CS5275 Lecture 1: Basic Tools

Jonathan Scarlett

February 21, 2025

Useful references:

- "Methods of Proof" blog posts: `https://www.jeremykun.com/primers/`
- CMU lecture on Big-O: `https://www.youtube.com/watch?v=_gKb855_3bk`
- CMU lecture on online resources: `https://www.youtube.com/watch?v=qP4XEZ54eSc`
- Inequality sheet: `https://www.lkozma.net/inequalities_cheat_sheet/ineq.pdf`

Note on examination: Nothing in this lecture will be examined *directly*, but many of the tools/concepts will be used when studying other topics.

1 General Background

This lecture gives some very broad background, much of which might already be familiar. After this one, the subsequent lectures will be more focused and specific.

The main assumed CS background for this course is:

- Algorithms and data structures
- Some exposure to tools like optimization, graphs, etc.
- Some exposure to computational complexity

The main assumed mathematical background for this course is:

- Probability *(see background document)*
- Linear algebra *(see background document)*
- Basic calculus
- Some exposure to tools like Taylor expansion, limits, binomial coefficients, etc.

2 Proof Techniques

Some of the most common proof techniques are outlined as follows with one example each.

Proof technique 1: Direct proof

- Idea: Just find a sequence of steps that leads directly from the assumptions to the desired statement
- Often this is done via a (possibly long) sequence of steps, e.g.:
  - To show $A \implies B$, show $A \implies S_1$, then $S_1 \implies S_2$, etc., up to $S_{k-1} \implies S_k$ and $S_k \implies B$
  - To show $a \le b$, just show $a \le \dots \le b$ (or similarly for $a = b$ or $a \ge b$)
- Example: Consider the claim "The product of 3 consecutive positive integers is always a multiple of 6." We can simply combine the following direct observations:
  - Fix $n \ge 1$. Since one (or two) of $n, n+1, n+2$ must be even, we find that $n(n+1)(n+2)$ is an even number (i.e., a multiple of 2).
  - Similarly, exactly one of $n, n+1, n+2$ is a multiple of 3, so $n(n+1)(n+2)$ is a multiple of 3.
  - Since $n(n+1)(n+2)$ is a multiple of both 2 and 3, it is a multiple of 6.

Proof technique 2: Contrapositive

- Idea: To show something of the form $A \implies B$, show the equivalent statement $B^c \implies A^c$, where $(\cdot)^c$ means the complement ("not").
- Example:
  - Consider the claim "$\sqrt{n}$ is irrational for any non-square positive integer n".
  - We can re-word this into "$A \times B$" form as follows: "For any positive integer n, if n is not a square number, then $\sqrt{n}$ is irrational."
  - The contrapositive statement is "For any positive integer n, if $\sqrt{n}$ is rational, then n is a square number". This turns out to be more convenient to prove.
  - The details:
    - ∗ If $\sqrt{n}$ is rational, then there must exist integers a, b such that $\sqrt{n} = \frac{a}{b}$, or equivalently $nb^2 = a^2$.
    - ∗ Now factor a, b, n into a product of primes (uniquely). Since both a^2 and b^2 have every prime appearing an even number of times, the same must be true of n. Thus, n is a square number.

Proof technique 3: Contradiction

- Idea: To show that a claim is true, start by assuming that it is false, and show that this leads to a mathematical contradiction.
  - Note: This is not the same as the contrapositive technique (in which nothing contradictory is assumed), and it can be applied to other statements beyond those of the form "$A \implies B$"

- Example:
  - Consider proving the claim "There are infinitely many prime numbers" (many proofs are known!)
  - Proceed with a proof by contradiction, assuming that there are only finitely many prime numbers. That is, there exists a finite N such that the prime numbers are $p_1, \dots, p_N$.
  - Define $M = p_1 \times p_2 \times \dots \times p_M$, and consider the integer $M+1$. Since every integer can be written as a product of primes, there must exist some p_i that is a factor of $M+1$.
  - But p_i is also trivially a factor of M.
  - Being a factor of both M and $M+1$ is impossible, since $p_i \geq 2$ – a contradiction.

