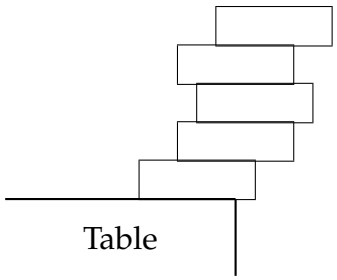
Sums, Approximations, and Asymptotics II

1 Block Stacking

How far can a stack of identical blocks overhang the end of a table without toppling over? Can a block be suspended *entirely beyond* the table's edge?



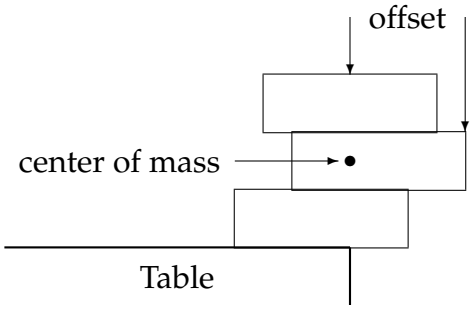
Physics imposes some constraints on the arrangement of the blocks. In particular, the stack falls off the desk if its center of mass lies beyond the desk's edge. Moreover, the center of mass of the top k blocks must lie above the $(k+1)$-st block; otherwise, the top k would fall over.

In order to find the best configuration of blocks satisfying these constraints, we'll need a fact about centers of mass.

Fact 1. *If two objects have masses m_1 and m_2 and centers-of-mass at positions z_1 and z_2, then the center of mass of the two objects together is at position:*

$$\frac{z_1 m_1 + z_2 m_2}{m_1 + m_2}$$

Define the *offset* of a particular configuration of blocks to be the horizonal distance from its center of mass to its rightmost edge.



The offset determines precisely how far that configuration can extend beyond the desk since at best the center of mass lies right above the desk's edge. So hereafter we can focus on maximizing the *offset* of a stack of n blocks rather than the *overhang*. As a result, we need only be concerned with whether the stack is internally stable and not with whether the whole configuration is too far to the right and falls off the table.

We can establish the greatest possible offset of a stack of n blocks with an inductive argument. This is an instance where induction not only serves as a proof technique, but also turns out to be a nice tool for reasoning about the problem.

Theorem 1. *The greatest possible offset of a stack of $n \geq 1$ blocks is:*

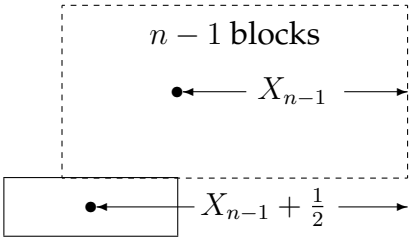
$$X_n = \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \ldots + \frac{1}{2n}$$

Proof. We use induction on n, the number of blocks. Let $P(n)$ be the proposition that the greatest possible offset of a stack of $n \geq 1$ blocks is $1/2 + 1/4 + \ldots + 1/(2n)$.

Base case: For a single block, the center of mass is distance $X_1 = 1/2$ from its rightmost edge. So the offset is $1/2$ and $P(1)$ is true.

Inductive step: For $n \geq 2$, we assume that $P(n-1)$ is true in order to prove $P(n)$. A stack of n blocks consists of the bottom block together with a stack of $n-1$ blocks on top.

In order to acheive the greatest possible offset with n blocks, the top $n-1$ blocks should themselves have the greatest possible offset, which is X_{n-1}; otherwise, we could do better by replacing the top $n-1$ blocks with a different configuration that has greater offset. Furthermore, the center of mass of the top $n-1$ blocks should lie directly above the right edge of the bottom block; otherwise, we could do better by sliding the top $n-1$ blocks farther to the right.



We've now identified the configuration of n blocks with the greatest possible offset. What remains is to determine what that offset is. To that end, we'll apply Fact 1 where positions are measured relative to the rightmost edge of the n-block stack. In particular, we have $n-1$ blocks with center of mass at position X_{n-1}, and 1 block with center of mass at position $X_{n-1} + \frac{1}{2}$. So Fact 1 implies that the maximum possible offset of a stack of n

blocks is:

$$\begin{aligned} X_n &= \frac{X_{n-1} \cdot (n-1) + (X_{n-1} + \frac{1}{2}) \cdot 1}{n} \\ &= X_{n-1} + \frac{1}{2n} \\ &= \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \ldots + \frac{1}{2n} \end{aligned}$$

We use the assumption $P(n-1)$ in the last step. This proves $P(n)$.

