6.1200: Mathematics for Computer Science

Ace Chun

January 16, 2026

Contents

1 Logic and Proofs

A mathematical proof is a formal method of showing that a particular statement is true within a particular "universe" determined by a set of *axioms*, or base assumptions. A proof consists of a series of logical deductions from known true statements, following a set of inference rules.

A *proposition* is a statement that is either definitively true or false. A *predicate* ($P(n)$) is a proposition whose truth value depends on the value of one or more variables (n).

Logical operations take propositions (as boolean T/F values) and string them together, creating new propositions. The most fundamental operations are `NOT` ($\neg$ or $\overline{A}$), `AND` ($\wedge$), and `OR` ($\vee$). They are defined in terms of truth tables, as outlined below:

A	B	$\overline{A}$	$A \wedge B$	$A \vee B$
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

In addition, we have *implications* ($A \implies B$), which can be thought of as an "if" statement: if A is true, then B must also be true. If A is false, the statement is never executed, so we don't necessarily care about the truth value of B. The truth table for an implication looks like the following:

A	B	$A \implies B$
T	T	T
T	F	F
F	T	T
F	F	T

The statement $A \implies B$ is *logically equivalent* to $\overline{A} \vee B$ — they exhibit the same truth table for any arbitrary A and B. Logical inference rules allow us to take these operations and declare new propositions to be true by applying them. For example,

$$\frac{P \wedge (P \implies Q)}{Q}$$

The statement above the line is the *antecedent*, which can be composed using the inference rule to conclude the statement below the line, the *consequent*. The above inference rule is termed *modus ponens*, and it is essentially a way to *eliminate* implication statements; if we know that P implies Q, and we also know that P is true, then the only conclusion that satisfies both statements is for Q to be true as well. We can therefore conclude that Q is also a true proposition.

In addition, implications can be composed:

$$\frac{(P \implies Q) \wedge (Q \implies R)}{(P \implies R)}$$

We can transform $\wedge$ statements into $\vee$ statements and vice-versa using De Morgan's laws:

$$\overline{P} \vee \overline{Q} \iff \overline{(P \wedge Q)}$$
$$\overline{P} \wedge \overline{Q} \iff \overline{(P \vee Q)}$$

From $P \implies Q$, we can define three other kinds of statements:

1. **Converse:** $Q \implies P$
2. **Inverse:** $\overline{P} \implies \overline{Q}$
3. **Contrapositive:** $\overline{Q} \implies \overline{P}$

An implication is equivalent to its contrapositive, but it *is not* equivalent to its converse or its inverse. However, when $P \implies Q \wedge Q \implies P$, we call this an "if and only if" (iff) statement, and notate this as $P \iff Q$.

P	Q	$P \implies Q$	$Q \implies P$	$P \iff Q$
T	T	T	T	T
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

Note that the inverse is equivalent to the converse, as it is the contrapositive of the converse.

1.1 Quantifiers

Given some predicate $P(n)$, with n as a member of some set or domain of mathematical objects of its type. Predicate quantifiers allow us to make some statement over this domain.

The existential quantifier, abbreviated $\exists$, corresponds to the statement that there exists at least one n for which $P(n)$ is true.

$$\exists n[P(n)]$$

The universal quantifier, abbreviated $\forall$, corresponds to the statement that all possible n satisfy $P(n)$.

$$\forall n[P(n)]$$

In a sense, $\exists$ is an "infinite" $\vee$, and $\forall$ is an "infinite" $\wedge$. There are similar De Morgan's laws for existential and universal quantifiers.

$$\neg[\exists x P(x)] \leftrightarrow \forall x[\neg P(x)]$$

$$\neg[\forall x P(x)] \leftrightarrow \exists x[\neg P(x)]$$

Notationally, $\exists!$ may be used to indicate the existence of a *unique* n satisfying $P(n)$.

1.2 Direct Proof

Often, we want to prove that the statement $P \implies Q$ is *in itself* true, where P defines some world of assumptions that we take on in order to prove that Q is a logical consequence of those assumptions.

A direct proof simply follows the flow of the implication as stated; we first assume that P is true, and then use logical deductions to show that Q must also be true. This kind of proof implicitly uses the intuition behind modus ponens.

1.3 Contrapositive

The *contrapositive* of an implication is logically equivalent to the implication itself.

$$(P \implies Q) \iff (\overline{Q} \implies \overline{P})$$

Therefore, when we want to prove $P \implies Q$, it suffices to prove $\overline{Q} \implies \overline{P}$. In some scenarios, this may be easier than proving the implication directly;

we are, in some sense, finding what *must not* be true in P if Q were to be false.

1.4 Contradiction

Suppose we would like to prove some proposition $\overline{P}$, and we know that $P \implies Q$ and $\overline{Q}$ (i.e. P logically leads to Q, but Q is false). If we assume that P is true (and $\overline{P}$ is false), then, by modus ponens, we conclude that Q must be true. However, we arrive at a contradiction, since we know that $\overline{Q}$ must also be true. Therefore, the premises of the argument are unsound, and P must be false.

$$\frac{(P \implies Q) \wedge (\overline{Q})}{\overline{P}}$$

Another way to look at this is via the contrapositive. Since $P \implies Q$ is logically equivalent to $\overline{Q} \implies \overline{P}$, and since we know $\overline{Q}$, we can conclude $\overline{P}$.

1.5 Induction

Induction is most often used to prove that some predicate $P(n)$ holds for all items n over some set (often, $\mathbb{N}$). For example, if we want to show that the proposition $\forall n\ P(n)$ is true, where $P(n)$ is a statement about some natural number n, we can utilize the induction principle:

$$\frac{P(n_0) \wedge [\forall n \geq n_0\ (P(n) \implies P(n+1))]}{\forall n \geq n_0\ P(n)}$$

An inductive proof consists of two parts: the *base case* and the *inductive step*.

The base case is a demonstration that the statement is true for some *starting point* n_0. The inductive step then shows $P(n) \implies P(n+1)$ for any $n \geq n_0$, first by assuming $P(n)$ (the *inductive hypothesis*), then demonstrating that $P(n+1)$ must also be true. This gives us a "chain" of implications, which we can compose together to make a statement about all

n. For example, if $n_0 = 0$,

$$\begin{aligned} P(0) &\implies P(1) \\ P(1) &\implies P(2) \\ P(2) &\implies P(3) \\ &\vdots \end{aligned}$$

This, paired with $P(0)$, gives us the ability to demonstrate $P(n)$ for any n.

However, it is sometimes more convenient to assume that $P(n)$ holds for all smaller n, rather than just the preceding element. We may modify our inductive hypothesis slightly:

$$[P(0) \land P(1) \land \cdots \land P(n)] \implies P(n+1)$$

The above is referred to as the *strong inductive hypothesis*, in contrast to the *weak inductive hypothesis*. Formally, the strong induction rule is stated as

$$\frac{P(n_0) \land \left[\left(\bigwedge_{i=n_0}^{n} P(i) \right) \implies P(n+1) \right]}{\forall n \geq n_0 P(n)}$$

1.6 Casework

Often, it can be more convenient to prove a proposition by breaking the scenario at hand into simpler cases. For example, if we can prove that both

$$\begin{aligned} P &\implies Q \\ \overline{P} &\implies Q \end{aligned}$$

are true, then we may conclude that Q is *always true*, regardless of the value of P (as $P \lor \overline{P}$ is always true — this is the *law of the excluded middle*). In some sense, proof by casework is like distributing an implication over several difference scenarios that are $\lor$'d together.

2 Sets and Relations

2.1 Sets

A set is a mathematical object that is an unordered grouping of other mathematical objects: these can be anything, from integers, to irrational numbers,

to sets themselves. Objects within a set are called the elements, or members, of a set. We notate set membership of an object x inside some set A as

$$x \in A$$

Conversely, non-membership is notated as

$$x \notin A$$

A set B is said to be a subset of A if every element of B is a member of A:

$$B \subseteq A : \ \forall x[x \in B \rightarrow x \in A]$$

A set B is said to be a proper subset of A if B is a subset of A, and, in addition, there is at least one element of A that is not a member of B.

$$B \subset A : \ \forall x[x \in B \rightarrow x \in A] \wedge \exists y[y \in A \wedge y \notin B]$$

Two sets A and B are said to be equal iff A is a subset of B and B is a subset of A.

$$A = B \iff (A \subseteq B) \wedge (B \subseteq A)$$

The empty set, denoted $\varnothing$, is the set containing no elements.