Proof technique 4: Induction

- Idea: To prove that a claim C_n holds for all integers $n \geq n_0$ (where n_0 is typically 0 or 1), first prove C_{n_0} (base case) and then prove that $C_n \implies C_{n+1}$ (induction step).
- Example: In the Towers of Hanoi problem, there are n disks stacked from largest (bottom) to smallest (top):

  The goal is to move all the disks to the right-hand side pole, but only one ring can be moved at a time, and *a larger disk can never be above a smaller disk*.

- Claim: This problem can be solved in $2^n - 1$ moves (in fact this is optimal, but we won't prove that)
- Proof:
  - The base case is $n = 1$, in which case $2^n - 1 = 1$ and the result is trivial.
  - For the induction step, we need to show that if $2^n - 1$ moves suffices with n disks, then $2^{n+1} - 1$ moves suffices with $n+1$ disks.
  - To see this, we perform 3 steps:
    * Use the n-disk method to move the n smallest disks to the middle pole ($2^n - 1$ moves)
    * Move the largest disk to the right pole (1 move)
    * Use the n-disk method to move the n smallest disks to the right pole ($2^n - 1$ moves)

    The total number of moves is $2(2^n - 1) + 1 = 2^{n+1} - 1$, as desired.

Other proof techniques:

- Proof by cases (a special case being proof by exhaustion)
- Proof by example for "there exists..." statements (or by counter-example for "not all..." statements)
- The probabilistic method (which we will cover later)

- Visual proofs
- Proof of equality by showing $\geq$ and $\leq$ separately
- Other logic-based proofs (e.g., to show $A \implies (B_1 \text{ or } B_2)$, show that when A is true, $B_1^c \implies B_2$)

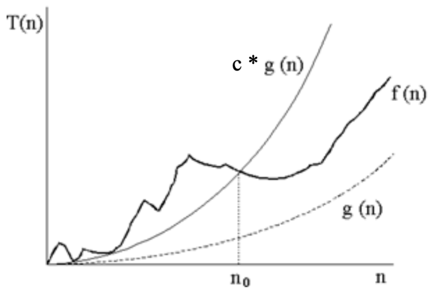
Errors in proofs

- There are endless ways that proofs can contain errors (even when the final result is correct), and they sometimes enter in very subtle or unexpected ways.
- Example 1: It is easy to accidentally take $(a^b)^c$ and a^{bc} to be the same. This is not always true!
  - Particular care should be taken with complex numbers, e.g.:, for any $\theta \in \mathbb{R}$ we know that $e^{i\theta} = \cos\theta + i\sin\theta$, whereas
  $$(e^{2\pi i})^{\frac{\theta}{2\pi}} = 1^{\frac{\theta}{2\pi}} = 1$$
  which is clearly different from $\sin\theta + i\cos\theta$ except for some very specific θ values.
  - Even without complex numbers, $a < 0$ can cause similar issues (e.g., $a = -1$, $b = 2$, $c = \frac{1}{2}$)
  - On the other hand, if a, b, c are all real-valued and $a > 0$, then indeed $(a^b)^c = a^{bc}$.
  - (Note: This problem could be alleviated by treating both ± 1 as square roots of 1, or more generally taking $e^{2\pi i/n}$ to be an n-th root of 1 for every $i = 1, \dots, n$.)
- Example 2: Consider a setup where we repeatedly roll a 6-sided die until we observe a 6. What's the average number of rolls (including the roll that gave 6) conditioned on only seeing even numbers?
  - Incorrect intuition: 3, because without even numbers we have $\{2, 4, 6\}$ being equally likely. *(This intuition would be correct if we were to condition on an* ***infinite*** *sequence of rolls containing no even numbers.)*
  - Correct intuition: 1.5, because we can think of this as repeatedly rolling until we see anything except $\{2, 4\}$ (and if it's odd, we reset the experiment). Hence, the stopping time has a Geometric(2/3) distribution, whose mean is 1.5.
  - (This isn't a rigorous argument, but a formal proof via the definition of conditional probability is also possible, and gives 1.5.)

