s_1, s_2)$. Similarly, we can sample from the marginal distribution $D_2 = f_2(D)$ over S_2 by sampling $(s_1, s_2) \sim D$ and ignoring s_1.

On the right is an example joint distribution D over $\{a, b\} \times \{c, d\}$. In the marginal distribution D_1, an outcome's probability is the sum of the entries in a row of D. In the marginal distribution D_2, an outcome's probability is the sum of the entries in a column of D. The probabilities for D_1 and D_2 are written "in the margins," which explains where the word "marginal" came from.

D	c	d
a	0.1	0.2
b	0.3	0.4

D_1
0.3
0.7

D_2	0.4	0.6

The notions of joint and marginal distributions generalize to cartesian products of more than two sets.

We sometimes use a countably infinite sample space such as $\mathbb{N}$ or $\mathbb{Z}$, in which case the theory is not substantially different from the finite case.

A.6.2 Continuous probability spaces

Occasionally we use continuous probability spaces that are simple enough to be equivalent to discrete probability spaces. The continuous perspective is sometimes more intuitive.

A continuous sample space is an interval of real numbers $[a,b] = \{x \in \mathbb{R} : a \leq x \leq b\}$ with endpoints $a < b$. Since the interval contains infinitely many points, the uniform distribution would assign probability 0 to each outcome. Rather than considering individual outcomes, we consider events that are subintervals. The probability of a subinterval $[c,d] \subseteq [a,b]$ (under the uniform distribution) is $(d-c)/(b-a)$, which is the length of $[c,d]$ divided by the length of $[a,b]$. If an event is the union of finitely many subintervals that are disjoint (except possibly at their endpoints), then its probability is the sum of the probabilities of those subintervals.

Example. With the uniform distribution over the sample space $[2,12]$, the event $[3,4.5] \cup [6.5,9]$ has probability:

$$\frac{4.5-3}{12-2} + \frac{9-6.5}{12-2} = \frac{1.5}{10} + \frac{2.5}{10} = 0.4$$

♦

Consider a scenario where every event is a union of some of the subintervals $[z_0,z_1]$, $[z_1,z_2]$, $\ldots$, $[z_{n-1},z_n]$ for some finite sequence of points $a = z_0 < z_1 < z_2 < \cdots < z_n = b$. This is equivalent to a discrete probability space with sample space $[n]$, where outcome $i \in [n]$ has probability $(z_i - z_{i-1})/(b-a)$. If we throw a dart uniformly at the 1-dimensional dartboard $[a,b]$, it only matters which region $[z_{i-1},z_i]$ the dart lands in (ignoring the possibility that the dart lands exactly on a boundary z_i, which has probability 0).

It's more subtle to define nonuniform continuous distributions and events that aren't unions of finitely many subintervals.

A.6.3 Conditioning and independence

Suppose someone runs a random experiment and tells us event A occurred, but doesn't say which specific outcome $s \in A$ occurred. How should we revise our view of the probability space? With the visual "probability = area" analogy, we should erase everything outside the oval corresponding to event A, then "zoom in" or "scale up" this oval until it has area exactly 1, as if it becomes the entire new probability space. Everything inside the oval gets its probability (area) scaled accordingly, by dividing by $\mathbf{Pr}[A]$. This is called *conditioning* on A. It is probability theory's version of "if-then." The notation for the updated probabilities uses a vertical bar: $\mathbf{Pr}[s \mid A]$ (probability of s "given A" or "conditioned on A") for an outcome s, and $\mathbf{Pr}[B \mid A]$ for an event B. The definition is:

$$\mathbf{Pr}[s \mid A] = \begin{cases} \mathbf{Pr}[s]/\mathbf{Pr}[A] & \text{if } s \in A \\ 0 & \text{if } s \notin A \end{cases} \qquad \text{and} \qquad \mathbf{Pr}[B \mid A] = \textstyle\sum_{s \in B} \mathbf{Pr}[s \mid A] = \mathbf{Pr}[B \cap A]/\mathbf{Pr}[A]$$

If $\mathbf{Pr}[A] = 0$ then conditioning on A is disallowed due to division by 0. If D is the original distribution function, then $(D \mid A)$ denotes the conditioned distribution $(D \mid A)(s) = \mathbf{Pr}_D[s \mid A]$, whose support is contained in A.

With a uniform probability space, $\mathbf{Pr}[B \mid A] = \big(|B \cap A|/|S|\big)/\big(|A|/|S|\big) = |B \cap A|/|A|$.

Example. If we throw a die and let A be the event that s is odd and B be the event that $s \geq 5$, then $\mathbf{Pr}[B \mid A] = |B \cap A|/|A| = 1/3$ and $\mathbf{Pr}[A \mid B] = |A \cap B|/|B| = 1/2$. ♦

Example. Suppose we shuffle a deck of cards and let A be the event that the first card is a ♠ and B be the event that the second card is a ♠. These events only depend on the first two cards, so we can filter the irrelevant information out of the sample space by applying a function that returns the first two cards of the outcome. This lets us work with the uniform distribution over ordered pairs of distinct cards, and assume A and B are subsets of this new sample space. We have $\mathbf{Pr}[A] = \mathbf{Pr}[B] = (13 \cdot 51)/(52 \cdot 51) = 13/52 = 1/4$ and $\mathbf{Pr}[B \mid A] = \mathbf{Pr}[A \mid B] = (13 \cdot 12)/(13 \cdot 51) = 12/51 < 1/4$. ♦

Conditioning on an event A, and then further conditioning on an event B, is equivalent to conditioning on the intersection: $((D \mid A) \mid B) = (D \mid (A \cap B))$ because for every outcome $s \in A \cap B$:

$$\begin{aligned}
\mathbf{Pr}_{(D \mid A) \mid B}[s] &= \mathbf{Pr}_{D \mid A}[s]/\mathbf{Pr}_{D \mid A}[B] \\
&= \big(\mathbf{Pr}_D[s]/\mathbf{Pr}_D[A]\big)/\mathbf{Pr}_D[B \mid A] \\
&= \mathbf{Pr}_D[s]/\mathbf{Pr}_D[A \cap B] \\
&= \mathbf{Pr}_{D \mid (A \cap B)}[s]
\end{aligned}$$

Events A and B are *independent* (of each other) when $\mathbf{Pr}[A \cap B] = \mathbf{Pr}[A] \cdot \mathbf{Pr}[B]$. If $\mathbf{Pr}[A] = 0$ or $\mathbf{Pr}[B] = 0$ then they are certainly independent; otherwise, we can rearrange the definition to $\mathbf{Pr}[B \mid A] = \mathbf{Pr}[B]$ or equivalently $\mathbf{Pr}[A \mid B] = \mathbf{Pr}[A]$. This says if we know that one of the events occurred, it doesn't change the probability of the other occurring. Events A and B are *conditionally independent* given an event C when $\mathbf{Pr}[A \cap B \mid C] = \mathbf{Pr}[A \mid C] \cdot \mathbf{Pr}[B \mid C]$, in other words, A and B are independent under the distribution $(D \mid C)$.

Example. Continuing our previous example, $\mathbf{Pr}[B \mid A] < \mathbf{Pr}[B]$ so A and B are not independent: Knowing the first card is a ♠ decreases the probability that the second is a ♠, because there are fewer remaining ♠s but no fewer of other suits. However, A and B are conditionally independent given the event C that the two cards have different ranks: $\mathbf{Pr}[A \cap B \mid C] = \mathbf{Pr}[A \mid C] \cdot \mathbf{Pr}[B \mid C]$ since the left side is $(13 \cdot 12)/(52 \cdot 48) = (1/4)^2$, and each factor on the right side is $(13 \cdot 48)/(52 \cdot 48) = 1/4$. Intuitively, the marginal distribution on the pair of suits becomes uniform over $\{♠, ♡, ♣, ♢\}^2$ once we know the ranks are different. ♦

Example. It's often useful to consider a biased coin with $\mathbf{Pr}[\mathrm{H}] = p$ and $\mathbf{Pr}[\mathrm{T}] = 1 - p$ for some $p \neq 1/2$. Suppose $p = 1/3$ and we toss the coin twice (or toss two such coins once each), so an outcome is $s \in \{\mathrm{H}, \mathrm{T}\}^2$ (and we view s as a string, to avoid writing parentheses and commas). Since the tosses intuitively don't affect each other, we define the joint distribution D so any outcome of the first toss is independent of any outcome of the second toss. For example, $\mathbf{Pr}_D[\mathrm{HT}] = \mathbf{Pr}_{D_1}[\mathrm{H}] \cdot \mathbf{Pr}_{D_2}[\mathrm{T}] = (1/3) \cdot (2/3) = 2/9$ by definition. Here's the distribution function and some example calculations involving conditioning:

s	$D(s)$
HH	1/9
HT	2/9
TH	2/9
TT	4/9

$A = \{s : s_1 = s_2\}, \;\; B = \{s : s_1 = \mathrm{H}\}$

$\mathbf{Pr}[A] = 1/9 + 4/9 = 5/9$

$\mathbf{Pr}[B] = 1/9 + 2/9 = 1/3$

$\mathbf{Pr}[B \mid A] = (1/9)/(5/9) = 1/5$

$\mathbf{Pr}[A \mid B] = (1/9)/(1/3) = 1/3$ ♦

Two functions $f_1 : S \to T_1$ and $f_2 : S \to T_2$ are independent when for all $(t_1, t_2) \in T_1 \times T_2$, the events $f_1^{-1}(t_1) = \{s : f_1(s) = t_1\}$ and $f_2^{-1}(t_2) = \{s : f_2(s) = t_2\}$ are independent.

Example. We saw that the suits of the first two cards in a shuffled deck are not independent. Similarly, the ranks of the first two cards are not independent. But let's show that the first card's suit and the second card's rank are independent. This is an assertion about many pairs of events. Formally, consider $f_1 : S \to \{\spadesuit, \heartsuit, \clubsuit, \diamondsuit\}$ and $f_2 : S \to [13]$ where $f_1(s)$ is the first card's suit and $f_2(s)$ is the second card's rank. For all $(t_1, t_2) \in \{\spadesuit, \heartsuit, \clubsuit, \diamondsuit\} \times [13]$, say $(t_1, t_2) = (\heartsuit, 7)$, we have $\mathbf{Pr}\big[f_1(s) = \heartsuit \text{ and } f_2(s) = 7\big] = 51/(52 \cdot 51) = 1/52$ because out of the $52 \cdot 51$ possibilities for the first two cards, there are 51 possibilities where the first is a $\heartsuit$ and the second is a 7. This is because out of the 52 possibilities of (first card's rank, second card's suit) $\in [13] \times \{\spadesuit, \heartsuit, \clubsuit, \diamondsuit\}$, only $(7, \heartsuit)$ does not occur in this event. We also have $\mathbf{Pr}\big[f_1(s) = \heartsuit\big] = 1/4$ and $\mathbf{Pr}\big[f_2(s) = 7\big] = 1/13$, so:

$$\mathbf{Pr}\big[f_1(s) = \heartsuit \text{ and } f_2(s) = 7\big] \;=\; \mathbf{Pr}\big[f_1(s) = \heartsuit\big] \cdot \mathbf{Pr}\big[f_2(s) = 7\big] \qquad \blacklozenge$$

More than two events $A_1, \ldots, A_m$ are *fully* (or *mutually*) independent when not only $\mathbf{Pr}[A_1 \cap A_2 \cap \cdots \cap A_m] = \mathbf{Pr}[A_1] \cdot \mathbf{Pr}[A_2] \cdots \mathbf{Pr}[A_m]$ holds, but a similar equation holds for every sub-collection of the m events:

$$\text{for every nonempty } I \subseteq [m]\text{:}\;\; \mathbf{Pr}\big[\textstyle\bigcap_{i \in I} A_i\big] = \textstyle\prod_{i \in I} \mathbf{Pr}[A_i]$$

Functions $f_1 : S \to T_1, \ldots, f_m : S \to T_m$ are fully independent when for all $(t_1, \ldots, t_m) \in T_1 \times \cdots \times T_m$, the events $f_1^{-1}(t_1), \ldots, f_m^{-1}(t_m)$ are fully independent.

A.6.4 Independence by design

For any two distributions D_1 over S_1 and D_2 over S_2, the joint distribution D over $S_1 \times S_2$ defined by $D(s_1, s_2) = D_1(s_1) \cdot D_2(s_2)$ has D_1 and D_2 as its marginals. We describe D as "sample $s_1 \sim D_1$ and $s_2 \sim D_2$ independently," because any event A_1 depending only on s_1 is independent of any event A_2 depending only on s_2 (which implies that any function depending only on s_1 is independent of any function depending only on s_2): If $A_1 = B_1 \times S_2$ and $A_2 = S_1 \times B_2$ for some $B_1 \subseteq S_1$ and $B_2 \subseteq S_2$, then:

$$\begin{aligned} D(A_1 \cap A_2) \;=\; D(B_1 \times B_2) &= \textstyle\sum_{s_1 \in B_1} \sum_{s_2 \in B_2} D_1(s_1) \cdot D_2(s_2) \\ &= \textstyle\sum_{s_1 \in B_1} \big(D_1(s_1) \cdot \sum_{s_2 \in B_2} D_2(s_2)\big) \\ &= \textstyle\sum_{s_1 \in B_1} D_1(s_1) \cdot D_2(B_2) \\ &= D_1(B_1) \cdot D_2(B_2) \;=\; D(A_1) \cdot D(A_2) \end{aligned}$$

In the last step, we used

$$\begin{aligned} D(A_1) &= \textstyle\sum_{s_1 \in B_1} \sum_{s_2 \in S_2} D_1(s_1) \cdot D_2(s_2) \\ &= \textstyle\sum_{s_1 \in B_1} \big(D_1(s_1) \cdot \sum_{s_2 \in S_2} D_2(s_2)\big) \\ &= \textstyle\sum_{s_1 \in B_1} D_1(s_1) \cdot 1 \\ &= D_1(B_1) \end{aligned}$$

and similarly $D(A_2) = D_2(B_2)$.

A common probability space is n tosses of a coin with heads probability p. We intuitively expect different tosses to be fully independent since the tosses don't affect each other. For any outcome $s \in \{\mathrm{H}, \mathrm{T}\}^n$ with k heads and $n-k$ tails, we define $\mathbf{Pr}[s] = p^k(1-p)^{n-k}$. As a sanity check, let's make sure these probabilities sum to 1: There are $\binom{n}{k}$ outcomes with k heads, so

$$\textstyle\sum_s \mathbf{Pr}[s] \;=\; \sum_{k=0}^{n}\binom{n}{k}p^k(1-p)^{n-k} \;=\; (p+(1-p))^n \;=\; 1^n \;=\; 1$$

by the binomial formula (§A.5.2). This distribution's definition ensures that events depending on different tosses are indeed fully independent; here's an example to illustrate why.

Example. Suppose A_1, A_2, A_3 are the events that the 1st toss is H, the 2nd toss is T, and the 3rd toss is T, respectively. Since $\binom{n-3}{k}$ many outcomes s are such that $s_1s_2s_3 = \mathrm{HTT}$ and the other $n-3$ tosses have k heads:

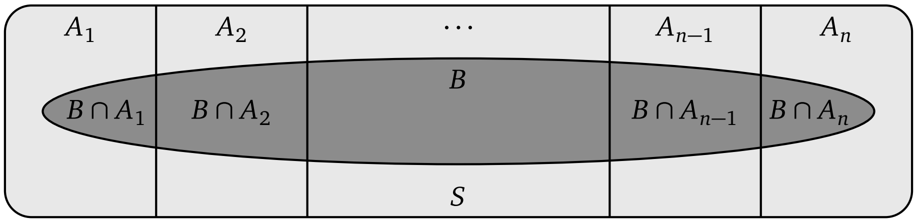
$$\begin{aligned}\mathbf{Pr}[A_1 \cap A_2 \cap A_3] &= \textstyle\sum_{k=0}^{n-3}\binom{n-3}{k}p(1-p)(1-p)\cdot p^k(1-p)^{n-3-k}\\ &= p(1-p)^2\cdot(p+(1-p))^{n-3}\\ &= p(1-p)^2\end{aligned}$$

Similar reasoning shows that $\mathbf{Pr}[A_1] = p$ and $\mathbf{Pr}[A_2] = \mathbf{Pr}[A_3] = 1-p$ and $\mathbf{Pr}[A_1 \cap A_2] = \mathbf{Pr}[A_1 \cap A_3] = p(1-p)$ and $\mathbf{Pr}[A_2 \cap A_3] = (1-p)^2$. Hence A_1, A_2, A_3 are fully independent. In fact, this reasoning shows that the marginal distribution on any m of the n tosses is the same as the distribution in which there are only m tosses total. ♦

It's generally cumbersome to verify that a collection of m events is fully independent, since there are almost 2^m equations to check (one for each $I \subseteq [m]$). Fortunately, we can always design probability spaces with full independence "built in": For any distributions $D_1, \ldots, D_n$ over sets $S_1, \ldots, S_n$, the joint distribution D over $S_1 \times \cdots \times S_n$ defined by $D(s_1, \ldots, s_n) = \prod_{i=1}^{n} D_i(s_i)$ has $D_1, \ldots, D_n$ as its marginals. We describe D as "for each $i \in [n]$ independently, sample $s_i \sim D_i$." (For example, if we have a loaded die with a particular distribution over $[6]$, then "n independent throws of the die" means the joint distribution is defined in this way.) Like when $n = 2$, if $A_1, \ldots, A_m$ are events depending on disjoint subsets of coordinates (say, A_i only depends on coordinates $I_i \subseteq [n]$, and $I_i \cap I_j = \emptyset$ for $i \neq j$), then these events are fully independent. This implies that any functions depending on disjoint subsets of coordinates are fully independent.

A.6.5 Decomposing events

If events $A_1, \ldots, A_n$ form a partition of S, then $(B \cap A_1), \ldots, (B \cap A_n)$ form a partition of B (possibly with some empty parts), so $\mathbf{Pr}[B] = \sum_{i=1}^{n} \mathbf{Pr}[B \cap A_i]$ by the probability analogue of the addition principle for counting. Recall that $\mathbf{Pr}[B \cap A_i] = \mathbf{Pr}[A_i] \cdot \mathbf{Pr}[B \mid A_i]$ if $\mathbf{Pr}[A_i] > 0$. Thus we derive the *law of total probability*: $\mathbf{Pr}[B] = \sum_{i=1}^{n} \mathbf{Pr}[A_i] \cdot \mathbf{Pr}[B \mid A_i]$. (If $\mathbf{Pr}[A_i] = 0$ then the ith term would be omitted from the sum.) Pictorially, this says B's area is the sum, over all parts of the partition of S, of the part's area times the fraction of the part occupied by B.



This gives a technique for calculating a complicated event's probability: Design an appropriate partition and apply the law of total probability. For partitions with two parts, we phrase the law as $\mathbf{Pr}[B] = \mathbf{Pr}[A] \cdot \mathbf{Pr}[B \mid A] + \mathbf{Pr}[\overline{A}] \cdot \mathbf{Pr}[B \mid \overline{A}]$ where $\overline{A} = S \setminus A$ is the complement event.

Example. If we toss a fair coin n times, what's the probability of getting an odd number of heads? The tosses are fully independent by definition, so an outcome with k heads and $n-k$ tails has probability $(1/2)^k(1/2)^{n-k} = 1/2^n$, which is the same for all outcomes. Thus the distribution is uniform over $S = \{\mathrm{H}, \mathrm{T}\}^n$, so this is "just a counting problem." However, the probabilistic perspective is often helpful, so we illustrate it here. Let B be the event of getting an odd number of heads. For each $r \in \{\mathrm{H}, \mathrm{T}\}^{n-1}$, let A_r be the event that the first $n-1$ tosses are exactly r. If r has an even number of heads, then $r\mathrm{H} \in B$ and $r\mathrm{T} \notin B$ and these two outcomes each have probability $1/2$ conditioned on A_r (by independence), so $\mathbf{Pr}[B \mid A_r] = 1/2$. Similarly, if r has an odd number of heads, then $r\mathrm{H} \notin B$ and $r\mathrm{T} \in B$ and $\mathbf{Pr}[B \mid A_r] = 1/2$. Since $\mathbf{Pr}[B \mid A_r] = 1/2$ for every $r \in \{\mathrm{H}, \mathrm{T}\}^{n-1}$, and these A_r sets partition S, we have $\mathbf{Pr}[B] = \sum_r \mathbf{Pr}[A_r] \cdot \mathbf{Pr}[B \mid A_r] = \frac{1}{2} \cdot \sum_r \mathbf{Pr}[A_r] = 1/2$ by the law of total probability. (The specific values of $\mathbf{Pr}[A_r]$ didn't matter here, but they're all $1/2^{n-1}$ by independence.) Using 1 and 0 instead of H and T, this means exactly half of all length-n bit strings have odd weight. ♦

Suppose $A_1, \ldots, A_n$ are arbitrary events (not a partition of S). What's the probability they all occur simultaneously? If they are fully independent, it's simply $\prod_{i=1}^n \mathbf{Pr}[A_i]$. Otherwise we can use the *chain rule*:

$$\mathbf{Pr}[A_1 \cap \cdots \cap A_n] = \mathbf{Pr}[A_1] \cdot \mathbf{Pr}[A_2 \mid A_1] \cdot \mathbf{Pr}[A_3 \mid A_1 \cap A_2] \cdots \mathbf{Pr}[A_n \mid A_1 \cap \cdots \cap A_{n-1}]$$

This holds since the numerator in the i^{th} factor on the right side cancels out the denominator in the $(i+1)^{\text{st}}$ factor.

Example. The chain rule yields an elegant way to calculate the probability that a shuffled deck's first five cards are ♠s: For $i \leq 5$, let A_i be the event that the i^{th} card is a ♠. Then $\mathbf{Pr}[A_{i+1} \mid A_1 \cap \cdots \cap A_i] = (13-i)/(52-i)$ since no matter which ♠s the first i cards are, $13-i$ many ♠s remain among the other $52-i$ cards. The answer is $\prod_{i=0}^4 (13-i)/(52-i) = (13!/8!)/(52!/47!)$. ♦

A.6.6 Decomposing distributions

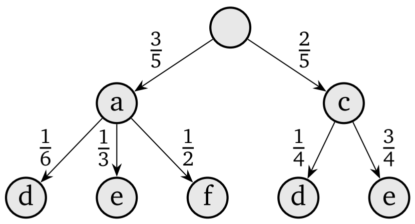
Suppose D is a joint distribution over $S_1 \times \cdots \times S_n$. We can sample $(s_1, \ldots, s_n) \sim D$ with an n-step process: First sample s_1, then sample s_2 conditioned on what we got for s_1, then sample s_3 conditioned on what we got for s_1 and s_2, and so on. This process yields any particular outcome s with the correct probability $D(s)$ by the chain rule, where A_i is the event that the i^{th} coordinate equals s_i. We represent this process with an out-tree: We label the root's children with the elements of S_1 and label the root's outgoing edges with the corresponding probabilities of the marginal D_1, but we omit any child that would have probability 0. For each child of the root, say labeled s_1, we label its children with the elements of S_2 and its outgoing edges with the corresponding probabilities of the distribution $(D \mid A_1)_2$, which is D's 2nd marginal after conditioning on the event A_1 that the 1st coordinate equals s_1, but again we omit any child that would have probability 0. In general, for a node whose path from the root has node labels $s_1, \dots, s_i$, if $i = n$ then it's a leaf, otherwise we label its children using S_{i+1} and its outgoing edges using $(D \mid A_1 \cap \cdots \cap A_i)_{i+1}$, omitting any 0-probability children, where A_j is the event that the jth coordinate equals s_j.

Example. With $n = 2$:

D	d	e	f
a	0.1	0.2	0.3
b	0	0	0
c	0.1	0.3	0

D_1
0.6
0
0.4

D_2	0.2	0.5	0.3



♦

The process follows a random path down the tree. In each step, it picks a random outgoing edge according to the probabilities labeling these edges. Each outcome $s \in S_1 \times \cdots \times S_n$ with $D(s) > 0$ corresponds to a leaf, and its probability $D(s)$ is the product of the edge labels along the root-to-leaf path.

If the coordinates of D are fully independent (that is, $D(s) = \prod_{i=1}^{n} D_i(s_i)$ for every s), then all nodes across the same level of the tree have the same distribution labeling their outgoing edges. This is because if A_j is the event that the jth coordinate equals some particular s_j, then $\mathbf{Pr}[A_{i+1} \mid A_1 \cap \cdots \cap A_i] = \prod_{j=1}^{i+1} D_j(s_j) / \prod_{j=1}^{i} D_j(s_j) = D_{i+1}(s_{i+1})$, which only depends on s_{i+1}, not on $s_1, \dots, s_i$.

When D is a distribution over $S_1 \times S_2$, the process decomposes D_2 into a *mixture* of conditioned distributions: It samples $s_1 \sim D_1$ followed by $s_2 \sim D^{s_1}$, where the distribution D^{s_1} over S_2 is defined by $D^{s_1}(s_2) = D(s_1, s_2)/D_1(s_1)$, which is D's 2nd marginal after conditioning on the 1st coordinate being s_1. (We make s_1 a superscript since subscripts are for marginals.) We write this compactly as $D_2 = \sum_{s_1} D_1(s_1) \cdot D^{s_1}$ where the sum is over all s_1 in D_1's support. Thus, we form D_2 by mixing together all the distributions D^{s_1}, with coefficients (or "weights") given by D_1. In other words, the coefficients are the root's outgoing edge labels, and each constituent distribution D^{s_1} is node s_1's outgoing edge labels. To calculate the probability of any event $B \subseteq S_2$ under D_2, we can calculate B's probability under each D^{s_1} and combine these probabilities using the mixture's coefficients: $D_2(B) = D(S_1 \times B) = \sum_{s_1} D_1(s_1) \cdot D^{s_1}(B)$ by the law of total probability with $S_1 \times S_2$ partitioned into the events $A^{s_1} = \{s_1\} \times S_2$ for all $s_1 \in S_1$.

Example. Continuing the previous example, if we think of distributions over $\{\text{d}, \text{e}, \text{f}\}$ as tuples $(\mathbf{Pr}[\text{d}], \mathbf{Pr}[\text{e}], \mathbf{Pr}[\text{f}])$, then $D_2 = (\frac{1}{5}, \frac{1}{2}, \frac{3}{10})$ is decomposed into the mixture $D_1(\text{a}) \cdot D^{\text{a}} + D_1(\text{c}) \cdot D^{\text{c}} = \frac{3}{5} \cdot (\frac{1}{6}, \frac{1}{3}, \frac{1}{2}) + \frac{2}{5} \cdot (\frac{1}{4}, \frac{3}{4}, 0)$. By the law of total probability, we can calculate:

$$D_2(\{\text{d}, \text{f}\}) = D_1(\text{a}) \cdot D^{\text{a}}(\{\text{d}, \text{f}\}) + D_1(\text{c}) \cdot D^{\text{c}}(\{\text{d}, \text{f}\}) = \tfrac{3}{5} \cdot (\tfrac{1}{6} + \tfrac{1}{2}) + \tfrac{2}{5} \cdot (\tfrac{1}{4} + 0) = \tfrac{1}{2}$$ ♦

Above, we used a joint distribution D to express D_2 as a mixture. In the reverse direction, we can use a mixture to define a joint distribution: Suppose $D_2 = \sum_{s_1} D_1(s_1) \cdot D^{s_1}$ where D_1 is some distribution over S_1, and D^{s_1} is some distribution over D_2 for each s_1 in D_1's support. We can define a joint distribution D over $S_1 \times S_2$ by $D(s_1, s_2) = D_1(s_1) \cdot D^{s_1}(s_2)$, which has marginals D_1 and D_2. This corresponds to sampling $s_1 \sim D_1$ followed by $s_2 \sim D^{s_1}$.

A technique for studying any distribution D_2 is to design a joint distribution D with D_2 as the 2nd marginal, thereby decomposing D_2 into a mixture of other distributions that may be individually easier to analyze than D_2 itself is.

Example. Suppose D_2 is the following distribution over $S_2 = [4k]$ for some positive integer k:

$$D_2(x) = \tfrac{1}{12k} \text{ if } 0 < x \le k, \qquad D_2(x) = \tfrac{1}{3k} \text{ if } k < x \le 3k, \qquad D_2(x) = \tfrac{1}{4k} \text{ if } 3k < x \le 4k$$

There's hidden structure to uncover: D_2 is the mixture $\frac{1}{4} \cdot D^{\mathrm{H}} + \frac{3}{4} \cdot D^{\mathrm{T}}$ where D^{H} is uniform over $[3k]$ and D^{T} is uniform over $\{k+1, \ldots, 4k\}$. In other words, if we toss a coin with heads probability $1/4$ and tails probability $3/4$ and let $s_1 \in S_1 = \{\mathrm{H}, \mathrm{T}\}$ be the outcome, and then sample $s_2 \sim D^{s_1}$, the joint distribution of (s_1, s_2) indeed has D_2 as its 2nd marginal. Note that D^{H} and D^{T} are *flat* distributions (uniform over their supports), which are simpler to understand than D_2. Consider the event $B = [2k] \subseteq S_2$. It's straightforward to compute $D_2(B) = k \cdot \frac{1}{12k} + k \cdot \frac{1}{3k} = \frac{5}{12}$ directly from the definition of D_2, by partitioning B into $[k] \cup \{k+1, \ldots, 2k\}$. But the mixture yields another method (which may be simpler or more intuitive in other examples): $D^{\mathrm{H}}(B) = 2/3$ and $D^{\mathrm{T}}(B) = 1/3$ since B contains a $2/3$ and $1/3$ fraction of the supports of D^{H} and D^{T} respectively; thus $D_2(B) = \frac{1}{4} \cdot \frac{2}{3} + \frac{3}{4} \cdot \frac{1}{3} = \frac{5}{12}$. ♦

A.7 Random variables

Consider any discrete probability space, with sample space S and distribution D. A *random variable* is a function $X : S \to \mathbb{R}$, which assigns a number to each outcome. For example, if $S = \{\mathrm{H}, \mathrm{T}\}^n$ then the outcome's number of heads is a random variable: $X(\mathrm{HTHHT}) = 3$. "Random variable" is a misnomer since a function X is neither random nor a variable. This terminology evokes the idea of a "variable" that can't be plugged into at will—it's automatically assigned a random value according to a distribution $X(D)$ over some numbers.

A.7.1 Expectation

The *expectation* (or *expected value* or *mean* or *weighted average*) of a random variable X under a distribution D is $\mathbf{E}[X] = \sum_s \mathbf{Pr}_D[s] \cdot X(s)$ (that is, the sum, over all outcomes, of the random variable's value "weighted" by the probability). An alternative form of the definition sums over outcomes of the distribution $X(D)$ rather than over outcomes of D: $\mathbf{E}[X] = \sum_x \mathbf{Pr}_{X(D)}[x] \cdot x$. This follows from the original form by collecting all s with the same value of $X(s)$, say x, and factoring out the x, leaving the probability of the event $X^{-1}(x) = \{s : X(s) = x\}$. Slightly abusing notation, we may denote the latter event "$X = x$".