The cardinality, or size, of a set is defined as the number of elements in the set. The cardinality of some set A is denoted $|A|$. A set A is said to be infinite if there exists some $A_n \subset A$ where $|S_n| = n$ for all natural numbers.

Given some domain, the complement of a set (denoted $\bar{A}$) is the set of all elements within the domain that are not contained within A.

A partition of a set S is a list of subsets $S_1, S_2, \cdots$, such that each element in S belongs to exactly one subset S_i.

2.1.1 Operations

The intersection of two sets A and B returns a new set that contains all x that are members of both A and B.

$$A \cap B = \{x | x \in A \wedge x \in B\}$$

The union of two sets A and B returns a new set that contains all x that are either members of A or members of B, or both.

$$A \cup B = \{x | x \in A \vee x \in B\}$$

Note that these two definitions can be extended to multiple sets. For $\{S_i | i \in \Gamma\}$ for a set of indexes Γ,

$$\bigcap_{i \in \Gamma} = \{s | s \in S_j, \text{ for every } j \in \Gamma\}$$

$$\bigcup_{i \in \Gamma} = \{s | s \in S_j, \text{ for some } j \in \Gamma\}$$

The difference $A - B$ is the set of all elements in A that are not in B.

$$A - B = \{x | x \in A \vee x \notin B\}$$

A side note: two sets are disjoint if $A \cap B = \varnothing$.

A set $B = S_i | i \in \Gamma$ of nonempty subsets of A is said to be a partition of A if

$$A = \bigcup_{i \in \Gamma} S_i$$

and

$$S_i \cap S_j = \varnothing$$

Essentially, every element of A must only be a member of some S_j.

The cartesian product of n sets is the set, of all ordered n-tuples where each index of a coordinate a_i is a member of S_i.

$$S_1 \times S_2 \times \cdots \times S_n = \{(a_1, a_2, \ldots, a_n) | a_i \in S_i\}$$

The power set of a set, denoted $\mathcal{P}(S)$, is the set of all subsets of S (including $\varnothing$ and S itself).

A symmetric difference between two sets is defined as

$$A \Delta B = (A - B) \cup (B - A)$$

2.2 Well Ordering Principle

The Well Ordering Principle is the statement that every non-empty set of non-negative integers necessarily has a *smallest* element. Since the principle holds for all such sets, it is often useful for proving that some predicate $P(n)$ applies to every non-negative integer. The template, by contradiction, is outlined as:

Let C be the set of all natural numbers n for which $P(n)$ is not true. For the sake of contradiction, suppose C is non-empty.

$$C = \{n \mid n \in \mathbb{N},\ \overline{P(n)}\}$$

By the well-ordering principle, this means that C will have some minimum element — we can call this n'. If we demonstrate that either $P(n')$ is actually true (so n' is not a member of C, and therefore cannot be its minimum element), or that there exists an element of C that is smaller than n', then we reach a contradiction — the well-ordering principle does not hold for C, meaning that C must be empty.

2.3 Relations

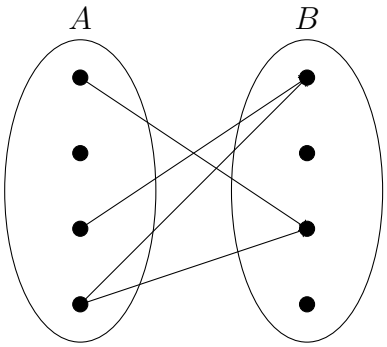
A (binary) relation R is a set $R \subseteq A \times B$ determined by:

- A, a domain
- B, a codomain
- $R \subseteq A \times B$, a specification of the ordered pairs encompassed by R

A relation is a general construct that maps some sort of association between the elements of A and the elements of B. For $a \in A$ and $b \in B$, a is said to *relate to* b if the pair $(a, b) \in R$. Notationally, this can be written in a couple of different ways:

$$(a, b) \in R \iff aRb \iff R(a, b)$$

Relations can be visualized in a kind of directed graph, with arrows coming out of elements in A and going into its related element(s) in B.



R is said to be a *function* iff $\forall a \in A$, a relates to at most one element in B (so there is at most one arrow pointing out of each element of A). We have the following notation for functions:

$$R : A \to B,\ (a, b) \in R \iff R(a) = b$$

R is said to be *total* if every element in the domain A is related to at least one element in B (so there is at last one arrow pointing out of each element of A). If R is a *total function*, every element of A is related to exactly one element of B.

We have similar characterizations pertaining to arrows into B.

- R is *injective* if every element of B is mapped to at most once (so there is at most one arrow pointing into each element of B).
- R is *surjective* if every element of B is mapped to at least once (so there is at least one arrow pointing into each element of B).
- R is *bijective* if R is a total function that is both injctive and surjective (so there is exactly one arrow connecting every element to A to every element of B).

These characterizations can be used to compare sizes of sets. If R is total relation,

- R is injective $\implies |A| \leq |B|$
- R is surjective $\implies |A| \geq |B|$
- R is bijective $\implies |A| = |B|$

2.3.1 Equivalence

Relations are often defined over a single set, such that $R \subseteq A \times A$. This is referred to as a *relation on* A. With this, we can define some notion of equivalence between the members of A, which requires three properties:

- **Reflexivity:** $\forall a \in A,\ aRa$ (every element is equivalent to itself)
- **Symmetry:** $aRb \implies bRa$ (equivalence goes both ways)
- **Transitivity:** $aRb \wedge bRc \implies aRc$ (equivalence is preserved)

If R satisfies the properties above, it is called an *equivalence relation*. Equivalently, R partitions A into subsets $A_1, A_2, \cdots$ called *equivalence classes*, such that each $a \in A$ belongs to exactly one A_i and aRb implies that a and b are members of the same A_i.

2.3.2 Ordering

We can also define the notion of an ordering or comparison between members on A. A *weak partial ordering* R satisfies the following:

- **Reflexivity:** $\forall a \in A,\ aRa$
- **Antisymmetry:** $aRb \wedge bRa \implies a = b$
- **Transitivity:** $aRb \wedge bRc \implies aRc$

In a weak partial ordering, there may exist a pair of elements (a, b) that are not directly comparable, so the ordering is not defined between those elements. A *total ordering* is an ordering in which *every pair* is comparable to each other.

2.4 Counting

Counting the sizes of sets rely on drawing equivalences/bijections to other sets that are somehow easier to count. We have a few basic counting rules:

- **Product rule:** For finite sets $A_1, A_2, \cdots, A_n$,
$$|A_1 \times A_2 \times \cdots \times A_n| = |A_1||A_2| \cdots |A_n|$$
- **Sum rule:** For pairwise disjoint finite sets $A_1, A_2, \cdots, A_n$,
$$|A_1 \sqcup A_2 \sqcup \cdots \sqcup A_n| = |A_1| + |A_2| + \cdots + |A_n|$$
- **Bijection rule:** If there exists a bijection $R \subseteq A \times B$, then
$$|A| = |B|$$
- **Division rule:** If there exists a k-to-1 correspondence between A and B for some constant k (i.e., every element of b is mapped to by exactly k elements of A), then
$$|B| = \frac{|A|}{k}$$

The product rule and sum rule are usually applied when counting the number of elements of a set according to a set of criteria. The product rule roughly corresponds to taking the `AND` of these criteria, and the sum rule corresponds to taking the `OR` of these criteria. The division rule is used when the elements of B are strategically overcounted by a constant factor k in counting $|A|$, which can be accounted for by dividing by that factor.

A *permutation* of a set A is a sequence that contains each element in A exactly once. If $|A| = n$, then there are $n!$ permutations of the elements of A.

If, instead, we wanted ordered sequences of exactly $r < n$ elements, the total quantity $n!$ overcounts by a factor of $(n-r)!$ (representing the number of ways to order the elements not included in the sequence, which we do not care about). Therefore, the number of length-r sub-permutations of A is

$$\frac{n!}{(n-r)!}$$

Further, if we did not care about the ordering of the r elements, and instead just wanted to count the number of ways to *choose* r elements from the set, we would have to account for the additional factor of $r!$.