3 Asymptotic Notation

Big-O notation:

- For two real-valued sequences f_n and g_n indexed by $n \in \{1, 2, 3, \dots\}$, we say that $f_n = O(g_n)$ if there exist constants $C > 0$ and $n_0 > 0$ such that $|f_n| \leq C g_n$ for all $n \geq n_0$.
  - More concisely: f_n is upper bounded by a constant times g_n when n is large enough
  - An illustration (from UWash CSE373 slides):
- Notes on notation:



- Technically $O(\cdot)$ is a *set of sequences*, so notation like $f_n \in O(g_n)$ would be more precise. But more often we just write $f_n = O(g_n)$.
- We also tend to combine other operations with $O(\cdot)$, e.g., writing things like $f_n = n^2 + e^{O(\sqrt{\log n})}$. It can get even more confusing when there are multiple variables involved (more on this below).

- Other limits: We have introduced $O(\cdot)$ with respect to asymptotics in the limit of a parameter $n \to \infty$, and we will mostly focus on this below. But the notation is also used with respect to other limits, such as some $\epsilon \to 0$ (e.g., $f(\epsilon) = 1 + \epsilon + O(\epsilon^2)$). The context should always make the precise limit clear, otherwise it should be stated explicitly.

Standard variations:

- (Ω) The statement $f_n = \Omega(g_n)$ is equivalent to $g_n = O(f_n)$. That is, f_n is lower bounded by a constant times g_n (when n is large enough).
- (Θ) The statement $f_n = \Theta(g_n)$ is equivalent to having both $f_n = O(g_n)$ and $g_n = O(f_n)$. That is, f_n and g_n coincide to within a constant factor.
- (o) The statement $f_n = o(g_n)$ is equivalent to $\lim_{n\to\infty} \frac{f_n}{g_n} = 0$. That is, f_n is strictly smaller than g_n asymptotically.
- (ω) The statement $f_n = \omega(g_n)$ is equivalent to $g_n = o(f_n)$. That is, f_n is strictly larger than g_n asymptotically.

Variations omitting log factors:

- The statement $f_n = \widetilde{O}(g_n)$ is equivalent to there existing some constant c such that $f_n = O(g_n(\log g_n)^c)$. This notion is useful when the goal is to get the "leading terms" while treating log factors as less significant.
  - For example, $n^2 \log^3 n = \widetilde{O}(n^2)$, and $2^n n^2 \log n = \widetilde{O}(2^n)$.
  - Statements like "$f_n = O(g_n(\log g_n)^c)$ for some c" are also often written as $f_n = O(g_n \text{poly}(\log g_n))$
- Note: While you can think of $\widetilde{O}(g_n)$ as "like $O(g_n)$ but ignoring log factors", it should always be remembered that this means factors that are *logarithmic in g_n, not necessarily in n.* For example:
  - $\widetilde{O}(n)$ and $\widetilde{O}(n^2)$ hide $(\log n)^c$ factors;
  - $\widetilde{O}(2^n)$ or $\widetilde{O}(e^n)$ hides n^c factors, because $\log_2(2^n) = \log_e(e^n) = n$.
  - $\widetilde{O}(\log n)$ is only allowed to hide $(\log \log n)^c$ factors.

  - $\widetilde{O}(1)$ makes no sense!

- We can similarly consider $\widetilde{\Omega}$ and $\widetilde{\Theta}$ (whereas $\widetilde{o}$ and $\widetilde{\omega}$ are not standard notions.)

Examples:

- We tend to state algorithm runtimes in $O(\cdot)$ notation, especially when the precise constants depend on implementation details (e.g., $O(n \log n)$ time for sorting).
- Big-O notation is often closely tied to Taylor expansions. For example, we can write $\sqrt{1+x} = 1+O(x)$ as $x \to 0$, or include higher-order terms like $\sqrt{1+x} = 1 + \frac{x}{2} + O(x^2)$ as $x \to 0$.
  - To highlight the importance of different limits, note that the limit $x \to \infty$ gives the completely different behavior of $\sqrt{1+x} = O(\sqrt{x})$.
  - To get a more precise expression as $x \to \infty$ we could also write $\sqrt{1+x} = \sqrt{x} \cdot \sqrt{1+1/x}$ and then use the above findings, e.g.,

$$\sqrt{1+x} = \sqrt{x} \cdot \left(1 + \frac{1}{2x} + O\Big(\frac{1}{x^2}\Big)\right) = \sqrt{x} + \frac{1}{2\sqrt{x}} + O(x^{-3/2}).$$