The theorem follows by the principle of induction. □

1.1 Harmonic Numbers

Sums similar to the one in Theorem 1 come up all the time in computer science. In particular,

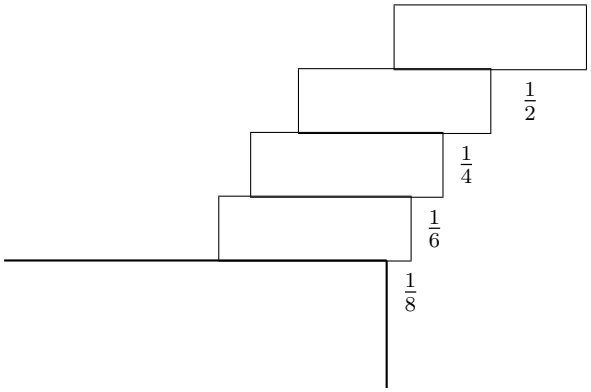
$$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

is called a ***harmonic sum***. Its value is called the **n*-th harmonic number*** and is denoted H_n. In these terms, the greatest possible offset of a stack of n blocks is $\frac{1}{2}H_n$.

We can tabulate the maximum achievable overhang with $n = 1$, 2, 3 and 4 blocks by computing the first few harmonic numbers:

# of blocks	maximum overhang
1	$\frac{1}{2}H_1 = \frac{1}{2}(\frac{1}{1}) = \frac{1}{2}$
2	$\frac{1}{2}H_2 = \frac{1}{2}(\frac{1}{1} + \frac{1}{2}) = \frac{3}{4}$
3	$\frac{1}{2}H_3 = \frac{1}{2}(\frac{1}{1} + \frac{1}{2} + \frac{1}{3}) = \frac{11}{12}$
4	$\frac{1}{2}H_4 = \frac{1}{2}(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4}) = \frac{25}{24} > 1$

The last line reveals that we can suspend the *fourth* block beyond the edge of the table! Here's the configuration that does the trick:



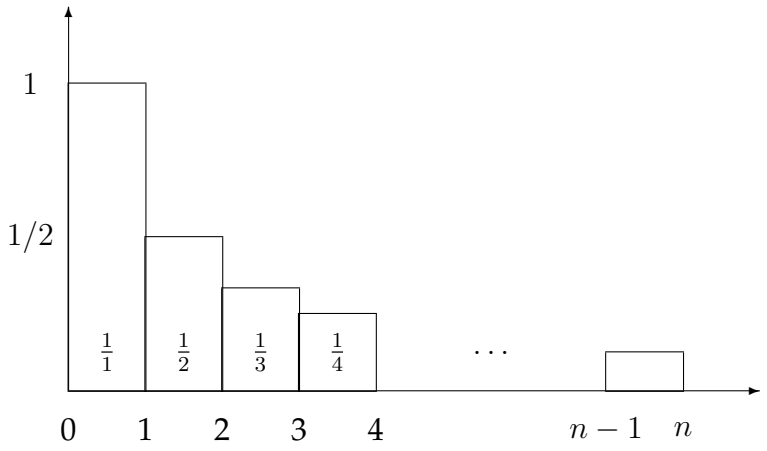
1.2 Bounding a Sum with an Integral

We need to know more about harmonic sums to determine what can be done with a large number of blocks. Unfortunately, there is no closed form for H_n. But, on the bright side, we can get good lower and upper bounds on harmonic sums using a general technique involving integration that applies to many other sums as well.