Example. If $S = \{\mathrm{H}, \mathrm{T}\}$, D is a fair coin toss, and $X(\mathrm{H}) = 0$ and $X(\mathrm{T}) = 10$, then $\mathbf{E}[X] = \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 10 = 5$. "Expected value" is a misnomer since in this example, we never actually expect X to have value 5—it is always either 0 or 10. ♦

$\mathbf{E}[X]$ is a single-number summary of the distribution $X(D)$. In the extreme case where $X(s)$ is the same for all s, $\mathbf{E}[X]$ is that common value.

Example. Revisiting the example from §A.6.3 of two coin tosses with heads probability 1/3 and tails probability 2/3, let X be the number of heads:

s	$D(s)$	$X(s)$
HH	1/9	2
HT	2/9	1
TH	2/9	1
TT	4/9	0

$$\begin{aligned}\mathbf{E}[X] &= \tfrac{1}{9}\cdot 2+\tfrac{2}{9}\cdot 1+\tfrac{2}{9}\cdot 1+\tfrac{4}{9}\cdot 0 && \text{(original form)}\\ &= \tfrac{1}{9}\cdot 2+(\tfrac{2}{9}+\tfrac{2}{9})\cdot 1+\tfrac{4}{9}\cdot 0 && \text{(alternative form)}\\ &= 6/9\\ &= 2/3\end{aligned}$$

♦

Such ad hoc calculations (using the definition of $\mathbf{E}$) get cumbersome for larger examples. Let's develop techniques to make expectations simpler to evaluate.

A common type of random variable is the *indicator* of an event $A \subseteq S$, defined by $X(s) = 1$ if $s \in A$, and $X(s) = 0$ if $s \notin A$ (so X's value indicates whether A occurred). An indicator's expectation is the event's probability: $\mathbf{E}[X] = \mathbf{Pr}[A]\cdot 1 + \mathbf{Pr}[\overline{A}]\cdot 0 = \mathbf{Pr}[A]$.

For any random variable X and constant $c \in \mathbb{R}$, define the random variable cX ("X scaled by factor c") by $(cX)(s) = c \cdot X(s)$ for all outcomes s. The expectation also gets scaled by c since we can factor c out of the definition of $\mathbf{E}$ by the distributive law:

- $\mathbf{E}[cX] = c \cdot \mathbf{E}[X]$ for any random variable X and constant c.

If X_1 and X_2 are random variables on the same probability space, we call them *joint* random variables. We define the random variable X_1+X_2 by $(X_1+X_2)(s) = X_1(s)+X_2(s)$ for all outcomes s. This generalizes to summing more than two joint random variables. The expectation of a sum equals the sum of the expectations, by the distributive law:

- $\mathbf{E}[X_1+\cdots+X_n] = \mathbf{E}[X_1]+\cdots+\mathbf{E}[X_n]$ for any joint random variables $X_1,\dots,X_n$.

The above two bullets together are called *linearity of expectation*. We combine them into a single, general form:

- $\mathbf{E}[c_1X_1+\cdots+c_nX_n] = c_1\cdot\mathbf{E}[X_1]+\cdots+c_n\cdot\mathbf{E}[X_n]$ for any joint random variables $X_1,\dots,X_n$ and constants $c_1,\dots,c_n$.

Linearity of expectation is elementary but surprisingly useful.

Example. Consider n tosses of a coin with heads probability p, so $S = \{\mathrm{H},\mathrm{T}\}^n$ and $D(s) = p^k(1-p)^{n-k}$ where k is the number of heads in s. (Each H contributes a factor p, and each T contributes a factor $1-p$, since the tosses are independent.) Letting the random variable X be the number of heads, $X(D)$ is called the *binomial distribution* and we denote it $B_{n,p}$. For $k \in \{0,1,\dots,n\}$, we have $B_{n,p}(k) = \mathbf{Pr}[X = k] = \binom{n}{k}p^k(1-p)^{n-k}$ since $\binom{n}{k}$ many outcomes have exactly k heads, and they all have the same probability. Now $\mathbf{E}[X] = \sum_{k=0}^{n} B_{n,p}(k)\cdot k$ is tricky to evaluate directly, but it's much simpler when we exploit the power of linearity: For $i \in [n]$, let X_i be the indicator random variable for the event that the i^{th} toss is heads, and note that $\mathbf{E}[X_i] = \mathbf{Pr}_{s\sim D}[s_i = \mathrm{H}] = p$. We have $X = X_1+\cdots+X_n$ (that is, for every outcome s, the random variables X and $X_1+\cdots+X_n$ have the same value as each other) since the sum has a 1 for each

H in s. Now, we calculate

$$\mathbf{E}[X] = \mathbf{E}[X_1]+\cdots+\mathbf{E}[X_n] = p+\cdots+p = np$$

which agrees with our earlier calculation yielding $2/3$ when $n=2$ and $p=1/3$. ♦

Example. If a committee of k people is chosen uniformly at random out of n people, what's the probability you'll be on the committee? Intuitively, since k out of n people get selected, each individual should have a k/n chance of being selected. This is correct, but the reason is subtle since the inclusions of different people are not independent events: Conditioning on person 1 being picked decreases the probability of person 2 being picked, since there are fewer remaining slots to fill. If we instead tossed a coin with heads probability k/n for each person to determine whether they're selected (H = on committee), then the answer to our question would be obvious but the committee's size wouldn't necessarily be k (it would be random with expectation $n \cdot k/n = k$ as shown in the previous example). We need some care to analyze the "fixed committee size" setting.

Formally, for some positive integers $k \le n$, we sample a uniformly random size-k subset of $[n]$. The sample space has size $\binom{n}{k}$, and $D(s) = 1/\binom{n}{k}$ for each $s \subseteq [n]$ of size k. For any particular $i \in [n]$, what's the probability that $i \in s$? The number of outcomes s with $i \in s$ is $\binom{n-1}{k-1}$ since they correspond to choosing $k-1$ of the remaining elements $[n] \setminus \{i\}$ to form the set s with i. Thus

$$\mathbf{Pr}_{s\sim D}[i \in s] = \binom{n-1}{k-1}\Big/\binom{n}{k} = \frac{(n-1)!}{(k-1)!(n-k)!}\Big/\frac{n!}{k!(n-k)!} = k/n$$

since the $(n-k)!$ factors cancel, and $k!/(k-1)!$ cancels except a k in the numerator, and $(n-1)!/n!$ cancels except an n in the denominator.

A slicker way to calculate this exploits linearity: For $i \in [n]$, let X_i be the indicator random variable for the event that $i \in s$, so $\mathbf{Pr}_{s\sim D}[i \in s] = \mathbf{E}[X_i]$. The random variable $X_1+\cdots+X_n$ has value k on every outcome s since it counts the number of elements included in s, which is always $|s| = k$. Thus $\mathbf{E}[X_1]+\cdots+\mathbf{E}[X_n] = \mathbf{E}[X_1+\cdots+X_n] = k$. Due to the symmetry among elements of $[n]$ (nothing is special about any particular i), $\mathbf{E}[X_i]$ is the same for all i. Since the sum of n copies of the same number equals k, this number must be k/n. ♦

It's not generally true that $\mathbf{E}[X_1X_2] = \mathbf{E}[X_1]\cdot\mathbf{E}[X_2]$ where X_1X_2 is the random variable defined by $(X_1X_2)(s) = X_1(s)\cdot X_2(s)$. For example, consider a single fair coin toss, and let X_1 be the indicator for heads and X_2 be the indicator for tails: $\mathbf{E}[X_1] = \mathbf{E}[X_2] = 1/2$ but $\mathbf{E}[X_1X_2] = 0 \neq 1/4$.

A.7.2 Conditioning

For any random variable X and event A with nonzero probability, the *conditional expectation* $\mathbf{E}[X \mid A]$ is X's expectation under the conditioned distribution $(D \mid A)$:

$$\mathbf{E}_D[X \mid A] = \mathbf{E}_{D\mid A}[X] = \textstyle\sum_s \mathbf{Pr}_{D\mid A}[s]\cdot X(s) = \sum_x \mathbf{Pr}_{X(D\mid A)}[x]\cdot x$$

(Recall that $\mathbf{Pr}_{D\mid A}[s] = \mathbf{Pr}_D[s \mid A]$ and $\mathbf{Pr}_{X(D\mid A)}[x] = \mathbf{Pr}_D[X = x \mid A]$.)

The *law of total expectation* (analogous to the law of total probability) states that if $A_1,\dots,A_n$ form a partition of S, then $\mathbf{E}[X]=\sum_{i=1}^n \mathbf{Pr}[A_i]\cdot\mathbf{E}[X\,|\,A_i]$ (omitting any terms for which $\mathbf{Pr}[A_i]=0$). To verify this law, note that $\mathbf{E}[X\,|\,A_i]=\sum_{s\in A_i}(\mathbf{Pr}[s]/\mathbf{Pr}[A_i])\cdot X(s)$ by the definition of conditioning, and plugging this into the right side of the law and cancelling $\mathbf{Pr}[A_i]$ factors yields $\sum_{i=1}^n\sum_{s\in A_i}\mathbf{Pr}[s]\cdot X(s)$, which equals $\mathbf{E}[X]$ since each s is accounted for exactly once.

This gives a technique for calculating a complicated random variable's expectation: Design an appropriate partition and apply the law of total expectation.

Example. Let's apply this technique to the uniform distribution over an interval of integers: $S=\{a,a+1,\dots,b-1,b\}$, and $D(s)=1/(b-a+1)$ and $X(s)=s$ for each $s\in S$. Intuitively, the expectation should be the middle of the interval, $\mathbf{E}[X]=(a+b)/2$, because the distribution is symmetric about this point. To formalize this, we partition S into parts $A_0=\{a,b\}$, $A_1=\{a+1,b-1\}$, $A_2=\{a+2,b-2\}$, $\dots$, and if $|S|$ is odd then there's also a singleton part $A_{(b-a)/2}=\{(a+b)/2\}$. Conditioned on $A_i=\{a+i,b-i\}$, those two outcomes each have probability $1/2$, so $\mathbf{E}[X\,|\,A_i]=\frac{1}{2}\cdot(a+i)+\frac{1}{2}\cdot(b-i)=(a+b)/2$. If $|S|$ is odd and $i=(b-a)/2$ then also $\mathbf{E}[X\,|\,A_i]=(a+b)/2$. Thus $\mathbf{E}[X]=\sum_i\mathbf{Pr}[A_i]\cdot\mathbf{E}[X\,|\,A_i]=\frac{a+b}{2}\cdot\sum_i\mathbf{Pr}[A_i]=(a+b)/2$. The specific values of $\mathbf{Pr}[A_i]$ didn't matter, so this reasoning applies to any symmetric distribution. ♦

Example. Suppose we repeatedly toss a coin with heads probability $p>0$ until we see a heads, and let X be the total number of tosses (for example, $X(\text{TTTH})=4$). The theory of finite sample spaces still works even though this sample space is infinite. Technically, "never seeing a heads" ($X=\infty$) is a possible outcome, but it has probability $(1-p)^\infty=0$ since $p>0$, so we ignore it.

If $X(s)=k$ (so s is $k-1$ Ts followed by one H) then $\mathbf{Pr}_{X(D)}[k]=\mathbf{Pr}_D[s]=(1-p)^{k-1}p$ since the tosses are independent. As a sanity check, let's make sure these probabilities sum to 1. We use the telescoping trick for evaluating a geometric series (§A.5.1). Letting $q=1-p$ for brevity:

$$\begin{aligned}\textstyle\sum_s\mathbf{Pr}[s] \;=\; \sum_{k=1}^\infty(1-p)^{k-1}p \;&=\; (1-q)\textstyle\sum_{k=1}^\infty q^{k-1}\\ &=\; \big(q^0+q^1+q^2+\cdots\big)-\big(q^1+q^2+q^3+\cdots\big) \;=\; q^0 \;=\; 1\end{aligned}$$

(We let $0^0=1$ if $p=1$.) Because of this geometric series, $X(D)$ is called the *geometric distribution* with parameter p.

What is the geometric random variable's expectation? The definition gives the series $\mathbf{E}[X]=\sum_{k=1}^\infty(1-p)^{k-1}p\cdot k$, which is tricky to evaluate directly. Instead of wading through that morass, let's partition the sample space into the event $A=\{\text{H}\}$ that the first toss is already heads, and the event $\overline{A}$ that more than one toss occurs. We have $\mathbf{E}[X\,|\,A]=X(\text{H})=1$. A key observation is that if the first toss is tails, then the number of additional tosses to see a heads is distributed exactly as in the original experiment: $\mathbf{E}[X\,|\,\overline{A}]=1+\mathbf{E}[X]$. Plugging this in, we get

$$\mathbf{E}[X] \;=\; \mathbf{Pr}[A]\cdot\mathbf{E}[X\,|\,A]+\mathbf{Pr}[\overline{A}]\cdot\mathbf{E}[X\,|\,\overline{A}] \;=\; p\cdot 1+(1-p)\cdot(1+\mathbf{E}[X]) \;=\; 1+(1-p)\cdot\mathbf{E}[X]$$

which rearranges to $\mathbf{E}[X]=1/p$. (Caution: Some random variables on infinite probability spaces have expectation ∞ since the series in the definition may diverge. For the geometric random variable, the expectation is finite since $p>0$.) ♦

Recall from §A.6.6 that sampling (s_1,s_2) from a joint distribution D over $S_1\times S_2$ is equivalent to sampling $s_1\sim D_1$ and then $s_2\sim D^{s_1}$ where $D^{s_1}(s_2)=D(s_1,s_2)/D_1(s_1)$ is D's 2nd marginal after conditioning on the 1st coordinate being s_1.

For any random variable $X\colon S_1 \times S_2 \to \mathbb{R}$, we can express $\mathbf{E}[X]$ as an "expectation of expectations": For each $s_1 \in S_1$, define the random variable $X^{s_1}\colon S_2 \to \mathbb{R}$ by $X^{s_1}(s_2) = X(s_1, s_2)$ (hardcoding s_1 as the 1st argument). Define the random variable $X_1\colon$ (D_1's support) $\to \mathbb{R}$ by $X_1(s_1) = \mathbf{E}_{D^{s_1}}[X^{s_1}] = \mathbf{E}_D[X \mid A^{s_1}]$ where $A^{s_1} = \{s_1\} \times S_2$ (so X_1's values are themselves expectations). The punchline is $\mathbf{E}_D[X] = \mathbf{E}_{D_1}[X_1]$, in other words:

$$\mathbf{E}_{(s_1,s_2)\sim D}\big[X(s_1,s_2)\big] \;=\; \mathbf{E}_{s_1\sim D_1}\Big[\mathbf{E}_{s_2\sim D^{s_1}}\big[X(s_1,s_2)\big]\Big]$$

This provides a technique for computing X's expectation: For each s_1 separately, find X's expectation conditioned on s_1, and then combine these values by taking the expectation over the random choice of s_1. In particular, if X is the indicator of an event $B \subseteq S_1 \times S_2$, this lets us express $\mathbf{Pr}_D[B]$ as an "expectation of probabilities":

$$\mathbf{Pr}_{(s_1,s_2)\sim D}\big[(s_1,s_2)\in B\big] \;=\; \mathbf{E}_{s_1\sim D_1}\Big[\mathbf{Pr}_{s_2\sim D^{s_1}}\big[(s_1,s_2)\in B\big]\Big]$$

The $\mathbf{Pr}$ inside the $\mathbf{E}$ is the value of $X_1(s_1)$.

A.7.3 Independence

Joint random variables X_1 and X_2 are independent when for every pair of numbers (x_1, x_2), the events "$X_1 = x_1$" and "$X_2 = x_2$" are independent (§A.6.3). This generalizes to full independence of more than two joint random variables. In a probability space with built-in full independence—that is, $D(s_1,\ldots,s_n) = \prod_{i=1}^n D_i(s_i)$—any random variables $X_1,\ldots,X_n$ where X_i only depends on s_i are fully independent. For example, if we roll a die n times and let X_i be the ith roll mod 3, then $X_1,\ldots,X_n$ are fully independent.

It's not generally true that $\mathbf{E}\big[\prod_{i=1}^n X_i\big] = \prod_{i=1}^n \mathbf{E}[X_i]$ (expectation of product equals product of expectations), but this does hold if $X_1,\ldots,X_n$ are fully independent. To verify this, we use neither the original definition of $\mathbf{E}$ (sum over outcomes of the underlying probability space) nor the alternative form (sum over outcomes of the random variable $\prod_{i=1}^n X_i$) but rather an intermediate form that sums over all tuples of outcomes of the individual random variables $X_1,\ldots,X_n$:

$$\begin{aligned}
\mathbf{E}\big[\textstyle\prod_{i=1}^n X_i\big] &= \textstyle\sum_{x_1,\ldots,x_n} \mathbf{Pr}\big[X_1 = x_1 \text{ and } \cdots \text{ and } X_n = x_n\big]\cdot \prod_{i=1}^n x_i \\
&= \textstyle\sum_{x_1,\ldots,x_n} \prod_{i=1}^n (\mathbf{Pr}[X_i = x_i]\cdot x_i) \\
&= \textstyle\prod_{i=1}^n \sum_{x_i} \mathbf{Pr}[X_i = x_i]\cdot x_i \\
&= \textstyle\prod_{i=1}^n \mathbf{E}[X_i]
\end{aligned}$$

The second line follows from the first by full independence of the "$X_i = x_i$" events, and the third line equals the second by completely expanding the product-of-sums (picking one summand from each factor, in all possible ways) to a sum-of-products. Let's see an application.

Example. Consider n tosses of a coin with heads probability p. What's the probability of getting an odd number of heads (that is, a sample from the binomial distribution $B_{n,p}$ is odd)? In §A.6.5 we showed the answer is $1/2$ if $p = 1/2$, but that method doesn't apply when the coin is biased ($p \neq 1/2$). The new insight is to translate the question about oddness into a question about products: If we multiply together some factors of -1, then each factor makes the result flip between 1 and -1, so the result is -1 if there are an odd number of factors, and is 1 otherwise. Defining $X_i(s) = -1$ if $s_i = \mathrm{H}$, and $X_i(s) = 1$ if $s_i = \mathrm{T}$, we have $(\prod_i X_i)(s) = -1$ if s has an odd number of heads, and $(\prod_i X_i)(s) = 1$ if s has an even number of heads. We have $\mathbf{E}[X_i] = p \cdot (-1) + (1-p) \cdot 1 = 1 - 2p$. Letting A be the event of getting an odd number of heads, $\mathbf{E}\big[\prod_i X_i\big] = \mathbf{Pr}[A] \cdot (-1) + (1 - \mathbf{Pr}[A]) \cdot 1 = 1 - 2 \cdot \mathbf{Pr}[A]$. Since $X_1, \ldots, X_n$ are fully independent:

$$1 - 2 \cdot \mathbf{Pr}[A] \;=\; \mathbf{E}\big[\textstyle\prod_i X_i\big] \;=\; \prod_i \mathbf{E}[X_i] \;=\; \prod_i (1-2p) \;=\; (1-2p)^n$$

Rearranging yields $\mathbf{Pr}[A] = \frac{1}{2} - \frac{1}{2} \cdot (1-2p)^n$. If tails is more likely than heads ($p < 1/2$), then the probability is less than $1/2$ but approaches $1/2$ as n increases (unless $p = 0$). If heads is more likely than tails ($p > 1/2$), then the probability alternates between above and below $1/2$ but approaches $1/2$ as n increases (unless $p = 1$). ♦

A.8 Finite fields

A.8.1 Modular arithmetic

Example. With the 24-hour clock ("military time"), hours are numbered $0, 1, 2, \ldots, 23$. If it's 20 o'clock, what time will it be 31 hours from now? Adding $20 + 31$ gives 51, which isn't a valid time. Starting at 20 and incrementing 31 times: After 4 increments we hit 24, which "wraps around" to 0, then after 24 more increments it wraps around to 0 again, and finally 3 more increments brings us to 3 o'clock. We write this as $(20+31) \bmod 24 = 3$ where "mod" is the *modulo* operator. This also goes the other way: If it's 20 o'clock, what time was it 31 hours ago? After 21 decrements we hit -1, which wraps around to 23, and then 10 more decrements brings us to 13 o'clock, so $(20-31) \bmod 24 = 13$. ♦

Arrange the numbers $0, 1, 2, \ldots, m-1$ (for some integer $m > 1$) clockwise around a circle. Then $a \bmod m$ (for any integer a) is the number we end at if we start at 0 and take a steps clockwise—or if a is negative, this means $|a|$ steps counter-clockwise. Another perspective: If we interpret "a divided by m" as decomposing $a = q \cdot m + r$ with *quotient* $q \in \mathbb{Z} = \{\ldots, -2, 1, 0, 1, 2, \ldots\}$ and *remainder* $r \in \mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$, then $r = a \bmod m$. If a is nonnegative, then q is the number of full trips around the circle, and r is the number of remaining steps after the last visit to 0. If a is negative, then q is negative and $|q|$ is the number of full counter-clockwise trips we would take if, after landing on r (after $|a|$ steps), we finish the last cycle by continuing around to 0 (if we're not already there). For our earlier example, $20 + 31 = 51 = 2 \cdot 24 + 3$ and $20 - 31 = -11 = -1 \cdot 24 + 13$.

All major programming languages provide a mod operator, but they don't all agree with the mathematical definition on negative numbers: In Java, `(-10) % 3` evaluates to `-1`, but mathematically it should be $(-10) \bmod 3 = 2$ since $-10 = -4 \cdot 3 + 2$ (and the remainder must be in $\mathbb{Z}_3 = \{0, 1, 2\}$).

Example.

a	$\cdots$	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	$\cdots$
$a \bmod 4$	$\cdots$	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	$\cdots$

♦

"mod m" partitions the set of all integers $\mathbb{Z}$ into m subsets: For each $r \in \mathbb{Z}_m$, the r^{th} subset $\{\ldots, -2m+r, -m+r, 0+r, m+r, 2m+r, \ldots\}$ contains all integers with remainder r upon division by m, and r itself serves as the "representative" from this subset. All integers from the same subset are equivalent to each other for the purposes of plus, minus, and times, so we might as well always use the representative. More precisely:

$$\begin{aligned}(a+b) \bmod m &= \big((a \bmod m)+(b \bmod m)\big) \bmod m\\ (a-b) \bmod m &= \big((a \bmod m)-(b \bmod m)\big) \bmod m\\ (a \cdot b) \bmod m &= \big((a \bmod m) \cdot (b \bmod m)\big) \bmod m\end{aligned}$$

To verify this, write $a = q_1 m + r_1$ (so $a \bmod m = r_1$) and $b = q_2 m + r_2$ (so $b \bmod m = r_2$). For the addition property, writing $r_1 + r_2 = q_3 m + r_3$ (so $(r_1+r_2) \bmod m = r_3$) we get:

$$a+b \;=\; (q_1+q_2)m+(r_1+r_2) \;=\; (q_1+q_2+q_3)m+r_3 \quad (\text{so } (a+b) \bmod m = r_3)$$

The subtraction property is similar. For the multiplication property, writing $r_1 \cdot r_2 = q_4 m + r_4$ (so $(r_1 \cdot r_2) \bmod m = r_4$) we get:

$$a\cdot b \;=\; (q_1q_2m+q_1r_2+q_2r_1)m+(r_1r_2) \;=\; (q_1q_2m+q_1r_2+q_2r_1+q_4)m+r_4 \quad (\text{so } (a\cdot b) \bmod m = r_4)$$

Upshot: We can simplify a "runaway" calculation by "confining" it to $\mathbb{Z}_m$.

Example. Let's evaluate $((19+13)\cdot 17) \bmod 7$. Instead of calculating $544 \bmod 7 = 5$, we can first mod the individual numbers by 7 to get $((5+6)\cdot 3) \bmod 7$, then simplify $(5+6) \bmod 7 = 11 \bmod 7 = 4$ to get $(4 \cdot 3) \bmod 7$, which finally yields $12 \bmod 7 = 5$. ♦

A.8.2 Fields

The set of real numbers $\mathbb{R}$ forms a *field*: It contains two special numbers 0 and 1 and is equipped with four arithmetic operators $+, -, \cdot, /$ (with division by 0 disallowed) obeying these familiar *axioms* for all numbers a, b, c (not necessarily distinct):

Commutative:	$a+b=b+a$ and $a\cdot b=b\cdot a$
Associative:	$a+(b+c)=(a+b)+c$ and $a\cdot(b\cdot c)=(a\cdot b)\cdot c$
Distributive:	$a\cdot(b+c)=(a\cdot b)+(a\cdot c)$
Identity:	$a+0=a$ and $a\cdot 1=a$
Additive inverse:	$a-a=0$ and $b-a=b+(0-a)$
Multiplicative inverse:	$a/a=1$ and $b/a=b\cdot(1/a)$ assuming $a\neq 0$

It's possible (though not entirely trivial) to see that these axioms imply other familiar properties such as $a\cdot 0=0$, and $a\cdot(b-c)=(a\cdot b)-(a\cdot c)$, and $(a+b)/c=(a/c)+(b/c)$, and $a\cdot b\neq 0$ when $a\neq 0$ and $b\neq 0$. So we need not include those properties in the list of axioms.

The notations $-a$ and a^{-1} are shorthands for the numbers $0-a$ and $1/a$. The inverse axioms imply that the $-$ and $/$ operators are redundant: Subtracting a means adding $-a$, and dividing by a means multiplying by a^{-1}. It also follows from the axioms that inverses are unique (if

$a+b=0$ then $b=-a$, and if $a\cdot b=1$ then $b=a^{-1}$) and that $-(-a)=a=(a^{-1})^{-1}$ and $-(a+b)=(-a)+(-b)$ and $(a\cdot b)^{-1}=a^{-1}\cdot b^{-1}$.

The set of rationals $\mathbb{Q}$ also forms a field, since applying any of the four arithmetic operators to two rational numbers always produces a rational. The set of integers $\mathbb{Z}$, however, does not form a field: Though the first five axioms hold, the multiplicative inverse axiom is problematic because $/$ isn't well-defined on $\mathbb{Z}$ (since b/a might not be an integer when b and $a\neq 0$ are integers).

For computer science applications, an issue with $\mathbb{R}$ and $\mathbb{Q}$ is that they're infinite: We can't represent arbitrary elements exactly in a computer with finite memory. Approximate representations such as floating point numbers unfortunately introduce precision and rounding issues. Some applications require the arithmetic properties of all six axioms, but greatly benefit from having only finitely many numbers to juggle, so that the numbers can always be specified exactly. This is where *finite fields* enter the scene.

Consider the finite set $\mathbb{Z}_m=\{0,1,2,\ldots,m-1\}$ for some integer $m>1$. Define arithmetic operators $+,-,\cdot:\mathbb{Z}_m\times\mathbb{Z}_m\to\mathbb{Z}_m$ by:

$$a+b=(a+b)\bmod m \qquad a-b=(a-b)\bmod m \qquad a\cdot b=(a\cdot b)\bmod m$$

The $+,-,\cdot$ notation is overloaded here: On the left sides of these equalities, $+,-,\cdot$ are the new operators being defined, and on the right sides, $+,-,\cdot$ are the ordinary integer operators. What about $/$? Though the first five field axioms hold for $\mathbb{Z}_m$ like for $\mathbb{Z}$, division is still problematic. Let's not throw out the baby with the bathwater. The last axiom says to think of division by a as multiplication by the multiplicative inverse a^{-1}, which is some field element such that $a\cdot a^{-1}=1$. From this perspective, division in $\mathbb{Z}_m$ can actually make sense, though unfortunately some nonzero elements might have no multiplicative inverse.

Example. In the context of $\mathbb{Z}_9$, we simply write $5+7=3$ and $5-7=7$ and $5\cdot 7=8$ without specifying "mod 9" everywhere. We would ordinarily think of $5/7$ as $0.714\cdots$, which is not an element of $\mathbb{Z}_9$. (Modular arithmetic doesn't accommodate non-integers.) Since $7\cdot 4=1$ in $\mathbb{Z}_9$ (and 4 is the only such element of $\mathbb{Z}_9$), we deem 4 the multiplicative inverse of 7 and denote $7^{-1}=4$. Now, we treat division by 7 as multiplication by 4: We have $5/7=5\cdot 4=2$ in $\mathbb{Z}_9$. Unfortunately, 3 has no multiplicative inverse in $\mathbb{Z}_9$: There's no c such that $3\cdot c=1$. ♦

In summary, in $\mathbb{Z}_m$ we interpret b/a as $b\cdot a^{-1}$ where a^{-1} is some element of $\mathbb{Z}_m$ such that $a\cdot a^{-1}=1$ (all mod m, of course). Fortunately, if $p>1$ is a *prime* number (not divisible by any positive integer besides 1 and p itself), then every nonzero $a\in\mathbb{Z}_p$ does have a (unique) multiplicative inverse. We prove this non-obvious fact in §B.8.1 and take it for granted for now.

Example. $p=7$ is prime. Here are the multiplicative inverses in $\mathbb{Z}_7$:

a	0	1	2	3	4	5	6
a^{-1}		1	4	5	2	3	6

For example, $3^{-1}=5$ since $3\cdot 5=1$ (that is, $15 \bmod 7=1$), and $6^{-1}=6$ since $6\cdot 6=1$ (that is, $36 \bmod 7=1$). Thus in $\mathbb{Z}_7$:

$$2/(4/3)\;=\;2/(4\cdot 5)\;=\;2/6\;=\;2\cdot 6\;=\;5$$

In $\mathbb{R}$ we could rearrange $2/(4/3)$ to $(2\cdot 3)/4$ to avoid doing two divisions, and the same is possible in $\mathbb{Z}_7$: $(4/3)^{-1} = 3/4$ because $(4/3)\cdot(3/4) = (4\cdot 3^{-1})\cdot(3\cdot 4^{-1}) = (4\cdot 4^{-1})\cdot(3\cdot 3^{-1}) = 1\cdot 1 = 1$ (by commutativity and associativity), and thus $2/(4/3) = 2\cdot(3/4) = (2\cdot 3)/4 = 6\cdot 2 = 5$. The point is that such familiar manipulations follow from the field axioms, letting us do finite field arithmetic with the comfort we're accustomed to. ♦

Example. Here are the tables for the operators $+, -, \cdot, / : \mathbb{Z}_5 \times \mathbb{Z}_5 \to \mathbb{Z}_5$, where $/$ is a partial function whose value is undefined when the second argument is 0 (hence the 0 column is blank).

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$-$	0	1	2	3	4
0	0	4	3	2	1
1	1	0	4	3	2
2	2	1	0	4	3
3	3	2	1	0	4
4	4	3	2	1	0

$\cdot$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

$/$	0	1	2	3	4
0		0	0	0	0
1		1	3	2	4
2		2	1	4	3
3		3	4	1	2
4		4	2	3	1

The 0 row of the $-$ table shows the additive inverse of every field element, and the 1 row of the $/$ table shows the multiplicative inverses. We can evaluate arithmetic expressions in the field $\mathbb{Z}_5$ by consulting these tables, for example $(1-2)/(2-4) = 4/3 = 3$. ♦

The smallest field is $\mathbb{Z}_2$, in which $\cdot$ is logical "and" (viewing 1 as T and 0 as F) and $+$ is xor. This is why the symbol $\oplus$ for xor (§A.3.1) has a plus sign inside it.