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

$\binom{n}{r}$ is also referred to as a *binomial coefficient*, which comes from the *binomial theorem*:

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

Binomial coefficients are symmetric:

$$\binom{n}{r} = \binom{n}{n-r}$$

This is because choosing a set of r objects to *include* is equivalent to choosing a set of $n-r$ objects to *exclude*. In effect, $\binom{n}{r}$ counts the number of size-2 partitions of the set, each with a fixed number of members. If we instead wanted k different partitions with $r_1, r_2, \cdots, r_k$ members each, we obtain a multinomial coefficient:

$$\binom{n}{r_1, r_2, \cdots, r_k} = \frac{n!}{r_1! r_2! \cdots r_k}$$

where

$$r_1 + r_2 + \cdots + r_k = n$$

The corresponding multinomial theorem states

$$(x_1 + x_2 + \cdots x_k)^n = \sum_{r_1+\cdots+r_k=n} \binom{n}{r_1, r_2, \cdots, r_k} x_1^{r_1} x_2^{r_2} \cdots x_k^{r_k}$$

2.4.1 Inclusion-Exclusion Principle

If we want to sum over non-disjoint sets that may have overlapping elements, we have to compensate for double-counted elements.

$$\begin{aligned}\left|\bigcup_{i=1}^{n} A_i\right| = \sum_i |A_i| - \sum_{i_1<i_2} |A_{i_1} \cap A_{i_2}| + \cdots \\ + (-1)^{r+1} \sum_{i_1<\cdots<i_r} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_r}| + \cdots \\ + (-1)^{n+1} |A_1 \cap A_2 \cap \cdots \cap A_n|\end{aligned}$$

2.4.2 Pigeonhole Principle

If there exists a total function $f : A \to B$ for sets such that $|A| > |B|$, then f cannot be injective; there exists $a_1, a_2 \in A$ such that $f(a_1) = f(a_2)$, but $a_1 \neq a_2$. Equivalently, there must exist at least 2 elements of A that map to the same element in B.

More generally, if $|A| > k \cdot |B|$, then there must exist at least $k + 1$ elements in A that map to the same element in B.

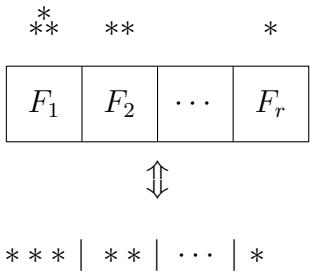
Importantly, the pigeonhole principle is a statement about *existence* — it doesn't say anything about which particular elements of A will collide, only the fact that they exist.

2.4.3 Stars and Bars

The number of ways to place n indistinguishable objects into r separate bins is

$$\binom{n + r - 1}{r - 1}$$

To see why, we can map the r bins as $r - 1$ *fences* that can be placed in between the n objects in a row.



The objects and the fences together form a string of length $n+r-1$. Out of these spots, we are then choosing $r-1$ spots to place the fences (or, equivalently, n spots to place the objects). Therefore, the number of configurations is

$$\binom{n+r-1}{r-1} = \binom{n+r-1}{n}$$

2.4.4 Combinatorial Proofs

A combinatorial proof, contrary to an algebraic proof of a combinatorial identity, aims to "intuit" the meaning of an identity. We must come up with some sort of situation that would be described by both sides of the combinatorial identity, and then explain why both sides describe the same situation (essentially, constructing and proving a bijection between the two scenarios).

There are a few rules of thumb when handling these proofs:

1. Sum indicates a partition, or a proof by cases. Often, the use of a sum will mean that we must consider casework of different situations.

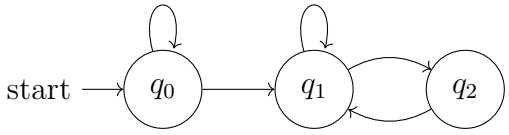
2. Products indicate independent choices. The multiplication of entities indicates that there are different combinations of choices being considered.

3. Binomial coefficients indicate that we are choosing some subset of objects from a larger subset.

3 Program Analysis and Combinatorics

3.1 State Machines

It is often convenient to model processes as a sequence of *states* and *transitions* between states. We can diagram states as nodes, interconnected by arrows for each transition. For example,



State q_0 can transition to itself or q_1. q_1 can transition to itself or q_2. q_2 may only transition back to q_1. States can model the current status of some system, and transitions model ways in which that status can *change* according to the rules of the system.

We refer to the set of states of the machine as Q, the start state as q_0, and the transitions T as a set of pairs:

$$T = \{r \mapsto s \mid r \text{ can transition to } s\}$$

An *execution* of a state machine is a sequence of valid transitions that the system can take, beginning with its start state. A state is *reachable* if there exists a valid execution that leads to that state.

Now, consider some predicate $P(q)$ over the states of the system. We say that $P(q)$ is *preserved* if transitions do not change the truth value of the predicate; that is, if $P(q)$ is true for some q, and there exists a transition $q \mapsto r$, then $P(r)$ is also true.

$P(q)$ is an invariant predicate if it is true for all *reachable* states. Implicitly, this encodes two pieces of information: $P(q_0)$ (for the start state q_0), and $P(q)$ is a preserved predicate. This is known as the *invariance principle*. In some sense, the invariance principle is a form of induction; if we prove $P(q_0)$, where q_0 is our base case, and that $P(q) \implies P(r)$ for any $q \mapsto r$ transition, then $P(q)$ holds for all reachable states *because* we have followed all possible ways to move from q_0 over all possible path extensions.

The invariance principle can be used to prove that a state (say, q_r) is unreachable from a given start state. To do so, we may state and claim that some predicate $P(q)$ is an invariant *given that* we start at some state q_0, for which $P(q_0)$ holds. It then suffices to demonstrate that $P(q_r)$ is *false*; by the

invariance principle, q_r must therefore be unreachable via valid transitions from q_0.

3.2 Correctness and Termination

In analyzing a program (often articulated as a state machine), there are two properties that we'd like to maintain in order to verify that it executes as desired: *partial correctness* and *termination*.

A state q_t is a *final state* if there are no valid transitions leaving it — if the state machine happens to hit q_t in the middle of an execution, it will stop running. A state machine is said to *terminate* if there are no infinite executions (so, it hits a final state at some point).

Partial correctness is the verification that all final states of the program are also correct (in that they achieve the desired property) — for example, if our program was a sorting algorithm and the states were permutations of some input sequence, then partial correctness would require that all final states listed the elements of the sequence in correctly sorted order. Another way to frame this is in terms of some predicate $P(q)$ over the states that must be preserved for any state that can lead to a correct conclusion. We term this *partial* correctness, as it only guarantees that the program is correct *if* it terminates. In order to ensure total correctness, we must then also prove that it does, in fact, reach a final state.

To prove termination, we refer to the method of *derived variables* (also called *potential functions*). A derived variable f is simply a function that maps the states of the machine to the real numbers; we can think of it as some kind of "score" that we assign to the states that encodes some kind of useful property.

$$f : Q \to \mathbb{R}$$

f is said to be *strictly decreasing* iff

$$r \mapsto s \implies f(r) > f(s)$$

that is, for any transition, the value of the derived value can only *go down*. If f is strictly decreasing and its range is defined over some discrete set with a minimum element, then there must be a point at which f can no longer decrease; the program must terminate at some point. We can also consider the contrapositive: if we suppose that the program does not terminate, it must have an infinitely long execution. However, this would mean that there

is a strictly decreasing sequence of values of f that is infinitely long, which is a contradiction with our initial assumptions about f.

This idea motivates the following statement: if f is a strictly decreasing derived variable that takes values over $\mathbb{N}$, then the length of any execution starting at some state q must be at most $f(q)$. This theorem generalizes to any *well ordered set*, as the Well Ordering Principle states that any well ordered set must have a well-defined minimum element.

3.3 Sums

Modeling a quantity as an iterated sum of several smaller quantities over some indexed variable is useful for many modeling approaches, especially with regards to determining algorithm runtimes and complexity. However, we would like to be able to take an expanded sum that iterates over a variable to find a *closed form* formula that expresses the same quantity.