  - Note: Wolfram Alpha is a useful tool for getting Taylor expansions without doing them by hand
- *Stirling's approximation* is often written using asymptotic notation, e.g.,

$$\log(n!) = n \log n - n + O(\log n).$$

  This leads to similar sorts of statements for the binomial coefficients $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, e.g.:
  - (Coarse behavior) $\log \binom{n}{k} = O\big(k \log \frac{n}{k}\big)$
  - (Precise behavior when $k = \alpha n$ with $\alpha \in (0,1)$) $\log \binom{n}{k} = nH_2(\alpha)(1+o(1))$ where $H_2(\alpha) = \alpha \log \frac{1}{\alpha} + (1-\alpha) \log \frac{1}{1-\alpha}$ is the binary entropy function.
  - (Precise behavior when $k = o(n)$) $\log \binom{n}{k} = \big(k \log \frac{n}{k}\big)(1+o(1))$.
- We often use $o(\cdot)$ notation to hide "lower-order terms". For example, suppose that we want to show that some probability is exponentially decaying in n, and we get a bound of the form

$$\mathbb{P}[\text{event}] \le e^{-nC}\left(1 + O\Big(\frac{1}{n}\Big)\right) + e^{-n \log n}.$$

  The second term is $e^{-\omega(n)}$ (i.e., it decays faster than exponential). So we might summarize the above by the simpler statement

$$\mathbb{P}[\text{event}] \le e^{-nC}(1+o(1)).$$

  In contrast, note that if we started with the expression $e^{-nC(1+o(1))}$, we could *not* simplify it to $e^{-nC}(1+o(1))$. This is because $e^{-nC(1+o(1))} = e^{-nC} \times e^{o(n)}$, but $e^{o(n)}$ could still by very large or very small (e.g., n^{100} or n^{-100}, or something even more significant like $e^{\sqrt{n}}$ or $e^{-\sqrt{n}}$).

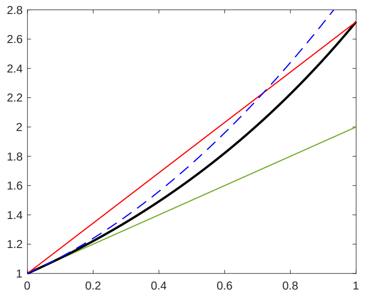
Cautionary notes:

- As evidenced by the last example above, we should be careful with non-linear functions, e.g., if $y = O(x)$ then we can say that $y^2 = O(x^2)$ but we *cannot* say that $e^y = O(e^x)$ (i.e., $e^{O(x)}$ and $O(e^x)$ are not equivalent).
  - Along similar lines, another common mistake is incorrect cancellations, e.g., if $f_n = e^{n+o(n)}$ and $g_n = e^{n+o(n)}$ then we have $\frac{f_n}{g_n} = e^{o(n)}$, but we cannot say something similar about $f_n - g_n$ (e.g., consider $f_n = 2e^n = e^{n+\ln 2}$ and $g_n = e^n$).
- If limits are being taken with respect to multiple variables, then $O(\cdot)$ notation could become unclear, because limits taken in different orders can give different results. I tend to think of it as follows: To make a statement like $f(m,\delta,\epsilon) = O\big(\frac{m}{\epsilon^2}\log\frac{1}{\delta}\big)$ with $m \to \infty$, $\delta \to 0$, and $\epsilon \to 0$, it should be the case that *any* sequence $\{(m_n,\delta_n,\epsilon_n)\}_{n=1}^{\infty}$ with $m_n \to \infty$, $\delta_n \to 0$ and $\epsilon_n \to 0$ should satisfy $f(m_n,\delta_n,\epsilon_n) = O\big(\frac{m_n}{\epsilon_n^2}\log\frac{1}{\delta_n}\big)$ as $n \to \infty$.