We mentioned that $\mathbb{Z}_9$ is not a field. In fact, $\mathbb{Z}_m$ is never a field if m is not prime. To see why not, suppose a is a nontrivial divisor of m, that is, $1 < a < m$ and $m = da$ for some integer d. Then a can't have a multiplicative inverse: For every $c \in \mathbb{Z}$, we have $ac \bmod m \neq 1$ because if we write $ac = qm + r$ where $r = ac \bmod m$ is the remainder, and we plug in $m = da$ and rearrange, then we get $r = (c - qd)a$ which is a multiple of $a > 1$ and therefore can't be 1.

What other finite fields exist besides $\mathbb{Z}_p$ for prime p? For nonprime m, $\mathbb{Z}_m$ isn't a field, but maybe a field with exactly m elements still exists? In fact, if m is a prime power ($m = p^e$ for a prime p and an integer exponent $e \geq 1$) then a field of size m exists, and if m isn't a prime power then no field of size m exists.

A.9 Polynomials

A.9.1 Univariate polynomials

A *polynomial* is an expression consisting of a sum of *monomials*, where each monomial is a coefficient times a variable x raised to a natural number power. When we say a polynomial P is over a particular field $\mathbb{F}$, this means:

- Every coefficient is an element of $\mathbb{F}$.
- The elements of $\mathbb{F}$ may be plugged in for x.
- We evaluate P using $\mathbb{F}$'s arithmetic operators.

A polynomial P over $\mathbb{F}$ expresses a function $P : \mathbb{F} \to \mathbb{F}$. We slightly abuse notation by letting P denote both the expression and the function.

Example. In $P(x) = 4x^3 + 3x^2 - 5x + 6$ we view $-5x$ as $+(-5)x^1$ and view 6 as $6 \cdot 1 = 6x^0$. If we view $Q(x) = x^2 + 3x + 1$ as a polynomial over $\mathbb{R}$, we can evaluate $Q(2) = 2^2 + 3 \cdot 2 + 1 = 11$. If we instead view the same $Q(x)$ as a polynomial over $\mathbb{Z}_5$, we get $Q(2) = 1$ because $11 \bmod 5 = 1$. ⧫

If several monomials have the same power of x, we can collect them into a single monomial by summing the coefficients. A polynomial P's *degree*, denoted $\deg(P)$, is the highest power of x with a nonzero coefficient, after collecting terms so that all monomials have different powers of x. (The word "degree" is overloaded since it has a different meaning for nodes in graphs.)

Example. $2x^3 + 5x^3 + x^2 + 4x^2 + 5x^2$ has degree 3 over $\mathbb{R}$ since it equals $(2+5)x^3 + (1+4+5)x^2 = 7x^3 + 10x^2$, but it has degree 2 over $\mathbb{Z}_7$ since it equals $0x^3 + 3x^2 = 3x^2$. ⧫

A polynomial over a specified field has a *canonical form*

$$P(x) \;=\; c_d \cdot x^d \,+\, c_{d-1} \cdot x^{d-1} \,+\, \cdots \,+\, c_2 \cdot x^2 \,+\, c_1 \cdot x \,+\, c_0$$

where d is the degree, $c_d \neq 0$ is the *leading coefficient*, and c_0 is the *constant term* (the coefficient of x^0, where $0^0 = 1$). A polynomial has degree 0 when its canonical form is just a nonzero constant term. The polynomial whose coefficients are all 0 is the *zero polynomial* and has degree $-\infty$, for reasons that we clarify shortly.

When polynomials have the same canonical form as each other, we consider them the same polynomial, even if their expressions look different. Over $\mathbb{R}$, it turns out that polynomials with different canonical forms express different functions, but over finite fields, that's not always true. For example, x^2 and x are distinct canonical-form polynomials over $\mathbb{Z}_2$, but they express the same function (which maps 0 to 0 and 1 to 1). However, over the field $\mathbb{Z}_p$ (for a prime p), distinct canonical-form polynomials of degree $< p$ indeed express different functions—we explain why shortly.

We can add and multiply polynomials with each other. Suppose $P(x) = \sum_k b_k \cdot x^k$ and $Q(x) = \sum_k c_k \cdot x^k$ are two canonical-form polynomials over the same field. We naturally define $P + Q$ as the polynomial $\sum_k (b_k + c_k)x^k$, so that $(P + Q)(x) = P(x) + Q(x)$ no matter what value of x we plug in. Summing polynomials can't make a new power of x appear with a nonzero coefficient:

- $\deg(P + Q) \leq \max\bigl(\deg(P), \deg(Q)\bigr)$

This "$\leq$" becomes "$=$" if P and Q have different degrees, or they're both zero, or they have the same degree and their leading coefficients aren't additive inverses of each other. Otherwise, the leading coefficients would cancel out, and the degree would go down. The above inequality holds even when P or Q is the zero polynomial of degree $-\infty$. Subtraction of polynomials is completely analogous to addition.

Next, we discuss multiplication of P and Q. To multiply two monomials, we multiply the coefficients and add the powers: $(b_i x^i) \cdot (c_j x^j) = (b_i c_j) x^{i+j}$. We obtain the polynomial $P \cdot Q$ by multiplying each pair of monomials—picking one monomial from P and one from Q, in all possible ways—summing the results, and collecting monomials with the same power: $\sum_k (b_k c_0 + b_{k-1} c_1 + b_{k-2} c_2 + \cdots + b_0 c_k) \cdot x^k$. This ensures $(P \cdot Q)(x) = P(x) \cdot Q(x)$ no matter what value of x we plug in.

Example. Over $\mathbb{R}$:

$$\begin{aligned}(3x^2+2x+1)\cdot(5x+4) &= (3\cdot 5)x^3+(3\cdot 4+2\cdot 5)x^2+(2\cdot 4+1\cdot 5)x+(1\cdot 4)\\ &= 15x^3+22x^2+13x+4\end{aligned}$$ ♦

If P and Q are nonzero, then the product of their leading coefficients (which are nonzero) is nonzero and forms the leading coefficient of $P\cdot Q$:

- $\deg(P\cdot Q)=\deg(P)+\deg(Q)$

This equality holds even when P or Q is the zero polynomial, since $-\infty$ plus anything (a natural number or $-\infty$) is still $-\infty$.

A *root* of a polynomial is any field element a such that $P(a)=0$.

Example. x^3+x^2+2x+2 has only one root $x=-1$ over $\mathbb{R}$ but has two roots $x=1$ and $x=2$ over $\mathbb{Z}_3$. ♦

A classic fact, which we prove in §B.8.2, is that every nonzero polynomial of degree d has at most d roots in its field. Over $\mathbb{R}$, this means the plot of a polynomial P crosses or touches the x-axis at most $\deg(P)$ times. If P and Q are distinct polynomials of degree $\le d$ over the same field, then $P(a)=Q(a)$ holds for at most d many field elements a, because if $P(a)-Q(a)=0$ then a is a root of the nonzero polynomial $P-Q$, and there are at most d many such roots since $\deg(P-Q)\le d$. In particular, over the field $\mathbb{Z}_p$, distinct polynomials of degree $<p$ express different functions, because $P(a)=Q(a)$ holds for at most $p-1$ of the p possible values of a, and so $P(a)\ne Q(a)$ for at least one $a\in\mathbb{Z}_p$.

We just discussed *univariate* polynomials, which have only one variable.

A.9.2 Multivariate polynomials

A *multivariate* polynomial $P(x_1,x_2,\ldots,x_n)$ has multiple variables. Each monomial is a coefficient times a product of variables raised to natural number powers. For example, $P(x_1,x_2,x_3)=5x_1x_2^4x_3-6x_2x_3^3+7x_1^2x_2$ has three variables and three monomials. A variable that doesn't appear in a monomial has exponent 0 in that monomial. A polynomial P in n variables over a field $\mathbb{F}$ expresses a function $P\colon\mathbb{F}^n\to\mathbb{F}$ (letting P denote both the expression and the function): We evaluate $P(a_1,\ldots,a_n)$ by plugging in $x_i=a_i$ (for each i) and using $\mathbb{F}$'s arithmetic.

If several monomials have the same pattern of powers of the variables, then we can collect them into a single monomial by summing the coefficients and picking a unique order to write the variables, for example $x^6y^7z^8+3y^7z^8x^6=4x^6y^7z^8$. This lets us write a polynomial in its canonical form, where each pattern of powers of variables appears in only one monomial (that is, it has a unique coefficient, possibly 0). We consider two polynomials to be the same when they have the same canonical form, meaning that for each pattern of powers, the coefficients are the same. Over $\mathbb{R}$, distinct polynomials (on the same variables but with distinct canonical forms) express different functions, but that's not always true over finite fields.

Defining "degree" for multivariate polynomials is subtler than in the univariate setting, because different variables can have different powers in a monomial. One useful definition is *total degree*. A monomial's total degree is the sum of the exponents of the variables. A polynomial's total degree is the maximum total degree of any monomial with a nonzero coefficient in the canonical form. A nonzero constant has total degree 0. The zero polynomial has total degree $-\infty$.

Example. $P(x,y) = 9x^4y^3 + 2y^3x^4 + xy^5$ has total degree 7 over $\mathbb{R}$ because it equals $11x^4y^3 + xy^5$ and the sum of exponents is $4+3=7$ for the first monomial and only $1+5=6$ for the second monomial. But this $P(x,y)$ has total degree 6 over $\mathbb{Z}_{11}$ because it equals $0x^4y^3 + xy^5 = xy^5$. ◆

A particular variable's *individual degree* is the highest power the variable has in any monomial with a nonzero coefficient in the polynomial's canonical form. When focusing on a particular variable, it's often useful to rearrange the expression so it looks like a univariate polynomial whose "coefficients" are multivariate polynomials in the other variables.

Example. In $5x^2yz + 7xy^4 + 9xy^2z^3 + 8$ the individual degrees of x, y, z are 2, 4, 3 respectively, and the total degree is 6. Singling out x and rearranging yields $(5yz)\cdot x^2 + (7y^4 + 9y^2z^3)\cdot x + 8$. The degree of x in the latter "univariate perspective" is the individual degree of x in the original multivariate polynomial. ◆

Over a finite field $\mathbb{Z}_p$ (for a prime p), distinct polynomials express different functions if every variable has individual degree $< p$ in both polynomials.

We let $\deg(P)$ denote the total degree and $\deg_{x_i}(P)$ denote the individual degree of a variable x_i. Adding and multiplying multivariate polynomials (that have the same variables as each other) is analogous to the univariate case.

- $\deg(P+Q) \le \max\big(\deg(P), \deg(Q)\big)$ and $\deg_{x_i}(P+Q) \le \max\big(\deg_{x_i}(P), \deg_{x_i}(Q)\big)$
- $\deg(P \cdot Q) = \deg(P) + \deg(Q)$ and $\deg_{x_i}(P \cdot Q) = \deg_{x_i}(P) + \deg_{x_i}(Q)$

For $\deg(P \cdot Q) = \deg(P) + \deg(Q)$, although it's elementary to see that "$\le$" holds (since the same holds if P and Q are monomials), it's not obvious that equality always holds (since it's plausible that after multiplying each monomial of P by each monomial of Q, all the resulting monomials of highest total degree might collectively cancel each other out). The verification of this property is Exercise A.27.

When multiplying more than two multivariate polynomials, we can obtain the canonical form by selecting one monomial from each factor, multiplying the selected monomials (by multiplying the coefficients and summing the powers for each variable), and doing this in all possible ways and summing the resulting monomials.

Example. Over $\mathbb{Z}_7$, we can expand

$$(x+2y)(x+4z)(y+5z) \;=\; x^2y + 5x^2z + 4xyz + 6xz^2 + 2xy^2 + 3xyz + y^2z + 5yz^2$$

and the xyz monomial disappears from the canonical form since its coefficient is $4+3=0$. ◆

One connection to computer science is that a polynomial expression is a kind of algorithm for computing a function. Just as one algorithm may be more efficient than another for the same problem, one expression may be evaluated using fewer operations than another equivalent expression. For example, we can evaluate $P(x_1, x_2, \ldots, x_n) = (x_1+1)(x_2+1)\cdots(x_n+1)$ using n additions (each $x_i + 1$) and $n-1$ multiplications, whereas its canonical form $\sum_{S\subseteq[n]}\prod_{i\in S} x_i$ has 2^n monomials and thus involves $2^n - 1$ additions (and many multiplications).

A.10 Matrices

A.10.1 Matrix structure

A *vector* is a tuple of numbers (a 1-dimensional array). A *scalar* is an individual number (a "0-dimensional array"). A *matrix* is a rectangular 2-dimensional array of numbers. We write matrices inside brackets. An $m \times n$ matrix has m rows and n columns. The *entries* are the individual numbers in a matrix A, and $A_{i,j}$ is the entry in row i and column j (where row 1 is the top, row m is the bottom, column 1 is the leftmost, and column n is the rightmost). The *transpose* of an $m \times n$ matrix A is the $n \times m$ matrix $A^\top$ such that $A^\top_{j,i} = A_{i,j}$ for each $i \in [m]$ and $j \in [n]$. In other words, $A^\top$ is A mirrored so upper-right and lower-left are swapped.

Example. Here are a 3×4 matrix A and its 4×3 transpose $A^\top$:

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} \qquad A^\top = \begin{bmatrix} 0 & 4 & 8 \\ 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \end{bmatrix}$$

Two of A's entries are $A_{1,4} = 3$ and $A_{3,2} = 9$. ♦

We encountered matrices for representing functions of two arguments: Suppose $f : S_1 \times S_2 \to \mathbb{R}$ is a function, and each set S_i has an ordering (one element is designated as the 1st, another as the 2nd, and so on). Define a $|S_1| \times |S_2|$ matrix F by $F_{i,j} = f\big(i^{\text{th}}$ element of S_1, j^{th} element of $S_2\big)$. In particular, we may view a joint probability distribution over sample space $S_1 \times S_2$ as a matrix in this way.

We can treat vectors as matrices. Two orientations are possible: A $1 \times n$ matrix is a *row vector*. An $m \times 1$ matrix is a *column vector*. A row vector's transpose is a column vector, and vice versa. If v is a row vector, we abbreviate $v_{1,j}$ as v_j. If v is a column vector, we abbreviate $v_{i,1}$ as v_i. We can treat a scalar as a 1×1 matrix.

In a *zero* vector or matrix, all entries are 0. In a *binary* vector or matrix, all entries come from $\{0,1\}$. A *square* matrix has the same number of rows as columns (size $n \times n$). A square matrix A is *symmetric* if $A = A^\top$, that is, $A_{i,j} = A_{j,i}$ for all indices i and j. A square matrix's *diagonal* is all the entries $A_{i,i}$ along the line from upper-left to lower-right. The $n \times n$ *identity matrix* has 1s on its diagonal and 0s for all other entries.

A.10.2 Matrix arithmetic

A matrix's entries are usually elements of a field such as $\mathbb{R}$ or $\mathbb{Z}_p$. We extend the field's arithmetic to "arithmetic-like" operators on matrices. The simplest is addition (and similarly, subtraction): If A and B are $m \times n$ matrices, then $A + B$ is the $m \times n$ matrix obtained by "entry-wise addition": $(A+B)_{i,j} = A_{i,j} + B_{i,j}$ for all indices i and j (where $+$ on the left is matrix addition, and $+$ on the right is the field's scalar addition). The normal meaning of "matrix multiplication" is trickier than "entry-wise multiplication." We develop the definition gradually.

For starters, the *dot product* (also known as *scalar product* or *inner product*) of two vectors v and w (with the same number of components) is $v \cdot w = \sum_i (v_i \cdot w_i)$ (where $\cdot$ on the right is the field's multiplication). To express this with matrix notation, we write v as a row vector and w as a column vector, and juxtapose them with no visible operator between.

Example. $(1,2,3)\cdot(4,5,6) = 1\cdot4+2\cdot5+3\cdot6 = 32$ over $\mathbb{R}$. In matrix notation:

$$\begin{bmatrix}1 & 2 & 3\end{bmatrix}\begin{bmatrix}4\\5\\6\end{bmatrix} = 32$$ ♦

If v and w are both column vectors, then $v\cdot w = v^\top w$ (transposing v to a row vector). If v and w are both row vectors, then $v\cdot w = vw^\top$ (transposing w to a column vector).

Let's move from vector-vector products to matrix-vector products. If A is an $m\times n$ matrix and v is an $n\times1$ column vector, then Av is the $m\times1$ column vector whose i^{th} entry is the dot product of A's i^{th} row and v. An equivalent perspective is that we obtain Av by using the entries of v as coefficients in a *linear combination* of A's columns, which means we sum A's columns after multiplying the j^{th} column's entries by v_j. In particular, if the column vector v is all 0s except $v_j = 1$, then Av is A's j^{th} column.

Example.

$$\begin{aligned}\begin{bmatrix}0 & 2 & 4\\1 & 3 & 5\end{bmatrix}\begin{bmatrix}6\\7\\8\end{bmatrix} &= \begin{bmatrix}(0,2,4)\cdot(6,7,8)\\(1,3,5)\cdot(6,7,8)\end{bmatrix} && \text{(dot products with rows)}\\ &= \begin{bmatrix}0\\1\end{bmatrix}\cdot6+\begin{bmatrix}2\\3\end{bmatrix}\cdot7+\begin{bmatrix}4\\5\end{bmatrix}\cdot8 && \text{(linear combination of columns)}\\ &= \begin{bmatrix}46\\67\end{bmatrix}\end{aligned}$$ ♦

Vector-matrix products are analogous: If A is an $m\times n$ matrix and v is a $1\times m$ row vector, then vA is the $1\times n$ row vector whose j^{th} entry is the dot product of v and A's j^{th} column. Alternatively, vA is a linear combination of A's rows, with v's entries as coefficients. In particular, if the row vector v is all 0s except $v_i = 1$, then vA is A's i^{th} row.

Example.

$$\begin{aligned}\begin{bmatrix}6 & 7\end{bmatrix}\begin{bmatrix}0 & 2 & 4\\1 & 3 & 5\end{bmatrix} &= \begin{bmatrix}(6,7)\cdot(0,1) & (6,7)\cdot(2,3) & (6,7)\cdot(4,5)\end{bmatrix} && \text{(dot products with columns)}\\ &= 6\cdot\begin{bmatrix}0 & 2 & 4\end{bmatrix}+7\cdot\begin{bmatrix}1 & 3 & 5\end{bmatrix} && \text{(linear combination of rows)}\\ &= \begin{bmatrix}7 & 33 & 59\end{bmatrix}\end{aligned}$$ ♦

Vector-matrix-vector multiplication is associative: If we multiply a matrix by a row vector on the left and by a column vector on the right, the result is a scalar (1×1 matrix), and it doesn't matter which multiplication we do first. If A is $m\times n$, and v is $1\times m$, and w is $n\times1$, then

$$(vA)w = \textstyle\sum_j\left(\sum_i v_i\cdot A_{i,j}\right)\cdot w_j = \sum_i v_i\cdot\left(\sum_j A_{i,j}\cdot w_j\right) = v(Aw)$$

so we might as well drop the parentheses and write $vAw = \sum_{i,j}(v_i \cdot A_{i,j} \cdot w_j)$ where the sum is over $(i,j) \in [m] \times [n]$.

We can interpret vector-matrix-vector products over $\mathbb{R}$ in terms of *zero-sum games*. For a fixed $m \times n$ matrix A, consider a two-player game where the "row player" R picks a row index i, and simultaneously the "column player" C picks a column index j, and then C must pay $A_{i,j}$ dollars to R (and if $A_{i,j} < 0$, this means R pays $|A_{i,j}|$ dollars to C). Thus R wants to maximize her profit $A_{i,j}$, while C wants to maximize his profit $-A_{i,j}$. Such a game is called "zero-sum" because one player's gain is the other player's loss, so the sum of their profits is 0. The players may use *mixed strategies*, which means R randomly samples i from a distribution over $[m]$, and independently C randomly samples j from a distribution over $[n]$. Let the row vector v contain the probabilities in R's distribution. Let the column vector w contain the probabilities in C's distribution. Then by independence, $v_i \cdot w_j$ is the probability of outcome (i,j). Hence the expectation of R's profit is $\sum_{i,j}(v_i \cdot w_j) \cdot A_{i,j} = vAw$.

Finally, we come to matrix-matrix products: If A is an $\ell \times m$ matrix and B is an $m \times n$ matrix, then AB is the $\ell \times n$ matrix whose entry at position (h,j) is the dot product of A's h^{th} row and B's j^{th} column: $(AB)_{h,j} = \sum_{i=1}^{m}(A_{h,i} \cdot B_{i,j})$. That is, an individual column of AB is a matrix-vector product (A times the corresponding column of B), and an individual row of AB is a vector-matrix product (the corresponding row of A, times B). If A's number of columns doesn't match B's number of rows, then the product AB is undefined (a "syntax error").

Example. Over the field $\mathbb{Z}_2$, with $\ell = 3$, $m = 4$, and $n = 2$:

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} (0,0,1,0)\cdot(0,1,0,1) & (0,0,1,0)\cdot(1,0,1,1) \\ (1,0,0,1)\cdot(0,1,0,1) & (1,0,0,1)\cdot(1,0,1,1) \\ (1,1,1,1)\cdot(0,1,0,1) & (1,1,1,1)\cdot(1,0,1,1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \blacklozenge$$

Matrix multiplication is associative: $(AB)C = A(BC)$ and so we may just write ABC (assuming the sizes of the matrices allow the products to be defined). To see this, let's determine the entry at position (i,j): Letting v be A's i^{th} row and w be C's j^{th} column,

$$\big((AB)C\big)_{i,j} = (vB)w = v(Bw) = \big(A(BC)\big)_{i,j}$$

since we already know vector-matrix-vector multiplication is associative.

Matrix multiplication is not commutative: AB doesn't always equal BA. Supposing A is $\ell \times m$ and B is $m \times n$: If $\ell \neq n$ then BA isn't even defined. If $\ell = n$ but $m \neq n$ then BA isn't even the same size as AB. If $\ell = m = n$ then AB may or may not equal BA.

Matrix multiplication is distributive: $(A+B)C = AC + BC$ (if the sizes allow) because:

$$\begin{aligned} \big((A+B)C\big)_{h,j} &= \textstyle\sum_i\big((A+B)_{h,i} \cdot C_{i,j}\big) && \text{(matrix multiplication)} \\ &= \textstyle\sum_i\big((A_{h,i}+B_{h,i}) \cdot C_{i,j}\big) && \text{(matrix addition)} \\ &= \big(\textstyle\sum_i(A_{h,i} \cdot C_{i,j})\big) + \big(\textstyle\sum_i(B_{h,i} \cdot C_{i,j})\big) && \text{(field axioms)} \\ &= (AC)_{h,j} + (BC)_{h,j} && \text{(matrix multiplication)} \\ &= (AC+BC)_{h,j} && \text{(matrix addition)} \end{aligned}$$

Similarly, $A(B+C) = AB + AC$ (if the sizes allow).

Also, $(AB)^\top = B^\top A^\top$ because $((AB)^\top)_{i,j} = (AB)_{j,i}$ is the dot product of A's j^{th} row and B's i^{th} column, and $(B^\top A^\top)_{i,j}$ is the dot product of $B^\top$'s i^{th} row (which is B's i^{th} column) and $A^\top$'s j^{th} column (which is A's j^{th} row).

If A has size $m \times n$ and I_n denotes the $n \times n$ identity matrix, then $AI_n = A$ and similarly $I_m A = A$, so the identity matrices play the role of the number 1. If A has size $n \times n$ and $k \in \mathbb{N}$, we let A^k denote k copies of A multiplied together (in particular, $A^0 = I_n$ and $A^1 = A$). The notation $A^k_{i,j}$ is ambiguous since $(A^k)_{i,j}$ doesn't generally equal $(A_{i,j})^k$.

How about matrix division? Inspired by how division works in fields, we define the multiplicative inverse of an $n \times n$ matrix A as an $n \times n$ matrix A^{-1} such that $AA^{-1} = I_n$. Not every nonzero $n \times n$ matrix A has a multiplicative inverse, but if it does then A^{-1} is unique and $A^{-1}A = I_n$ also holds. In that case, there are two possible ways to define "B divided by A" where B has size $n \times n$: BA^{-1} is the solution to the equation $XA = B$, and $A^{-1}B$ is the solution to the equation $AX = B$.

A.10.3 Walks in graphs

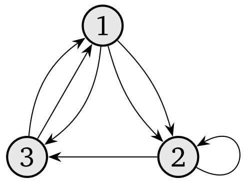
Consider any directed graph G with set of nodes $[n]$. If self-loops are allowed but multi-edges aren't, then G's *adjacency matrix* is the $n \times n$ matrix A where $A_{i,j} = 1$ if G has an edge from node i to node j, and $A_{i,j} = 0$ if G has no such edge. If multi-edges (including multi-self-loops) are allowed, then $A_{i,j}$ is the number of (i,j) edges. The sum of row i's entries is node i's outdegree. The sum of column j's entries is node j's indegree.

An undirected graph's adjacency matrix is the adjacency matrix of the directed version in which each edge $\{i,j\}$ is replaced with a pair of edges (i,j) and (j,i). Thus, if $i \neq j$ then $A_{i,j}$ and $A_{j,i}$ equal the number of $\{i,j\}$ edges, and $A_{i,i}$ equals 2 times the number of undirected self-loops on node i. An undirected graph's adjacency matrix is symmetric, and the sum of the entries in row i (or in column i) is node i's degree.

Recall that a walk goes from node to node to node (and so on) by following edges, and may reuse edges. A walk's length is the number of edge traversals (steps).

Example.

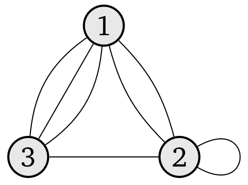
Directed graph



Adjacency matrix

$$\begin{bmatrix} 0 & 2 & 1 \\ 0 & 1 & 1 \\ 2 & 0 & 0 \end{bmatrix}$$

Undirected graph



Adjacency matrix

$$\begin{bmatrix} 0 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 1 & 0 \end{bmatrix}$$

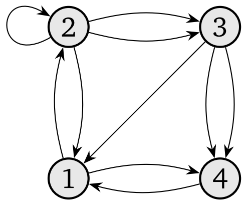
In the directed graph on the left (which is not the directed version of the undirected graph on the right), the number of distinct walks of length 3 that visit nodes 3, 1, 3, 1 in that order is $2 \cdot 1 \cdot 2 = 4$: There are two choices for the first edge, one choice for the second edge, and two choices for the third edge (either reusing the walk's first edge or using the other $(3,1)$ edge). ♦

For the rest of this section, we focus on directed graphs since they're more general.

Consider any n-node directed graph with multi-edges and self-loops allowed. Let A be its adjacency matrix. By definition, $A_{i,j}$ is the number of length-1 walks that start at node i and end at node j. The connection with matrix multiplication is that for every integer $\ell \geq 1$, $(A^\ell)_{i,j}$ is the number of length-ℓ walks that start at i and end at j. To see why, first consider $\ell = 2$: For any node k, the number of walks that visit nodes i, k, j (in that order) equals $A_{i,k} \cdot A_{k,j}$ by the multiplication principle (since there are $A_{i,k}$ many (i,k) edges and $A_{k,j}$ many (k,j) edges). By the addition principle, we can partition the set of all length-2 walks from i to j according to their second node k, to see that the total number of such walks is $\sum_{k=1}^n (A_{i,k} \cdot A_{k,j}) = (A^2)_{i,j}$. We repeat this reasoning to handle $\ell = 3, 4, \ldots$: Assuming we already know that the entries of $A^{\ell-1}$ count walks of length $\ell - 1$, we count the number of length-ℓ walks from i to j by using the addition principle (partitioning the set of such walks according to their second node k) and multiplication principle to get $\sum_{k=1}^n (A_{i,k} \cdot (A^{\ell-1})_{k,j}) = (AA^{\ell-1})_{i,j} = (A^\ell)_{i,j}$.

In a directed graph, a *random walk* of length ℓ starts at a designated node i, and in each step, it follows a uniformly random outgoing edge from the current node. The graph's *transition matrix* is the $n \times n$ matrix T where $T_{i,j} = (\text{number of } (i,j) \text{ edges}) / \operatorname{outdeg}(i)$ is the probability of being at node j after one step from node i. The sum of the entries in any individual row of T is 1.

Example.



Adjacency matrix

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Transition matrix

$$\begin{bmatrix} 0/2 & 1/2 & 0/2 & 1/2 \\ 1/4 & 1/4 & 2/4 & 0/4 \\ 1/3 & 0/3 & 0/3 & 2/3 \\ 1/1 & 0/1 & 0/1 & 0/1 \end{bmatrix}$$

Taking a random walk of length ℓ from node i is generally not the same as picking a uniformly random one of all such walks. For instance, with $\ell = 2$ and $i = 1$, the above graph has five distinct walks of length 2 from node 1, four of which have 2 as the second node, and one of which has 4 as the second node. A random walk, however, does not assign probability 1/5 to each of these walks: It first picks one of the two edges out of node 1 uniformly at random, so the walk with second node 4 has probability 1/2, while each of the walks with second node 2 has probability 1/8. ♦

$(A^\ell)_{i,j}$ is the number of length-ℓ walks that start at i and end at j, but this information isn't enough to determine probabilities with a random walk. For that, the transition matrix is useful. It turns out $(T^\ell)_{i,j}$ is the probability of being at node j after taking a random walk of length ℓ from i. This holds by definition when $\ell = 1$, so let's next consider $\ell = 2$: For any node k, the probability that the random walk visits nodes i, k, j (in that order) equals

$$\mathbf{Pr}[\text{second node is } k] \cdot \mathbf{Pr}\Big[\text{last node is } j \,\Big|\, \text{second node is } k\Big] = T_{i,k} \cdot T_{k,j}$$

by the chain rule (since conditioned on the second node being k, the marginal distribution of the second edge is the same as a random walk of length 1 from k). By the law of total probability, we can partition the set of all length-2 walks from i according to their second node k, to get:

$$\mathbf{Pr}[\text{last node is } j] = \textstyle\sum_{k=1}^n \mathbf{Pr}[\text{second node is } k \text{ and last node is } j] = \sum_{k=1}^n (T_{i,k} \cdot T_{k,j}) = (T^2)_{i,j}$$

We repeat this reasoning to handle $\ell = 3,4,\ldots$: Assuming we already know that the entries of $T^{\ell-1}$ are the relevant probabilities for walks of length $\ell-1$, we calculate the probability of being at j after a length-ℓ random walk from i by using the law of total probability (partitioning the set of such walks according to their second node k) and chain rule to get $\sum_{k=1}^{n}(T_{i,k} \cdot (T^{\ell-1})_{k,j}) = (TT^{\ell-1})_{i,j} = (T^{\ell})_{i,j}$. Again, a key observation was that conditioned on the second node being k, the marginal distribution of the last $\ell-1$ steps is the same as a random walk of length $\ell-1$ from k.