3.3.1 Common Forms

A *geometric series* takes the form

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n$$

for some common ratio x. The closed form of this sum is

$$\sum_{k=0}^{n} x^k = \frac{1 - x^{n+1}}{1 - x}$$

As $n \to \infty$, the sum converges if $x < 1$, and diverges (goes to ∞) if not.

$$\sum_{k=0}^{n} x^k = \frac{1}{1 - x}, \ x < 1$$

Another common sum is

$$\sum_{k=0}^{n} k = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

In many cases, it is possible to break a more complicated sum into simpler forms that we already know the closed form for, which we may then compose

back into a larger formula. Based on the form of the summand and from computing a few terms, it is also possible to guess a formula and check its correctness via induction. The general structure of an inductive proof for proving a sum formula is as follows:

Suppose $S_n = \sum_{k=0}^{n} f(k)$, and we would like to prove that $S_n = F(n)$ for some function of n.

1. **Base Case:** Demonstrate that $S_0 = F(0)$ (or, more generally, $S_a = F(a)$ for some starting point a).
2. **Inductive Step:** Demonstrate

$$S_n = F(n) \implies S_{n+1} = F(n+1)$$

We can break S_{n+1} into $S_n + f(n+1)$. Using the inductive hypothesis, it then suffices to demonstrate that

$$F(n+1) = F(n) + f(n+1)$$

3.3.2 Perturbation

The perturbation method compares a sum to a slightly modified or rewritten version of itself. For example, the geometric sum may be rewritten in the following way:

$$\begin{aligned} S &= 1 + x + x^2 + \cdots + x^n \\ xS &= \quad x + x^2 + \cdots + x^n + x^{n+1} \end{aligned}$$

S and xS share all of their terms, with the exception of their endpoints. We observe that

$$S - xS = 1 - x^{n+1} = (1-x)S$$

and therefore, the total sum must be equal to

$$S = \frac{1 - x^{n+1}}{1 - x}$$

As another example, the sum $\sum_{k=0}^{n} k$ can be rewritten *backwards*:

$$S = 1 + \quad 2 \quad + \quad 3 \quad + \cdots + n$$
$$S = n + (n-1) + (n-2) + \cdots + 1$$

We then note that when we add the top and bottom sum together, all n of the terms combine to $n+1$.

$$2S = \underbrace{(n+1) + (n+1) + \cdots + (n+1)}_{n \text{ terms}} = n \cdot (n+1) \implies S = \frac{n(n+1)}{2}$$

3.3.3 Ansatz Method

The ansatz method takes an "educated guess" about the general form of the closed form solution, with arbitrary constants, and then backsolves for the values of the constants by constructing a system of equations. For example, we might guess that the sum

$$\sum_{k=0}^{n} k^2$$

evolves roughly according to a cubic in n, as we are adding n terms that are bounded by n^2. We can hypothesize that

$$S_n = \sum_{k=0}^{n} k^2 = an^3 + bn^2 + cn + d$$

From manual calculation, we find that

$$S_0 = 0 = d$$
$$S_1 = 1 = a + b + c + d$$
$$S_2 = 5 = 8a + 4b + 2c + d$$
$$S_3 = 14 = 27a + 9b + 3c + d$$

Solving the system of equations yields

$$a = 1/3,\ b = 1/2,\ c = 1/6,\ d = 0$$

so

$$\sum_{k=0}^{n} k^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n = \frac{n(n+1)(2n+1)}{6}$$

which we may verify for all n (not just from 0 to 3) through induction.

3.3.4 Nested sums

We can resolve nested summations by first resolving the innermost sums (according to known formulas or methods listed above) in terms of the index variables of the outermost sums. In effect, we are eliminating index variables sequentially. For example,

$$\sum_{i=0}^{n}\sum_{j=0}^{i} j = \sum_{i=0}^{n} \frac{i(i+1)}{2}$$

We can then resolve this expression by breaking apart the inner summand into $\frac{i^2}{2}$ and $\frac{i}{2}$.

It is also sometimes useful to reverse the order of summation to make the inner sum more convenient to resolve in this manner. For example, the inner sum of

$$\sum_{i=0}^{n}\sum_{j=i}^{n} j$$

cannot be resolved directly with any convenient formulas. However, note that this is equivalent to summing over all indices i, j such that $0 \leq i \leq j \leq n$. In particular, i is always bound between 0 and j. Therefore,

$$\sum_{i=0}^{n}\sum_{j=i}^{n} j = \sum_{j=0}^{n}\sum_{i=0}^{j} j = \sum_{j=0}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$$

3.3.5 Integral Approximation

Even if we are unable to find an exact value for a sum, it is still useful to find an approximation and corresponding error bound for estimation purposes.

Let $f(x)$ be a weakly increasing function, and we want to find the discrete sum

$$S = \sum_{k=1}^{n} f(k)$$

Recalling the definition of an integral as the limit of a Riemann sum, observe

that

$$I = \int_1^n f(x)dx \geq \int_1^n f(\lfloor x \rfloor)dx = \sum_{k=1}^{n-1} f(k) = S - f(n)$$
$$I \leq \int_1^n f(\lceil x \rceil)dx = \sum_{k=2}^{n} f(k) = S - f(1)$$

where the directions of the inequalities are obtained by noting that, since $f(x)$ is *increasing*, $f(\lfloor x \rfloor) \leq f(x)$ and $f(\lceil x \rceil) \geq f(x)$. Therefore,

$$I + f(1) \leq S \leq I + f(n)$$

If $f(x)$ is a weakly decreasing function, we retain the same conceptual idea, but we instead have to switch the bounds (as $f(\lfloor x \rfloor) \geq f(x)$ and $f(\lceil x \rceil) \leq f(x)$).

$$I + f(n) \leq S \leq I + f(1)$$

3.4 Asymptotics

In many applications within computer science, we are interested in the *asymptotic* behavior of functions and sums; i.e., what happens to the value of a function as its input grows. In tandem, we would like to be able to compare the long-term behavior of two mathematical functions in relation to each other.

For functions $f(x)$ and $g(x)$, we have the following relations:

$$\begin{aligned}
f(x) \sim g(x) &\iff \lim_{x\to\infty} \frac{f(x)}{g(x)} = 1 \\
f(x) \in O(g(x)) &\iff \exists c \exists x_0 \ s.t. \ \forall x \geq x_0, \ |f(x)| \leq c \cdot g(x) \\
f(x) \in o(g(x)) &\iff \lim_{x\to\infty} \frac{f(x)}{g(x)} = 0 \\
f(x) \in \Omega(g(x)) &\iff g(x) \in O(f(x)) \\
f(x) \in \omega(g(x)) &\iff \lim_{x\to\infty} \frac{f(x)}{gx} \to \infty \\
f(x) \in \Theta(g(x)) &\iff f(x) \in O(g(x)) \wedge g(x) \in O(f(x))
\end{aligned}$$

$\sim$ denotes functions that grow at the same rate as each other, including any constant factors — this is contrasted with $\Theta(g(x))$, which describes rough

equality *ignoring* constant factors. $O(g(x))$ describes functions that grow at the same rate as or slower than $g(x)$ itself, while $o(g(x))$ describes functions that grow *strictly slower* than $g(x)$. Conversely, $\Omega(g(x))$ describes functions that grow at the same rate as or faster than $g(x)$, and $\omega(g(x))$ requires strictly faster growth.

