9 Sums and Asymptotics

Sums and products arise regularly in the analysis of algorithms, financial applications, physical problems, and probabilistic systems. For example, we have already encountered the sum $1 + 2 + 4 + \cdots + N$ when counting the number of nodes in a complete binary tree with N inputs. Although such a sum can be represented compactly using the sigma notation

$$\sum_{i=0}^{\log N} 2^i, \tag{9.1}$$

it is a lot easier and more helpful to express the sum by its *closed form* value

$$2N - 1.$$

By *closed form*, we mean an expression that does not make use of summation or product symbols or otherwise need those handy (but sometimes troublesome) dots.... Expressions in closed form are usually easier to evaluate (it doesn't get much simpler than $2N - 1$, for example) and it is usually easier to get a feel for their magnitude than expressions involving large sums and products.

But how do you find a closed form for a sum or product? Well, it's part math and part art. And it is the subject of this chapter.

We will start the chapter with a motivating example involving annuities. Figuring out the value of the annuity will involve a large and nasty-looking sum. We will then describe several methods for finding closed forms for all sorts of sums, including the annuity sums. In some cases, a closed form for a sum may not exist and so we will provide a general method for finding good upper and lower bounds on the sum (which are closed form, of course).

The methods we develop for sums will also work for products since you can convert any product into a sum by taking a logarithm of the product. As an example, we will use this approach to find a good closed-form approximation to

$$n! ::= 1 \cdot 2 \cdot 3 \cdots n.$$

We conclude the chapter with a discussion of asymptotic notation. Asymptotic notation is often used to bound the error terms when there is no exact closed form expression for a sum or product. It also provides a convenient way to express the growth rate or order of magnitude of a sum or product.

9.1 The Value of an Annuity

Would you prefer a million dollars today or $50,000 a year for the rest of your life? On the one hand, instant gratification is nice. On the other hand, the *total dollars* received at $50K per year is much larger if you live long enough.

Formally, this is a question about the value of an annuity. An *annuity* is a financial instrument that pays out a fixed amount of money at the beginning of every year for some specified number of years. In particular, an n-year, m-payment annuity pays m dollars at the start of each year for n years. In some cases, n is finite, but not always. Examples include lottery payouts, student loans, and home mortgages. There are even Wall Street people who specialize in trading annuities.[1]

A key question is, "What is an annuity worth?" For example, lotteries often pay out jackpots over many years. Intuitively, $50,000 a year for 20 years ought to be worth less than a million dollars right now. If you had all the cash right away, you could invest it and begin collecting interest. But what if the choice were between $50,000 a year for 20 years and a *half* million dollars today? Now it is not clear which option is better.

9.1.1 The Future Value of Money

In order to answer such questions, we need to know what a dollar paid out in the future is worth today. To model this, let's assume that money can be invested at a fixed annual interest rate p. We'll assume an 8% rate[2] for the rest of the discussion.

Here is why the interest rate p matters. Ten dollars invested today at interest rate p will become $(1+p)\cdot 10 = 10.80$ dollars in a year, $(1+p)^2\cdot 10 \approx 11.66$ dollars in two years, and so forth. Looked at another way, ten dollars paid out a year from now is only really worth $1/(1+p)\cdot 10 \approx 9.26$ dollars today. The reason is that if we had the $9.26 today, we could invest it and would have $10.00 in a year anyway. Therefore, p determines the value of money paid out in the future.

So for an n-year, m-payment annuity, the first payment of m dollars is truly worth m dollars. But the second payment a year later is worth only $m/(1+p)$ dollars. Similarly, the third payment is worth $m/(1+p)^2$, and the n-th payment is worth only $m/(1+p)^{n-1}$. The total value, V, of the annuity is equal to the sum of the

[1] Such trading ultimately led to the subprime mortgage disaster in 2008–2009. We'll talk more about that in Section 19.5.3.