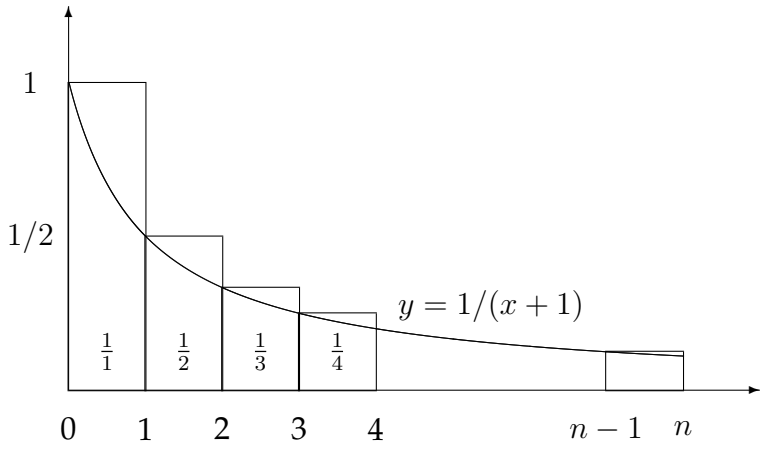
Here's how it works. First, we imagine a bar graph where the area of the k-th bar is equal to the k-th term in the sum. In particular, each bar has width 1 and height equal to the value of the k-th term. For example, the bar graph corresponding to the harmonic sum

$$H_n = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

is shown below.



Now the *value of the sum is equal to the area of the bars*, and we can estimate the area of the bars using integration. Suppose we draw a smooth curve that runs just below the bars; in fact, there's a natural choice: the curve described by the function $y = 1/(x+1)$.



The area of the bars is at least the area under this curve, so we have a lower bound on the n-th harmonic sum:

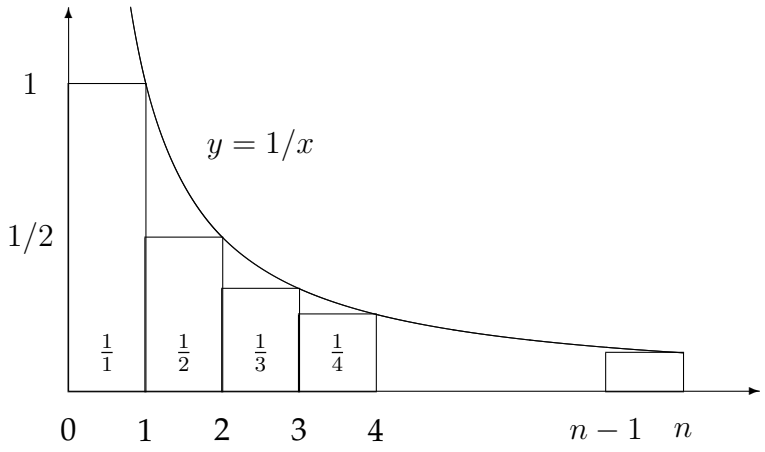
$$\begin{aligned} H_n &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \\ &\geq \int_0^n \frac{1}{x+1}\, dx \\ &= \ln(n+1) \end{aligned}$$

Remember that n blocks can overhang the edge of a table by $\frac{1}{2}H_n$ block widths. So if we had, say, a million blocks, then this lower bound implies that we could achieve an overhang of at least

$$\frac{\ln(1,000,000+1)}{2} = 6.907\ldots$$

block widths! In fact, since the lower bound of $\frac{1}{2}\ln(n+1)$ grows arbitrarily large, there is *no limit* on how far the stack can overhang. Of course, this assumes no breeze, deformation of the blocks, or gravitational variation as our stack grows thousands of miles high.

We can get an upper bound on the n-th harmonic number by playing essentially the same game. Now we need a curve that skims just above the bar graph. The curve defined by $y = 1/x$ fits the bill.



The area under this curve is an upper bound on the area of the bar graph and thus on the n-th harmonic sum. But there's a problem: the area under the curve is infinite because $y = 1/x$ has a bad asymptote at $x = 0$. This is a common problem when bounding sums with integrals and there's a simple solution: take the *exact* value of the first term $(1/1)$ and then bound the remaining terms $(1/2 + 1/3 + \ldots + 1/n)$ with an integral. In this case, we

get the upper bound:

$$\begin{aligned} H_n &= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \\ &\leq 1 + \int_1^n \frac{1}{x}\, dx \\ &= 1 + \ln n \end{aligned}$$

So even though there is no closed-form for the n-th harmonic sum, we now know that the harmonic numbers grow logarithmically:

$$\ln(n+1) \leq H_n \leq 1 + \ln n$$

There is a refinement to the integration method we've just employed, called *Euler-Maclaurin summation*, which yields a sequence of terms that correct errors in the inital estimate. This technique gives an absurdly accurate formula for harmonic numbers:

$$H_n = \ln n + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + \frac{\epsilon(n)}{120n^4}$$

The second term is ***Euler's constant***, $\gamma = 0.577215664\ldots$. This is probably the third most important mathematical constant behind e and π. It is somewhat mysterious; for example, no one knows whether γ is rational or irrational, though if it equals a ratio of integers, then the denominator must be at least $10^{242,080}$. In the final term of the formula above, $\epsilon(n)$ denotes some number between 0 and 1. You'll never need this kind of precision in this course or, probably, anywhere else.