Note that O, o, Ω, and ω delineate *sets of functions* — to say $O(g(x))$, for example, is to denote the *set of all functions* that are asymptotically upper-bounded by $g(x)$ (i.e. those that grow at the same rate as or slower than g). In addition, we have limit tests for $O(g(x))$ and $\Omega(g(x))$, though these are *one-sided implications*; if the limit does notexist,then the relationship is inconclusive.

$$f(x) \in O(g(x)) \implies \lim_{x\to\infty} \frac{|f(x)|}{g(x)} < \infty$$
$$f(x) \in \Omega(g(x)) \implies \lim_{x\to\infty} \frac{|f(x)|}{g(x)} > 0$$

We can make the following analogies to (in)equalities:

$$\begin{aligned} f \sim g &\iff \text{“} = \text{”} \\ f \in O(g) &\iff \text{“} \leq \text{”} \\ f \in o(g) &\iff \text{“} < \text{”} \\ f \in \Omega(g) &\iff \text{“} \geq \text{”} \\ f \in \omega(g) &\iff \text{“} > \text{”} \\ f \in \Theta(g) &\iff \text{“} = \text{”} \end{aligned}$$

Applying these to sums, we can find integral approximation bounds for a summation and take the limit of each bound. If they converge to the same function, the corresponding sum must grow accordingly by the squeeze theorem. For example, for an increasing function $f(x)$,

$$\lim_{n\to\infty} \int_1^n f(x)dx + f(1) \leq \lim_{n\to\infty} S_n \leq \lim_{n\to\infty} \int_1^n f(x)dx + f(n)$$

3.5 Recurrences

Similar to the motivation behind solving sums, certain quantities and sequences may be easier to describe recursively, rather than with a closed-form

function from the start. In particular, when analyzing the runtime $T(n)$ of a recursive algorithm with an input of size n, it is likely much more straightforward to describe $T(n)$ in terms of $T(n')$ for $n' < n$, the recursive subproblem in the algorithm.

In addition, every recurrence requires the specification of some number of base cases to signify where the sequence starts.

3.5.1 Linear Recurrences

$T(n)$ is said to be a linear recurrence if it is defined as a linear function of previous terms $T(n-c)$ in the sequence, for some constants c. For example, the Fibonacci sequence

$$F(n) = F(n-1) + F(n-2)$$

with initial conditions $F(0) = 0$ and $F(1) = 1$, is a linear recurrence, as $F(n)$ is a linear combination of $F(n-1)$ and $F(n-2)$. The order of a linear recurrence is the offset of the furthest element from n that is referred to in the definition of $T(n)$ — in the above, the furthest element of the sequence is $T(n-2)$, so the Fibonacci sequence is a second-order linear recurrence.

One strategy for solving linear recurrences is to just write out the first few terms and then observe a pattern, whose correctness can be verified by an inductive proof (similar to the format outlined for sums).

Another method for solving linear recurrences takes inspiration from the method used to solve homogeneous linear differential equations. The method relies on the ansatz that $T(n) = c_1 x_1^n + c_2 x_2^n + \cdots + c_k x_k^n$ for order k, some fixed ratios x_i and constants c_i (that are determined by initial conditions). Each of the x_i's are roots to the *characteristic polynomial* of $T(n)$, $p_T(x)$:

$$T(n) = a_1 T(n-1) + \cdots + a_k T(n-k) \iff p_T(x) = x^k - [c_1 x^{k-1} + \cdots + c_{k-1} x + c_k]$$

Taking a look at the Fibonacci recurrence above, we note that its characteristic equation is

$$x^2 - x - 1 = 0$$

which has solutions ϕ (the golden ratio) and its conjugate $\bar{\phi}$. We therefore might guess that

$$F(n) = a\phi^n + b\bar{\phi}^n$$

for constants a and b. Solving for a and b by plugging cases $F(0)$, $F(1)$, and so forth yields

$$F(n) = \frac{1}{\sqrt{5}}\left(\phi^n - \bar{\phi}^n\right)$$

The above formula is called *Binet's formula*, and is the closed form for the nth Fibonacci number.

Some caveats: if $p_T(x)$ has repeated roots (for example, the root x_i has a multiplicity of j), then the solution must take the form

$$T(n) = c_1 x_i^n + c_2 n x_i^n + \cdots c_j n^{j-1} x_i^n$$

for that root. If some solutions are complex numbers, then we must take their exponential form $x_j = r_j e^{i\theta_j} = r_j(\cos\theta_j + i\sin\theta_j)$ to substitute into the equation so that

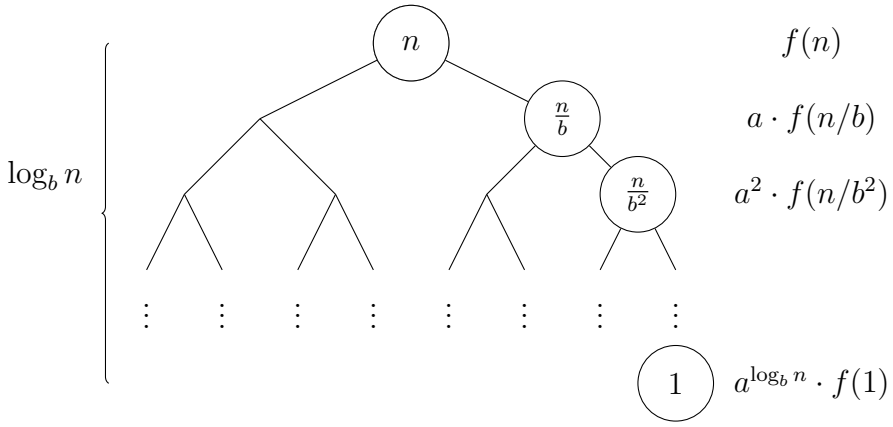
$$T(n) = r_j^n(c_1\cos(n\theta_j) + c_2\sin(n\theta_j))$$

3.5.2 Divide-and-Conquer Recurrences

Divide and conquer recurrences take the form

$$T(n) = aT\left(\left\lfloor\frac{n}{b}\right\rfloor\right) + f(n)$$

We can visualize this in a tree structure:



The recurrence at n is the total of the "work" spread out across this tree, such that

$$T(n) = \sum_{i=0}^{\log_b n} a^i f\left(\frac{n}{b^i}\right)$$

which we can bound asymptotically by using summing strategies and approximation bounds.

3.5.3 Master Theorem

The master theorem divides (applicable) divide-and-conquer recurrences into three cases, dependent on $f(n)$. We define $c^\star = \log_b a$.

1. If $f(n) \in O(n^{c^\star - \epsilon})$ for some $\epsilon > 0$, then $T(n) \in \Theta(n^{c^\star})$
2. If $f(n) \in \Theta(n^{c^\star})$ then $T(n) \in \Theta(n^{c^\star})$
3. If $f(n) \in \Omega(n^{c^\star + \epsilon})$ for some $\epsilon > 0$ and $af(n/b) < cf(n)$ for $c \in (0, 1)$, then $T(n) \in \Theta(f(n))$

These three cases correspond to when $f(n)$ either overpowers, is drowned out by, or matches the rate of growth of the recurrence itself, often visualized in a recurrence tree.

4 Number Theory

Number theory focuses on studying the set of integers, equipped with multiplication and addition.

4.1 Divisibility

An integer a is said to divide b if there exists some integer constant k such that $b = ak$. This is denoted "$a|b$".

$$a|b \iff \exists k \in \mathbb{Z} \; s.t. \; a \cdot k = b$$

We have some properties:

- If $a|b$ and $b|c$, then $a|c$.
- If $a|b$, then $a|bc$ for some c.
- If $a|b$ and $a|c$, then $a|sb + tc$ for some integers s and t.

The quantity $sb + tc$ is called an *integer linear combination* (ILC) of b and c.

4.1.1 Greatest Common Divisor

The common divisor of a and b is a shared factor between the two: an integer d such that $d|a$ and $d|b$. The greatest common divisor $\gcd(a, b)$ is the largest such shared factor (except $\gcd(0, 0)$, which is defined to be 0). a and b are said to be *coprime* if $\gcd(a, b) = 1$.

The following are true for all integers:

$$\gcd(a, 0) = |a| \tag{1}$$
$$\gcd(a, b) = \gcd(a, b - a) \tag{2}$$

The second statement follows from the proof that the set of common divisors of a, b is exactly equal to the set of common divisors of $a, b - a$. In addition, for any integer n and $d > 0$, there is a unique (q, r) pair with $0 \leq r < d$ such that

$$a = qb + r$$

Essentially, this is the statement that any number can be partitioned uniquely with respect to another number into a residue (r) and quotient (q). We say that $r = a \text{ rem } b$ and $q = a \text{ div } b$.

Even further, we recognize that we can apply statement (2) to a and b multiple times to obtain

$$\gcd(a, b) = \gcd(a, b \text{ rem } a) = \gcd(a \text{ rem } b, b)$$

These facts give rise to Euclid's algorithm for finding $\gcd(a, b)$:

1. If either $a = 0$ or $b = 0$, return the magnitude of the non-zero argument (from statement (1)).
2. Otherwise, recursively find $\gcd(a, b \text{ rem } a)$ (for $b > a$).