[2] U.S. interest rates have dropped steadily for several years, and ordinary bank deposits now earn around 1.5%. But just a few years ago the rate was 8%; this rate makes some of our examples a little more dramatic. The rate has been as high as 17% in the past thirty years.

payment values. This gives:

$$\begin{aligned} V &= \sum_{i=1}^{n} \frac{m}{(1+p)^{i-1}} \\ &= m \cdot \sum_{j=0}^{n-1} \left(\frac{1}{1+p}\right)^{j} && \text{(substitute } j = i-1\text{)} \\ &= m \cdot \sum_{j=0}^{n-1} x^{j} && \text{(substitute } x = 1/(1+p)\text{).} \end{aligned} \tag{9.2}$$

The goal of the preceding substitutions was to get the summation into a simple special form so that we can solve it with a general formula. In particular, the terms of the sum

$$\sum_{j=0}^{n-1} x^{j} = 1 + x + x^2 + x^3 + \cdots + x^{n-1}$$

form a *geometric series*, which means that the ratio of consecutive terms is always the same and it is a positive value less than one. In this case, the ratio is always x, and $0 < x < 1$ since we assumed that $p > 0$. It turns out that there is a nice closed-form expression for any geometric series; namely

$$\sum_{i=0}^{n-1} x^{i} = \frac{1-x^{n}}{1-x}. \tag{9.3}$$

Equation 9.3 can be verified by induction, but, as is often the case, the proof by induction gives no hint about how the formula was found in the first place. So we'll take this opportunity to describe a method that you could use to figure it out for yourself. It is called the *Perturbation Method*.

9.1.2 The Perturbation Method

Given a sum that has a nice structure, it is often useful to "perturb" the sum so that we can somehow combine the sum with the perturbation to get something much simpler. For example, suppose

$$S = 1 + x + x^2 + \cdots + x^{n-1}.$$

An example of a perturbation would be

$$xS = x + x^2 + \cdots + x^{n}.$$

The difference between S and xS is not so great, and so if we were to subtract xS from S, there would be massive cancellation:

$$\begin{aligned} S &= 1 + x + x^2 + x^3 + \cdots + x^{n-1} \\ -xS &= \quad - x - x^2 - x^3 - \cdots - x^{n-1} - x^n. \end{aligned}$$

The result of the subtraction is

$$S - xS = 1 - x^n.$$

Solving for S gives the desired closed-form expression in Equation 9.3:

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

We'll see more examples of this method when we introduce *generating functions* in Chapter 12.

9.1.3 A Closed Form for the Annuity Value

Using Equation 9.3, we can derive a simple formula for V, the value of an annuity that pays m dollars at the start of each year for n years.

$$V = m\left(\frac{1 - x^n}{1 - x}\right) \qquad \text{(by Equations 9.2 and 9.3)} \tag{9.4}$$

$$= m\left(\frac{1 + p - (1/(1+p))^{n-1}}{p}\right) \qquad \text{(substituting } x = 1/(1+p)\text{)}. \tag{9.5}$$

Equation 9.5 is much easier to use than a summation with dozens of terms. For example, what is the real value of a winning lottery ticket that pays \$50,000 per year for 20 years? Plugging in $m = \$50{,}000$, $n = 20$, and $p = 0.08$ gives $V \approx \$530{,}180$. So because payments are deferred, the million dollar lottery is really only worth about a half million dollars! This is a good trick for the lottery advertisers.

9.1.4 Infinite Geometric Series

The question we began with was whether you would prefer a million dollars today or \$50,000 a year for the rest of your life. Of course, this depends on how long you live, so optimistically assume that the second option is to receive \$50,000 a year *forever*. This sounds like infinite money! But we can compute the value of an annuity with an infinite number of payments by taking the limit of our geometric sum in Equation 9.3 as n tends to infinity.

Theorem 9.1.1. *If* $|x| < 1$*, then*

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}.$$

Proof.

$$\begin{aligned}
\sum_{i=0}^{\infty} x^i &::= \lim_{n\to\infty} \sum_{i=0}^{n-1} x^i \\
&= \lim_{n\to\infty} \frac{1-x^n}{1-x} && \text{(by Equation 9.3)} \\
&= \frac{1}{1-x}.
\end{aligned}$$

The final line follows from that fact that $\lim_{n\to\infty} x^n = 0$ when $|x| < 1$. ■

In our annuity problem, $x = 1/(1+p) < 1$, so Theorem 9.1.1 applies, and we get

$$\begin{aligned}
V &= m \cdot \sum_{j=0}^{\infty} x^j && \text{(by Equation 9.2)} \\
&= m \cdot \frac{1}{1-x} && \text{(by Theorem 9.1.1)} \\
&= m \cdot \frac{1+p}{p} && (x = 1/(1+p)).
\end{aligned}$$

Plugging in $m = \$50{,}000$ and $p = 0.08$, we see that the value V is only \$675,000. Amazingly, a million dollars today is worth much more than \$50,000 paid every year forever! Then again, if we had a million dollars today in the bank earning 8% interest, we could take out and spend \$80,000 a year forever. So on second thought, this answer really isn't so amazing.