2 The Factorial Function

One of the most common elements in messy mathematical expressions is the factorial function:

$$n! = 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n$$

Factorial comes up particularly often in combinatorics and probability, which happen to be the major topics in the remainder of 6.042. Within a few weeks, you are going to have factorials coming out your ears.

A good way to deal with any product is to covert it into a sum by taking a logarithm:

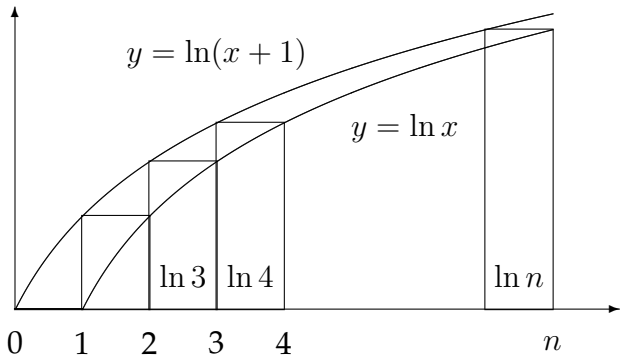
$$\ln\left(\prod_{k=1}^{n} f(k)\right) = \sum_{k=1}^{n} \ln f(k)$$

Then we can apply all our summing tools and exponentiate at the end to undo the effect of the log. Let's use this strategy to get bounds on $n!$. First, we take a logarithm to make

the product into a sum:

$$\begin{aligned}\ln n! &= \ln(1 \cdot 2 \cdot 3 \cdots n) \\ &= \ln 1 + \ln 2 + \ln 3 + \ldots + \ln n\end{aligned}$$

This sum is rather nasty, but we can still get bounds by the integration method. A picture can be *extremely helpful* in working out the proper bounding functions and limits of integration.



In this case, we get the bounds:

$$\int_1^n \ln x \, dx \;\leq\; \ln n! \;\leq\; \int_0^n \ln(x+1) \, dx$$

$$\begin{aligned} x \ln x - x\Big|_1^n \;&\leq\; \ln n! \;\leq\; (x+1)\ln(x+1) - (x+1)\Big|_0^n \\ n \ln n - n + 1 \;&\leq\; \ln n! \;\leq\; (n+1)\ln(n+1) - (n+1) + 1 \end{aligned}$$

Finally, we exponentiate to get bounds on $n!$.

$$\left(\frac{n}{e}\right)^n \cdot e \;\leq\; n! \;\leq\; \left(\frac{n+1}{e}\right)^{n+1} \cdot e$$

This gives some indication how big $n!$ is: about $(n/e)^n$. This estimate is often good enough. If you need a little more accuracy, you can add one more term to get ***Stirling's Formula***:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

Stirling's formula is worth committing to memory. We'll often use it to rewrite expressions involving $n!$. Now, one might object that the expression on the *left* actually looks a lot better than the expression on the *right*. But that's just an artifact of notation. If you actually wanted to compute $n!$, you'd need $n-1$ multiplications. However, the expression on the right is a closed form; evaluating it requires only a handful of basic operations, regardless of the value of n. Furthermore, when $n!$ appears inside a larger expression,

you usually can't do much with it. It doesn't readily cancel or combine with other terms. In contrast, the expression on the right looks scary, but melds nicely into larger formulas. So don't be put off: Stirling's Formula is your friend.

(Stepping back a bit, Stirling's formula is fairly amazing. Who would guess that the product of the first n positive *integers* could be so precisely described by a formula involving both e and π?)

If you're feeling a little crazy, you might pull out these even-more-precise bounds:

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/(12n+1)} \;\le\; n! \;\le\; \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/(12n)}$$

These bounds are ridiculously close. For example, if $n = 100$, then we get:

$$100! \ge \left(\frac{100}{e}\right)^{100} \sqrt{200\pi}\ \underbrace{e^{1/1201}}_{=1.000832\ldots}$$
$$100! \le \left(\frac{100}{e}\right)^{100} \sqrt{200\pi}\ \underbrace{e^{1/1200}}_{=1.000833\ldots}$$

The upper bound is less than 7 hundred-thousandths of 1% greater than the lower bound!