This algorithm maintains the invariant that any reachable state (x, y) from starting numbers (a, b), $\gcd(x, y) = \gcd(a, b)$. The algorithm additionally terminates because the arguments to successive recursive calls get *smaller*: for any (x, y), $x + y \text{ rem } x \leq x + y$.

Bezout's identity says that $\gcd(a, b)$ is an ILC of a and b:

$$\exists s, t \ s.t. \ gcd(a, b) = sa + tb$$

Furthermore, $\gcd(a, b)$ is the *smallest positive integer* that can be written as an ILC of a and b.

An integer n can be written as an ILC of a and b iff it is a multiple of $\gcd(a, b)$.

4.2 Modular Arithmetic

Modular arithmetic examines congruences between numbers, up to remainders (relative to some defined base n). In particular,

$$a \equiv_n b \iff n|(a-b)$$

so a and b are the same "distance" from any multiple of n.

$$a \equiv_n (a \text{ rem } n)$$

All possible remainders from dividing by n (the numbers $0, \cdots, n-1$) each define *equivalence classes* over this residual relation.

$$a \equiv_n b \iff a \text{ rem } n = b \text{ rem } n$$

Notationally, a rem n is commonly written as a mod n — they both refer to the (positive) residual quantity when as many multiples of n are subtracted from a as possible.

If $a \equiv_n b$, then for any c,

1. $a + c \equiv_n b + c$
2. $ac \equiv_n bc$
3. $a - c \equiv_n b - c$
4. $c - a \equiv_n c - b$
5. $a^c \equiv_n b^c$ (for $c \geq 1$)

We also have notions of *multiplicative inverses* for a with respect to n. A multiplicative inverse is described as some number a^{-1} such that

$$a \cdot a^{-1} \equiv_n 1$$

However, there are required conditions for such an a^{-1} to exist.

$$\exists a^{-1} \iff \gcd(a, n) = 1$$

This statement is proven by rewriting

$$a \cdot a^{-1} \equiv_n 1 \implies a \cdot a^{-1} - 1 = nk \implies a \cdot a^{-1} - nk = 1$$

so 1 is an ILC of n and a. As any ILC of n and a must be divisible by $\gcd(a, n)$, and $\gcd(a, n)$ is the smallest positive number that can be written as an ILC, we must conclude that $\gcd(a, n) = 1$.

Furthermore, if n is prime, $\gcd(a, n) = 1$ if a is not itself a direct multiple of n. For any prime p,

$$a \not\equiv_p 0 \implies \exists a^{-1} \; s.t. \; a^{-1}a \equiv_p 1$$

This leads to Fermat's Little Theorem, which states that

$$a \not\equiv_p 0 \implies \exists a^{-1} \; s.t. \; a^{p-1} \equiv_p 1$$

for some prime p.

4.3 RSA Cryptography

The RSA cryptosystem is a public-key cryptographic scheme whose security relies on the hardness of prime factorization.

The scheme begins with two primes p and q which are kept secret. We define a number $n = pq$, which is published as a public key alongside a larger number e coprime to $(p-1)(q-1)$.

$$k_p = (n, e)$$

We then compute d as the multiplicative inverse of e with respect to $(p-1)(q-1)$, such that

$$de \equiv_{(p-1)(q-1)} 1$$

d is a *secret key*:

$$k_s = (n, d)$$

Now, we take a message m. To transmit, m is encrypted with the public key k_p:

$$E(m, k_p) = m^e \text{ rem } n = c$$

Now, to decrypt c, we compute

$$D(c, k_s) = c^d \text{ rem } n$$

The claim, then, is that

$$(m^e)^d = m^{ed} \equiv_n m$$

This is a consequence of Fermat's little theorem. We first know that

$$ed = 1 + k(p-1)(q-1)$$

for some integer k, since d is the modular inverse of e with respect to $(p-1)(q-1)$. Then,

$$m^{ed} = m^{1+k(p-1)(q-1)} = m \cdot (m^{p-1})^{k(p-1)}$$

From Fermat's Little Theorem, we know that

$$m^{p-1} \equiv_p 1$$

provided that m is not a multiple of p. Then,

$$m \cdot (m^{p-1})^{k(q-1)} \equiv_p m \cdot 1^{k(q-1)} \equiv_p m \cdot 1 \equiv_p m$$

so

$$m^{ed} \equiv_p m$$

By symmetry, we can perform the same process with q. Therefore,

$$p|(m^{ed} - m) \wedge q|(m^{ed} - m)$$

Since p and q are both prime, then, the only way for both of them to divide $m^{ed} - m$ is for the quantity to contain both p and q in its prime factorization. Therefore,

$$pq|(m^{ed} - m) \implies m^{ed} - m \equiv_{pq} 0 \implies m^{ed} \equiv_{pq} m$$

4.4 Chinese Remainder Theorem

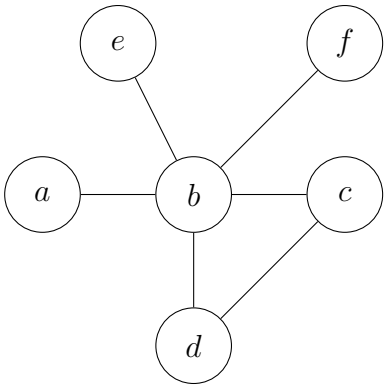
If p and q are coprime, there exists a unique $x,\ 0 \leq x < pq$, for a pair $a, b \in \mathbb{Z}$ such that

$$x \equiv_p a,\ x \equiv_q b$$

If there are two different integer solutions to the congruences x_1 and x_2, then they are equivalent modulo pq.

5 Graphs

A graph is a mathematical structure that consists of a set of *vertices* V (otherwise called nodes) and *edges* E, which connect between vertices. For example,



$$G = (V, E)$$
$$V = \{a, b, c, d, e, f\}$$
$$E = \{\{a, b\}, \{b, c\}, \{b, d\}, \{b, e\}, \{b, f\}, \{c, d\}\}$$

The graph above is an *undirected* graph, meaning that the edges are bidirectional (and, for example, the edge $\{b, c\} = \{c, b\}$). We can define related constructs with directed edges; these are termed digraphs.

This class only considers *simple graphs*, meaning that there are no self-loops from a vertex to itself and there are no duplicate/parallel edges.

Graphs are primarily used to map out relationships between distinct entities or states. In fact, a state machine can be thought of as a type of graph, equipped with a defined starting point.

Two vertices u and v are said to be *adjacent* if there exists an edge between them ($\{u, v\} \in E$). An edge $\{u, v\}$ is *incident* to its endpoints u and v.

The degree of a vertex $\deg(v)$ is the number of edges incident to that vertex.

$$\deg(v) = |\{\{u, v\} \in E\}|$$

The Handshake lemma for undirected graphs states that

$$\sum_{v \in V} \deg(v) = 2|E|$$

Intuitively, summing the degrees over all of the vertices in a graph double-counts each *relation* (edge) between two vertices. The upshot of this lemma is that total sum over all degrees in an undirected graph must be even; a degree set of, say, $(2, 2, 2, 1)$ is impossible.

5.1 Coloring

To color a graph is to assign color labels to each vertex in the graph such that no edge connects two vertices with the same color. In particular, a graph is said to be k-colorable if there is a way to partition V into at most k subsets such that no edge connects a vertex in one subset to another vertex in that same subset — we call these partitions a k-coloring.

Formally, a proper k-coloring of $G = (V, E)$ is a function $f : V \to C$ such that $|C| \leq k$, and

$$\forall \{u, v\} \in E,\ f(u) \neq f(v)$$

The chromatic number $\chi(G)$ is the smallest number of colors required to produce a valid coloring of G. In general, proving that a graph has a particular chromatic number requires the demonstration of an upper bound and a lower bound — if we wanted to prove that some $\chi(G)$ must be some number k, we must demonstrate

1. G is k-colorable in the first place — produce a k-coloring of G.
2. G cannot be colored in less than k colors — this is often proven by contradiction.