9.1.5 Examples

Equation 9.3 and Theorem 9.1.1 are incredibly useful in computer science. In fact, we already used Equation 9.3 implicitly when we claimed in Chapter 6 than an N-input complete binary tree has

$$1 + 2 + 4 + \cdots + N = 2N - 1$$

nodes. Here are some other common sums that can be put into closed form using Equation 9.3 and Theorem 9.1.1:

$$1 + 1/2 + 1/4 + \cdots = \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = \frac{1}{1-(1/2)} = 2 \tag{9.6}$$

$$0.99999\cdots = 0.9\sum_{i=0}^{\infty} \left(\frac{1}{10}\right)^i = 0.9\left(\frac{1}{1-1/10}\right) = 0.9\left(\frac{10}{9}\right) = 1 \tag{9.7}$$

$$1 - 1/2 + 1/4 - \cdots = \sum_{i=0}^{\infty} \left(\frac{-1}{2}\right)^i = \frac{1}{1-(-1/2)} = \frac{2}{3} \tag{9.8}$$

$$1 + 2 + 4 + \cdots + 2^{n-1} = \sum_{i=0}^{n-1} 2^i = \frac{1-2^n}{1-2} = 2^n - 1 \tag{9.9}$$

$$1 + 3 + 9 + \cdots + 3^{n-1} = \sum_{i=0}^{n-1} 3^i = \frac{1-3^n}{1-3} = \frac{3^n-1}{2} \tag{9.10}$$

If the terms in a geometric sum grow smaller, as in Equation 9.6, then the sum is said to be *geometrically decreasing*. If the terms in a geometric sum grow progressively larger, as in Equations 9.9 and 9.10, then the sum is said to be *geometrically increasing*. In either case, the sum is usually approximately equal to the term in the sum with the greatest absolute value. For example, in Equations 9.6 and 9.8, the largest term is equal to 1 and the sums are 2 and 2/3, both relatively close to 1. In Equation 9.9, the sum is about twice the largest term. In Equation 9.10, the largest term is 3^{n-1} and the sum is $(3^n - 1)/2$, which is only about a factor of 1.5 greater. You can see why this rule of thumb works by looking carefully at Equation 9.3 and Theorem 9.1.1.

9.1.6 Variations of Geometric Sums

We now know all about geometric sums—if you have one, life is easy. But in practice one often encounters sums that cannot be transformed by simple variable substitutions to the form $\sum x^i$.

A non-obvious, but useful way to obtain new summation formulas from old is by differentiating or integrating with respect to x. As an example, consider the following sum:

$$\sum_{i=1}^{n-1} i x^i = x + 2x^2 + 3x^3 + \cdots + (n-1)x^{n-1}$$

This is not a geometric sum, since the ratio between successive terms is not fixed, and so our formula for the sum of a geometric sum cannot be directly applied. But suppose that we differentiate Equation 9.3:

$$\frac{d}{dx}\left(\sum_{i=0}^{n-1} x^i\right) = \frac{d}{dx}\left(\frac{1-x^n}{1-x}\right). \tag{9.11}$$

The left-hand side of Equation 9.11 is simply

$$\sum_{i=0}^{n-1} \frac{d}{dx}(x^i) = \sum_{i=0}^{n-1} i x^{i-1}.$$

The right-hand side of Equation 9.11 is

$$\begin{aligned}\frac{-nx^{n-1}(1-x)-(-1)(1-x^n)}{(1-x)^2} &= \frac{-nx^{n-1}+nx^n+1-x^n}{(1-x)^2}\\ &= \frac{1-nx^{n-1}+(n-1)x^n}{(1-x)^2}.\end{aligned}$$