3 Asymptotic Notation

Our final topic is a special notation— called ***asymptotic notation***— designed to sweep mathematical messes under the rug. Asymptotic notation is sort of like the Force from Star Wars. It's everywhere. Tt's an essential part of the language of computer science. And mostly it is a tool for good. At best, asymptotic notation suppresses irrelevant detailss, while allowing the essential features of a mathematical analysis to shine through. However, as we'll see, asymptotic notation also has an alluring dark side.

3.1 Asymptotic Notation: The Jedi Perspective

Suppose you want to know how long a computer takes to multiply two $n \times n$ matrices. You *could* tally up all the multiplications and additions and loop variable increments and comparisons and perhaps factor in hardware-specific considerations such as page faults and cache misses and branch mispredicts and floating-point unit availability and all this would give you one sort of answer. (Whew!) Such an answer would be very accurate, but not very insightful. Given a very complicated formula for the running time, we would be hard-pressed to answer basic questions: How would doubling the size of the matrices alter the running time? What is the biggest matrix we can handle? Furthermore, a minor change in the procedure or hardware would invalidate the answer.

On the other hand, each of the n^2 entries in the product matrix takes about n steps to compute. So the running time is *roughly* $n^2 \cdot n = n^3$. This answer is certainly less precise, but it was easy to derive and is easy to interpret. For example, we can see that doubling the size of the matrices from $n \times n$ to $2n \times 2n$ would increase the running time from about n^3 to about $(2n)^3 = 8n^3$— a factor of 8. And, assuming a computer performs billions of operations in a reasonable time (as opposed to millions or trillions), the largest matrices we could handle would be roughly 1000×1000. Furthermore, this approximate answer is independent of tiny implementation and hardware details. It remains valid even after you upgrade your computer.

So approximate answers are quite attractive. And asymptotic notation allows one to formulate vague statements like "roughly n^3" in a very precise way.

3.2 Six Funny-Lookin' Symbols

Asymptotic notation involves six weird little symbols:

O	Ω	Θ	o	ω	$\sim$
oh	omega	theta	little-oh	little-omega	tilde

We'll focus on O, which is the most widely used. The others are about equally popular, except for ω, which is the Jar-Jar of the lot.

Suppose that f and g are functions. Then the statement

$$f(x) = O(g(x))$$

means that there exist constants x_0 and $c > 0$ such that

$$|f(x)| \leq c \cdot g(x)$$

for all $x > x_0$. Now this definition is pretty hairy. But what it's trying to say, with all its cryptic little constants, is that *f grows no faster than g*. A bit more precisely, it says that f is at most a constant times greater than g, except maybe for small values of x. For example, here's a true statement:

$$5x + 100 = O(x)$$

This holds because the left side is only about 5 times larger than the right. Of course, for small values of x (like $x = 1$) the left side is many times larger than the right, but the definition of O is cleverly designed to sweep aside such inconvenient details.

Let's work carefully through a sequence of examples involving O in order to better understand this definition.

Claim 2. $5x + 100 = O(x)$

Proof. We must show that there exist constants x_0 and $c > 0$ such that $|5x + 100| \leq c \cdot x$ for all $x > x_0$. Let $c = 10$ and $x_0 = 20$ and note that:

$$|5x + 100| \leq 5x + 5x = 10x \qquad \text{for all } x > 20$$

□

Here's another claim that points out a very common error involving O notation.

Claim 3. $x = O(x^2)$

Proof. We must show that there exist constants x_0 and $c > 0$ such that $|x| \leq c \cdot x^2$ for all $x > x_0$. Let $c = 1$ and $x_0 = 1$ and note that

$$|x| \leq 1 \cdot x^2 \qquad \text{for all } x > 1$$

□

Many people fail to realize that a statement of the form $f(x) = O(g(x))$ only places an *upper bound* on the growth of the function f. So, for example, you'll often hear people say things like, "I can't use an algorithm that runs in $O(x^2)$ steps because that's too slow." People who say things like that are *dumb* and you should make fun of them until they cry. Okay, maybe not. But they are incorrect; we just proved that a fast algorithm that requires only x steps also runs in time $O(x^2)$! One properly expresses a *lower bound* using the Ω notation, which we'll come to presently.