4 Inequalities

Inequalities are ubiquitous in theoretical computer science; some common ones are outlined as follows (see the sheet linked on p1 for many more):

- Basic inequalities, e.g., $1 + x \le e^x$ or equivalently $\log(1+x) \le x$
  - Even when there is no hope of analogous reverse inequalities that hold for all x (e.g., $1 + cx$ can never be a universal upper bound on e^x, no matter how large c is), we can often get those for a *restricted range of* x:



  - The bold curve is e^x, the solid straight lines are $1 + x$ and $1 + (e-1)x$, and the dashed curve is $1 + x + x^2$. Restricted to $x \in [0,1]$, the latter two are upper bounds on e^x.
- Cauchy-Schwarz inequality: $|\langle \mathbf{u},\mathbf{v}\rangle| \le \|\mathbf{u}\| \cdot \|\mathbf{v}\|$
  - Generalizes to Hölder's inequality: $|\langle \mathbf{u},\mathbf{v}\rangle| \le \|\mathbf{u}\|_p\|\mathbf{v}\|_q$ when $p,q \ge 1$ satisfy $\frac{1}{p} + \frac{1}{q} = 1$ (e.g., $|\langle \mathbf{u},\mathbf{v}\rangle| \le \|\mathbf{u}\|_1\|\mathbf{v}\|_\infty$). Here $\|\mathbf{u}\|_p = \big(\sum_i |u_i|^p\big)^{1/p}$ is the "p-norm", and $\|\mathbf{u}\|_\infty = \max_i |u_i|$.
  - Both generalize beyond real-valued vectors, e.g., $\mathbb{E}[|XY|] \le \big(\mathbb{E}[|X|^p]\big)^{1/p}\big(\mathbb{E}[|Y|^q]\big)^{1/q}$.
- Triangle inequality: $|a+b| \le |a| + |b|$, or for vectors $\|\mathbf{u}+\mathbf{v}\| \le \|\mathbf{u}\| + \|\mathbf{v}\|$.
- Bounds on binomial coefficient, e.g.,

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{ne}{k}\right)^k$$

(This is less precise than Stirling's approximation, but convenient due to being non-asymptotic.)

- Union bound (AKA Boole's inequality):

$$\mathbb{P}\left[\bigcup_{i=1}^{N} A_i\right] \le \sum_{i=1}^{n} \mathbb{P}[A_i].$$

- Arithmetic mean vs. geometric mean: $\left(\prod_{i=1}^{n} x_i\right)^{1/n} \le \frac{1}{n}\sum_{i=1}^{n} x_i$.

- Inequalities between norms, e.g., for $\mathbf{u} \in \mathbb{R}^n$:

$$\|\mathbf{u}\|_2 \le \|\mathbf{u}\|_1 \le \sqrt{n}\|\mathbf{u}\|_2$$
$$\|\mathbf{u}\|_\infty \le \|\mathbf{u}\|_2 \le \sqrt{n}\|\mathbf{u}\|_\infty.$$

  Note that $\|\mathbf{u}\|_1 = \sum_i |u_i|$ and $\|\mathbf{u}\|_\infty = \max_i |u_i|$, and $\|\mathbf{u}\|_2 = \sqrt{\sum_i u_i^2}$ is the regular Euclidean norm.

- We will not need many (if any) matrix inequalities in this course, but for an extensive list, search "Matrix Cookbook" in Google.

- Jensen's inequality: $f(\mathbb{E}[X]) \le \mathbb{E}[f(X)]$ for convex f (see the later lecture on convex optimization)

- Probabilistic bounds and concentration inequalities (next lecture): Markov, Chebyshev, Bernstein, Chernoff, Hoeffding, McDiarmid, etc.

5 Note on Computational Complexity

- This course will be fairly light on computational complexity (consider taking CS5230!), but here we pause to mention a model that is widely used *implicitly* and is worth seeing more explicitly at least once (though we will avoid giving a very formal description).

- Motivating example:

  - Suppose that we have a sorted list of n numbers, and let's say each of them as a positive integer in the range $\{1, 2, \dots, n^{10}\}$ (the number 10 here is arbitrary). What's the computational complexity of finding the first number to exceed a specified threshold γ? (e.g., in the list $\{1, 5, 9, 10, 53, 94, 1024, 1999\}$, find the first number greater than 100)
  - This can be solved using *binary search* – check the middle element, then move to the left half or right half depending on whether it exceeds γ, and continue recursively.
  - We recurse $O(\log n)$ times before narrowing down to a single element, so it's natural to say that the computational complexity is $O(\log n)$.
  - However, this raises the following concern: *Representing an integer in* $\{1, 2, \dots, n^{10}\}$ *requires* $O(\log n)$ *bits.* If we read those bits one-by-one to compare two numbers, and each bit takes a constant amount of time to read, then each comparison takes $O(\log n)$ time and thus the total time is $O((\log n)^2)$.