Hence, Equation 9.11 means that

$$\sum_{i=0}^{n-1} i x^{i-1} = \frac{1-nx^{n-1}+(n-1)x^n}{(1-x)^2}.$$

Often, differentiating or integrating messes up the exponent of x in every term. In this case, we now have a formula for a sum of the form $\sum i x^{i-1}$, but we want a formula for the series $\sum i x^i$. The solution is simple: multiply by x. This gives:

$$\sum_{i=1}^{n-1} i x^i = \frac{x-nx^n+(n-1)x^{n+1}}{(1-x)^2} \tag{9.12}$$

and we have the desired closed-form expression for our sum[3]. It's a little complicated looking, but it's easier to work with than the sum.

Notice that if $|x| < 1$, then this series converges to a finite value even if there are infinitely many terms. Taking the limit of Equation 9.12 as n tends infinity gives the following theorem:

[3]Since we could easily have made a mistake in the calculation, it is always a good idea to go back and validate a formula obtained this way with a proof by induction.

Theorem 9.1.2. *If* $|x| < 1$*, then*

$$\sum_{i=1}^{\infty} i x^i = \frac{x}{(1-x)^2}.$$

As a consequence, suppose that there is an annuity that pays im dollars at the *end* of each year i forever. For example, if $m = \$50{,}000$, then the payouts are \$50,000 and then \$100,000 and then \$150,000 and so on. It is hard to believe that the value of this annuity is finite! But we can use Theorem 9.1.2 to compute the value:

$$\begin{aligned} V &= \sum_{i=1}^{\infty} \frac{im}{(1+p)^i} \\ &= m \cdot \frac{1/(1+p)}{(1-\frac{1}{1+p})^2} \\ &= m \cdot \frac{1+p}{p^2}. \end{aligned}$$

The second line follows by an application of Theorem 9.1.2. The third line is obtained by multiplying the numerator and denominator by $(1+p)^2$.

For example, if $m = \$50{,}000$, and $p = 0.08$ as usual, then the value of the annuity is $V = \$8{,}437{,}500$. Even though the payments increase every year, the increase is only additive with time; by contrast, dollars paid out in the future decrease in value exponentially with time. The geometric decrease swamps out the additive increase. Payments in the distant future are almost worthless, so the value of the annuity is finite.

The important thing to remember is the trick of taking the derivative (or integral) of a summation formula. Of course, this technique requires one to compute nasty derivatives correctly, but this is at least theoretically possible!

9.2 Power Sums

In Chapter 3, we verified the formula

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}. \tag{9.13}$$

But the source of this formula is still a mystery. Sure, we can prove it is true using well ordering or induction, but where did the expression on the right come from in

the first place? Even more inexplicable is the closed form expression for the sum of consecutive squares:

$$\sum_{i=1}^{n} i^2 = \frac{(2n+1)(n+1)n}{6}. \tag{9.14}$$

It turns out that there is a way to derive these expressions, but before we explain it, we thought it would be fun[4] to show you how Gauss proved Equation 9.13 when he was a young boy.[5]

Gauss's idea is related to the perturbation method we used in Section 9.1.2. Let

$$S = \sum_{i=1}^{n} i.$$

Then we can write the sum in two orders:

$$\begin{aligned} S &= 1 + \quad 2 \quad + \ldots + (n-1) + n, \\ S &= n + (n-1) + \ldots + \quad 2 \quad + 1. \end{aligned}$$

Adding these two equations gives

$$\begin{aligned} 2S &= (n+1) + (n+1) + \cdots + (n+1) + (n+1) \\ &= n(n+1). \end{aligned}$$

Hence,

$$S = \frac{n(n+1)}{2}.$$

Not bad for a young child. Looks like Gauss had some potential....

Unfortunately, the same trick does not work for summing consecutive squares. However, we can observe that the result might be a third-degree polynomial in n, since the sum contains n terms that average out to a value that grows quadratically in n. So we might guess that

$$\sum_{i=1}^{n} i^2 = an^3 + bn^2 + cn + d.$$

If the guess is correct, then we can determine the parameters a, b, c, and d by plugging in a few values for n. Each such value gives a linear equation in a, b,

[4] Remember that we are mathematicians, so our definition of "fun" may be different than yours.