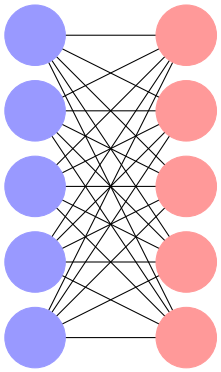
5.1.1 Bipartite Graphs

A *bipartite* graph is a graph whose vertices can be exactly partitioned into two sets such that every edge has exactly one endpoint in each set.

$$V = L \sqcup R,\ E = \{\{\ell, r\} \mid \ell \in L,\ r \in R\}$$

In particular, no edge can connect two vertices in the same subset. For bipartite graphs in particular, we can specify the Handshake lemma further:

$$\sum_{\ell \in L} \deg(\ell) = \sum_{r \in R} \deg(r) = |E|$$



Every 2-colorable graph is a bipartite graph by definition.

5.1.2 Approximation

In general, graph coloring is a hard (in fact, NP-complete) problem. However, it is possible to approximate an optimal solution by using the following greedy algorithm:

1. Order the vertices $v_1, v_2, \cdots, v_n$ in the order that they will be processed.
2. Order the colors $c_1, c_2, \cdots$.
3. For each vertex in the defined ordering, assign it the smallest legal color in the color sequence. If there are no such legal colors, add a new color to the set.

This algorithm uses at most $d + 1$ colors, for $d = \max_{v \in V}(\deg(v))$. This can be proven by induction over the number of vertices in the graph.

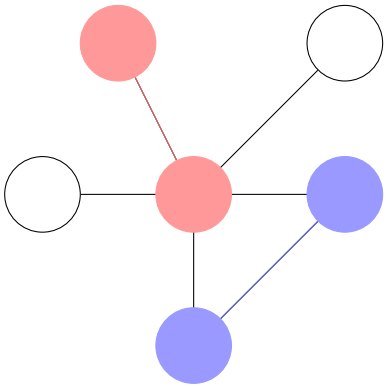
Let $P(n)$ = "The greedy algorithm produces a coloring of at most $d+1$ colors for every graph with n vertices and maximum degree d."

1. $P(1)$: A 1-vertex graph has no edges, so its maximum degree is 0. Furthermore, we can assign a single color to the singular vertex in the graph, so we use $d + 1 = 0 + 1 = 1$ colors.

2. $P(n) \implies P(n+1)$: Suppose $P(n)$ is true. The algorithm, when run on a graph with vertices $v_1, \cdots, v_n$ with a maximum degree of d, will produce a coloring using at most $d+1$ colors. Now, take any graph with vertices $v_1, \cdots, v_n, v_{n+1}$. We can color the subgraph containing only the vertices $v_1, \cdots, v_n$ with $d + 1$ colors (ignoring the last vertex, v_{n+1}) as per the inductive hypothesis. Then, when we add v_{n+1} back in, we know that its degree must be at most d, so it has at most d neighbors. However, we have $d + 1$ colors available to use, so we may still color v_{n+1} with a color that is not assigned to any of v_{n+1}'s neighbors. This produces a valid coloring, and we maintain the upper bound of $d + 1$ colors. Therefore, $P(n) \implies P(n + 1)$.

5.2 Matching

A matching in a graph G is a subgraph M of G in which every vertex has a degree of exactly 1. M consists of direct pairs of vertices that are connected by edges in G, and no vertex can be in two pairs at once. For example, the colored vertices in the graph below form a matching:



A *maximal* matching in G is a matching that exhausts all possible ways

to pair the vertices up while still being a valid matching. That is, it is not a subset of any other matching M' of G.

M is a *maximum* matching if it is the most comprehensive matching that can be made in G, such that $|M'| \leq |M|$ for any other matching M'.

M is a *perfect* matching if it uses all of the vertices in G. Equivalently, the matching is perfect if it contains $\frac{|V|}{2}$ edges.

Often, the graph can be modified to include weights on each of the edges. This induces an optimization problem: namely, finding a matching that minimizes (or maximizes) the total sum of the weights of the edges taken in the matching. This is known as the *minimum weight matching problem.*

5.2.1 Stability

The matching problem is often used to solve problems relating to ranked-preference matching between the nodes of a graph. Each node has a permuted list of the other vertices it is connected to; if it is matched with their last choice, it is unhappiest. Preferences are not necessarily symmetric.

In this context, a rogue couple in a matching M is a pair (x, y) such that $(x, y) \neq M$ but x and y prefer each other to their respective partners in M. M is a *stable matching* iff it has no rogue couples. The Gale-Shapley algorithm produces such a matching in a bipartite graph with n members of each partition.

5.3 Connectivity

A walk over a graph is a sequence of vertices $(v_0, v_1, \cdots, v_k)$ that follows valid edges in a graph (i.e. $\{v_i, v_{i+1}\} \in E \forall i$). The length of a walk is described as the number of edges it traverses. A *trail* is a walk that does not repeat edges, and a *path* is a walk that does not repeat edges or vertices.

A walk is *closed* if it starts and ends at the same vertex. A *tour* is a closed trail, and a *cycle* is a closed path (i.e. does not repeat vertices, except at its endpoints). A graph is said to be *Eulerian* if there exists a tour that uses every edge exactly once. As it turns out, a graph G is Eulerian iff all of the degrees of its vertices are even.

Two vertices $u, v \in V$ are *connected* iff there exists a walk in G from u to v. If edges are undirected, the existence of a path from u to v is equivalent to the existence of a path from v to u. A graph is (pairwise) *connected* if all pairs of vertices in the graph are connected. More specifically, a *connected*

component of a graph is a subgraph in which all vertices are connected. The connected component of a vertex v is the subgraph induced by set of vertices reachable from v. All of the connected components of a graph form a partition of V.

5.4 Directed Graphs

A directed graph (or digraph) is a graph in which edges are directed: an edge $(u, v) \in E$ is not equivalent to the edge (v, u). Instead of a unified notion of degree, digraphs have notions of in-degrees (the number of incoming edges) and out-degrees (the number of outgoing edges) respectively.

$$\deg^-(v) = |\{(u, v) \in E\}|$$

$$\deg^+(v) = |\{(v, u) \in E\}|$$

The Handshake lemma for digraphs states

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

The notions of walks remain the same, just with directions required for each edge traversal.

Notably, in digraphs, the existence of a path from u to v does not guarantee the existence of a path from v to u. Instead, u and v are said to be *strongly connected* if v can reach u and u can reach v. G is strongly connected if every pair of vertices is strongly connected to each other.

A strongly connected component (SCC) of a vertex v is the subgraph induced by taking all vertices u that are strongly connected to v. We notate this component as $[v]$.

The condensation graph G^C of G collapses all of the vertices in a single SCC into one vertex, and maps out directional relations between SCCs.

$$\begin{aligned} G^C &= (C, E') \\ C &= \{[v] : v \in V\} \\ E' &= \{([u], [v]) : [u] \neq [v], (u, v) \in E\} \end{aligned}$$

5.4.1 DAGs

A directed acyclic graph (DAG) is a digraph that has no cycles. A source in a digraph is a vertex that has incoming edges, and a sink is a vertex that has no outgoing edges. A topological ordering of a DAG is a permutation of its vertices such that every vertex appears earlier in the permutation than any other vertex reachable from it.

A topological ordering is a partial ordering of the vertices in a DAG, as it ensures that the relationship between all *directly comparable* vertices is preserved in the ordering, while not necessarily caring about the ordering between non-comparable vertices. u and v are said to be comparable if v is reachable from u, or vice versa. A *chain* is a set of vertices in which every pair is comparable. An *antichain* is a set of vertices in which every pair is *not* comparable.

6 Probability

6.1 Events and Probability Spaces

A *probability space* is a set $\mathcal{S}$ (called the *sample space*) equipped with a probability measure Pr.

$$\Pr : \mathcal{S} \to [0, 1]$$

is a total function that must satisfy

$$\sum_{\omega \in \mathcal{S}} \Pr[\omega] = 1$$

for elements ω. In addition, of $\mathcal{S}$ is countable, then

$$\Pr[\{\omega_1, \omega_2, \cdots, \omega_n\}] = \Pr[\omega_1] + \Pr[\omega_2] + \cdots + \Pr[\omega_n]$$

We can also define probabilities of *events* $A \subseteq \mathcal{S}$.

$$\Pr[A] := \sum_{\omega \in A} \Pr[\omega]$$

We define the *complement* of A, $\overline{A}$, as all elements ω that are not in A, such that

$$\Pr[\overline{A}] = 1 - \Pr[A]$$

A few other rules follow.