Another perspective: Suppose instead of starting at a designated node i, the random walk starts at a random node sampled from some distribution over nodes, represented by a $1 \times n$ row vector v where v_i is the probability of starting at node i. Then the row vector vT represents the distribution over nodes after one step of the random walk. That is, $(vT)_j$ is the probability of being at node j after starting at a random node i distributed according to v and following a uniformly random edge out of i, because

$$\begin{aligned}(vT)_j &= \textstyle\sum_{i=1}^{n}(v_i \cdot T_{i,j}) = \sum_{i=1}^{n}\big(\mathbf{Pr}[\text{first node is } i] \cdot \mathbf{Pr}\big[\text{second node is } j \,\big|\, \text{first node is } i\big]\big) \\ &= \textstyle\sum_{i=1}^{n} \mathbf{Pr}[\text{first node is } i \text{ and second node is } j] = \mathbf{Pr}[\text{second node is } j]\end{aligned}$$

by the chain rule and law of total probability. What's the distribution over nodes after two steps of the random walk? Since the distribution after one step is vT, the distribution after another step will be $(vT)T = vT^2$. Repeating this reasoning shows that the distribution vector after ℓ steps is $(vT^{\ell-1})T = vT^{\ell}$ by associativity. That is, $(vT^{\ell})_j$ is the probability of being at node j after a length-ℓ random walk with starting node distributed according to v. If $v_i = 1$ and $v_h = 0$ for all $h \neq i$, we recover the result from the previous paragraph: $(T^{\ell})_{i,j} = (vT^{\ell})_j$ is the probability that a length-ℓ random walk from node i winds up at node j.

Exercises

A.1 Which of the following properties are guaranteed to hold for all subsets S and T of a universe U? Justify your answers.

(a) $S \setminus T = \overline{T} \setminus \overline{S}$

(b) $\overline{S \setminus T} = \overline{S} \setminus \overline{T}$

(c) $\overline{S \setminus T} = \overline{S} \cup T$

(d) $S \setminus S = T \setminus T$

(e) $S \setminus (S \setminus T) = T \setminus (T \setminus S)$

(f) $(U \setminus T) \setminus S = U \setminus (T \setminus S)$

(g) $(S \setminus T) \cup (T \setminus S) \subseteq U \setminus (S \cap T)$

(h) If $S \subseteq T$ then $\overline{S} \subseteq \overline{T}$.

(i) If $T \subseteq \overline{S} \cap T$ then $S \subseteq \overline{T} \cap S$.

A.2 The *power set* of a set S is $\text{pow}(S) = \{T : T \subseteq S\}$, the set of all subsets of S. For example, $\text{pow}(\{1,2,3\}) = \big\{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\big\}$.

(a) Write all the elements of $\text{pow}(\text{pow}(\{1\}))$.

(b) Is it true that $S \cap T \in \text{pow}(S) \cap \text{pow}(T)$ for all sets S and T? Justify your answer.

(c) Is it true that $S \cup T \in \text{pow}(S) \cup \text{pow}(T)$ for all sets S and T? Justify your answer.

A.3 Which of the following properties are guaranteed to hold for all sets Q, R, S, T? Justify your answers.

(a) $(Q \times R) \cap (S \times T) = (Q \cap S) \times (R \cap T)$

(b) $(Q \times R) \cup (S \times T) = (Q \cup S) \times (R \cup T)$

(c) $(Q \times R) \setminus (S \times T) = (Q \setminus S) \times (R \setminus T)$

A.4 This is about basic components in digital logic design. For this exercise, assume a length-n bit string has positions indexed from 0 to $n-1$, and view the indices as binary (base two) numbers. For example, $0101_{10} = 0$ because the subscript 10 means index two, which is the third position from the left, which is a 0 in 0101.

(a) Write the 2-dimensional table for the *2-to-1 multiplexer* function:

$$\text{Mux}\colon \{0,1\}^2 \times \{0,1\} \to \{0,1\} \quad \text{where} \quad \text{Mux}(x,i) = x_i$$

(b) Write the 2-dimensional table for the *1-to-2 demultiplexer* function:

$$\text{Demux}\colon \{0,1\}\times\{0,1\} \to \{0,1\}^2 \text{ where } \text{Demux}(y,i) = \text{bit string with } y \text{ at index } i \text{ and 0s elsewhere}$$

(c) Write the table for the *2-to-4 decoder* function:

$$\text{Dec}\colon \{0,1\}^2 \to \{0,1\}^4 \quad \text{where} \quad \text{Dec}(i) = \text{bit string with 1 at index } i \text{ and 0s elsewhere}$$

(d) Write the table for the *4-to-2 priority encoder* partial function:

$$\text{Enc}\colon \{0,1\}^4 \to \{0,1\}^2 \quad \text{where} \quad \text{Enc}(x) = \text{index of the first 1 in } x \text{ or undefined if } x \text{ has no 1}$$

A.5 Recall the lexicographic order on $\{0,1\}^n$. We extend this to the lexicographic order on all functions $f\colon \{0,1\}^n \to \{0,1\}$: We say f lexicographically precedes g when $f(x) = 0$ and $g(x) = 1$ for the lexicographically lowest x such that $f(x) \neq g(x)$.

For $i \in [16]$, define $f_i\colon \{0,1\}^2 \to \{0,1\}$ to be the i^{th} such function in lexicographic order. Write the 2-dimensional table for $F\colon \{0,1\}^2 \times [16] \to \{0,1\}$ where $F(x,i) = f_i(x)$.

A.6 Here are some graph calisthenics.

(a) Give an example of a connected, undirected graph (without multi-edges or self-loops) such that every node has degree exactly 3 but there is no cycle of length exactly 3.

(b) Give an example of a strongly connected, directed graph (without multi-edges or self-loops) such that every node has indegree exactly 2 and outdegree exactly 2, but there is no cycle of length exactly 2.

A.7 Determine which of the following hold. For each, if equivalent then write the shared truth table, and if inequivalent then give a counterexample truth assignment.

(a) $\neg(P \Leftrightarrow Q) \equiv \neg P \Leftrightarrow Q$

(b) $\neg(P \Rightarrow Q) \equiv \neg P \Rightarrow Q$

(c) $(P \vee Q) \Rightarrow R \equiv (P \Rightarrow R) \wedge (Q \Rightarrow R)$

(d) $(P \wedge Q) \Rightarrow R \equiv (P \Rightarrow R) \vee (Q \Rightarrow R)$

(e) $(P \Leftrightarrow Q) \Rightarrow R \equiv (P \Rightarrow R) \Leftrightarrow (Q \Rightarrow R)$

A.8 This exercise explores relationships between sets and logic. Which of the following statements are guaranteed to hold for all sets Q, R, S, T? Justify your answers.

(a) $(R \cup S) \subseteq T \iff (R \subseteq T \ \wedge\ S \subseteq T)$

(b) $(R \cap S) \subseteq T \iff (R \subseteq T \ \vee\ S \subseteq T)$

(c) $Q \times R \subseteq S \times T \iff (Q \subseteq S \ \wedge\ R \subseteq T)$

A.9 Let $P(x, y)$ be the predicate "$x < y < x + 1$". For each of the following propositions, determine whether it's true or false, and justify your answer.

(a) $(\forall x \in \mathbb{Z})\,(\exists y \in \mathbb{R})\, P(x, y)$

(b) $(\forall x \in \mathbb{R})\,(\exists y \in \mathbb{Z})\, P(x, y)$

(c) $(\exists y \in \mathbb{R})\,(\forall x \in \mathbb{Z})\, \neg P(x, y)$

(d) $(\exists y \in \mathbb{Z})\,(\forall x \in \mathbb{R})\, \neg P(x, y)$

A.10 For each of the following propositions, determine whether it's true or false, and justify your answer. (The *floor* function $\lfloor x \rfloor$ rounds x down to an integer toward $-\infty$, and the *ceiling* function $\lceil x \rceil$ rounds x up to an integer toward $+\infty$.)

(a) $(\exists x \in \mathbb{R})\,(\forall y \in \mathbb{R})\,(x + y = xy)$

(b) $(\exists x \in \mathbb{R})\,(\forall y \in \mathbb{R})\,(x + y \geq 2xy)$

(c) $(\exists x \in \mathbb{R})\,(\forall y \in \mathbb{R})\,(\lceil y \rceil = \lfloor y + x \rfloor)$

(d) $(\forall x \in \mathbb{Z})\,(\exists y \in \mathbb{N})\,(y^2 \geq x^2 \ \Rightarrow\ x \geq 0)$ Hint: Use $P \Rightarrow Q \equiv \neg P \vee Q$.

A.11 Determine which of the following hold. For each, if equivalent then informally explain why, and if inequivalent then exhibit particular predicates that form a counterexample. Assume the domains are always non-empty.

(a) $\exists x\,(P(x) \wedge \forall y\, Q(y)) \equiv \forall y\, \exists x\,(P(x) \wedge Q(y))$

(b) $\forall x\, \exists y\,(P(x, y) \vee Q(x, y)) \equiv (\forall x\, \exists y\, P(x, y)) \vee (\forall x\, \exists y\, Q(x, y))$

(c) $(\exists x\, \forall y\, P(x, y)) \Rightarrow (\forall y\, \exists x\, P(x, y)) \equiv (\exists x\, \forall y\, P(x, y)) \Rightarrow (\exists x\, \exists y\, P(x, y))$

A.12 Supposing $P(n)$ is a predicate over the natural numbers, write formulas expressing the following statements. You may use standard quantifiers ($\exists$ and $\forall$), logical connectives, arithmetic, and equality/inequalities.

(a) There are exactly two values that make P true.

(b) There are no three consecutive values for which P has the same truth value.

(c) There are infinitely many pairs of distinct values that make P true and that are within some fixed distance of each other. (For the specific case where $P(n) =$ "n is prime", this is a deep and celebrated theorem of number theory.)

A.13 This exercise shows that "$f = o(g)$" is not always the same as "$f = O(g)$ and $f \neq \Omega(g)$", even for monotonically nondecreasing functions ($f(n) \leq f(n+1)$ and $g(n) \leq g(n+1)$ for all $n \in \mathbb{N}$). Consider the functions $f(n) = \lfloor n/2 \rfloor!$ and $g(n) = \lceil n/2 \rceil!$. (Here ! means factorial, and Exercise A.10 has a reminder about the meaning of $\lfloor\ \rfloor$ and $\lceil\ \rceil$.) Verify that $f \neq o(g)$, $f = O(g)$, $f \neq \Omega(g)$, and f and g are monotonically nondecreasing.

A.14 Show that the following properties hold, using the definitions of big-O and little-o.

(a) For every constant $a > 0$ (not necessarily an integer), if $f = O(g)$ then $f^a = O(g^a)$.

(b) If $f_1 = O(g_1)$ and $f_2 = o(g_2)$ then $f_1 f_2 = o(g_1 g_2)$.

(c) $O_\vee$ is equivalent to O_+, defined as follows: $f(m,n) = O_+(g(m,n))$ when:

$$(\exists c > 0)\ (\exists n_0 \in \mathbb{N})\ (\forall (m,n) \in \mathbb{N}^2)\ \big(m + n \geq n_0 \ \Rightarrow\ f(m,n) \leq c \cdot g(m,n)\big)$$

A.15 Suppose passwords are strings where the alphabet has 10 digits, 26 lower-case letters, 26 upper-case letters, and 15 special symbols.

(a) How many length-n passwords contain at least two of the four types of characters?

(b) How many length-n passwords have no two adjacent identical characters?

(c) How many length-n passwords contain at least two of the four types of characters and have no two adjacent identical characters?

A.16 For some $k \leq n$, suppose we sample a uniformly random $n \times n$ matrix with k many 1s and $n^2 - k$ many 0s.

(a) What's the probability that all 1s are in the same row?

(b) What's the probability that all 1s are in different rows?

A.17 Here are some calisthenics for probability calculations.

(a) Two fair dice are rolled. Given that the sum of the dice is at most 4, what's the conditional probability of getting doubles (that is, the two dice show the same value)?

(b) You have a bag with three coins: an ordinary fair coin, a coin with heads on both sides, and a coin with tails on both sides. You close your eyes, select a coin uniformly at random from the bag, toss it, then see that it landed with heads showing. What's the conditional probability that the other side of the coin you selected is also heads?

(c) A computer system has n machines. There are $k \leq n$ jobs, and the system assigns each job to one of the machines selected uniformly at random, independent of the other jobs. What's the probability that at least one machine gets at least two jobs?

(d) Two fair n-sided dice are rolled (with faces labeled by numbers in $[n]$). What's the probability that the sum of the dice is an even number? Hint: There are two cases, depending on whether n is even or odd.

A.18 Suppose each new child's gender is a fair coin toss, independent of all other children (and there are no twins, triplets, and so on).

(a) A particular family has two children. Given that at least one child is a boy, what's the conditional probability that the other child is also a boy?

(b) Further suppose that the day of the week (Sunday, Monday, ...) of any particular child's birth is uniformly distributed, independent of the gender and independent of all other children. A particular family has two children. Given that at least one child is a boy born on a Tuesday, what's the conditional probability that the other child is also a boy? (It may be surprising that the answer is not the same as for Exercise A.18.a.)

A.19 Show that for events A, B in any probability space with $\mathbf{Pr}[A] > 0$:

$$\mathbf{Pr}[B \mid A] - \mathbf{Pr}[\overline{A}] \;\le\; \mathbf{Pr}[B] \;\le\; \mathbf{Pr}[B \mid A] + \mathbf{Pr}[\overline{A}]$$

(Thus, if $\mathbf{Pr}[\overline{A}]$ is small then $\mathbf{Pr}[B] \approx \mathbf{Pr}[B \mid A]$.)

A.20 For the following joint distribution over $\{\mathrm{a}, \mathrm{b}, \mathrm{c}\} \times \{\mathrm{d}, \mathrm{e}, \mathrm{f}\}$, find both marginal distributions and draw the tree representing the two-stage process of sampling an outcome s by sampling s_1 followed by s_2 conditioned on s_1.

D	d	e	f
a	0	0.1	0.1
b	0.1	0.3	0.1
c	0.1	0	0.2

A.21 Here are some calisthenics for wielding linearity of expectation.

(a) A coin with heads probability p is tossed n times. A "run" is defined as a maximal sequence of consecutive tosses that all show the same side of the coin (for example, HHTHHHTTH has five runs). What's the expectation of the number of runs?

(b) A uniformly random permutation of the set $[n]$ is given to the bubble sort algorithm, which sorts the numbers into ascending order by repeatedly swapping adjacent pairs that are inverted (that is, the larger number immediately precedes the smaller of the two). What's the expectation of the number of swaps that occur? (It turns out the number of swaps doesn't depend on the order in which swaps are done, only on the input permutation.)

A.22 **(a)** Let X be any random variable that only takes values in $\mathbb{N} = \{0, 1, 2, \ldots\}$. (Binomial and geometric random variables have this property.) Show $\mathbf{E}[X] = \sum_{k=1}^{\infty} \mathbf{Pr}[X \ge k]$.

(b) Use Exercise A.22.a to give an alternative derivation of the fact that the expectation of the geometric random variable with parameter p equals $1/p$.

A.23 **(a)** Someone gives you a peculiar die that has three sides, which are labeled 2, 4, and 6. You roll the die repeatedly until seeing a 6. What's the expectation of the number of rolls?

(b) Now, you roll a normal 6-sided die repeatedly until seeing a 6. What's the probability that all rolls (up through the first 6) are even numbers?

(c) Again you roll a normal 6-sided die repeatedly until seeing a 6. Conditioned on the event that all rolls (up through the first 6) are even numbers, what's the expectation of the number of rolls? (It may be surprising that the answer is not the same as for Exercise A.23.a.)

A.24 Here are some calisthenics for modular arithmetic.

(a) Find 59 mod 13 and (-59) mod 13, as well as the corresponding quotients.

(b) Evaluate $(21\cdot(13-43)) \bmod 9$ by doing "mod 9" on all the intermediate expressions. Show all steps.

(c) Find the multiplicative inverse of each nonzero element of the field $\mathbb{Z}_{11}$.

A.25 Do the following calculations in the field $\mathbb{Z}_3$:

(a) Write the table for the function $P\colon \mathbb{Z}_3 \to \mathbb{Z}_3$ expressed by the univariate polynomial $P(x) = 2x^2 + x + 1$.

(b) Write the 2-dimensional table for the function $Q\colon \mathbb{Z}_3 \times \mathbb{Z}_3 \to \mathbb{Z}_3$ expressed by the multivariate polynomial $Q(x, y) = x^2y + 2xy^2 + 2x + y$.

(c) Evaluate $\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 0 & 2 \\ 2 & 1 \end{bmatrix}$ over $\mathbb{Z}_3$.

A.26 We obtain the composition of two univariate polynomials $P(y)$ and $Q(x)$ (over the same field) by substituting a complete copy of $Q(x)$ in for each occurrence of y in P. This results in a univariate polynomial $P(Q(x))$ expressing the function $P \circ Q$.

(a) Find the canonical form of $P(Q)$ where $P(y) = 3y^2 + 4y + 5$ and $Q(x) = x^2 + 1$ over $\mathbb{R}$.

(b) Show that $\deg(P(Q)) = \deg(P)\cdot\deg(Q)$, assuming P and Q are both nonzero.

(c) If $P(y_1, \ldots, y_m)$ and $Q_1(x_1, \ldots, x_n)$, $\ldots$, $Q_m(x_1, \ldots, x_n)$ are multivariate polynomials (over the same field), the composition $P(Q_1, Q_2, \ldots, Q_m)$ is a polynomial with variables $x_1, \ldots, x_n$, obtained by substituting a complete copy of $Q_i(x_1, \ldots, x_n)$ in for each occurrence of y_i in P (for each $i \in [m]$). Recalling that deg() denotes total degree, show that if $\deg(P) \le a$ and $\deg(Q_i) \le b$ for each i, then $\deg(P(Q_1, \ldots, Q_m)) \le a \cdot b$, assuming P and each Q_i are nonzero.

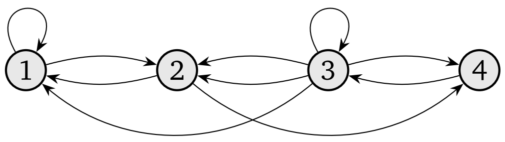
A.27 Show that $\deg(P\cdot Q) = \deg(P) + \deg(Q)$ for all multivariate polynomials P and Q (with the same variables and over the same field), where deg() denotes total degree. Hint: Letting the variables be ordered $x_1, x_2, \ldots, x_n$, look at the exponent tuple $(e_1, \ldots, e_n) \in \mathbb{N}^n$ of any monomial $c \cdot x_1^{e_1} \cdots x_n^{e_n}$. Recall the lexicographic ordering of such tuples: $(a_1, \ldots, a_n)$ lexicographically precedes $(b_1, \ldots, b_n)$ when for some i, $a_i < b_i$ and $a_k = b_k$ for all $k < i$. Show that if you take the lexicographically earliest among P's highest-total-degree monomials, and the lexicographically earliest among Q's highest-total-degree monomials, then their product is a monomial of $P\cdot Q$ that's not cancelled out by any other monomial.

A.28 Explain why $A^\top A$ is symmetric for every matrix A.

A.29 **(a)** You sample vectors v and w from $(\mathbb{Z}_2)^n$, uniformly at random and independent of each other. What's the probability that $v \cdot w = 1$ (where the dot product uses $\mathbb{Z}_2$ arithmetic)?

(b) You sample matrices A of size $\ell \times n$ and B of size $n \times m$ over $\mathbb{Z}_2$, uniformly at random and independent of each other. What's the expectation of the number of 1s in AB (where the matrix product uses $\mathbb{Z}_2$ arithmetic)?

A.30 Consider the following graph:



(a) Write the adjacency matrix and the transition matrix.

(b) Calculate the distribution on nodes after one step of a random walk where the starting node is sampled according to $\begin{bmatrix}\mathbf{Pr}[1] & \mathbf{Pr}[2] & \mathbf{Pr}[3] & \mathbf{Pr}[4]\end{bmatrix} = \begin{bmatrix}\frac{1}{5} & \frac{1}{10} & \frac{1}{2} & \frac{1}{5}\end{bmatrix}$.

Notes

The following rules are collectively known as "De Morgan's laws."

$$\begin{aligned}
\overline{S \cup T} &= \overline{S} \cap \overline{T} & \overline{P \vee Q} &\equiv \overline{P} \wedge \overline{Q} & \neg(\exists x\ P(x)) &\equiv \forall x\ (\neg P(x)) \\
\overline{S \cap T} &= \overline{S} \cup \overline{T} & \overline{P \wedge Q} &\equiv \overline{P} \vee \overline{Q} & \neg(\forall x\ P(x)) &\equiv \exists x\ (\neg P(x))
\end{aligned}$$

The set difference operator $\smallsetminus$ is also denoted $-$.

Injections are also called "one-to-one functions." Surjections are also called "onto functions." Bijections are also called "one-to-one correspondences."

Another notation to specify a function f is $x \mapsto y$, which means $f(x) = y$. This notation enables discussion of functions such as $x \mapsto x^2$ on the fly, without giving them names ("anonymous functions," like lambdas in programming languages).

When discussing graphs, what we call paths are sometimes called "simple paths," while what we call trails are sometimes called paths. What we call cycles are sometimes called "simple cycles," while what we call closed trails are sometimes called cycles or "circuits."

What we call binary trees are sometimes called "full binary trees" to distinguish them from trees where nodes may have exactly one child (or zero or two children). Out-trees are also called "arborescences."

The binomial coefficients $\binom{n}{k}$ have several alternative notations, such as $C(n,k)$ and ${}_nC_k$.

A bit string's weight (number of 1s) is often called its "Hamming weight."

In probability theory, it's somewhat more traditional to denote the sample space by Ω and an individual outcome by ω. The chain rule is also called the "general product rule." The term "random variable" is sometimes also used for functions with arbitrary codomains, not just $\mathbb{R}$ or a subset thereof. Indicator random variables are also called "Bernoulli random variables."

Finite fields are also known as "Galois fields" and so $\mathbb{Z}_p$ is also denoted $GF(p)$.

Matrices are sometimes written with enclosing parentheses rather than brackets.

In an undirected graph's adjacency matrix, it's unfortunately somewhat common to use 1 rather than 2 for the diagonal entry representing a self-loop. Our definition (with 2) is consistent with treating undirected edges as pairs of directed edges: From a node's "local perspective," an undirected self-loop provides two ways to depart the node and thus contributes 2 to the degree. A diagonal entry of 1 should be viewed as a directed self-loop in an otherwise undirected graph.

"Markov chains" are more general than random walks in directed graphs: They let each node have a possibly-nonuniform distribution over its outgoing edges.

Exercise A.18.b is attributed to Gary Foshee. Exercise A.23.c is known as "Mossel's dice paradox."

Chapter B

Proof Techniques

The only way to know for sure that a mathematical proposition is true is to find a *proof*. This can involve creativity, insight, and hard work. There's no recipe or flowchart, but some techniques are widely applicable. The satisfaction from designing a correct proof is like the satisfaction from writing a significant computer program and seeing it work.

B.1 What is a proof?

A proof uses logical reasoning to convince people that a particular proposition holds (is true). A criminal trial typically requires proof beyond a reasonable doubt, but math requires proof beyond any doubt whatsoever. An "airtight" proof is the only way to ascertain a proposition's truth.

A proof is a sequence of statements, each of which is an obvious fact (such as $x + y \geq x$ if $y \geq 0$) or follows from previous statements by an obvious logical rule (such as if $P \vee Q$ and $\neg P$ have both been established, then Q can be deduced). Though each step may feel routine, the whole proof might be surprising and elegant. A melody's individual notes may be uninteresting in isolation but beautiful when combined in the right way.

Common mistakes involve steps that seem obvious but are invalid upon closer inspection. What counts as "obvious" varies from person to person, but some basic facts and rules are agreed to be self-evident by essentially everyone familiar with abstract math. Strict adherence to these facts and rules is known as *rigor*. Seeing lots of examples is a good way to get accustomed to the contemporary standard of rigor. Furthermore, active practice in developing proofs is essential for understanding and appreciating more advanced proofs—this is not a spectator sport.

Proofs are intimidating to some people but are mostly just everyday reasoning applied to abstract things. With sufficient experience, reading and writing proofs are like having ordinary conversations, using the notation and terminology from Chapter A as vocabulary. As with ordinary conversations, a proof's level of detail should be tailored to the audience—it may take less detail to convince an expert than a novice.

B.1.1 Writing style and conventions

Since the audience is humans, a proof should mainly be full English sentences, spelling out the logical flow with words such as "for all" instead of $\forall$, "and" instead of $\wedge$, and "if ... then" instead of $\Rightarrow$. Notation and formulas such as "$x + 1$" should be parts of the sentences. There's a tradeoff between math notation and prose: A proof is hard to follow if it's too dense with notation where plain English would do, but a proof feels sluggish if it has too much prose where crisp, succinct notation would do. (Sometimes, notation such as $\forall$, $\wedge$, $\Rightarrow$ helps condense technical parts of proofs.) There's also a tradeoff in understandability: Excessive detail and formality can make a proof tedious and pedantic, but informal writing might render a proof ambiguous and unconvincing. Proof authors should aim for the "sweet spot" of readability.

It often helps to begin with a "proof idea" conveying the intuition for why the proposition is true, an outline or "game plan" for the proof structure, the thought process behind the discovery of the proof, or anything else people might find illuminating. Then, proceed to an official proof with enough detail and precision as to be indisputable. The *tombstone* ■ marks the end of a proof. We also mark the end of "proof idea" sections with ■.

Once we *prove* (or *argue* or *show*) a proposition, it's a *theorem*. Here's our first example.

Theorem B.1. *For all real numbers $x_1, x_2, \dots, x_n$, we have $|x_1 + \cdots + x_n| \leq |x_1| + \cdots + |x_n|$.*

Before thinking about why it's true, let's understand what this proposition says. Each x_i is an arbitrary element of $\mathbb{R}$, and these numbers may or may not be different from each other. They're components of a tuple $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$. The "we have" just visually separates the quantification on the left from the predicate on the right. The bars in the predicate mean absolute value ($|y| = y$ if $y \geq 0$, and $|y| = -y$ if $y < 0$). A sum $x_1 + \cdots + x_n$ can also be written $\sum_{i=1}^n x_i$, but sometimes the $\cdots$ notation is easier on the eye.

What's the value of n? When a variable (such as n in Theorem B.1) is not bound to a quantifier, by convention it is universally quantified ($\forall n$) over a domain that's inferred from context. Here, since n is a tuple's length, it only makes sense as a nonnegative integer. (When $n = 0$, a sum with no summands is simply 0.) A completely formal version of Theorem B.1 would be "$(\forall n \in \mathbb{N})\ (\forall x \in \mathbb{R}^n)\ \left|\sum_{i=1}^n x_i\right| \leq \sum_{i=1}^n |x_i|$" while a pure English version would be "For every tuple of real numbers, the absolute value of their sum is at most the sum of their absolute values." The former looks dense and the latter looks verbose. Our chosen statement of Theorem B.1 is perhaps the quickest way to grasp the meaning.

Proof idea. A number's absolute value is its "distance from 0." Imagine we start at 0 and walk on the number line: We first walk x_1 units (which is to the right if $x_1 > 0$, and to the left if $x_1 < 0$). From there, we walk x_2 units, and so on. After all n phases, we're at location $x_1 + \cdots + x_n$. Our final distance from the starting point is $|x_1 + \cdots + x_n|$, which cannot be greater than our "total walking distance": Ignoring directions, our feet have walked a total of $|x_1| + \cdots + |x_n|$ units. The farthest away from 0 we could possibly be is if all phases were in the same direction: $|x_1 + \cdots + x_n| = |x_1| + \cdots + |x_n|$ if x_i is nonnegative for every i, or if x_i is nonpositive for every i. If some phases move right and others move left, the backtracking would put us closer to 0 than if we walked the same lengths ($|x_1|, \dots, |x_n|$) all in the same direction.

The above paragraph may be convincing enough, but let's express this more rigorously. Each x_i equals either $|x_i|$ or $-|x_i|$, so $-|x_i| \leq x_i \leq |x_i|$. Summing these inequalities over all indices i yields:

$$\underbrace{(-|x_1|) + \cdots + (-|x_n|)}_{\text{leftmost possible ending spot}} \quad \leq \quad \underbrace{x_1 + \cdots + x_n}_{\text{actual ending spot}} \quad \leq \quad \underbrace{|x_1| + \cdots + |x_n|}_{\text{rightmost possible ending spot}}$$

We used the basic fact that if we increase each summand then the whole sum also increases (if $a \leq b$ and $c \leq d$ then $a + c \leq b + d$, and similarly with more than two summands). In the streamlined official proof, we show that this implies the theorem by separately considering two cases, for whether the actual ending spot is to the right or left of 0. In the latter case, we use another basic fact: Negating both sides of an inequality "flips" the inequality's direction (if $a \ge b$ then $-a \le -b$). ■

Proof. Let $x_1, x_2, \dots, x_n$ be arbitrary real numbers. Consider two cases: If $x_1 + \dots + x_n \ge 0$ then $|x_1 + \dots + x_n| = x_1 + \dots + x_n \le |x_1| + \dots + |x_n|$ since $x_i \le |x_i|$ for each index i. If $x_1 + \dots + x_n < 0$ then $|x_1 + \dots + x_n| = -(x_1 + \dots + x_n) = (-x_1) + \dots + (-x_n) \le |x_1| + \dots + |x_n|$ since $x_i \ge -|x_i|$ and thus $-x_i \le |x_i|$ for each index i. ■

A *lemma* is a theorem intended for use in the proof of a more important theorem, rather than to stand alone. Like a method/procedure/subroutine in a computer program, there are several reasons to carve out part of a proof into a lemma:

- Reusability: A lemma can be invoked multiple times, without duplicating the lemma's proof. This is like extracting duplicated code into a subroutine. Redundancies in a program or proof can make it unwieldy, error-prone, and hard to adapt.
- Modularity: A proof is easier to digest when it's broken into pieces that interface nicely with each other. Once the correctness of a subroutine or lemma has been verified, it can be treated as a black box, and the details of how it works can be forgotten.

When a lemma has a short or simple proof, it may be called a *claim*. A *corollary* is a theorem that follows easily from another theorem. A *conjecture* is a proposition that some people believe is true, but is not yet proven (and so may be false).

B.1.2 Quantifiers, examples, and counterexamples

A proposition $\forall x\ P(x)$ (where $P(x)$ is a predicate) is written "for all x" or "for every x" or "for each x." A proposition $\exists x\ P(x)$ is written "there exists an x such that" or "for some x" or "for at least one x." Beware the word "any": "Does $P(x)$ hold for any x?" could mean either $\forall x\ P(x)$ or $\exists x\ P(x)$. In some situations though, "any" is unambiguous: "$P(x)$ doesn't hold for any x" means $\forall x\ \neg P(x)$, and "consider any x" means $\forall x$.

To prove $\forall x\ P(x)$, we consider an arbitrary (unspecified) element x from the domain, and argue that $P(x)$ holds. We have no control over x's value and can't make any assumptions about it, except that it's from the domain. Imagine someone descends from the clouds and presents a value of x—we must be prepared to handle whatever it might be.

To prove $\exists x\ P(x)$, one way is to describe a particular element x from the domain, and argue that $P(x)$ holds. Such a value is an *example* demonstrating the truth of $\exists x\ P(x)$. In contrast to this *constructive* approach (specifying an explicit value for x), some *nonconstructive* techniques can prove $\exists x\ P(x)$ without pinpointing any particular value of x (see §B.4 and §B.7).