What about the reverse of the previous claim? Is $x^2 = O(x)$? On an informal basis, this means x^2 grows no faster than x, which is false. Let's prove this formally.

Claim 4. $x^2 \neq O(x)$

Proof. We argue by contradiction; suppose that there exist constants x_0 and c such that:

$$|x^2| \leq c \cdot x \qquad \text{for all } x > x_0$$

Dividing both sides of the inequality by x gives:

$$x \leq c \qquad \text{for all } x > x_0$$

But this is false when $x = 1 + \max(x_0, c)$. □

We can show that $x^2 \neq O(100x)$ by essentially the same argument; intuitively, x^2 grows quadratically, while $100x$ grows only linearly. Generally, changes in multiplicative constants do not affect the validity of an assertion involving O. However, constants in exponentials can be critical:

Claim 5.

$$4^x \neq O(2^x)$$

Proof. We argue by contradiction; suppose that there exist constants x_0 and $c > 0$ such that:

$$|4^x| \leq c \cdot 2^x \qquad \text{for all } x > x_0$$

Dividing both sides by 2^x gives:

$$2^x \leq c \qquad \text{for all } x > x_0$$

But this is false when $x = 1 + \max(x_0, \log c)$. □

3.3 Asymptotic Notation and Limits

The remaining symbols, Θ, Ω, o, ω, and $\sim$, all have definitions similar to O: "There exist constants blah and blah such that for all blah, such-and-such holds." This may prompt a sense of deja vu: these definitions are all quite similar to the definition of a *limit*. In fact, this similarity has a happy consequence: all these symbols have simple, intuitive interpretations in terms of limits. These are summarized in the table below.

Notation	**Intuition**	$\lim_{x\to\infty} f/g$	**Example**
$f = O(g)$	f grows no faster than g	$\neq \infty$	$4x + 7 = O(x^2)$
$f = \Omega(g)$	f grows at least as fast as g	$\neq 0$	$9x^2 = \Omega(x)$
$f = \Theta(g)$	f and g grow at about the same rate	$\neq 0, \infty$	$8x^2 + x = \Theta(x^2)$
$f \sim g$	f and g grow at the same rate	$= 1$	$x^2 + \sqrt{x} \sim x^2$
$f = o(g)$	f grows slower than g	$= 0$	$1/n = o(1)$
$f = \omega(g)$	f grows faster than g	$= \infty$	$n^2 = \omega(n)$

This summary of asymptotic notation is valid for essentially all functions encountered in computer science. However, in the rare case when $\lim_{n\to\infty} f/g$ does not exist or is negative, one must consult the formal, nit-picky definitions of these symbols. These definitions are provided in the notes for the recitation following this lecture.

3.4 Enter the Dark Side

So when someone talks asymptotics, why should you reach for your light saber?

Asymptotic notation is so widely used to analyze algorithm running times that there is a temptation to "design for the notation" rather than for true usefulness. Remember that asymptotic notation conceals some information about an algorithm's performance while highlighting other aspects. So when a researcher designs an algorithm, he or she might make tradeoffs so that the *revealed* part look good even though the *concealed* parts make the algorithm undesirable.

Indeed, there are some notable examples of algorithms where asymptotic notation obscures Deathstar-sized problems. For example, there is an ingenious algorithm that multiplies two $n \times n$ matrices in $O(n^{2.376})$ steps instead of the obvious $O(n^3)$. However, the O symbol conceals a constant so enormous that the naive multiplication algorithm is preferable up to at least $n = 10^{20}$. No one seems to have figured out exactly where the crossover point is, but we're unlikely to come anywhere near it in this century at least. Another example involves an algorithm that finds an approximate solution to the *Traveling Salesman Problem* in "nearly linear time". The running time to find an approximation within 10% of optimal is around $O(n \log^{400} n)$. Now, in a sense, the author's claim that this is "nearly linear" is correct; for example:

$$n \log^{400} n = O(n^{1.01})$$

But this is just a tad misleading since

$$n \log^{400} n \gg n^{1.01}$$

for all $n < 10^{100,000}$. These extreme examples are well-known, but whether misuse of asymptotic notation drives an wedge between algorithmic theory and practice more generally is a question you might ponder. The moral is: use asymptotic notation to clarify, not conceal.

Avoid the Dark Side, you must.