[5] We suspect that Gauss was probably not an ordinary boy.

c, and d. If we plug in enough values, we may get a linear system with a unique solution. Applying this method to our example gives:

$$\begin{aligned} n = 0 \quad &\text{implies} \quad 0 = d \\ n = 1 \quad &\text{implies} \quad 1 = a + b + c + d \\ n = 2 \quad &\text{implies} \quad 5 = 8a + 4b + 2c + d \\ n = 3 \quad &\text{implies} \quad 14 = 27a + 9b + 3c + d. \end{aligned}$$

Solving this system gives the solution $a = 1/3$, $b = 1/2$, $c = 1/6$, $d = 0$. Therefore, *if* our initial guess at the form of the solution was correct, then the summation is equal to $n^3/3 + n^2/2 + n/6$, which matches Equation 9.14.

The point is that if the desired formula turns out to be a polynomial, then once you get an estimate of the *degree* of the polynomial, all the coefficients of the polynomial can be found automatically.

Be careful! This method let's you discover formulas, but it doesn't guarantee they are right! After obtaining a formula by this method, it's important to go back and *prove* it using induction or some other method, because if the initial guess at the solution was not of the right form, then the resulting formula will be completely wrong![6]

9.3 Approximating Sums

Unfortunately, it is not always possible to find a closed-form expression for a sum. For example, consider the sum

$$S = \sum_{i=1}^{n} \sqrt{i}.$$

No closed form expression is known for S.

In such cases, we need to resort to approximations for S if we want to have a closed form. The good news is that there is a general method to find closed-form upper and lower bounds that work for most any sum. Even better, the method is simple and easy to remember. It works by replacing the sum by an integral and then adding either the first or last term in the sum.

[6] Alternatively, you can use the method based on generating functions described in Chapter 12, which does not require any guessing at all.

Theorem 9.3.1. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a nondecreasing[7] continuous function and let*

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

and

$$I = \int_1^n f(x)\,dx.$$

Then

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

Similarly, if f is nonincreasing, then

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

Proof. Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a nondecreasing function. For example, $f(x) = \sqrt{x}$ is such a function.

Consider the graph shown in Figure 9.1. The value of

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

is represented by the shaded area in this figure. This is because the ith rectangle in the figure (counting from left to right) has width 1 and height $f(i)$.

The value of

$$I = \int_1^n f(x)\,dx$$

is the shaded area under the curve of $f(x)$ from 1 to n shown in Figure 9.2.

Comparing the shaded regions in Figures 9.1 and 9.2, we see that S is at least I plus the area of the leftmost rectangle. Hence,

$$S \geq I + f(1) \tag{9.15}$$

This is the lower bound for S. We next derive the upper bound.

Figure 9.3 shows the curve of $f(x)$ from 1 to n shifted left by 1. This is the same as the curve $f(x + 1)$ from 0 to $n - 1$ and it has the same area I.

Comparing the shaded regions in Figures 9.1 and 9.3, we see that S is at most I plus the area of the rightmost rectangle. Hence,

$$S \leq I + f(n). \tag{9.16}$$

[7] A function f is *nondecreasing* if $f(x) \geq f(y)$ whenever $x \geq y$. It is *nonincreasing* if $f(x) \leq f(y)$ whenever $x \geq y$.

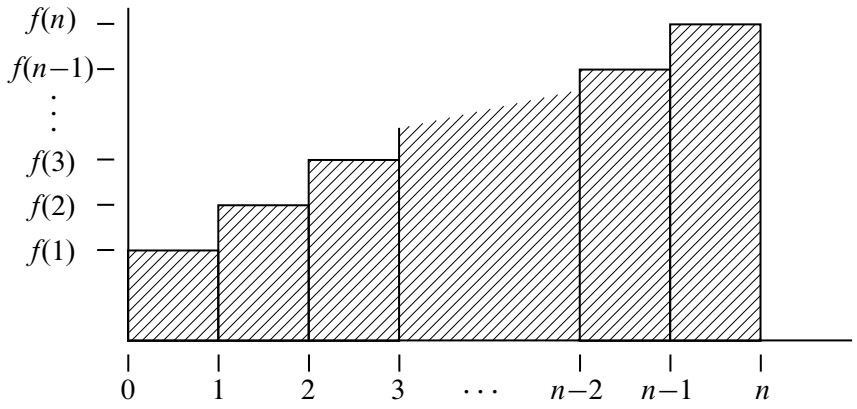


Figure 9.1 The area of the ith rectangle is $f(i)$. The shaded region has area $\sum_{i=1}^{n} f(i)$.