Disproving $\forall x\ P(x)$ means proving $\neg\forall x\ P(x)$, which is equivalent to $\exists x\ \neg P(x)$. One approach is to construct a *counterexample*—a specific value of x for which $P(x)$ is false. Disproving $\exists x\ P(x)$ means proving $\neg\exists x\ P(x)$, which is equivalent to $\forall x\ \neg P(x)$.

Let's illustrate these concepts. The *distance* $\text{dist}(x, y)$ between two bit strings x and y (of the same length) is the number of indices i for which $x_i \ne y_i$ (the number of bits we'd need to flip to change x to y). Suppose we want a large set of bit strings (of a common length) all at distance > 1 from each other. In other words, to get from one string in the set to another, we'd have to flip more than one bit. For example, {1000, 1101, 0110, 0011} fits the criterion: $\mathrm{dist}(1000, 1101) = 2 > 1$, and $\mathrm{dist}(1000, 0110) = 3 > 1$, and so on for the other pairs of strings from the set. This set contains only a quarter (4/16) of all length-4 bit strings, and if we add any other string to this set, we'd violate the criterion: No matter which string we add, it would have distance ≤ 1 to at least one current string. Is there a larger set (that's not a superset of {1000, 1101, 0110, 0011}) fitting the criterion? Let's be methodical. We're interested in subsets of $\{0,1\}^n$. It turns out a good target size is half of all bit strings. Perhaps we conjecture that it's always possible to find such a large set fitting the criterion:

Proposition B.2. *For all $n \geq 2$, there exists a set S containing half of $\{0,1\}^n$ such that every pair of distinct strings in S differ in more than one bit position.*

Proposition B.2 has the form $\forall n\, P(n)$. Suppose n has been fixed to some value. Containing half of $\{0,1\}^n$ means $|S| = |\{0,1\}^n|/2 = 2^{n-1}$, so the domain of $\exists S$ is all subsets of $\{0,1\}^n$ of size 2^{n-1}. Next, it says "every pair of distinct strings in S." Writing $\forall x, y \in S$ would mean $(\forall x \in S)(\forall y \in S)$, which includes the possibility that $y = x$. To make the distinctness explicit, we could write either $(\forall x \in S)(\forall y \in S \setminus \{x\})$ or $(\forall (x,y) \in S^2 : x \neq y)$, but it's more succinct to write $(\forall x, y \in S : x \neq y)$. In summary, here's a translation into formal notation:

$$P(n) = \big(\exists S \subseteq \{0,1\}^n : |S| = 2^{n-1}\big)\big(\forall x, y \in S : x \neq y\big)\big(\mathrm{dist}(x,y) > 1\big)$$

To prove $P(n)$, let's look for an example S. Maybe we try $S = \big\{x \in \{0,1\}^n : x_n = 0\big\}$ (all strings with 0 as the last bit) since clearly $|S| = 2^{n-1}$. Does $\big(\forall x, y \in S : x \neq y\big)\big(\mathrm{dist}(x,y) > 1\big)$ hold for this S? No, because $(x,y) = (000\cdots0, 100\cdots0)$ is a counterexample: This x and y are both in S, but $\mathrm{dist}(x,y) = 1$ since they differ only in the first bit.

If we try some more candidate sets S and fail to find an example for $P(n)$, we might shift gears and try to disprove $P(n)$ by proving the negation, $\neg P(n)$. It often helps to pull a $\neg$ all the way inside a logic formula, and here's an opportunity to practice doing so. The $\neg$ goes on a rampage across the formula for $P(n)$, flipping all the quantifiers in its wake, until it turns $>$ into $\leq$. Omitting the domains for clarity:

$$\begin{aligned}
\neg P(n) &= \neg\,(\exists S)\quad(\forall x, y)\quad(\mathrm{dist}(x,y) > 1)\\
&= \quad(\forall S)\,\neg\,(\forall x, y)\quad(\mathrm{dist}(x,y) > 1)\\
&= \quad(\forall S)\quad(\exists x, y)\,\neg\,(\mathrm{dist}(x,y) > 1)\\
&= \quad(\forall S)\quad(\exists x, y)\quad(\mathrm{dist}(x,y) \leq 1)
\end{aligned}$$

The rampage continues inside the predicate: Since $\mathrm{dist}(x,y) > 1$ means x and y differ on at least two distinct coordinates i and j, we can negate this as follows. Omitting the domain $(i, j \in [n] : i \neq j)$ for clarity:

$$\begin{aligned}
\neg(\mathrm{dist}(x,y) > 1) &= \neg\,(\exists i, j)\quad(\ x_i \neq y_i \ \wedge\ x_j \neq y_j\ )\\
&= \quad(\forall i, j)\,\neg\,(\ x_i \neq y_i \ \wedge\ x_j \neq y_j\ )\\
&= \quad(\forall i, j)\quad(\neg(x_i \neq y_i) \vee \neg(x_j \neq y_j))\\
&= \quad(\forall i, j)\quad(\ x_i = y_i \ \vee\ x_j = y_j\ )
\end{aligned}$$

We used the rule that $\neg(Q \wedge R)$ is equivalent to $\neg Q \vee \neg R$. Thus $\text{dist}(x, y) \leq 1$ means for every pair of distinct coordinates, x and y agree on at least one of them. "Pulling the negation inside" is so ubiquitous, it's usually done tacitly (even subconsciously). The negation of Proposition B.2 is:

Proposition B.3. *There exists an $n \geq 2$ such that for every set S containing half of $\{0,1\}^n$, there exists a pair of distinct strings in S that differ in only one bit position.*

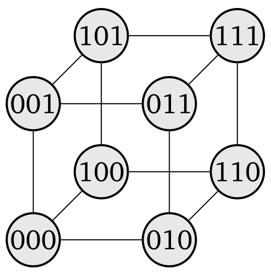
Enough beating around the bush! Proposition B.2 is true, and Proposition B.3 is false:

Proof idea. A bit string's *weight* is the number of 1s in it. Let's include the all-0 string in our set S. This precludes every string of weight 1 because those have distance only 1 from the all-0 string. But it doesn't preclude strings of weight 2, so let's throw all those into S. This would preclude strings of weight 3 (since for every such string, we could flip a single 1 to 0 to get a string of weight 2), but it wouldn't preclude strings of weight 4. This brainstorming inspires us to try the set of all strings of even weight for S. We saw in §A.6.5 that $|S| = 2^{n-1}$ because for every bit string of length $n-1$, can append a unique bit get an even-weight string. It remains to carefully verify that distinct strings in S have distance > 1 from each other. So consider any distinct even-weight strings $x, y \in \{0,1\}^n$. We can reason about x and y even though we don't know exactly what they are. If we flip just a single bit of x, the weight goes either up by one (if flipping a 0 to 1) or down by one (if flipping a 1 to 0), so the weight becomes odd. To get to y, which has even weight, at least one more bit flip is needed. To express this more rigorously in the official proof, we use the basic fact that sums and differences of even numbers are even. ■

Proof of Proposition B.2 / Disproof of Proposition B.3. Consider any $n \geq 2$. Let $S \subseteq \{0,1\}^n$ be all even-weight strings. We already know $|S| = 2^{n-1}$. Consider any two distinct strings $x, y \in S$. We claim $\text{dist}(x, y) \geq 2$. Let a be the number of indices i such that $x_i = 1$ and $y_i = 0$. Let b be the number of indices i such that $x_i = y_i = 1$. Let c be the number of indices i such that $x_i = 0$ and $y_i = 1$. Then $a + b = \text{weight}(x)$ (which is even), and $b + c = \text{weight}(y)$ (which is even), and $a + c = \text{dist}(x, y)$. It follows that $(a+b)+(b+c) = a+2b+c$ is even, and subtracting the even number $2b$ shows that $a + c$ is also even. Since x and y are distinct, $\text{dist}(x, y) \neq 0$, so $\text{dist}(x, y) \geq 2$ since it is even. ■

By similar reasoning, the set of all odd-weight bit strings is also an example for Proposition B.2 / counterexample for Proposition B.3. The sets of even-weight strings and odd-weight strings are actually the only two such examples (Exercise B.6), and if we ask for a set of size $> 2^{n-1}$ then no such example would exist (Theorem B.6), so Proposition B.2 is "tight."

Another perspective: The n-dimensional *hypercube* is an undirected graph whose set of nodes is $\{0,1\}^n$ (each node is labeled by a length-n bit string) and where two nodes are neighbors when they differ in only one bit position ($\{x, y\}$ is an edge when $\text{dist}(x, y) = 1$). When $n = 3$, the graph looks like a cube. (The 1st bit position corresponds to front versus back, the 2nd to left versus right, and the 3rd to bottom versus top.) If $S \subseteq \{0,1\}^n$ is such that $\text{dist}(x, y) > 1$ for every pair of distinct $x, y \in S$, that means no edge of the hypercube has both endpoints in S.



An *independent set* in an undirected graph is a subset of nodes S such that no edge has both endpoints in S. The upshot of our discussion is that the n-dimensional hypercube has an independent set of half the nodes. (In fact, it has exactly two independent sets of size 2^{n-1} and no independent set of size $> 2^{n-1}$.)

B.2 Proof by counting

B.2.1 Counting in two different ways

A technique for proving that two quantities are equal is to show that they count the same thing in different ways. Here's a classic result that exemplifies this.

Theorem B.4. *For every undirected graph (with multi-edges and self-loops allowed), the sum of the degrees of all nodes is an even number.*

Proof idea. If we visualize an arbitrary undirected graph, the sum of the degrees counts the "total number of places where an edge touches a node." Let's count the same quantity in an edge-centric rather than node-centric way: Each edge touches nodes in two places, so the quantity is the sum of several copies of 2 (specifically, one copy of 2 for each edge). This is even since it equals 2 times an integer. ■

Proof. Consider an arbitrary undirected graph $G = (V, E)$. Each non-self-loop edge contributes 1 to the degrees of two nodes, and each self-loop contributes 2 to the degree of one node, so each edge contributes 2 to $\sum_{v\in V} \deg(v)$. Thus $\sum_{v\in V} \deg(v) = 2 \cdot |E|$, which is even. ■

This proof also showed the handy result that for every undirected graph, the number of edges equals half the sum of the degrees. For example, let's count the edges in the n-dimensional hypercube discussed at the end of §B.1.2. Each node $x \in \{0,1\}^n$ has degree n since we can reach any neighbor by flipping one of the n bits in x. There are 2^n nodes, so the sum of the degrees is $n2^n$, and the number of edges is $\frac{1}{2} \cdot n2^n = n2^{n-1}$.

For a different illustration of the counting technique, let's see why the sum of the first n powers of 2 is exactly 1 less than the next power of 2.

Theorem B.5. *For all $n \in \mathbb{N}$, we have $2^{n-1} + 2^{n-2} + \cdots + 2^1 + 2^0 = 2^n - 1$.*

Recalling the "telescoping" trick for evaluating a geometric sum, we have a quick but perhaps unenlightening proof:

First proof. $\sum_{k=0}^{n-1} 2^k = (2-1)\sum_{k=0}^{n-1} 2^k = (2^n + \cdots + 2^1) - (2^{n-1} + \cdots + 2^0) = 2^n - 2^0 = 2^n - 1$. ■

A different approach is to think about what the two sides of the equation are counting:

Second proof. Of course, $2^n - 1$ counts the size of the set $[2^n - 1] = \{1, 2, 3, \ldots, 2^n - 1\}$. Let's partition this set into intervals: $S_0 = \{1\}$, $S_1 = \{2, 3\}$, $S_2 = \{4, 5, 6, 7\}$, and generally $S_k = \{2^k, 2^k + 1, \ldots, 2^{k+1} - 1\}$ for each $k \in \{0, \ldots, n-1\}$. Since $S_k = [2^{k+1} - 1] \setminus [2^k - 1]$, we have $|S_k| = (2^{k+1} - 1) - (2^k - 1) = 2 \cdot 2^k - 2^k = 2^k$. By the addition principle, $2^n - 1 = |S_{n-1}| + |S_{n-2}| + \cdots + |S_0| = 2^{n-1} + 2^{n-2} + \cdots + 2^0$. ■

For a different perspective on this proof, recall that the binary (base two) representation of a natural number x is a bit string $x_\ell\, x_{\ell-1} \cdots x_1\, x_0$, which means $x = x_\ell \cdot 2^\ell + x_{\ell-1} \cdot 2^{\ell-1} + \cdots + x_1 \cdot 2^1 + x_0 \cdot 2^0$ (the locations of the 1s indicate which powers of 2 to sum to get x). Here, ℓ is not unique since there could be any number of leading 0s. In binary, 2^k is 1 followed by k many 0s. In our second proof, the interval S_k contains all numbers whose binary representation is 1 followed by a bit string of length k, and there are 2^k many such strings. Thus we partitioned $[2^n - 1]$ according to the most significant 1's location. Thinking purely in terms of strings, we partitioned $\{0,1\}^n \smallsetminus \{0\}^n$ (all bit strings of length n except the all-0 string) according to the leftmost 1's location.

Using arithmetic with binary numbers, we can view Theorem B.5 from yet another angle:

Third proof. Summing the binary representations yields a 1 in each column:

$$\begin{array}{r} 2^{n-1} \\ +\, 2^{n-2} \\ +\, 2^{n-3} \\ \vdots \\ +\, 2^1 \\ +\, 2^0 \\ \hline =\, ? \end{array} \qquad\qquad \begin{array}{r} \overbrace{1000\cdots000}^{n-1 \text{ many}} \\ +\, 0100\cdots000 \\ +\, 0010\cdots000 \\ \vdots \\ +\, 0000\cdots010 \\ +\, 0000\cdots001 \\ \hline =\, \underbrace{1111\cdots111}_{n \text{ many}} \end{array}$$

If we add 1 to this number, all of its 1s flip to 0s as the carry propagates to the left end, resulting in 1 followed by n many 0s, which is 2^n. To see this propagation in action:

$$\cdots + 8 + 4 + 2 + 1 + 1 \;=\; \cdots + 8 + 4 + 2 + 2 \;=\; \cdots + 8 + 4 + 4 \;=\; \cdots + 8 + 8$$

and so on, which ends with $2^{n-1} + 2^{n-1} = 2^n$. ■

Care is needed when saying "and so on" because it sometimes hides subtle mistakes or ambiguities. Our third proof was a simple example of "proof by algorithm" (§ B.5): Adding 1 to the binary number $111\cdots1$ is handled by the basic addition algorithm, which loops over the bits from least significant to most significant—the "and so on" at the end of our third proof corresponds to this loop.

Even more proofs, yielding a result more general than Theorem B.5, are covered in §B.5 and Exercise B.26.

B.2.2 Pigeonhole principle

The self-evident *pigeonhole principle* says that if more than n pigeons fly into n holes, then at least one hole gets more than one pigeon, since there aren't enough holes to go around. Stated abstractly: For all finite sets S and T with $|S| > |T|$ and every function $f : S \to T$, there exist distinct $x, y \in S$ such that $f(x) = f(y)$. Elements of S are pigeons and elements of T are holes—pigeon x flies into hole $f(x)$. Recall that an injection is a function $f : S \to T$ where each codomain element has at most one preimage, in other words, $f(x) \neq f(y)$ for all distinct $x, y \in S$. The pigeonhole principle says no injection has domain larger than codomain.

An application is that lossless data compression is impossible in general: We cannot compress a completely arbitrary n-bit long file to a shorter file and hope to recover the original file with certainty from the compressed version. In other words, for every compression scheme, there will exist two distinct files that get compressed to the same shorter file, rendering it impossible to determine which was the true original. To see why, let's view the set of all possible n-bit files as $\{0,1\}^n$, and denote the set of all files that are less than n bits long as:

$$\{0,1\}^{<n} \;=\; \{0,1\}^{n-1} \cup \{0,1\}^{n-2} \cup \cdots \cup \{0,1\}^1 \cup \{0,1\}^0$$

(The set $\{0,1\}^0$ contains only the empty string.) A compression scheme would be a function $f\colon \{0,1\}^n \to \{0,1\}^{<n}$. By Theorem B.5:

$$|\{0,1\}^n| \;=\; 2^n \;>\; 2^{n-1} + 2^{n-2} + \cdots + 2^1 + 2^0 \;=\; |\{0,1\}^{<n}|$$

Since a compression scheme's domain is larger than its codomain, the pigeonhole principle says it can't be an injection. "Pigeons" are original files (elements of $\{0,1\}^n$). "Holes" are compressed files (elements of $\{0,1\}^{<n}$). Pigeon x flying into hole $f(x)$ means $f(x)$ is the compressed version of x. "Loss" occurs since there must exist $x \neq y$ such that $f(x) = f(y)$: A recovery procedure wouldn't know whether x or y was the original.

Real-world lossless compression schemes (such as .zip files) shrink many typical files, but some files (albeit rare in practice) don't get shrunk by even a single bit.

Next, let's revisit Proposition B.2 from §B.1.2: We proved there exists a set S containing half of $\{0,1\}^n$ such that every pair of distinct strings in S differ in more than one bit position. Using the pigeonhole principle, we prove this is tight: There does not exist such a set containing more than half of $\{0,1\}^n$.

Theorem B.6. *For all $n \geq 2$ and every $S \subseteq \{0,1\}^n$ with $|S| > 2^{n-1}$, there exist distinct $x, y \in S$ such that* $\text{dist}(x, y) = 1$.

Proof idea. S contains more than 2^{n-1} "pigeons." We want them to fly into 2^{n-1} "holes" so that pigeons in the same hole must correspond to strings that differ in only one bit position. Which bit position? Let's try for the last position. We let holes be elements of $\{0,1\}^{n-1}$, and send pigeon x to the hole corresponding to x's first $n-1$ bits. Pigeons in the same hole agree on their first $n-1$ bits and hence differ in only their last bit. Nothing is special about the last position—the argument would work if we singled out another position. ■

Proof. Consider any such S. Define $f\colon S \to \{0,1\}^{n-1}$ by $f(x) = x_1 \cdots x_{n-1}$. Since $|S| > |\{0,1\}^{n-1}|$, the pigeonhole principle says f is not an injection. Thus there exists a string in $\{0,1\}^{n-1}$ with at least two distinct preimages: $x, y \in S$ with $f(x) = f(y)$. Since $x_1 \cdots x_{n-1} = y_1 \cdots y_{n-1}$, we have $x_n \neq y_n$ and $\text{dist}(x, y) = 1$. ■

Some variants of the pigeonhole principle are equally self-evident: If fewer than n pigeons fly into n holes, then some hole is pigeonless. If exactly n pigeons fly into n holes and every hole gets at most one pigeon, then every hole gets exactly one pigeon. If exactly n pigeons fly into n holes and every hole gets at least one pigeon, then every hole gets exactly one pigeon. Let's summarize all the variants abstractly:

Fact B.7 (Pigeonhole principle). *For every function $f : S \to T$ where S and T are finite:*

- *If $|S| > |T|$ then f is not an injection.*
- *If $|S| < |T|$ then f is not a surjection.*
- *If $|S| = |T|$ and f is an injection, then f is a bijection.*
- *If $|S| = |T|$ and f is a surjection, then f is a bijection.*

B.3 Proving implications

B.3.1 Direct proof

The most basic way to prove propositions of the form "if P then Q" (implications) is *direct proof*: First suppose P is true, and then after some logical reasoning, deduce that Q must also be true. P is the *assumption* or *hypothesis*, and Q is the *conclusion*. Here's an example.

Theorem B.8. *For every undirected graph G with n nodes (and no multi-edges or self-loops), if G has a node with degree $n-1$ then G is connected.*

Proof idea. First, let's understand what we want to prove. Slightly shortening the statement:

$$\overbrace{\text{For every graph } G \text{ with } n \text{ nodes,}}^{\forall G}\ \text{if}\ \overbrace{G \text{ has a node with degree } n-1}^{P(G)}\ \overbrace{\text{then}}^{\Rightarrow}\ \overbrace{G \text{ is connected.}}^{Q(G)}$$

Thus the logical structure is $\forall G\ (P(G) \Rightarrow Q(G))$ where:

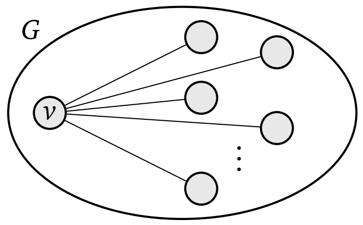
- G's domain is the set of all undirected graphs with no multi-edges or self-loops, and n is defined as the number of nodes in G.
- $P(G)$ is the predicate "$\exists v\ (\deg(v) = n-1)$" where v's domain is G's set of nodes.
- $Q(G)$ is the predicate "$\forall u, w\ (\exists$ walk between u and $w)$" where u's and w's domain is G's set of nodes. (This is the definition of "G is connected.") Though the $\exists$ quantifier here is not written in the form ($\exists$ variable) (predicate), it just asserts that the set of all walks in G between u and w is nonempty.

With all the quantifiers laid out, we see the flow of a proof:

- Consider an arbitrary G with n nodes. (In the official proof, we won't bother writing this, since it's automatic when a theorem begins with $\forall$.)
- To give a direct proof of "$P(G) \Rightarrow Q(G)$":
  - Assume there exists a node with degree $n-1$ in G, and call this special node v. If there exist more than one node with degree $n-1$, it doesn't matter which one we pick for v.
  - To deduce $Q(G)$:
    - Consider arbitrary nodes u and w in G.
    - Exhibit a walk between u and w in G.

Achieving the last bullet takes some insight, but the rest of the plan is just "following instructions."

Let's envision G if v has degree $n-1$. Since G has no multi-edges or self-loops, the most edges v could have would be one edge to each of the other $n-1$ nodes (besides v itself). Thus $\deg(v)$ can't exceed $n-1$, and the only way it could equal $n-1$ is if each of those $n-1$ edges is present. So v must have an edge to each other node. (In this illustration, there could be edges between pairs of non-v nodes, but we don't draw those edges since we won't need them.)



Formally, this is by the pigeonhole principle (specifically, Fact B.7's third bullet): The $n-1$ pigeons are the edges incident to v, and the $n-1$ holes are the nodes other than v. Pigeon $\{v,u\}$ flies into hole u. No hole gets more than one pigeon, so every hole gets a pigeon.

Now, we just need to find a walk between arbitrary nodes u and w. We need not consider the case $u=w$, since every node has a walk of length 0 to itself. (A walk's length is its number of edges.) So assume $u \neq w$. There might be many ways to get from u to w, but one option is "via v": Take the edge $\{u,v\}$ and then $\{v,w\}$. Let's write this walk succinctly as $u — v — w$. To be careful, we should separately consider the possibility that u or w is v itself, in which case there's a walk of length 1 (a single edge) rather than length 2. The lengths of the walks don't matter, but we mention them anyway in the official proof. ■

Proof. Assume some node v has degree $n-1$. Since G has no multi-edges or self-loops, v must have an edge to each of the other $n-1$ nodes, by the pigeonhole principle. To see that G is connected, consider any two nodes $u \neq w$. If $u=v$ or $w=v$ then $u — w$ is a walk (of length 1). If $u \neq v$ and $w \neq v$ then $u — v — w$ is a walk (of length 2). ■

A direct proof of $\forall x\ (P(x) \Rightarrow Q(x))$ shows $\forall x\ Q(x)$ where x's domain is restricted to values for which $P(x)$ holds. Also, we can write a proposition of the form $\exists x\ P(x)$ as just $\exists x$ where x's domain is restricted to values for which $P(x)$ holds.

B.3.2 Contrapositive proof

The *contrapositive* of $P \Rightarrow Q$ is $\neg Q \Rightarrow \neg P$, which is equivalent. For example, "if it's raining then it's cloudy" means the same as "if it's not cloudy then it's not raining."

A *contrapositive proof* of $P \Rightarrow Q$ is a direct proof of $\neg Q \Rightarrow \neg P$: First suppose Q is false, and then after some logical reasoning, deduce that P must also be false. For the following example, recall that $\binom{n}{k} = \frac{n\cdot(n-1)\cdots(n-k+1)}{k\cdot(k-1)\cdots 1}$ is the number of ways to choose a set of k objects from n objects.

Theorem B.9. *For every undirected graph G with n nodes (and no multi-edges or self-loops), if G has more than $\binom{n-1}{2}$ edges then every node in G has degree at least* 1.

Proof idea. Let's parse the statement:

$$\overbrace{\text{For every } G,}^{\forall G}\ \text{if}\ \overbrace{G \text{ has more than } \tbinom{n-1}{2} \text{ edges}}^{P(G)}\ \overbrace{\text{then}}^{\Rightarrow}\ \overbrace{\text{every node in } G \text{ has degree at least } 1.}^{Q(G)}$$

Thus the logical structure is $\forall G\ (P(G) \Rightarrow Q(G))$ where:

- $P(G)$ is the predicate "$|E| > \binom{n-1}{2}$" where E is G's set of edges.
- $Q(G)$ is the predicate "$\forall v\ (\deg(v) \geq 1)$" or equivalently "$\forall v\ \exists u\ (\{u,v\} \in E)$" where v's and u's domain is G's set of nodes.

What if we attempt a direct proof? Assume G has n nodes and more than $\binom{n-1}{2}$ edges. To prove $Q(G)$, we consider an arbitrary node v, and try to show that v has an edge to some node u. Which u? The trouble is that the hypothesis $|E| > \binom{n-1}{2}$ is a "global" property that doesn't tell us which node u to focus on (which is something "local" to v).

Instead of a direct proof, we do a contrapositive proof. Assume $Q(G)$ is false. Pulling the negation inside $\neg Q(G)$, this means $\exists v\ (\deg(v) < 1)$. Since degrees are natural numbers, the only degree less than 1 is 0. So some special node v has no edges whatsoever touching it. (This is the extreme opposite of the situation in Theorem B.8.)

We want to prove $\neg P(G)$, which says $|E| \leq \binom{n-1}{2}$. For a graph with n nodes (and no multi-edges or self-loops), the maximum possible number of edges is $\binom{n}{2}$ since each edge is a set of two nodes. Since all of G's edges have non-v nodes for both endpoints, we can view G as a graph on $n-1$ nodes (plus the isolated node v), so it has at most $\binom{n-1}{2}$ edges.

In general, pulling the negation inside $\neg Q$ and $\neg P$ is so routine that it's not elaborated in proofs. ■

Proof. Suppose not every node has degree at least 1. That means some node has degree 0. Since every edge connects two of the other $n-1$ nodes, there are at most $\binom{n-1}{2}$ possibilities for edges in G. Thus G does not have more than $\binom{n-1}{2}$ edges. ■

Here's another example where contrapositive reasoning is the right tool for the job.

Theorem B.10. *For all real numbers $x_1, x_2, \ldots, x_n$ and y, if $x_1 + \cdots + x_n < y$ then $x_i < y/n$ holds for at least one index $i \in [n]$.*

Proof idea. The statement's logical structure is $\forall x, y\ (P(x,y) \Rightarrow Q(x,y))$ where:

- x's domain is $\mathbb{R}^n$ and y's domain is $\mathbb{R}$.
- $P(x,y)$ is the predicate "$\sum_{i=1}^{n} x_i < y$".
- $Q(x,y)$ is the predicate "$\exists i\ (x_i < y/n)$" where i's domain is $[n]$.

To do a contrapositive proof, we first assume $Q(x,y)$ is false. Pulling the negation inside $\neg Q(x,y)$ yields $\forall i\ (x_i \geq y/n)$. Our goal is to show $\neg P(x,y)$, which is $\sum_{i=1}^{n} x_i \geq y$. The latter follows by the basic fact that a sum of n numbers, each of which is $\geq y/n$, is $\geq y$. ■

Proof. Consider any real numbers $x_1, x_2, \ldots, x_n$ and y, and suppose $x_i \geq y/n$ holds for every index i. Summing these inequalities yields $\sum_{i=1}^{n} x_i \geq \sum_{i=1}^{n} y/n = n \cdot y/n = y$. ■

B.3.3 Proving "if and only if" propositions

The standard way to prove a proposition of the form "P if and only if Q" ("P iff Q" for short) is to separately prove the two implications "if P then Q" and "if Q then P." This works because $P \Leftrightarrow Q$ is equivalent to $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$ (they have the same truth table).

- The left-to-right direction $P \Rightarrow Q$ is the "only if" half of "if and only if":
  The *only* way P can be true is *if* Q is also true.
- The right-to-left direction $P \Leftarrow Q$ is the "if" half of "if and only if":
  P is true *if* Q is true.

These two implications are *converses* of each other, and they may require substantially different proofs. A proof of $Q \Rightarrow P$ generally isn't a proof of $P \Rightarrow Q$ "in reverse."

Theorem B.11. *For every binary matrix, there exists an all-*1 *column iff for every way of choosing one entry from each column, at least one chosen entry is* 1.

Proof idea. The logical structure is $\forall M\ (P(M) \Leftrightarrow Q(M))$ where M's domain is the set of all matrices with entries in $\{0,1\}$. Let's imagine M as a 2-dimensional array of pebbles, where 1 entries are black pebbles and 0 entries are white pebbles. $P(M)$ says at least one column has only black pebbles. $Q(M)$ says if we put one pebble from the first column into a bag, and put one pebble from the second column into the bag, and so on for all columns, then we're guaranteed to have a black pebble in the bag.

The left-to-right implication $P(M) \Rightarrow Q(M)$ is straightforward by direct proof: Assume there's an all-black column. No matter how we choose a pebble from each column, of course the pebble we choose from the all-black column will be black.

The right-to-left implication $Q(M) \Rightarrow P(M)$ is more subtle. Let's see why a direct proof fails: Assume that for every way of choosing a pebble from each column, at least one chosen pebble is black. For some particular way, the chosen black pebble might come from the first column, while for a different way, the chosen black pebble might come from the second column, and so on. All the different possible "ways" might not agree on which column their chosen black pebble comes from. So to prove $P(M)$, it wouldn't be clear which particular column should be all-black. However, it will become straightforward if we look at the contrapositive, $\neg P(M) \Rightarrow \neg Q(M)$. To help pull the negations inside, let's make the quantifiers more explicit. Defining m as the number of rows and n as the number of columns (so M is $m \times n$):

- $P(M)$ is "$\exists j\ \forall i\ (M_{i,j} = 1)$" where $j \in [n]$ is a column index and $i \in [m]$ is a row index.
- $Q(M)$ is "$\forall (i_1, i_2, \dots, i_n)\ \exists j\ (M_{i_j,j} = 1)$" where $i_1 \in [m], \dots, i_n \in [m]$ are row indices (saying which entry to choose from each column) and $j \in [n]$ is a column index.

Our assumption $\neg P(M)$ says $\forall j\ \exists i\ (M_{i,j} = 0)$ (every column has at least one white pebble), and our desired conclusion $\neg Q(M)$ says $\exists (i_1, i_2, \dots, i_n)\ \forall j\ (M_{i_j,j} = 0)$ (there's a way to choose a pebble from each column such that all chosen pebbles are white). Now, the implication is clear: If each column has a white pebble, then it's possible to choose a white pebble from each column. ■

Proof. Consider any binary matrix.