- The *word RAM model* overcomes this ambiguity by assuming that reading an *entire* number (and performing "basic" operations on them) in fact takes constant time, in which case the complexity is indeed $O(\log n)$ in the above example.
  - Examples of basic operations: Comparison, addition, bit-wise AND/OR/NOT, etc. (there is a formal class called AC^0 containing these)
  - Less basic operations like multiplication and division are also often considered to take constant time, though not always (as it is a bit more questionable than addition).
- It could be argued that it's strange to assume "words" can be stored and read in a manner that grows with the input size n, but this model is quite well-aligned with how modern computers work and the fact that 64-bit words suffice for most practical purposes. However, there are some subtle issues:
  - The word length w should be thought of being at least $\log n$ (so that we can at least index integers from 1 to n) but at most $O(\log n)$ (meaning each number can take one of at most $\text{poly}(n)$ values, like n^{10} in the example above). A word size $w = \omega(\log n)$ tends to be "disallowed", as this would mean being able to index super-polynomially many memory locations in constant time.
  - Strictly speaking a real-valued number requires infinitely many bits to store exactly, so extra care may be needed for algorithms that (conceptually) operate on real numbers. This can usually be ignored when the real numbers are being used in "reasonable" ways (e.g., we don't cheat by hiding information in the very insignificant bits of the real number).
  - Roughly speaking, if the algorithm works on "real" numbers, then they should be "sufficiently accurately" represented using $O(\log n)$ bits. This is usually a reasonable assumption in practice.

6 Useful Resources

For basic mathematical checks, calculations, etc., some useful resources are as follows:

- Wikipedia is a perfectly fine starting point for a concept you've never heard of (e.g., "Stirling numbers of the second kind" or "BPP complexity class"), though it might sometimes be lacking in detail and/or require more scrutiny compared to other sources.
- Wolfram Alpha (online) for basic integrals, Taylor expansions, limits, etc.
- Wolfram Mathematica (software) for more sophisticated calculations along similar lines
  - Maple is along similar lines.
  - MATLAB is common but usually used more for numerical computations.
- Inverse Symbolic Calculator (online) for looking up constants (e.g., try 0.57721566490), or similarly OEIS for integer sequences (e.g., try 1,1,2,5,14,42)
- Optimization libraries / packages (e.g., Gurobi, CPLEX, Yalmip, CVX)
- MathOverflow/StackOverflow for endless Q&A posts (e.g., try typing into Google "StackOverflow difference between NP and co-NP")
- If you find yourself having to search through research papers, clever exploration could save you time:

  - If the paper's result seems too complicated or "over-the-top" for your purposes, check the Related Work section (or reference list) for possible simplified versions done in earlier works
  - Conversely, if the paper's result isn't strong enough for your purposes, try searching the paper on Google Scholar, then clicking "Cited By", possibly followed by a search with "Search within citing articles" ticked.

See `https://www.youtube.com/watch?v=qP4XEZ54eSc` for a "Street Fighting Mathematics" lecture along these lines.

7 CS5275 Topics

The main topics we will cover are as follows:

- Concentration inequalities
- The probabilistic method
- Convexity and convex optimization
- Submodularity and discrete optimization
- Multiplicative weights algorithms
- Fourier transform
- Information theory
- Error-correcting codes
- Expander graphs
- Communication complexity

We will also have a lecture touching very briefly on "topics we didn't get to cover fully":

- Distance measures (particularly between probability distributions)
- Matrix decompositions
- Further probabilistic limit theorems
- Computational complexity
- Constraint satisfaction
- Sketching and streaming
- Hashing
- Derandomization and psuedorandomness
- Random graph theory
- Spectral graph theory

- Other graph algorithms/tools (e.g., sparsifiers, bouded treewidth)
- Cryptography