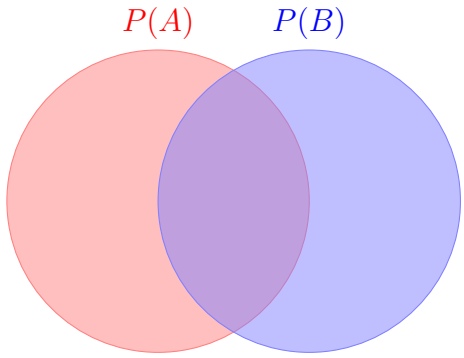
$$\begin{aligned}
\Pr[A \sqcup B] &= \Pr[A] + \Pr[B] \\
\Pr[A \setminus B] &= \Pr[A] - \Pr[A \cap B] \\
\Pr[A \cup B] &= \Pr[A] + \Pr[B] - \Pr[A \cap B] \\
&\leq \Pr[A] + \Pr[B] \\
A \subseteq B &\implies \Pr[A] \leq \Pr[B]
\end{aligned}$$

In general, the inclusion-exclusion principle applies as well. For events $E_1, \cdots, E_n$,

$$\begin{aligned}
P\left(\bigcup_{i=1}^{n} E_i\right) = &\sum_i P(E_i) - \sum_{i_1 < i_2} P(E_{i_1} E_{i_2}) + \cdots \\
&+ (-1)^{r+1} \sum_{i_1 < \cdots < i_r} P(E_{i_1} E_{i_2} \cdots E_{i_r}) + \cdots \\
&+ (-1)^{n+1} P(E_1 E_2 \cdots E_n)
\end{aligned}$$

6.2 Conditional Probability

Conditional probability asks about the probability that some event A occurs, *given that* another event B has already occurred. The effect of knowing that Bhas occurred is that it *shrinks* the possible sample space from the full set $\mathcal{S}$ to the set B, the set of all possible outcomes for which B is true.



For example, conditioning A on the event B, for example, would limit our probability sample space to the blue circle; the probability that A happens conditioned on B is the fraction of the blue circle that is occupied by the overlap with the red circle.

We express this as

$$\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]}$$

This also gives us a product rule for joint probabilities.

$$\Pr[A \cap B] = \Pr[A|B] \Pr[B]$$

By extension,

$$\begin{aligned} \Pr[A \cap B \cap C] &= \Pr[A|B \cap C] \Pr[B|C] \Pr[C] \\ \Pr[A \cap B|C] &= \Pr[A|B \cap C] \Pr[B|C] \end{aligned}$$

A and B are independent events if conditioning them on each other does not change their probabilities.

$$\Pr[A|B] = \Pr[A] \iff \Pr[B|A] = \Pr[B] \iff \Pr[A \cap B] = \Pr[A] \Pr[B]$$

6.2.1 Bayes' Rule

Bayes' rule substitutes the product rule back in to the definition of conditional probability to give us an equation for flipping conditions.

$$\Pr[B|A] = \frac{\Pr[A|B] \Pr[B]}{\Pr[A]}$$

For multiple events $B,\ C$ conditioned on A,

$$\frac{\Pr[B|A]}{\Pr[C|A]} = \frac{\Pr[A|B] \Pr[B]}{\Pr[A|C] \Pr[C]}$$

6.3 Random Variables

A random variable is a total function from the sample space to the real numbers.

$$X : \mathcal{S} \to \mathbb{R}$$

In general, they are used to assign some sort of "value" to events or outcomes in a probability space. We can also flip this perspective, and conceptualize *events themselves* as the probability that a random variable reaches a certain

value. For example, if we define the event to be all outcomes such that the random variable X is less than some value x:

$$\Pr[X \leq x] = \sum_{y \leq x} \Pr[X = y]$$

We can condition random variables on each other in a similar fashion:

$$\Pr[X = x | Y = y] = \frac{\Pr[X = x \cap Y = y]}{\Pr[Y = y]}$$

X and Y are *independent* if they are independent for all values in their range.

$$\forall x \forall y \Pr[X = x \cap Y = y] = \Pr[X = x] \Pr[Y = y]$$

6.3.1 Distributions

For a *discrete* random variable (i.e. a random variable whose outputs only take a discrete/countable set of values), we define a *probability mass function* (pdf)

$$f_X(x) = \Pr[X = x]$$

and a *cumulative distribution function* (cdf)

$$F_X(x) = \sum_{y \leq x} \Pr[X = y]$$

These functions describe how the values X takes on are spread across their domain. There a few special distributions:

Uniform

A uniform random variable on the set $\{1, \cdots, n\}$ has the cdf

$$f(i) = \frac{1}{n}, \; F(i) = \frac{i}{n}$$

for $i \in \{1, \cdots, n\}$.

Bernoulli

A Bernoulli random variable, also known as an *indicator* random variable, takes on the value 1 with probability p and 0 with probability $1 - p$, with p defined according to some event that we want to "indicate" (i.e. a weighted coin that flips heads with probability p).

$$f(0) = 1 - p, \;\; f(1) = p$$

$$F(0) = p, \;\; F(1) = 1$$

Binomial

A binomial random variable models n trials of a Bernoulli random variable (i.e. n weighted coin flips).

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

$$F(k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$$

6.4 Expectation and Variance

Given a random variable, we can define some notion of the "average value" it takes on across its domain, with each value weighted by the probability that the random variable takes it on. For a random variable X,

$$E[X] := \sum_{\omega \in \mathcal{S}} X(\omega) \Pr[\omega]$$

For example, the expectation of an indicator random variable is:

$$0 \cdot (1-p) + 1 \cdot p = p$$

The definition of expectation can also be phrased as

$$E[X] = \sum_{x \in \text{range}(X)} x \cdot \Pr[X = x]$$

Expectation also follows *linearity*, regardless of the relationship between the random variables in question. That is,

$$E[aX + bY] = aE[X] + bE[Y]$$

for constants $a,\ b$.

The variance of a random variable measures how much the random variable tends to deviate from its mean.

$$\text{Var}[X] := E[(X - E[X])^2]$$

Through some algebraic manipulations, we see that this is equal to

$$\text{Var}[X] = E[X^2] - E[X]^2$$

The *standard deviation* of X is the square root of its variance.

$$\text{SD}[X] = \sqrt{\text{Var}[X]}$$

We can define a similar notion for the relationship between two variables X and Y, capturing how much they tend to "move together". This is called the *covariance*.

$$\text{Cov}(X, Y) := E[(X - E[x])(Y - E[Y])] = E[XY] - E[X]E[Y]$$

Note that $\text{Cov}(X, X) = \text{Var}[X]$.]

If $X_1, X_2, \cdots, X_n$ are pairwise independent random variables,

$$\text{Var}[X_1 + X_2 + \cdots + X_n] = \text{Var}[X_1] + \text{Var}[X_2] + \cdots + \text{Var}[X_n]$$

6.5 Deviation Bounds

We can come up with bounds for how much a random variable is expected to deviate from its mean. Markov's inequality states that, for a non-negative random variable X,

$$\Pr[X \geq x] \leq \frac{E[X]}{x}$$

or otherwise:

$$\Pr[X \geq cE[X]] \leq \frac{1}{c}$$

Chebyshev's inequality, which holds for all random variables, states that, for a non-negative x,

$$\Pr[|X - E[X]| \geq x] \leq \frac{\text{Var}[X]}{x^2} = \left(\frac{\text{SD}[X]}{x}\right)^2$$

Alternatively,

$$\Pr[|X - E[X]| \geq c\text{SD}[X]] \leq \frac{1}{c^2}$$

The Chernoff bound is a statement about the *sums of independent random variables*. Let $T = T_1 + T_2 + \cdots + T_n$, for mutually independent $T_1, \cdots, T_n$. Then, $\forall c \geq 1$,

$$\Pr[T \geq cE[T]] \leq e^{-(c \ln c - c + 1)E[T]}$$