$\Rightarrow$: We give a direct proof. Assuming there's an all-1 column, clearly every way of choosing one entry from each column will have a 1 entry from the all-1 column.

$\Leftarrow$: We give a contrapositive proof. Assume there's no all-1 column. That means every column has at least one 0 entry. Hence there exists a way of choosing one entry from each column such that every chosen entry is 0. In other words, it's not the case that for every way of choosing one entry from each column, at least one chosen entry is 1. ■

"iff" propositions are relevant to showing that two sets S and T are equal: $\forall x\ (x \in S \Leftrightarrow x \in T)$. To prove $S = T$, we separately prove $\forall x\ (x \in S \Rightarrow x \in T)$ (that is, $S \subseteq T$) and $\forall x\ (x \in T \Rightarrow x \in S)$ (that is, $T \subseteq S$). We illustrate this with a theorem about subsets of a cartesian product $X \times Y = \{(x, y) : x \in X \text{ and } y \in Y\}$.

- $S \subseteq X \times Y$ is itself a cartesian product when $S = A \times B$ for some $A \subseteq X$ and $B \subseteq Y$.
- S is *mix-and-matchable* when for all $(x, y) \in S$ and all $(x', y') \in S$, we have $(x, y') \in S$.

Here, x' (pronounced "x prime") is a variable name that's separate from x but indicates that x' plays a role analogous to x (and similarly for y').

Theorem B.12. *For every $S \subseteq X \times Y$, S is a cartesian product iff S is mix-and-matchable.*

Proof idea. The left-to-right implication is straightforward from the definition of cartesian product. The right-to-left implication is more subtle: Assuming S is mix-and-matchable, we want to find sets $A \subseteq X$ and $B \subseteq Y$ such that $S = A \times B$. To accomplish this, we define $A \times B$ as the "smallest" cartesian product containing S (so $S \subseteq A \times B$ holds automatically), and argue that $A \times B \subseteq S$ holds by S's mix-and-matchability. ■

Proof. Consider any $S \subseteq X \times Y$.

$\Rightarrow$: We give a direct proof. Assume S is a cartesian product $A \times B$. To see that S is mix-and-matchable, consider any $(x, y) \in S$ and $(x', y') \in S$. Then $x \in A$ and $y' \in B$, so by definition $(x, y') \in A \times B = S$.

$\Leftarrow$: We give a direct proof. Assume S is mix-and-matchable. To show that S is a cartesian product, we argue that $S = A \times B$ where $A = \big\{x \in X : \text{there exists } y \text{ such that } (x, y) \in S\big\}$ and $B = \big\{y \in Y : \text{there exists } x \text{ such that } (x, y) \in S\big\}$.

Claim B.13. *For all $(x, y) \in X \times Y$, we have $(x, y) \in S$ iff $(x, y) \in A \times B$.*

Proof. Consider any $(x, y) \in X \times Y$.

$\Rightarrow$: We give a direct proof. Assume $(x, y) \in S$. Then $x \in A$ by the definition of A, and $y \in B$ by the definition of B, so $(x, y) \in A \times B$.

$\Leftarrow$: We give a direct proof. Assume $(x, y) \in A \times B$. Since $x \in A$, we have $(x, y') \in S$ for some y' by the definition of A. (The definition of A mentions y, but we rename that variable to y' since the name y is already in use.) Since $y \in B$, we have $(x', y) \in S$ for some x' by the definition of B. Since $(x, y') \in S$ and $(x', y) \in S$, we have $(x, y) \in S$ since S is mix-and-matchable.

This finishes the proof of Claim B.13. ■

This finishes the proof of Theorem B.12. ■

B.4 Proof by contradiction

If an assumption implies something that's false, then the assumption must have been wrong. The *proof by contradiction* technique embodies this idea. To prove a proposition P by contradiction: First suppose P is false, and then after logical reasoning, deduce that some proposition R must be both true and false (which is ridiculous). This form of argument is valid because $R \land \neg R$ is a *contradiction*—it can't be true—and therefore the assumption $\neg P$ couldn't have been true. Therefore P is true. This roundabout strategy is powerful and flexible due to the freedom to pick any $R \land \neg R$ for the contradiction.

B.4.1 Matrices, graphs, and parties

Theorem B.14. *For every binary matrix, either every row has at least one* 1 *or every column has at least one* 0*.*

Proof idea. Let's write the statement formally:

$$\forall M \; \big((\forall i \; \exists j \; (M_{i,j} = 1)) \;\lor\; (\forall j \; \exists i \; (M_{i,j} = 0)) \big)$$

M is a binary matrix, i is a row index, and j is a column index. To do a proof by contradiction, we start by negating the statement. Pulling the negation all the way inside, the $\lor$ flips to $\land$, and all quantifiers flip to the opposite types:

$$\exists M \; \big((\exists i \; \forall j \; (M_{i,j} = 0)) \;\land\; (\exists j \; \forall i \; (M_{i,j} = 1)) \big)$$

This says if Theorem B.14 were false, there would exist a matrix M with an all-0 row (because $\exists i \; \forall j \; (M_{i,j} = 0)$) and an all-1 column (because $\exists j \; \forall i \; (M_{i,j} = 1)$). But then the entry in this row and this column would be both 0 and 1. The contradiction is $R \land \neg R$ where R is "$M_{i,j} = 0$": R is true because every entry in row i is 0, and R is false because every entry in column j is 1. The assumption we made at the beginning—namely that Theorem B.14 is false—led to an impossible conclusion. Therefore Theorem B.14 is true. ■

Proof. Suppose for contradiction there exists a binary matrix M in which some row, say the i^{th}, is all-0 and some column, say the j^{th}, is all-1. Then $M_{i,j} = 0$ since this entry is in row i, but on the other hand, $M_{i,j} = 1$ since this entry is in column j. This is a contradiction since $M_{i,j}$ can't be both 0 and 1. ■

Here's a closely related result: Suppose each pair of people at a party are either friends with each other or enemies of each other. Then either everyone has at least one friend at the party, or everyone has at least one enemy at the party. Let's phrase this as a proposition about undirected graphs: Nodes are people. The presence of edge $\{u, v\}$ means u and v are friends. The absence of edge $\{u, v\}$ means u and v are enemies. Person v having at least one friend means $\deg(v) \geq 1$. Person v having at least one enemy means not all of the other $n-1$ people are friends with v, so $\deg(v) \leq n-2$.

Theorem B.15. *For every undirected graph with $n \geq 2$ nodes (and no multi-edges or self-loops), either every node has degree at least* 1 *or every node has degree at most $n-2$.*

Proof idea. The graph's adjacency matrix (§A.10.3) is a symmetric $n \times n$ binary matrix A where $A_{u,v} = 1$ means nodes u and v are neighbors ("friends"), and $A_{u,v} = 0$ means u and v are non-neighbors ("enemies"), assuming the nodes are numbered $1, 2, \dots, n$. A neighbor of v corresponds to a 1 in v's row (and a 1 in v's column). A non-neighbor of v corresponds to a 0 in v's column (and a 0 in v's row). So, it seems Theorem B.15 should be a corollary of Theorem B.14: Either every row has a 1 (everyone has a friend) or every column has a 0 (everyone has an enemy). We should get suspicious after noticing that every column has a 0 no matter what: The diagonal entries are all 0 since the graph has no self-loops. But a person isn't their own enemy—we should only consider non-neighbors of v besides v itself (off-diagonal 0s in v's column). We adjust the proof of Theorem B.14 to handle this issue by ignoring the diagonal. This illustrates that casual reasoning is susceptible to "gotchas" that lead to flawed proofs. ■

Proof. Suppose for contradiction there exists a graph G with $n \geq 2$ nodes such that some node u has degree 0, and some node v has degree $n-1$. Since $n-1 > 0$, u and v have different degrees and thus aren't the same node. By the pigeonhole principle, v is neighbors with all other nodes; in particular, $\{u, v\}$ is an edge in G. Since u has no neighbors, $\{u, v\}$ is not an edge in G. This is a contradiction, since $\{u, v\}$ can't be both an edge and a non-edge. ■

Corollary B.16. *For every undirected graph with $n \geq 2$ nodes (and no multi-edges or self-loops), there exist two different nodes with the same degree as each other.*

Proof idea. Think of nodes as pigeons and degrees as holes—pigeon v flies into hole $\deg(v)$. We'd like to find two pigeons in the same hole. The possible degrees are $0, 1, 2, \dots, n-1$ since the graph has no multi-edges or self-loops. There are as many holes as pigeons (n of each), so the pigeonhole principle doesn't immediately tell us some hole has multiple pigeons. However, Theorem B.15 says either the 0 hole or the $n-1$ hole is unoccupied. Either way, only $n-1$ possible degrees remain, so now we can invoke the standard pigeonhole principle. This is a nonconstructive argument: It doesn't directly say which two nodes have the same degree, it merely shows that two such nodes must exist. ■

Proof. Consider any graph with $n \geq 2$ nodes. By Theorem B.15, the nodes' degrees are either all in $\{1, 2, \dots, n-1\}$ or all in $\{0, 1, \dots, n-2\}$. In either case, only $n-1$ degrees are possible. Since there are more nodes than possible degrees, some two nodes must have the same degree, by the pigeonhole principle. ■

B.4.2 Proving implications by contradiction

$P \Rightarrow Q$ is equivalent to $(\neg P) \vee Q$ (they have the same truth table), and likewise $P \vee Q$ is equivalent to $(\neg P) \Rightarrow Q$. That is, implications are disjunctions ("or"s) and vice versa. To convert between an implication and a disjunction (in either direction), we just negate the left side.

So to prove a disjunction, we can view it as an implication and do a direct or contrapositive proof. For example, we can rephrase Theorem B.14 as: *For every binary matrix, if not every row has at least one* 1*, then every column has at least one* 0. Now it's amenable to a straightforward direct proof: Assuming some row is all-0, of course every column has a 0, namely in the all-0 row. Proof by contradiction wasn't necessary for Theorem B.14.

On the flip side, to prove an implication by contradiction, we can view it as a disjunction, negate the whole thing, and pull the negation inside: $\neg(P \Rightarrow Q)$ is equivalent to $P \wedge \neg Q$. (The only F row in the truth table of $P \Rightarrow Q$ is the row where P is T and Q is F.) So to prove "if P then Q" by contradiction: First suppose P is true and Q is false, and then deduce a contradiction $R \wedge \neg R$.

Direct and contrapositive proofs are special cases of proof by contradiction:

- In a direct proof "assume P, deduce Q":
  If we also assume $\neg Q$, then the proof reaches the contradiction $Q \wedge \neg Q$.
- In a contrapositive proof "assume $\neg Q$, deduce $\neg P$":
  If we also assume P, then the proof reaches the contradiction $P \wedge \neg P$.

Here's an example where the contradiction doesn't happen "at P" or "at Q" but at some other proposition. So unlike with Theorem B.14, this isn't just a direct or contrapositive proof in disguise:

Theorem B.17. *For every binary matrix, if every row is at least half* 1s *then some column is at least half* 1s.

Proof. Suppose for contradiction there exists an $m \times n$ binary matrix M in which every row is at least half 1s but no column is at least half 1s. The total number of 1s in M is $\geq mn/2$ since each of the m rows has $\geq n/2$ many 1s. The total number of 1s in M is $< mn/2$ since each of the n columns has $< m/2$ many 1s. This is a contradiction since the number of 1s can't be both $\geq mn/2$ and $< mn/2$. ■

B.5 Proof by algorithm

Math is useful for understanding algorithms, but the connection goes the other way too: Algorithms can be useful for proving theorems, even when the theorem statement has nothing to do with algorithms. For example, one way to prove $\forall x\ \exists y\ P(x, y)$ is to exhibit an algorithm that takes an arbitrary value of x as input, and outputs a value of y such that $P(x, y)$ holds. Such proofs are constructive because they not only show that a suitable y exists, but explain how to find such a y.

Two basic styles of algorithm are *iterative* (§B.5.1) and *recursive* (§B.5.2).

B.5.1 Iterative algorithms

An iterative algorithm's main control flow organization is a loop. Proving that an iterative algorithm is correct (it always produces the desired output) typically involves identifying an *invariant*: a property that holds at the boundaries between iterations of the loop. We want the invariant to hold at the end, so we show that it holds after every iteration along the way. A correctness proof shows:

- The invariant holds before the first iteration.
- The invariant is *maintained*: If it holds before an iteration, then it holds after the iteration.
- The loop is guaranteed to terminate.

- If the invariant holds when the loop terminates, then the output is correct.

Since the invariant holds before the first iteration, it also holds after the first iteration, which implies it also holds after the second iteration, which implies it also holds after the third iteration, and so on. (This proof strategy is called *induction*.) Since the invariant holds when the loop terminates, the algorithm outputs what we want.

Let's illustrate this framework. A walk in a graph goes from a node u to a node v by traversing a sequence of edges. A walk may repeat edges (and nodes), but a path may not repeat nodes (or edges).

Theorem B.18. *For every graph and nodes u and v, if there exists a walk from u to v then there exists a path from u to v.*

For conciseness, the statement doesn't say whether the graph is directed or undirected, because the theorem holds for both flavors. It also omits the fact that multi-edges and self-loops are allowed, because they aren't pertinent.

Proof idea. We do a direct proof. Suppose there's a walk from u to v. If the walk happens to be a path, we're done. Otherwise, it visits some node w at least twice. We might as well skip the segment of the walk between visits to w: Upon the first visit to w, we can instead have the walk continue as it would have after the subsequent visit to w. This is progress toward turning the walk into a path, because we eliminated a revisit. Some revisited nodes might remain on the walk, but we repeat this process until no node is visited more than once, in which case we have a path.

We just described an algorithm that finds a path from u to v, thereby proving that such a path exists. To make the proof more rigorous, we write pseudocode for the algorithm and prove that it maintains a certain invariant. We let P stand for the "current" walk from u to v (which becomes a path by the end). Thus, P is a variable that represents different walks at different times during the algorithm's execution. ■

Proof. Consider any directed graph. (The same argument works for an undirected graph.) Assume there exists a walk from u to v. This algorithm turns such a walk into a path from u to v:

```
let $P$ be a walk $(u \to \cdots \to v)$
while the nodes on $P$ are not all distinct:
    find a repeated node $w$ on $P = (u \to \cdots \to w \underbrace{\to \cdots \to w}_{S} \to \cdots \to v)$
    remove the segment $S$, so $P$ becomes $(u \to \cdots \to w \to \cdots \to v)$
output $P$
```

This algorithm maintains the invariant that P is a walk from u to v, because after removing S (which forms a closed walk containing w), P evidently remains a valid walk from u (even if $w = u$) to v (even if $w = v$). The invariant holds before the first iteration, so it holds after every iteration. The loop terminates because P's length (number of edges) decreases in every iteration (since S contains at least one edge, and no edges are ever added to P) but can't go below 0. When the loop terminates, P is a walk from u to v (by the invariant) and has no repeated nodes (by the loop's termination condition), so P is a path from u to v. In summary, we know a path from u to v exists because the algorithm outputs one. ■

By Theorem B.18, we could replace "walk" with "path" in the definitions of *connected* undirected graphs and *strongly connected* directed graphs (§A.2.1).

Next, let's see a proof that uses an algorithm to analyze something rather than to find something. A binary tree is a rooted tree in which each node has either zero children (a leaf) or exactly two children (an internal node).

Theorem B.19. *For every binary tree, the number of leaves is exactly one more than the number of internal nodes.*

Proof idea. We repeatedly shrink the tree so the numbers of leaves and internal nodes both decrease by exactly one. Eventually the tree is a single node—one leaf and zero internal nodes—so there must have been one more leaf than internal node all along. To shrink the tree, we delete two sibling leaves at the greatest *depth* (length of the path from the root). The algorithm outputs nothing because the invariant is all that matters. ■

In pseudocode, we generally use "←" for assigning to a variable, because the conventional "=" may misleadingly evoke "permanent mathematical equality." But for some algorithms dealing with directed graphs, "←" would be confusing since arrows represent directed edges.

Proof. Consider any binary tree T. Throughout the following algorithm, G changes, but T always refers to the original tree.

```
initialize $G \leftarrow T$
while $G$ has more than one node:
    find a deepest internal node $v$ in $G$
    delete this node's two children $u$ and $w$ (and their edges) from $G$
```

This algorithm maintains the invariant that G is a binary tree and (writing # for "number of"):

$$(\#\text{ leaves of } G) - (\#\text{ internal nodes of } G) = (\#\text{ leaves of } T) - (\#\text{ internal nodes of } T)$$

This holds at the beginning when $G = T$. To see that the invariant is maintained, suppose it holds at the beginning of some iteration. Both u and w are leaves (otherwise they would be internal nodes deeper than v), so their deletion decreases the number of leaves by 2, but v becomes a new leaf so the net change in number of leaves is -1. Also, the change in number of internal nodes is -1 since v is no longer internal. Thus $(\#\text{ leaves of } G) - (\#\text{ internal nodes of } G)$ is preserved. Furthermore G remains a binary tree after the deletion since v goes from having two children to having zero children, and all other nodes keep the same number of children. So the invariant holds after this iteration. The loop terminates since G's number of nodes decreases in each iteration. When G is a single node at the end, the invariant says $1 - 0 = (\#\text{ leaves of } T) - (\#\text{ internal nodes of } T)$. ■

Theorem B.19 yields yet another proof of Theorem B.5: In a *perfect binary tree* where all leaves have depth n, there are 2^n leaves and $2^{n-1} + 2^{n-2} + \cdots + 2^1 + 2^0$ many internal nodes.

B.5.2 Recursive algorithms

A recursive algorithm's main control flow organization is a subroutine that makes calls to itself. *Calling* or *invoking* the subroutine means passing it arguments and letting it run until it returns a value. Proving that a recursive algorithm is correct typically involves identifying an *invariant*: a relationship between the arguments and return value for any invocation of the subroutine. An invariant has the form: "For all arguments with this property, the return value has that property." A *base case* of a recursion is an invocation that makes no recursive calls (it spawns no further invocations). Every recursion needs base cases, since otherwise it wouldn't terminate. We want the invariant to hold for the "top level" invocation, so we show that it actually holds for every invocation throughout the whole execution. A correctness proof shows:

- The invariant holds for the base cases.
- The invariant is *maintained*: If it holds for each recursive call made by an invocation, then it also holds for that invocation.
- The recursion is guaranteed to terminate.
- If the invariant holds for the original invocation, then the return value yields a correct output.

Since the invariant holds for the base cases, it also holds for the invocations that only make base case calls. Since it holds for those invocations, it also holds for any invocation in which each recursive call is one of those or a base case. And so on for all invocations. (This is a fancy version of induction.) Since the invariant holds for the original invocation, the algorithm outputs what we want.

Let's illustrate this framework. A *round-robin tournament* has n teams, and every pair of teams plays one game against each other, so $\binom{n}{2}$ many games are played in total. In each game, one team wins and the other loses—there are no ties. We represent such a tournament as a directed graph where nodes are teams and for every pair of nodes u and v, either (u,v) is an edge (meaning team u beat team v in their game) or (v,u) is an edge (meaning v beat u) but not both. There may be cycles: If u beat v, and v beat w, that doesn't necessarily mean u beat w.

We prove that every round-robin tournament has at least one *full path*, which visits every node exactly once. That is, no matter how many teams, and no matter who beat whom, there will exist a way to order all the teams so that each team (except the last) beat the team that comes immediately after it in the order.

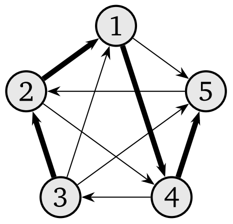
Theorem B.20. *Every round-robin tournament has a path that visits all nodes.*

Proof idea. Let's pick a node y and consider where to put y in the path. There are $n-1$ edges incident to y—one edge to or from each of the other nodes. If all these edges happen to leave y, then we can recursively find a path that visits all the other $n-1$ nodes, and prepend y to the path (since y has an edge to whichever node that path starts at). At the other extreme, if all of y's edges happen to enter y, then we can recursively find a path that visits all the other $n-1$ nodes, and append y to the path (since y has an edge from whichever node that path ends at).

In general, y might have some incoming edges and some outgoing edges. Let X be the set of nodes with edges to y, and Z be the set of nodes with edges from y (so $X \cup \{y\} \cup Z$ is all nodes). If we recursively find a path that visits all of X, and recursively find a path that visits all of Z, we can stitch the paths together using y: Whichever node the X path ends at, it has an edge to y. Whichever node the Z path starts at, it has an edge from y. We form a full path using the X path followed by y followed by the Z path.

The recursive subroutine's argument is a subset of nodes, and the return value is a path visiting exactly those nodes. The base cases correspond to individual nodes. ■

Example. In this round-robin tournament, 2 beat 1, and 1 beat 5, and 5 beat 2, and so on. The bold path $3 \to 2 \to 1 \to 4 \to 5$ is full. (Some other full paths also exist in this example.)



Suppose the algorithm picks $y = 1$, so $X = \{2, 3\}$ and $Z = \{4, 5\}$. One recursive call finds the path $3 \to 2$ visiting all of X, and the other recursive call finds the path $4 \to 5$ visiting all of Z, so the algorithm's final path stitches together $(3 \to 2) \to 1 \to (4 \to 5)$. ◆

Proof of Theorem B.20. Consider any round-robin tournament $G = (V, E)$. The following recursive subroutine maintains the invariant that for every nonempty $Y \subseteq V$, find-full-path(Y) returns a path in G that visits each node of Y and visits no node outside Y.

```
find-full-path($Y$):
    pick an arbitrary node $y \in Y$
    let $X = \{x \in Y : (x, y) \in E\}$ and if $X \neq \emptyset$: let $(x_1 \to \cdots \to x_k)$ = find-full-path($X$)
    let $Z = \{z \in Y : (y, z) \in E\}$ and if $Z \neq \emptyset$: let $(z_1 \to \cdots \to z_\ell)$ = find-full-path($Z$)
    return $(\underbrace{x_1 \to \cdots \to x_k \to}_{\text{omit if } X = \emptyset} y \underbrace{\to z_1 \to \cdots \to z_\ell}_{\text{omit if } Z = \emptyset})$
```

The base cases are when $Y = \{y\}$ has only one node, and the subroutine returns the trivial path (y) of length 0. To verify that find-full-path(Y) maintains the invariant, assume the invariant holds for find-full-path(X) and find-full-path(Z). That means $x_1 \to \cdots \to x_k$ is indeed a path in G that visits all of X (with $k = |X|$), and $z_1 \to \cdots \to z_\ell$ is indeed a path in G that visits all of Z (with $\ell = |Z|$). We assume $X \neq \emptyset$ and $Z \neq \emptyset$ here, but nothing is fundamentally different if one of these sets is empty. The edges $x_k \to y \to z_1$ are present in G since $x_k \in X$ and $z_1 \in Z$. Hence the return value is indeed a path in G that visits all of $X \cup \{y\} \cup Z = Y$. So the invariant holds for find-full-path(Y). The recursion terminates because each invocation picks a different distinguished node y, so there are only $|V|$ many invocations in total. In summary, we know a path visiting all of V exists because find-full-path(V) returns one. ■

Exercise B.27 asks for a proof of Theorem B.20 using an iterative algorithm, and for proofs of other interesting properties of round-robin tournaments.

As another example, let's reprove Theorem B.19 using a recursive algorithm.

Theorem B.19 (restated). *For every binary tree, the number of leaves is exactly one more than the number of internal nodes.*

Proof. Consider any binary tree. For every node v, let ℓ_v and k_v be the number of leaves and number of internal nodes (respectively) in the subtree rooted at v. The following recursive subroutine maintains the invariant that for every node v in the tree, count-nodes(v) returns (ℓ_v, k_v) and that $\ell_v - k_v = 1$.

```
count-nodes($v$):
    if $v$ is a leaf: return $(\ell_v, k_v) = (1, 0)$
    let $u$ and $w$ be $v$'s children
    let $(\ell_u, k_u) =$ count-nodes($u$)
    let $(\ell_w, k_w) =$ count-nodes($w$)
    return $(\ell_v, k_v) = (\ell_u + \ell_w,\ k_u + k_w + 1)$
```

The invariant trivially holds for the base cases (when v is a leaf). To verify that count-nodes(v) maintains the invariant, assume the invariant holds for count-nodes(u) and count-nodes(w). The leaves in v's subtree are the leaves in u's and w's subtrees, and there are $\ell_u + \ell_w = \ell_v$ such leaves. The internal nodes in v's subtree are the internal nodes in u's and w's subtrees along with v itself, and there are $k_u + k_w + 1 = k_v$ such nodes (the $+1$ is for v). Furthermore:

$$\ell_v - k_v \;=\; (\ell_u + \ell_w) - (k_u + k_w + 1) \;=\; (\ell_u - k_u) + (\ell_w - k_w) - 1 \;=\; 1 + 1 - 1 \;=\; 1$$

So the invariant holds for count-nodes(v). The recursion terminates because count-nodes(v) is only called once for each v. The theorem is true since the invariant holds for count-nodes(root). ■

B.6 Graphs without cycles

A tree is a connected undirected graph with no cycles. A dag (directed acyclic graph) is a directed graph with no cycles. In this section, we prove that trees and dags have interesting *characterizations*—properties that are equivalent to the definitions.

B.6.1 Characterizing trees

The next theorem provides two characterizations of trees. The theorem says "the following are equivalent ..." which is just a "multi-way iff." Saying "P, Q, R are equivalent" means either all three propositions are true or all three are false (that is, if one of them is true, then the other two are also true). Like viewing "P iff Q" as $P \Rightarrow Q \Rightarrow P$, we can prove "$P, Q, R$ are equivalent" by showing a "cycle of implications" $P \Rightarrow Q \Rightarrow R \Rightarrow P$. Or, we could do the cycle in the other direction if it's easier to prove $P \Leftarrow Q \Leftarrow R \Leftarrow P$.

Theorem B.21. *For every connected undirected graph (with multi-edges allowed but self-loops not allowed), the following are equivalent:*

(i) *For every two nodes u and v, there is only one path between u and v.*
(ii) *For every edge e, deleting e would disconnect the graph.*
(iii) *There is no cycle (that is, the graph is a tree).*

Proof. Consider any connected undirected graph.

(i) ⇒ (ii): We give a contrapositive proof. Suppose deleting some edge $e = \{u, v\}$ (with $u \neq v$ since e is not a self-loop) would not disconnect the graph. This means there exists a walk—and hence a path by Theorem B.18—between u and v that doesn't use e. Also, e itself is a path between u and v. Thus there is more than one path between u and v.

(ii) ⇒ (iii): We give a contrapositive proof. Suppose the graph has a cycle. Since there are no self-loops, the cycle has length at least 2. Let e be an edge on the cycle. We claim that deleting e would not disconnect the graph. For any two nodes u and v (not necessarily e's endpoints), we show that there exists a walk between u and v that doesn't use e. Since the original graph is connected, there exists a path P between u and v by Theorem B.18. If P doesn't use e then we're done, so assume P uses e. To get from u to v without using e: Follow P from u to an endpoint of e, then follow the cycle to e's other endpoint, then follow the rest of P to v. (Intuitively, we reroute P to avoid e by using the cycle as a detour.)

(iii) ⇒ (i): We give a contrapositive proof. Suppose there exist two nodes u and v with at least two different paths P_1 and P_2 between them.

First, we claim that some edge is on one of these two paths but not on the other. Suppose for contradiction P_1 and P_2 use exactly the same set of edges as each other but in different orders. Following P_1 and P_2 from u to v, consider the first step where they use different edges, say P_1 uses edge $\{w, x\}$ and P_2 uses edge $\{w, y\}$. Since P_1 and P_2 use the same set of edges, P_1 must eventually use the edge $\{w, y\}$, contradicting the fact that P_1 is a path and therefore doesn't revisit w. This proves the claim.

Say e is an edge on P_1 but not on P_2. There is a walk between e's endpoints that doesn't use e itself: Follow P_1 from one endpoint of e to u, then P_2 from u to v, then P_1 from v to e's other endpoint. By Theorem B.18 (using the graph with e deleted), there exists a path between e's endpoints that doesn't use e. Adding e to this path creates a cycle. ■

Exercise B.30 shows another characterization we could add to Theorem B.21: A connected undirected graph is a tree iff the number of nodes is exactly one more than the number of edges.

B.6.2 Characterizing dags

Our characterization of dags uses the following lemma. In a directed graph, a source is a node with indegree 0 (no incoming edges), and a sink is a node with outdegree 0 (no outgoing edges).

Lemma B.22. *Every dag has at least one source and at least one sink.*

Proof idea. If we walk from some node by blindly following outgoing edges, we must arrive at a sink because otherwise we would eventually revisit a node, thus revealing a cycle. Walking backward, we would arrive at a source. ■

Proof. To show that every dag has at least one sink, let's rephrase this as an implication: For every directed graph G, if G has no cycles then G has a sink. We give a contrapositive proof. Assume G has no sink, in other words, every node has at least one outgoing edge. This algorithm finds a cycle in G:

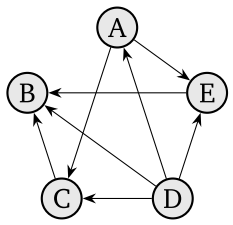
```
let $u_1$ be an arbitrary node
for $i = 1, 2, 3, \dots$:
    pick an arbitrary outgoing edge from $u_i$, and let $u_{i+1}$ be the other endpoint
    if $u_{i+1} = u_j$ for some $j \le i$: halt and output the cycle $(u_j \to u_{j+1} \to \cdots \to u_i \to u_j)$
```

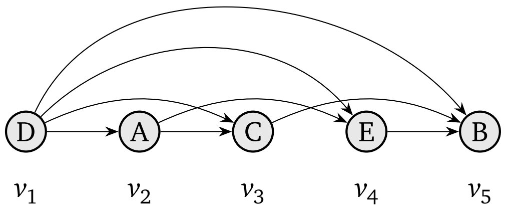
This algorithm maintains the invariant that $u_1 \to u_2 \to \cdots \to u_i$ is a path in G, since it terminates as soon as it revisits a node. The invariant holds before the first iteration (since (u_1) is a path of length 0), so it holds after every iteration. The loop terminates since there are only finitely many nodes, so revisiting is inevitable. When the loop terminates during the last iteration, the output is a cycle since $u_j \to u_{j+1} \to \cdots \to u_i$ is a path (by the invariant) and $u_i \to u_j$ is an edge (by the termination condition).

To see that every dag G has at least one source, define the directed graph G' by reversing the direction of each edge of G. Then G' is also a dag by contrapositive reasoning: Any cycle in G' would still be a cycle (in the reverse direction) in G. We know G' has a sink, which is a source in G. ■

A *topological order* of a directed graph is a way to order the nodes $v_1, v_2, \dots, v_n$ (picking a node to call v_1, picking another node to call v_2, and so on) such that every edge goes from a lower-index node to a higher-index node. Formally, if (v_i, v_j) is an edge then $i < j$. Informally, we arrange the nodes in a line so all edges point to the right. (In general, the word "topological" refers to spatial configurations.)

Example. On the left is a directed graph. On the right is one topological order $(v_1, v_2, v_3, v_4, v_5) = (\mathrm{D}, \mathrm{A}, \mathrm{C}, \mathrm{E}, \mathrm{B})$. (By the way, $(\mathrm{D}, \mathrm{A}, \mathrm{E}, \mathrm{C}, \mathrm{B})$ is a different topological order for this graph.)





♦

For some directed graphs, no topological order exists. We prove that dags are exactly the graphs that have a topological order. At a glance, it wasn't obvious that our example graph has no cycles, but we know it's a dag since it has a topological order.

Motivation: Suppose each node of a dag is a task to be done, and an edge (u, w) means task u must be done before task w. A topological order provides a schedule for doing all the tasks while obeying all constraints.

Theorem B.23. *For every directed graph (with multi-edges and self-loops allowed), the following are equivalent:*

(i) There exists a topological order.
(ii) There is no cycle (that is, the graph is a dag).

Proof idea. *(i)* says:

There exists an order $(v_1, \dots, v_n)$ such that for every edge (v_i, v_j), we have $i < j$.

Negating this, and pulling the negation inside, yields:

For every order $(v_1, \dots, v_n)$, there exists an edge (v_i, v_j) such that $i \geq j$.

The contrapositive of *(i)* ⇒ *(ii)* is intuitive: If the graph has a cycle, then no matter how we order the nodes, the cycle can't go to the right forever—it must eventually loop back to the left.

The key for *(ii)* ⇒ *(i)* is Lemma B.22. A dag has a source, and we might as well put a source first (leftmost) in a topological order: This node's outgoing edges will all point to the right, and we needn't worry about its incoming edges because it has none. Since we dealt with this node, we can delete it and repeat the process on the remaining dag. This is an iterative algorithm to find a topological order. ■

Proof. Consider any directed graph G with n nodes.

(i) ⇒ *(ii)*: We give a contrapositive proof. Suppose G has a cycle. To show that G has no topological order, we consider an arbitrary order $v_1, \dots, v_n$ and show that some edge (v_i, v_j) has $j \leq i$. Let i be the largest index of any node on the cycle, and let j be the index of the next node on the cycle. Thus v_i has an edge to v_j, but $j \leq i$ since v_j is on the cycle.

(ii) ⇒ *(i)*: We give a direct proof. Suppose G is a dag. The following algorithm finds a topological order. Throughout the algorithm, H changes, but G always refers to the original dag.

```
initialize $H$ to be $G$
for $i = 1, 2, \dots, n$:
    let $v_i$ be a source of $H$
    delete $v_i$ (and its outgoing edges) from $H$
```

The algorithm maintains the invariant that H is a dag, since deleting nodes and edges from a dag can't create a cycle. Lemma B.22 says H indeed has a source in every iteration (so the algorithm doesn't get stuck). To see that $v_1, v_2, \dots, v_n$ is a topological order of G, consider any edge (v_i, v_j) of G. We need to show that $i < j$, so suppose for contradiction that $j \leq i$. We can't have $j = i$, since a self-loop is a cycle and a dag has none of those. Thus $j < i$, which means when v_j is deleted, v_i still remains in H. Since (v_i, v_j) is an edge, v_j has indegree at least 1 and is thus not a source in H at the moment v_j is deleted, which contradicts how the algorithm works. ■

Example. Let's run the algorithm to find a topological order for our 5-node example dag from earlier. D is the only source, so it comes first. After deleting D (and its outgoing edges), the indegrees of other nodes decrease, but B, C, and E still have incoming edges (from non-D nodes). Now A is the only node with indegree 0, so it comes next. After deleting A, the indegrees of B, C, and E become 2, 0, and 0 respectively. Thus either C or E can be next. After that, B's indegree will be 1, so the other of C or E has to come next, followed finally by B. ♦

B.7 The probabilistic method

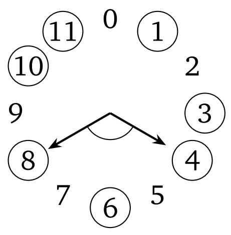
The idea of the *probabilistic method* for proving $\exists x\ P(x)$ is to design a distribution over x's domain and argue that $P(x)$ is true with positive probability for a randomly sampled value of x. This is nonconstructive because instead of explicitly describing a particular x, it shows that a suitable x exists without indicating how to find it efficiently. Even when the proposition has nothing to do with randomness, the probabilistic method can be quite effective because it lets us harness intuitions that would be more cumbersome to express without the language of probability theory.

Probabilities and expectations are often simpler to calculate when independence is involved, but many applications of the probabilistic method rely on union bounds (§B.7.1) or on linearity of expectation (§B.7.2), which don't require independence.

B.7.1 Using union bounds

The union bound says if $A_1, A_2, \ldots, A_n$ are events in a probability space, then their union's probability is at most the sum of their probabilities: $\mathbf{Pr}[A_1 \cup \cdots \cup A_n] \leq \mathbf{Pr}[A_1] + \cdots + \mathbf{Pr}[A_n]$. If the latter is < 1 then the former is < 1, so some outcome is in none of the events.

To demonstrate this technique, imagine a clock with $0, 1, \ldots, m-1$ equally spaced around it. Some numbers are circled. The "hour" and "minute" hands are rigidly affixed to each other with a particular angle, so when one of them rotates, the other rotates in sync. We'd like both hands to point at circled numbers simultaneously. Call the angle between consecutive numbers a "unit." In the example to the right with $m = 12$ and the hands at a 4-unit angle, the hands can point at $(4, 8)$ or $(6, 10)$ or $(11, 3)$. If the angle were 6 units, the hands could point at $(4, 10)$ or $(10, 4)$.



We prove that regardless of the angle between the hands, if more than half the numbers are circled, then it's possible to turn the hands so both hands point at circled numbers.

Proof idea. We could try to design an algorithm that finds a desired positioning of the hands, or prove the contrapositive by a counting argument, but these approaches are a little tricky to make rigorous. Instead, we do something lazy. Imagine flicking the hands, so they spin around and around in sync, eventually stopping at a uniformly random position. The hour hand (ignoring the minute hand) has probability $> 1/2$ of landing on a circled number, since more than half the numbers are circled. The minute hand (ignoring the hour hand) also has probability $> 1/2$ of landing on a circled number. If these events were independent, we could conclude that with probability $> (1/2) \cdot (1/2) = 1/4$, both hands land on circled numbers. The events aren't independent, but we can use a union bound on the complement events: The probability that at least one hand lands on a non-circled number is $< 1/2 + 1/2 = 1$. ■

We state this formally using modular arithmetic. The circled numbers are a subset of $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$. If the hour hand points at x and the angle is a units, then the minute hand points at $(x + a) \bmod m$.

Theorem B.24. *For every $S \subseteq \mathbb{Z}_m$ with $|S| > m/2$, and for every $a \in \mathbb{Z}_m$, there exists $x \in \mathbb{Z}_m$ such that $x \in S$ and $((x + a) \bmod m) \in S$.*

Proof. Consider any such S and a. Pick a uniformly random $x \in \mathbb{Z}_m$, and view $(x, x + a)$ as an outcome of a joint distribution over $\mathbb{Z}_m \times \mathbb{Z}_m$, where $x + a$ is shorthand for $(x + a) \bmod m$. We have $\mathbf{Pr}[x \notin S] = 1 - |S|/m < 1/2$, and similarly $\mathbf{Pr}[x + a \notin S] < 1/2$ since the marginal distribution of $x + a$ is uniform. By a union bound, $\mathbf{Pr}[x \notin S \text{ or } x + a \notin S] < 1/2 + 1/2 = 1$, so $\mathbf{Pr}[x \in S \text{ and } x + a \in S] > 0$. Hence there exists x such that $x \in S$ and $x + a \in S$. ■

The next theorem is a classic demonstration of the probabilistic method. For any $m \times n$ matrix M, deleting some rows and/or some columns yields a *submatrix*. For nonempty sets $I \subseteq [m]$ and $J \subseteq [n]$, let $M_{I,J}$ denote the submatrix that retains the entries with row indices in I and column indices in J.

Example.

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad M_{\{2,4\},\{1,3\}} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

♦

A binary matrix is *monochromatic* when either every entry is 0 or every entry is 1. Thinking of 0 as "white" and 1 as "black," monochromatic means the whole matrix is a single color. Also, recall that logs are base 2 when the base is unspecified.

Theorem B.25. *For every $n \geq 2$, there exists an $n \times n$ binary matrix with no monochromatic $k \times k$ submatrix where $k = \lceil 2 \log n \rceil$.*

Proof idea. Call a matrix "bad" if it has a monochromatic $k \times k$ submatrix, and "good" otherwise. We show that with positive probability (in fact, high probability), a uniformly random matrix is good. For each $k \times k$ set of positions, we show that the "bad event" that this submatrix is monochromatic has probability $< 1/(\text{the number of bad events})$. The union of all bad events therefore has probability < 1, so a random matrix is bad with probability < 1.

We could phrase the proof as "just a counting argument" (there are more matrices than bad matrices), but the probabilistic framing provides a clean intuition. ■

Proof. Pick a uniformly random $n \times n$ binary matrix M. For each $I \subseteq [n]$ and $J \subseteq [n]$ with $|I| = |J| = k$, let $A_{I,J}$ be the event that $M_{I,J}$ is monochromatic. $\mathbf{Pr}[A_{I,J}] = 2 \cdot 2^{-k^2} = 2^{-k^2+1}$ because we can partition $A_{I,J}$ into two events corresponding to $M_{I,J}$ being only 0s or only 1s, and both of these events have probability $(1/2)^{k^2}$ (since the k^2 many bits of $M_{I,J}$ are independent, and each has probability $1/2$ of being the target bit). There are $\binom{n}{k}$ possibilities for I and $\binom{n}{k}$ possibilities for J, so there are $\binom{n}{k}^2$ many $A_{I,J}$ events. By a union bound:

$$\mathbf{Pr}\Big[\bigcup_{I,J} A_{I,J}\Big] \le \sum_{I,J} \mathbf{Pr}[A_{I,J}] = \binom{n}{k}^2 \cdot 2^{-k^2+1}$$

We show that the latter bound is < 1, which implies that some outcome M is in none of the $A_{I,J}$ events and hence has no monochromatic $k \times k$ submatrix. To do this, we show that an even bigger (but tidier) bound is < 1. Let's work with

$$\binom{n}{k} = \frac{n\cdot(n-1)\cdots(n-k+1)}{k\cdot(k-1)\cdots 1}$$

first. This increases if we increase the numerator and decrease the denominator. We increase the numerator to n^k by changing each factor to n. We decrease the denominator to 2 by changing the k factor to 2 (using $k \ge 2$ since $n \ge 2$) and all other factors to 1. In summary, $\binom{n}{k} \le n^k/2$. The definition of k implies $k \ge 2\log n$, which rearranges to $n \le 2^{k/2}$. Combining these bounds, we have $\binom{n}{k} \le n^k/2 \le (2^{k/2})^k/2 = 2^{k^2/2-1}$. Putting everything together:

$$\mathbf{Pr}\Big[\bigcup_{I,J} A_{I,J}\Big] \le \binom{n}{k}^2 \cdot 2^{-k^2+1} \le 2^{k^2-2} \cdot 2^{-k^2+1} = 2^{k^2-k^2-1} = 1/2 < 1$$

■

This proof shows that not only some, but most $n \times n$ binary matrices M have no monochromatic $k \times k$ submatrix. Despite their abundance, such matrices are quite difficult to find! The probabilistic proof doesn't describe any particular such matrix. Nobody knows how to explicitly construct such a matrix (for arbitrary n). After herculean effort, researchers have found explicit constructions achieving $k = \log^{O(1)} n$. Exercise B.35.b shows that achieving $k = o(\log n)$ is impossible.

What does it mean for a matrix or other mathematical object to be "explicit"? It means an efficient algorithm outputs the object. This generic definition is a contribution of computer science to mathematics.

B.7.2 Using expectation

It's intuitive that the average of some numbers can't be greater than all those numbers. The following lemma is more general.

Lemma B.26. *For every random variable X on a discrete probability space, there exists an outcome s such that $X(s) \ge \mathbf{E}[X]$ and $\mathbf{Pr}[s] > 0$.*

Proof. Suppose for contradiction that $X(s) < \mathbf{E}[X]$ for every outcome s with $\mathbf{Pr}[s] > 0$. Thus, $\mathbf{Pr}[s] \cdot X(s) < \mathbf{Pr}[s] \cdot \mathbf{E}[X]$ for every s with $\mathbf{Pr}[s] > 0$ (since for all real numbers $a > 0$ and $b < c$, we have $ab < ac$). Outcomes s with $\mathbf{Pr}[s] = 0$ contribute $\mathbf{Pr}[s] \cdot X(s) = 0$ to the expectation, so we can sum over s with $\mathbf{Pr}[s] > 0$:

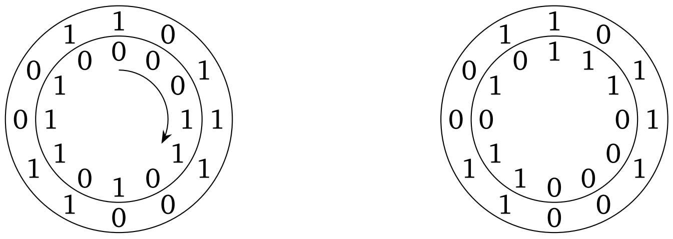
$$\mathbf{E}[X] = \sum_s \mathbf{Pr}[s] \cdot X(s) < \sum_s \mathbf{Pr}[s] \cdot \mathbf{E}[X] = \mathbf{E}[X] \cdot \sum_s \mathbf{Pr}[s] = \mathbf{E}[X] \cdot 1$$

This gives the contradiction $\mathbf{E}[X] < \mathbf{E}[X]$. ■

Lemma B.26 provides a technique to prove that an outcome with a certain property exists: Design a random variable X such that "$X(s) \geq \mathbf{E}[X]$" implies s has the property.

To demonstrate this technique, imagine a device like a rotary combination lock, but with n bits equally spaced around the stationary outside, and n bits equally spaced around the rotating inner dial. We're interested in turning the dial to maximize the number of positions that match (that is, the inner bit equals the adjacent outer bit).

Example. On the left, only five positions match $(1, 3, 4, 5, 8$ o'clock$)$. If we rotate the dial clockwise one or two or three units, in each case still only five positions match. If we instead rotate clockwise by four units (shown on the right) then seven positions match. No rotation does better in this example.



♦

We prove that if n is even and the dial is half 0s and half 1s, then it's possible to rotate the dial to make at least half of the positions match.

Proof idea. Spin the dial around and around so when it stops, the final rotation angle is a uniformly random number of units. The number of matching positions is a random variable X. Each position is equally likely to get a 0 or a 1 on the dial, and hence has probability 1/2 of matching (regardless of the outer bit in that position). We show that this implies $\mathbf{E}[X] \geq n/2$. The best rotation is at least as good as a random rotation. ■

Let's be more formal. Recall that a bit string's weight is its number of 1s, and the distance between two bit strings (of the same length) is the number of bit positions on which they differ. We index the positions of a bit string $y \in \{0,1\}^n$ using $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ (instead of the more conventional $[n] = \{1, 2, \ldots, n\}$) with y_0 being the leftmost bit. *Rotating* y to the right by k positions produces $z \in \{0,1\}^n$ such that $z_i = y_{(i-k) \bmod n}$ (the rightmost k bits of y become the leftmost k bits of z, and the rest of the bits shift over).

Example. With our earlier example, the outer bit string was $x = 101110011001$ and the inner bit string began as $y = 000110101110$, with $\operatorname{dist}(x, y) = 7$. Rotating y right by 4 positions yielded $z = 111000011010$ with $\operatorname{dist}(x, z) = 5$. ♦

Theorem B.27. *For all bit strings $x, y \in \{0,1\}^n$ such that* $\operatorname{weight}(y) = n/2$*, there exists a rotation z of y such that* $\operatorname{dist}(x, z) \leq n/2$*.*

Proof. Consider any such x and y. Pick a uniformly random $k \in \mathbb{Z}_n$, and define z by rotating y right by k positions. View (k, z) as an outcome of a joint distribution over $\mathbb{Z}_n \times \{0,1\}^n$. For each $i \in \mathbb{Z}_n$, the marginal distribution of $z_i = y_{(i-k) \bmod n}$ is uniform over $\{0,1\}$ because $(i-k) \bmod n$ is uniformly distributed over $\mathbb{Z}_n$ and $\operatorname{weight}(y) = n/2$ (so $y_{(i-k) \bmod n} = 0$ for half of all k, and $y_{(i-k) \bmod n} = 1$ for half of all k). Thus $\mathbf{Pr}[x_i = z_i] = 1/2$ regardless of whether x_i is 0 or 1. Defining the random variable X as the number of indices i for which $x_i = z_i$, we have $X = X_0 + \cdots + X_{n-1}$ where X_i is the indicator random variable for the event $x_i = z_i$. The X_i random variables are not independent, but linearity of expectation still applies:

$$\mathbf{E}[X] \;=\; \textstyle\sum_{i=0}^{n-1} \mathbf{E}[X_i] \;=\; \sum_{i=0}^{n-1} \mathbf{Pr}[x_i = z_i] \;=\; \sum_{i=0}^{n-1} 1/2 \;=\; n/2$$

By Lemma B.26, some outcome (k, z) has $X(k, z) \geq n/2$ and hence $\mathrm{dist}(x, z) = n - X(k, z) \leq n/2$ (and the outcome has positive probability, so z is a rotation of y). ■

B.8 Algebraic proofs

In algebraic proofs, properties of the four arithmetic operators $+, -, \cdot, /$ (the field axioms from §A.8.2) play central roles.

B.8.1 Multiplicative inverses modulo primes

In §A.8.2 we mentioned that if p is a prime number, then $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ forms a field using the "mod p" arithmetic operators. The field axioms were all straightforward except for one: the existence of a multiplicative inverse $a^{-1} = 1/a$ for every nonzero field element $a \in \mathbb{Z}_p$. Such an inverse is an $x \in \mathbb{Z}_p$ such that $ax \bmod p = 1$.

Theorem B.28. *For every prime p, every nonzero $a \in \mathbb{Z}_p$ has a unique* "mod p" *multiplicative inverse.*

We use the "proof by algorithm" technique to find a "mod p" multiplicative inverse of a, and we separately prove that a has only one such inverse. To facilitate the algorithm, we prove something more general. Two positive integers (a, b) are *coprime* (or *relatively prime*) to each other when they have no common divisor besides 1—that is, there's no integer $d > 1$ such that $a = q_1 d$ and $b = q_2 d$ for some integers q_1 and q_2. A "mod b" multiplicative inverse of a is an $x \in \mathbb{Z}_b$ such that $ax \bmod b = 1$. Since a prime p is coprime to every nonzero $a \in \mathbb{Z}_p$, Theorem B.28 is a corollary of the following:

Theorem B.29. *If $0 < a < b$ and (a, b) are coprime, then a has a unique* "mod b" *multiplicative inverse.*

Proof idea. We design a recursive subroutine to find such an inverse. (We could do it iteratively, but recursion is slightly more convenient.) Two lemmas tell us what arguments to pass to the recursive call, and what to do with the call's return value. We say a is "mod b" invertible when a has a "mod b" multiplicative inverse.

$$\begin{tikzcd}
\text{(a,b) are coprime} \arrow[d, Rightarrow, "\text{Lemma B.30}"'] & \text{a is ``mod b'' invertible} \\
\text{$(b \bmod a, a)$ are coprime} \arrow[r, Rightarrow, "\text{recursion}"'] & \text{$(b \bmod a)$ is ``mod a'' invertible} \arrow[u, Rightarrow, "\text{Lemma B.31}"']
\end{tikzcd}$$

■

Lemma B.30. *If $1 < a < b$ and (a, b) are coprime, then $0 < (b \bmod a) < a$ and $(b \bmod a, a)$ are coprime.*

Proof of Lemma B.30. Assuming $1 < a < b$, we show that if either $(b \bmod a) = 0$ or $(b \bmod a, a)$ aren't coprime, then (a, b) aren't coprime. If $(b \bmod a) = 0$ then $a > 1$ itself is a common divisor of a and b. Now, suppose $(b \bmod a, a)$ aren't coprime, so $(b \bmod a) = q_1 d$ and $a = q_2 d$ for some integers $d > 1$ and q_1 and q_2. Since $b = q_3 a + (b \bmod a)$ for some integer q_3, we have $b = q_3 q_2 d + q_1 d = (q_3 q_2 + q_1)d$. Thus d is also a divisor of both a and b, so (a, b) aren't coprime. ■

Lemma B.31. *If $1 < a < b$ and $(b \bmod a)$ is "mod a" invertible, then a is "mod b" invertible.*

Proof of Lemma B.31. Assume $1 < a < b$ and $(b \bmod a)$ has a "mod a" multiplicative inverse s, which means $(b \bmod a)s \bmod a = 1$, in other words $(b \bmod a)s = q_1 a + 1$ for some integer q_1. Combining this with $b = q_2 a + (b \bmod a)$ for some integer q_2, we get $(b - q_2 a)s = q_1 a + 1$. Rearranging this yields $a(-q_1 - q_2 s) = (-s)b + 1$, which says $at = q_3 b + 1$ where $t = -q_1 - q_2 s$ and $q_3 = -s$. This means $at \bmod b = 1$, so a has a "mod b" multiplicative inverse $(t \bmod b)$ since $a(t \bmod b) \bmod b = at \bmod b = 1$. ■

Proof of Theorem B.29. The following recursive subroutine maintains the invariant that if $0 < a < b$ and (a, b) are coprime, then find-inverse(a, b) returns a "mod b" multiplicative inverse of a.

```
find-inverse($a, b$):
    if $a = 1$: return 1
    $s \leftarrow$ find-inverse($b \bmod a, a$)
    return $(-q_1 - q_2 s) \bmod b$ where:
        $q_1$ is the integer quotient of $(b \bmod a)s$ divided by $a$
        $q_2$ is the integer quotient of $b$ divided by $a$
```

The base cases are when $a = 1$ and the subroutine returns 1, which is correct since $(1 \cdot 1) \bmod b = 1$. To prove that find-inverse(a, b) maintains the invariant when $a > 1$, we assume the invariant holds for find-inverse$(b \bmod a, a)$ and show that it also holds for find-inverse(a, b). Suppose $1 < a < b$ and (a, b) are coprime. Then $0 < (b \bmod a) < a$ and $(b \bmod a, a)$ are coprime by Lemma B.30. Thus s is a "mod a" multiplicative inverse of $(b \bmod a)$ by the invariant. The return value $(-q_1 - q_2 s) \bmod b$ is a "mod b" multiplicative inverse of a by the proof of Lemma B.31. The recursion terminates since the first argument decreases (but stays positive) in each successive call, so it eventually reaches a base case.

Now that we know a has at least one "mod b" multiplicative inverse (since find-inverse(a, b) returns one), we show that a has only one such inverse. Assuming t and u are "mod b" multiplicative inverses of a, we show that $t = u$. Doing all calculations "mod b": We have $a(t - u) = at - au = 1 - 1 = 0$. Multiplying both sides by t, the left side becomes $t(a(t-u)) = (ta)(t-u) = 1 \cdot (t - u) = t - u$, while the right side becomes $t \cdot 0 = 0$. In summary, $(t - u) \bmod b = 0$, which implies $t = u$ since $t, u \in \mathbb{Z}_b$. ■

We've just seen the tip of an iceberg. More general phenomena are explored in number theory and abstract algebra.

B.8.2 Roots of polynomials

We mentioned the following classic theorem in §A.9.1, and now we prove it. In this section, all polynomials are univariate. A root of a polynomial $P(x)$ is a field element a such that $P(a) = 0$.

Theorem B.32. *Over every field, every nonzero polynomial of degree d has at most d many roots.*

Corollary B.33. *Over every field, if P and Q are distinct polynomials of degree $\leq d$, then $P(a) = Q(a)$ holds for at most d many field elements a.*

Proof of Corollary B.33. The polynomial $P - Q$ is nonzero since $P \neq Q$, and $\deg(P - Q) \leq \max(\deg(P), \deg(Q)) \leq d$. By Theorem B.32, $P-Q$ has at most d many roots. The roots of $P-Q$ are the field elements a such that $P(a)-Q(a) = 0$, in other words, $P(a) = Q(a)$. ■

To prepare for the proof of Theorem B.32, recall integer division: For all integers p and $m > 0$, there exist unique integers q (the quotient) and r (the remainder) such that $p = q \cdot m + r$ and $0 \leq r < m$. Something similar holds for polynomial division:

Lemma B.34. *Over every field, for all polynomials P and $M \neq 0$, there exist unique polynomials Q (the quotient) and R (the remainder) such that $P = Q \cdot M + R$ and $\deg(R) < \deg(M)$.*

Proof of Lemma B.34. First, we prove the existence of Q and R. The following recursive subroutine maintains the invariant that for all polynomials P and $M \neq 0$, divide(P, M) returns a pair of polynomials (Q, R) such that $P = Q \cdot M + R$ and $\deg(R) < \deg(M)$.

```
divide($P, M$):
    if $\deg(P) < \deg(M)$: return $(0, P)$
    let $bx^d$ be $P$'s leading monomial (so $b \neq 0$ and $d = \deg(P) \geq 0$)
    let $cx^e$ be $M$'s leading monomial (so $c \neq 0$ and $e = \deg(M) \leq d$)
    $(Q', R) \leftarrow$ divide$(P', M)$ where $P' = P - (b/c)x^{d-e} \cdot M$
    return $(Q, R)$ where $Q = (b/c)x^{d-e} + Q'$
```

The invariant trivially holds for the base cases when $\deg(P) < \deg(M)$. To see that divide(P, M) maintains the invariant when $\deg(P) \geq \deg(M)$, assume the invariant holds for divide(P', M), that is, $P' = Q' \cdot M + R$ and $\deg(R) < \deg(M)$. Then by definition:

$$P \;=\; (b/c)x^{d-e} \cdot M + P' \;=\; (b/c)x^{d-e} \cdot M + \big(Q' \cdot M + R\big) \;=\; \big((b/c)x^{d-e} + Q'\big) \cdot M + R \;=\; Q \cdot M + R$$

The recursion terminates because $\deg(P)$ decreases in each successive call: $\deg(P') < \deg(P)$ because the leading monomial of $(b/c)x^{d-e} \cdot M$ is $(b/c)x^{d-e} \cdot cx^e = bx^d$, which cancels the leading monomial of P in the definition of P'. The invariant directly implies the existence of Q and R.

Now, we prove that Q and R are unique. Suppose for contradiction there exist some other polynomials S and T (either $S \neq Q$ or $T \neq R$) such that $P = S \cdot M + T$ and $\deg(T) < \deg(M)$. Then $0 = P - P = (Q \cdot M + R) - (S \cdot M + T) = (Q - S) \cdot M + (R - T)$. But the latter polynomial is nonzero because: If $S \neq Q$ then $\deg(R - T) < \deg(M) \leq \deg((Q - S) \cdot M)$. If $S = Q$ and $T \neq R$ then $0 \cdot M + (R - T)$ is nonzero. ■

Lemma B.35. *Over every field, if P is a polynomial and $P(a) = 0$, then $P(x) = Q(x) \cdot (x - a)$ for some polynomial Q.*

Proof of Lemma B.35. Suppose $P(a) = 0$. Divide P by $M(x) = x - a$ using Lemma B.34 to get $P = Q \cdot M + R$. Since $\deg(R) < \deg(M) = 1$, R is a constant. We have $R = 0$ and therefore $P = Q \cdot M$ because $0 = P(a) = Q(a) \cdot M(a) + R = Q(a) \cdot (a - a) + R = 0 + R = R$. ■

Proof of Theorem B.32. Rewriting the statement as an implication: For every nonzero polynomial P, if $\deg(P) = d$ then P has at most d many roots. We give a contrapositive proof. Assume P has at least $d+1$ many roots: $P(a_1) = P(a_2) = \cdots = P(a_{d+1}) = 0$ for some distinct field elements $a_1, a_2, \ldots, a_{d+1}$. We claim that $P = Q \cdot (x - a_1) \cdot (x - a_2) \cdots (x - a_{d+1})$ for some polynomial Q (which must be nonzero since P is nonzero), in which case

$$\deg(P) \;=\; \deg(Q) + \deg(x - a_1) + \cdots + \deg(x - a_{d+1}) \;\geq\; 0 + 1 + \cdots + 1 \;=\; d + 1$$

since the degree of a product equals the sum of the degrees. Here's an iterative algorithm to find Q, using Lemma B.35 to peel off the $(x - a_i)$ factors one at a time:

```
let $Q_{d+1} = P$
for $i \leftarrow d+1, d, d-1, \ldots, 1$:
    let $Q_{i-1}$ be a polynomial such that $Q_i = Q_{i-1} \cdot (x - a_i)$
output $Q = Q_0$
```

This maintains the invariant that $P = Q_i \cdot (x - a_{i+1}) \cdots (x - a_{d+1})$ and $a_1, \ldots, a_i$ are roots of Q_i. The invariant holds at the beginning ($i = d+1$) by assumption. To see that the invariant is maintained, suppose it holds for some $i \geq 1$. In particular, since $Q_i(a_i) = 0$, Lemma B.35 says that indeed $Q_i = Q_{i-1} \cdot (x - a_i)$ for some polynomial Q_{i-1} (so the algorithm doesn't get stuck). Thus:

$$P \;=\; Q_i \cdot (x - a_{i+1}) \cdots (x - a_{d+1}) \;=\; Q_{i-1} \cdot (x - a_i) \cdot (x - a_{i+1}) \cdots (x - a_{d+1})$$

Furthermore, $a_1, \ldots, a_{i-1}$ are roots of Q_{i-1} because for every $j \leq i-1$, we have $Q_{i-1}(a_j) = Q_i(a_j)/(a_j - a_i) = 0$ since a_j is a root of Q_i and $a_j \neq a_i$. So the invariant also holds for $i - 1$. The loop certainly terminates. In summary, since the invariant holds for $i = 0$, we have $P = Q \cdot (x - a_1) \cdot (x - a_2) \cdots (x - a_{d+1})$ as claimed. ■

B.8.3 An inequality for dot products

We present a classic inequality that makes occasional appearances in computational complexity proofs. In this section we deal exclusively with the field $\mathbb{R}$ (though some of what we say generalizes to other fields). Recall from §A.10 that a scalar is a single number, and a vector is a tuple of numbers, and the dot product of vectors $v \in \mathbb{R}^n$ and $w \in \mathbb{R}^n$ is the scalar $v \cdot w = \sum_{i=1}^{n} v_i w_i$.

Theorem B.36. $(v \cdot w)^2 \leq (v \cdot v)(w \cdot w)$ *for all vectors $v \in \mathbb{R}^n$ and $w \in \mathbb{R}^n$.*

First, we have two lemmas that encapsulate simple properties of dot products.

If c is a scalar and v is a vector, then cv denotes the vector where all components of v are multiplied ("scaled") by c: $(cv)_i = cv_i$. If v and w are vectors with the same number of components, then the vector $v+w$ is the entry-wise sum: $(v+w)_i = v_i + w_i$. Here's the linearity property of dot products (analogous to linearity of expectation from §A.7.1):

Lemma B.37. *For all vectors $u, v, w \in \mathbb{R}^n$ and every scalar $c \in \mathbb{R}$:*

(i) $(cv)\cdot w = c(v\cdot w)$
(ii) $u\cdot(v+w) = (u\cdot v)+(u\cdot w)$ *and* $u\cdot(v-w) = (u\cdot v)-(u\cdot w)$

Proof. For *(i)*,
$$(cv)\cdot w \;=\; \textstyle\sum_i (cv_i)w_i \;=\; c\sum_i v_i w_i \;=\; c(v\cdot w)$$
using the distributive axiom to factor out the c. For *(ii)*:
$$u\cdot(v+w) \;=\; \textstyle\sum_i u_i(v_i+w_i) \;=\; \sum_i(u_iv_i+u_iw_i) \;=\; \big(\sum_i u_iv_i\big)+\big(\sum_i u_iw_i\big) \;=\; (u\cdot v)+(u\cdot w)$$
Subtraction is similar. ■

Since dot product is commutative ($v\cdot w = w\cdot v$), we can generalize Lemma B.37: A scalar multiplier can be "pulled out" of either side of a dot product: $v\cdot(cw) = c(v\cdot w)$. Dot products "distribute" over addition in either direction: $(v+w)\cdot u = (v\cdot u)+(w\cdot u)$ and $(v-w)\cdot u = (v\cdot u)-(w\cdot u)$.

Lemma B.38. $v\cdot v \geq 0$ *for every vector* $v\in\mathbb{R}^n$*. Furthermore,* $v\cdot v = 0$ *iff* v *is the zero vector.*

Proof. This holds for scalars ($n=1$) because a real number's square is always nonnegative, and 0 is the only real number whose square is 0. For $n>1$, we have $v\cdot v = \sum_i v_i^2 \geq 0$ since $v_i^2 \geq 0$ for each index i. To verify the "furthermore" part:

$\Leftarrow$: We give a direct proof. If $v_i = 0$ for each index i, then $v\cdot v = \sum_i 0^2 = 0$.

$\Rightarrow$: We give a contrapositive proof. If $v_j \neq 0$ for some index j, then $v\cdot v = \sum_i v_i^2 \geq v_j^2 > 0$ since $v_i^2 \geq 0$ for each $i\neq j$. ■

Proof idea for Theorem B.36. Admittedly, this proof seems to come "out of the blue." Proper intuition comes from the subject of linear algebra. We provide a self-contained—but perhaps mysterious—proof.

The approach is to define a certain vector in terms of v and w, note that the dot product of this vector with itself is nonnegative, and then rearrange that inequality to get $(v\cdot w)^2 \leq (v\cdot v)(w\cdot w)$. The special vector has the form $x-y$ where x and y are scaled versions of v and w—that is, $x = cv$ and $y = dw$ for carefully chosen scalars c and d. The rearrangement uses linearity several times. For example:
$$x\cdot y \;=\; (cv)\cdot(dw) \;=\; c(v\cdot(dw)) \;=\; cd(v\cdot w)$$
$$\begin{aligned}(x-y)\cdot(x-y) \;&=\; \big((x-y)\cdot x\big)-\big((x-y)\cdot y\big)\\ &=\; \big((x\cdot x)-(y\cdot x)\big)-\big((x\cdot y)-(y\cdot y)\big) \;=\; (x\cdot x)-2(x\cdot y)+(y\cdot y)\end{aligned}$$
Such manipulations are so routine that we usually gloss over the individual steps.

With some foresight, the proof works if we choose the scalars $c = v\cdot w$ and $d = v\cdot v$. ■

Proof of Theorem B.36. If v is the zero vector, then $(v \cdot w)^2 \le (v \cdot v)(w \cdot w)$ because the left side is 0 (since $v \cdot w = \sum_i v_i w_i = \sum_i 0 w_i = \sum_i 0 = 0$) and the right side is also 0 (since $v \cdot v = 0$).

For the rest of the proof, assume v is nonzero, so $v \cdot v > 0$ by Lemma B.38. Define the vectors $x = (v \cdot w)v$ and $y = (v \cdot v)w$. By Lemma B.37.*(i)*:

$$x \cdot x = (v \cdot w)^2 (v \cdot v) \qquad x \cdot y = (v \cdot w)(v \cdot v)(v \cdot w) = (v \cdot w)^2 (v \cdot v) \qquad y \cdot y = (v \cdot v)^2 (w \cdot w)$$

Applying Lemma B.38 and Lemma B.37.*(ii)*, and noting that $x \cdot y = x \cdot x$:

$$0 \le (x - y) \cdot (x - y) = (x \cdot x) - 2(x \cdot y) + (y \cdot y) = -(x \cdot x) + (y \cdot y)$$

Rearranging yields $x \cdot x \le y \cdot y$. Plugging in our expressions for $x \cdot x$ and $y \cdot y$, we have $(v \cdot w)^2 (v \cdot v) \le (v \cdot v)^2 (w \cdot w)$. Dividing both sides by $v \cdot v$ (which is positive) produces $(v \cdot w)^2 \le (v \cdot v)(w \cdot w)$. ■

Corollary B.39. $\left(\sum_i |v_i|\right)^2 \le n \sum_i v_i^2$ *for every vector* $v \in \mathbb{R}^n$.

Proof. Define a vector $w \in \mathbb{R}^n$ such that $w_i = 1$ if $v_i \ge 0$, and $w_i = -1$ if $v_i < 0$. Note that $|v_i| = v_i w_i$ and $w_i^2 = 1$ for each index i. By Theorem B.36:

$$\left(\sum_i |v_i|\right)^2 = \left(\sum_i v_i w_i\right)^2 = (v \cdot w)^2 \le (v \cdot v)(w \cdot w) = \left(\sum_i v_i^2\right)\left(\sum_i w_i^2\right) = n \sum_i v_i^2$$ ■

B.8.4 Solving underdetermined systems

This is among the most fundamental results in linear algebra:

Theorem B.40. *Over every field, for every* $n \times (n+1)$ *matrix* A*, there exists a nonzero column vector* x *such that* Ax *is the zero vector.*

This says every underdetermined homogeneous system of linear equations has a nontrivial solution. Viewing the entries of x as unknowns and the entries of each row of A as coefficients, the system is:

$$\begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n+1} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n+1} \\ \vdots & \vdots & & \vdots \\ A_{n,1} & A_{n,2} & \cdots & A_{n,n+1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \begin{aligned} A_{1,1}x_1 + A_{1,2}x_2 + \cdots + A_{1,n+1}x_{n+1} &= 0 \\ A_{2,1}x_1 + A_{2,2}x_2 + \cdots + A_{2,n+1}x_{n+1} &= 0 \\ &\vdots \\ A_{n,1}x_1 + A_{n,2}x_2 + \cdots + A_{n,n+1}x_{n+1} &= 0 \end{aligned}$$

Underdetermined means there are more unknowns than equations. *Homogeneous* means the right-hand side is all 0s. The trivial solution would just assign $x_j = 0$ for all j. In a *nontrivial* solution, $x_j \ne 0$ for at least one j, which means the column vector x is nonzero.

To prove Theorem B.40, we argue that a standard algorithm for solving systems of linear equations is guaranteed to find a nonzero solution in the underdetermined homogeneous case.

Proof. Let A_i denote A's i^{th} row. The following recursive subroutine maintains the invariant that for every $n \times (n+1)$ matrix A, solve(A) returns a nonzero column vector x such that Ax is the zero vector.

```
solve(A):
    if A's last column is all 0s: return $x = \begin{bmatrix}0 & \cdots & 0 & 1\end{bmatrix}^\top$
    if $n = 1$: return $x = \begin{bmatrix}1 & -A_{1,1}/A_{1,2}\end{bmatrix}^\top$
    since A's last column is not all 0s, suppose $A_{n,n+1} \neq 0$ for simplicity of notation
    $y \leftarrow$ solve(B) where B is the $(n-1) \times n$ matrix such that for each $i \in [n-1]$:
        $B_i = A_i - (A_{i,n+1}/A_{n,n+1})A_n$ but with the last entry (which would be 0) deleted
    return $x = y$ but with $x_{n+1} = -\big(A_{n,1}y_1 + A_{n,2}y_2 + \cdots + A_{n,n}y_n\big)/A_{n,n+1}$ appended
```

For the base cases when A's last column is all 0s, the invariant holds by definition. For the base cases when $n = 1$ and $A_{1,2} \neq 0$, the invariant holds since:

$$Ax \;=\; A_{1,1}1 + A_{1,2}(-A_{1,1}/A_{1,2}) \;=\; A_{1,1} - A_{1,1} \;=\; 0$$

For all other A, to see that solve(A) maintains the invariant, we assume the invariant holds for solve(B) and argue that it also holds for solve(A): Assuming y is nonzero and By is zero, we show that x is nonzero and Ax is zero. Observe that x is nonzero since y is. Now, we argue that $A_i x = 0$ for each $i \in [n]$. When $i = n$:

$$A_n x \;=\; A_{n,1}y_1 + A_{n,2}y_2 + \cdots + A_{n,n}y_n + A_{n,n+1}x_{n+1} \;=\; 0$$

Now, suppose $i < n$. In the definition of B_i, the deleted last entry would have been:

$$A_{i,n+1} - (A_{i,n+1}/A_{n,n+1})A_{n,n+1} \;=\; 0$$

So regardless of x_{n+1}'s value,

$$0 \;=\; B_i y \;=\; \big(A_i - (A_{i,n+1}/A_{n,n+1})A_n\big)x \;=\; A_i x - (A_{i,n+1}/A_{n,n+1})A_n x \;=\; A_i x$$

by distributivity and $A_n x = 0$. The recursion terminates because B has fewer rows than A and so the recursion eventually reaches a base case. ■

Exercises

B.1 **(a)** Prove that for all real numbers $x_1, x_2, \ldots, x_n$:

$$\min(|x_1|, \ldots, |x_n|) \le |\max(x_1, \ldots, x_n)| \quad\text{and}\quad |\min(x_1, \ldots, x_n)| \le \max(|x_1|, \ldots, |x_n|)$$

(b) Prove that for all real numbers a, b, c, d, if $a \ge b$ and $c \ge d$ then $ac + bd \ge ad + bc$.

(c) Prove that for all nonnegative real numbers a, b, c, we have $\min(a, b) + \min(a, c) \ge \min(a, b + c)$.

B.2 Prove or disprove: For every directed graph (with no multi-edges or self-loops), there exist nodes u and v (possibly $u = v$) such that indeg(u) = outdeg(v).

B.3 Theorem B.6 says for every set S containing more than half the nodes in the n-dimensional hypercube, some node in S has at least one neighbor in S. Strengthen this to prove that some node in S has at least two neighbors in S. Hint: First, do the case $n = 2$.

B.4 Use the pigeonhole principle to prove that for every $n \geq 4$ and every set $U \subseteq [n]$ with $|U| \geq 2 \log n$, there exist disjoint nonempty subsets $S, T \subseteq U$ such that $\sum_{x \in S} x = \sum_{x \in T} x$.

B.5 **(a)** Use the pigeonhole principle to prove that for every $(n+1) \times \big(n\binom{n+1}{2} + 1\big)$ matrix M whose entries come from $[n]$, there exists a monochromatic 2×2 submatrix (that is, there exist row indices $i \neq j$ and column indices $k \neq \ell$ such that $M_{i,k} = M_{j,k} = M_{i,\ell} = M_{j,\ell}$).

(b) Design an $(n+1) \times \big(n\binom{n+1}{2}\big)$ matrix (for arbitrary n) whose entries come from $[n]$, with no monochromatic 2×2 submatrix.

B.6 Prove that for every $n \geq 2$, the sets of even-weight bit strings and odd-weight bit strings (of length n) are the only two examples for Proposition B.2. That is, if a set S contains half of $\{0,1\}^n$ and every pair of distinct strings in S differ in more than one bit position, then S is either the set of even-weight strings or the set of odd-weight strings.

B.7 Prove that for every undirected graph with n nodes (and no multi-edges or self-loops), if every node has degree $\geq (n-1)/2$ then the graph is connected.

B.8 Prove that for every string $x \in \{0,1\}^n$:

(a) If weight$(x) \geq n/2 + 1$ then x has two consecutive 1s somewhere.

(b) If x has an odd number of 0s and an odd number of 1s, then x is not a palindrome.

B.9 Prove that for every undirected graph with $n \geq 2$ nodes (and no multi-edges or self-loops), if every node has degree ≤ 3 then there do not exist two cycles that each have more than $n/2$ edges but which share no edges with each other.

B.10 Prove that for all nonnegative numbers $a_1, a_2, b_1, b_2, c_1, c_2$, if $\max(a_1+a_2,\ b_1+b_2) \leq c_1+c_2$ then there exists $i \in \{1,2\}$ such that $\max(a_i,\ b_i) \leq 2 \cdot c_i$. Hint: Give a contrapositive proof, and consider two cases: either $c_1 \geq c_2$ or $c_2 \geq c_1$.

B.11 **(a)** Prove that for every $m \times n$ binary matrix M and every $\varepsilon > 0$, if M has weight $\leq \varepsilon mn$ then $\leq \sqrt{\varepsilon} m$ rows have weight $\geq \sqrt{\varepsilon} n$ (where weight means "number of 1s").

(b) Let D be a joint distribution over a sample space $S_1 \times S_2$. Recall the notation from § A.6.6: Sampling $s \sim D$ is equivalent to sampling $s_1 \sim D_1$ and then $s_2 \sim D^{s_1}$, where D_1 is the marginal distribution on the 1st coordinate, and D^{s_1} is the marginal distribution on the 2nd coordinate after conditioning on the 1st coordinate being s_1. Prove that for every event $B \subseteq S_1 \times S_2$ and every $\varepsilon > 0$, if $\mathbf{Pr}[B] \leq \varepsilon$ then $\mathbf{Pr}_{s_1 \sim D_1}\big[\mathbf{Pr}_{s_2 \sim D^{s_1}}[s \in B] \geq \sqrt{\varepsilon}\big] \leq \sqrt{\varepsilon}$.

B.12 Here are some calisthenics for "iff" proofs. Prove that for every discrete probability space:

(a) There exists an event A with $1/3 \leq \mathbf{Pr}[A] \leq 2/3$ iff there exists an event B with

$1/3 \le \mathbf{Pr}[B] \le 1/2$.

(b) The support equals the whole sample space iff every event, other than the whole sample space, has probability < 1.

B.13 Define the *symmetric difference* of two sets A and B as $A \triangle B = (A \setminus B) \cup (B \setminus A)$ (that is, all elements that are in one of A or B but not in the other). Prove that for all sets A, B, C, we have $A \triangle B \subseteq C$ iff $A \cup C = B \cup C$.

B.14 Prove the following characterizations of strong connectedness of directed graphs (with multi-edges and self-loops allowed).

(a) For every directed graph G and every node s, G is strongly connected iff for every node v, there exist a walk from s to v and a walk from v to s in G.

(b) For every directed graph G with at least three nodes, G is strongly connected iff for every three distinct nodes u, v, w, there exists a closed walk containing all three.

(c) For every directed graph $G = (V, E)$ with at least two nodes, G is strongly connected iff for every way of partitioning V into two nonempty sets S and T, there exists an edge $(u, v) \in E$ such that $u \in S$ and $v \in T$.

B.15 Prove that for all real numbers a, b, c, we have $\min(a, b) \le c \le \max(a, b)$ iff there exists a real number $0 \le x \le 1$ such that $c = xa + (1-x)b$.

B.16 The *identity* function on a set maps each element to itself. Recall that a function $f : S \to T$ is *surjective* when each element of T has at least one preimage, and f is *injective* when each element of T has at most one preimage, and $\circ$ denotes function composition.

(a) Prove that f is surjective iff there exists $g : T \to S$ such that $f \circ g$ is the identity on T.

(b) Prove that f is injective iff there exists $g : T \to S$ such that $g \circ f$ is the identity on S.

B.17 Prove that for all $n \ge 3$, n is even iff there exists a connected undirected graph with n nodes such that every node has degree exactly 3.

B.18 Prove that for all sets A, B, C, D:

(a) $(A \times B) \cap (C \times D) = \emptyset$ iff either $A \cap C = \emptyset$ or $B \cap D = \emptyset$.

(b) $(A \times B) \cup (C \times D) = (A \cup C) \times (B \cup D)$ iff the following both hold: $A \subseteq C$ or $D \subseteq B$, and $C \subseteq A$ or $B \subseteq D$.

B.19 **(a)** Show that for all subsets S and T of a finite set U, we have $|S \cap T| \ge |S| + |T| - |U|$.

(b) In an undirected graph (with no multi-edges or self-loops), an *independent set* is a subset S of nodes such that for all distinct $u, v \in S$, $\{u, v\}$ is not an edge, and a *clique* is a subset T of nodes such that for all distinct $u, v \in T$, $\{u, v\}$ is an edge. Prove by contradiction that for every graph with n nodes, either every independent set has size $< n/2 + 1$ or every clique has size $< n/2 + 1$.

B.20 Prove by contradiction: For all positive numbers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$, there exists $i \in [n]$ such that $a_i/b_i \geq (a_1 + \cdots + a_n)/(b_1 + \cdots + b_n)$.

B.21 Prove the following implications by contradiction.

(a) For every directed graph, if there exists a node u with $\text{indeg}(u) > \text{outdeg}(u)$ then there exists a node v with $\text{outdeg}(v) > \text{indeg}(v)$. Does the converse hold for every directed graph?

(b) For every binary matrix, if at least half of the columns are each at least half 1s, then at least one row is at least a quarter 1s. Does the converse hold for every binary matrix?

B.22 For an undirected graph G (with no multi-edges or self-loops), let $\overline{G}$ denote the *complement graph*, which has the same set of nodes as G but the "opposite" set of edges: Each edge $\{u, v\}$ of G is a non-edge of $\overline{G}$, and each non-edge $\{u, v\}$ of G is an edge of $\overline{G}$. Prove or disprove: For every graph G, either G or $\overline{G}$ is connected.

B.23 Prove that for every partition of $[n]$ into sets $S_1, S_2, \ldots, S_k$, there exists $i \in [k]$ such that for every $T \subseteq [n]$ with $|T| > n/k$, there exists $j \in [k] \smallsetminus \{i\}$ such that $T \cap S_j \neq \emptyset$.

B.24 Prove that for every undirected graph (with no multi-edges or self-loops) in which each node has degree $\geq d$:

(a) There exists a path of length d.

(b) Either the number of nodes is $> d^2$ or there exists a cycle of length ≤ 4.

B.25 Prove that for every discrete probability space, every outcome has probability $\leq 2/3$ iff some event has probability $\leq 2/3$ and $\geq 1/3$.

B.26 In §B.5 we gave two proofs that for every binary tree, the number of leaves is exactly one more than the number of internal nodes. Here you'll develop another proof.

View the tree as a single-elimination tournament: Each leaf is a team, and each internal node is a game played between two teams. The loser of the game is eliminated from the tournament, while the winner advances to play the game at the parent node. The overall winner is the winner of the root game. Each game produces a unique loser, and every team except the overall winner loses exactly one game. Thus the number of games (internal nodes) equals the number of eventual losers, which equals the number of teams (leaves) minus one.

Turn this idea into a formal proof by exhibiting a bijection between the set of internal nodes and the set of all but one of the leaves.

B.27 Recall that a round-robin tournament is a directed graph such that for every pair of distinct nodes u and v, exactly one of (u, v) or (v, u) is an edge.

(a) Reprove Theorem B.20 (every round-robin tournament has a path that visits all nodes) using an iterative algorithm that starts with a path of length 0 and increases its length by 1 in each iteration.

(b) Give a proof by algorithm that for every round-robin tournament, if there exists a cycle then there exists a cycle of length 3.

(c) Prove that for every round-robin tournament, there exists a node u such that for every node v, there exists a path of length ≤ 2 from u to v.

B.28 A *cut* in a graph is a partition of the set of nodes into two sets S and T. If an edge has one endpoint in S and the other endpoint in T, we say the edge *crosses* the cut. Prove that for every undirected graph G, there exists a partition of G's edges into cycles iff every cut is crossed by an even number of G's edges.

B.29 This exercise yields an algorithm to determine whether a directed graph G has a *full trail* (which traverses each edge exactly once) from node s to node t, and to find such a trail if one exists.

Say G is *weakly connected* when the underlying undirected graph (after erasing the arrow directions) is connected. An *isolated* node v touches no edges: $\text{indeg}(v) = 0 = \text{outdeg}(v)$. Prove that G has a full trail from s to t iff: G is weakly connected except for isolated nodes, and $\text{outdeg}(s) = \text{indeg}(s) + 1$, and $\text{indeg}(t) = \text{outdeg}(t) + 1$, and $\text{indeg}(v) = \text{outdeg}(v)$ for each other node v. For the $\Leftarrow$ direction, use the "proof by algorithm" technique: Argue that if the right side of the "iff" holds, then the following algorithm finds a full trail from s to t.

```
start at $s$, and repeat until $t$ is reached:
    follow an arbitrary unused outgoing edge from the current node
while the current $s$-$t$ trail is not full (there exists an unused edge in $G$):
    find a node $u$ that's visited by the current trail and has an unused outgoing edge
    start at $u$, and repeat until $u$ is reached again:
        follow an arbitrary unused outgoing edge from the current node
    splice this closed trail (from $u$ to $u$) into the current $s$-$t$ trail:
        the new $s$-$t$ trail follows the current trail from $s$ to $u$,
                                then does the closed trail back to $u$,
                                then resumes the current trail from $u$ to $t$
output the current $s$-$t$ trail
```

In particular, prove that the first phase indeed reaches t without getting stuck, and that each outer iteration of the second phase succeeds in finding such a u and reaches u again without getting stuck.

B.30 Throughout this problem, consider undirected graphs with multi-edges allowed but self-loops not allowed.

(a) Prove that for every graph, deleting an edge can make the number of connected components go up by at most one.

(b) Prove that every connected graph with n nodes has at least $n-1$ edges.

(c) Prove that every tree with at least two nodes has at least one node with degree 1, using an argument similar to Lemma B.22 (every dag has a sink).

(d) Prove that for every connected graph with n nodes, it's a tree iff it has only $n-1$ edges.

B.31 A *topological layering* of a dag is a way of partitioning the set of nodes into "layers" $L_1, L_2, \dots$ such that every edge goes from a lower-index layer to a higher-index layer. Formally, if (u, v) is an edge and $u \in L_i$ and $v \in L_j$, then $i < j$. (A topological order is just a topological layering in which each layer is a single node.)

Prove that for every dag, the minimum number of layers of any topological layering equals one plus the length of a longest path. Hint: Prove $\geq$ and $\leq$ separately.

B.32 Use the probabilistic method to prove that for every prime p and every set $S \subseteq [p-1]$ with $|S| > \frac{2}{3}(p-1)$, there exists $x \in [p-1]$ such that x and $-x$ and x^{-1} are all in S, where $-x$ is the additive inverse and x^{-1} is the multiplicative inverse in the field $\mathbb{Z}_p$.

B.33 Use the probabilistic method to prove that for every $S \subseteq \{0,1\}^n$ with $|S| > (1 - \frac{1}{n+1})2^n$, there exists $x \in \{0,1\}^n$ such that x, as well as each of x's neighbors in the hypercube, are all in S.

B.34 Use the probabilistic method to prove that for every $n \geq 2$, there exists an undirected graph with n nodes (and no multi-edges or self-loops) in which every independent set and every clique has size $\leq \lceil 2 \log n \rceil$. (Compare this with Exercise B.19.b.)

B.35 **(a)** Give a proof by algorithm that for every $n \times n$ binary matrix, if every row is at least half 1s, then there exists a set of at most $\lfloor \log n \rfloor + 1$ many columns such that every row has a 1 in at least one of those columns.

(b) Give a proof by algorithm showing that Theorem B.25 is almost tight: For every sufficiently large n and every $n \times n$ binary matrix, there exists a monochromatic $k \times k$ submatrix where $k = \lfloor \frac{1}{2} \log n \rfloor$. Hint: If at least half the rows are each at least half 1s, then seek a monochromatic submatrix with only 1s, otherwise aim for only 0s. Then, take inspiration from the proof for Exercise B.35.a.

B.36 **(a)** Prove that for every bit string $x \in \{0,1\}^n$ with weight $< \sqrt{n}$, there exists a rotation y of x such that $x \wedge y$ (the "bitwise and") equals the all-0 string.

(b) Suppose the statement of Exercise B.36.a is modified to say $\text{weight}(x) < 2\sqrt{n}$ rather than $\text{weight}\, x < \sqrt{n}$. Disprove this stronger proposition by exhibiting a counterexample for every perfect square $n \geq 4$.

B.37 Prove that for all bit strings $x, y \in \{0,1\}^n$ there exists a rotation z of y such that $\text{dist}(x, z) \geq \text{weight}(x) + \text{weight}(y) - 2\,\text{weight}(x)\text{weight}(y)/n$.

B.38 **(a)** Prove that for every $x \in \{0,1\}^n$ where n is odd, if we sample $y \in \{0,1\}^n$ uniformly at random then $\mathbf{Pr}\big[\text{dist}(x, y) < n/2\big] = 1/2$.

(b) Prove that for every nonempty $S \subseteq \{0,1\}^n$ where n is odd, there exists $y \in \{0,1\}^n$ such that $\text{dist}(x, y) < n/2$ holds for at least half of the strings $x \in S$.

B.39 Prove that for every $x \in \{0,1\}^n$ with $\text{weight}(x) = n/2$ where n is even:

- **(a)** If x starts and ends with the same bit ($x_1 = x_n$), then there exists $i \in [n-1]$ such that $\text{weight}(x_1 x_2 \cdots x_i) = i/2$.
- **(b)** There exists a rotation y of x such that for all $i \in [n]$, $\text{weight}(y_1 y_2 \cdots y_i) \geq i/2$.

B.40 For integers $a \geq 0$ and $b > 0$, we let $\gcd(a,b)$ denote their *greatest common divisor*: the largest integer d such that $a = q_1 d$ and $b = q_2 d$ for some integers q_1 and q_2. Prove the following as a corollary of Theorem B.29: For all integers $0 < a < b$ there exists $x \in \mathbb{Z}_b$ such that $ax \bmod b = \gcd(a,b)$.

B.41 **(a)** Prove that for every prime p and every nonzero $a \in \mathbb{Z}_p$, we have $a^{p-1} = 1$ in the field $\mathbb{Z}_p$. Hint: Show that the field elements $a \cdot 1$, $a \cdot 2$, $\ldots$, $a \cdot (p-1)$ are all distinct and nonzero, and then multiply them together.

- **(b)** Use Exercise B.41.a to prove that for every prime p and every multivariate polynomial over $\mathbb{Z}_p$, there exists a multivariate polynomial over $\mathbb{Z}_p$ that expresses the same function and for which every variable's individual degree is $< p$.
- **(c)** Use a counting argument to prove that for every prime p and every function $f : \mathbb{Z}_p \to \mathbb{Z}_p$, there exists a unique univariate polynomial of degree $< p$ that expresses f. (Note that this implies the univariate case of Exercise B.41.b.)
- **(d)** Use Exercise B.41.c to prove that for every prime p, every $n \geq 1$, and every function $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$, there exists a unique multivariate polynomial with every individual degree $< p$ that expresses f. (This subsumes Exercise B.41.b.) Hint: First, consider functions that evaluate to 1 on a single input and evaluate to 0 on all other inputs.

B.42 Use the algorithm from the proof of Lemma B.34 to find the quotient and remainder of $x^5 + x^3 + 1$ divided by $x^3 + x^2 + 1$ over $\mathbb{Z}_2$.

B.43 A univariate polynomial P (over any particular field) is *irreducible* when there do not exist positive-degree polynomials Q_1 and Q_2 such that $P = Q_1 \cdot Q_2$. Give a proof by recursive algorithm that every univariate polynomial can be factored into a product of irreducible polynomials.

B.44 Prove that for all $(a_1, \ldots, a_{n+1}) \in (\{0,1\}^n)^{n+1}$, there exists a nonempty $I \subseteq [n+1]$ such that $\bigoplus_{i \in I} a_i$ (that is, the bitwise xor of the strings indexed by I) is the all-0 string.

B.45 Adapt the proof of Theorem B.36 to work with $x = (\sqrt{w \cdot w})v$ and $y = (\sqrt{v \cdot v})w$.

B.46 This exercise guides you through a proof by case analysis that $1 + x \leq \mathrm{e}^x$ for every real number x. Formally, e^x is defined as the infinite power series $1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$. You may use the familiar property $\mathrm{e}^{x+y} = \mathrm{e}^x \mathrm{e}^y$, which holds since:

$$\textstyle\sum_{i=0}^{\infty} \frac{(x+y)^i}{i!} = \sum_{i=0}^{\infty} \sum_{j=0}^{i} \binom{i}{j} \frac{x^j y^{i-j}}{i!} = \sum_{i=0}^{\infty} \sum_{j=0}^{i} \frac{x^j}{j!} \cdot \frac{y^{i-j}}{(i-j)!} = \Big(\sum_{j=0}^{\infty} \frac{x^j}{j!}\Big)\Big(\sum_{k=0}^{\infty} \frac{y^k}{k!}\Big)$$

(Such rearrangements of infinite series cannot be taken for granted, but basic real analysis implies that it works as expected in this case.)

(a) Prove $1 + x \leq \mathrm{e}^x$ for $x \geq 0$.

(b) Prove $1 + x \leq \mathrm{e}^x$ for $x \leq -1$. Hint: Show $\mathrm{e}^x > 0$ by considering the quantity $\mathrm{e}^x \mathrm{e}^{-x}$.

(c) Prove $1 + x \leq \mathrm{e}^x$ for $-1 < x < 0$. Hint: First, show that each term in the power series is smaller in absolute value than the previous term.

B.47 Similar to Exercise B.46, prove $\mathrm{e}^x \leq 1 + x + x^2$ for all $x \leq 1$ by considering the three cases $x \leq -1$ and $-1 < x < 0$ and $0 \leq x \leq 1$.

B.48 **(a)** Prove $1/1 + 1/2 + 1/3 + \cdots + 1/(n-1) \geq \ln n$ for every $n \geq 2$ by using $1 + x \leq \mathrm{e}^x$ for all x (Exercise B.46).

(b) Prove $1/2 + 1/3 + 1/4 + \cdots + 1/n \leq \ln n$ for every $n \geq 2$ by using $\mathrm{e}^x \leq 1 + x + x^2$ for all $x \leq 1$ (Exercise B.47).

Notes

The distance between two bit strings is often called the "Hamming distance."

Various versions of Theorem B.4 are called "the handshaking theorem."

Binary matrices with no small monochromatic submatrices (as in Theorem B.25) are sometimes called "bipartite Ramsey graphs."

Our proof of Theorem B.29 is a version of the "extended euclidean algorithm."

Theorem B.36 is most often called the "Cauchy–Schwarz inequality" and has also been attributed to Bunyakovsky.

The algorithm in our proof of Theorem B.40 is known as "gaussian elimination."

Exercise B.1.b is a basic version of the "rearrangement inequality."

Something stronger than Exercise B.7 is known: For every undirected graph with n nodes (and no multi-edges or self-loops), if every node has degree $\geq (n-1)/2$ then not only is the graph connected, but in fact the graph has a path that visits every node ("Dirac's theorem").

Exercise B.41.a is known as "Fermat's little theorem."

Index