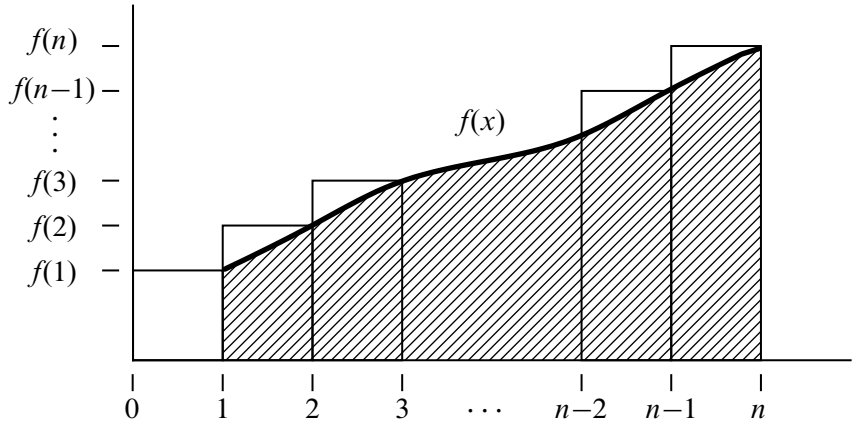


Figure 9.2 The shaded area under the curve of $f(x)$ from 1 to n (shown in bold) is $I = \int_1^n f(x)\,dx$.

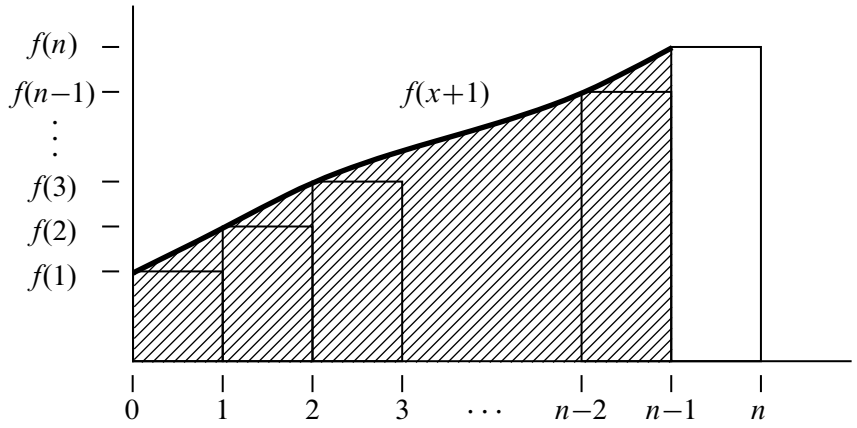


Figure 9.3 The shaded area under the curve of $f(x+1)$ from 0 to $n-1$ is I, the same as the area under the curve of $f(x)$ from 1 to n. This curve is the same as the curve in Figure 9.2 except that has been shifted left by 1.

Combining Equations 9.15 and 9.16, we find that

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

for any nondecreasing function f, as claimed

The argument for the case when f is nonincreasing is very similar. The analogous graphs to those shown in Figures 9.1–9.3 are provided in Figure 9.4. As you can see by comparing the shaded regions in Figures 9.4(a) and 9.4(b),

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

Similarly, comparing the shaded regions in Figures 9.4(a) and 9.4(c) reveals that

$$S \geq I + f(n).$$

Hence, if f is nonincreasing,

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

as claimed. ■

Theorem 9.3.1 provides good bounds for most sums. At worst, the bounds will be off by the largest term in the sum. For example, we can use Theorem 9.3.1 to bound the sum

$$S = \sum_{i=1}^{n} \sqrt{i}$$

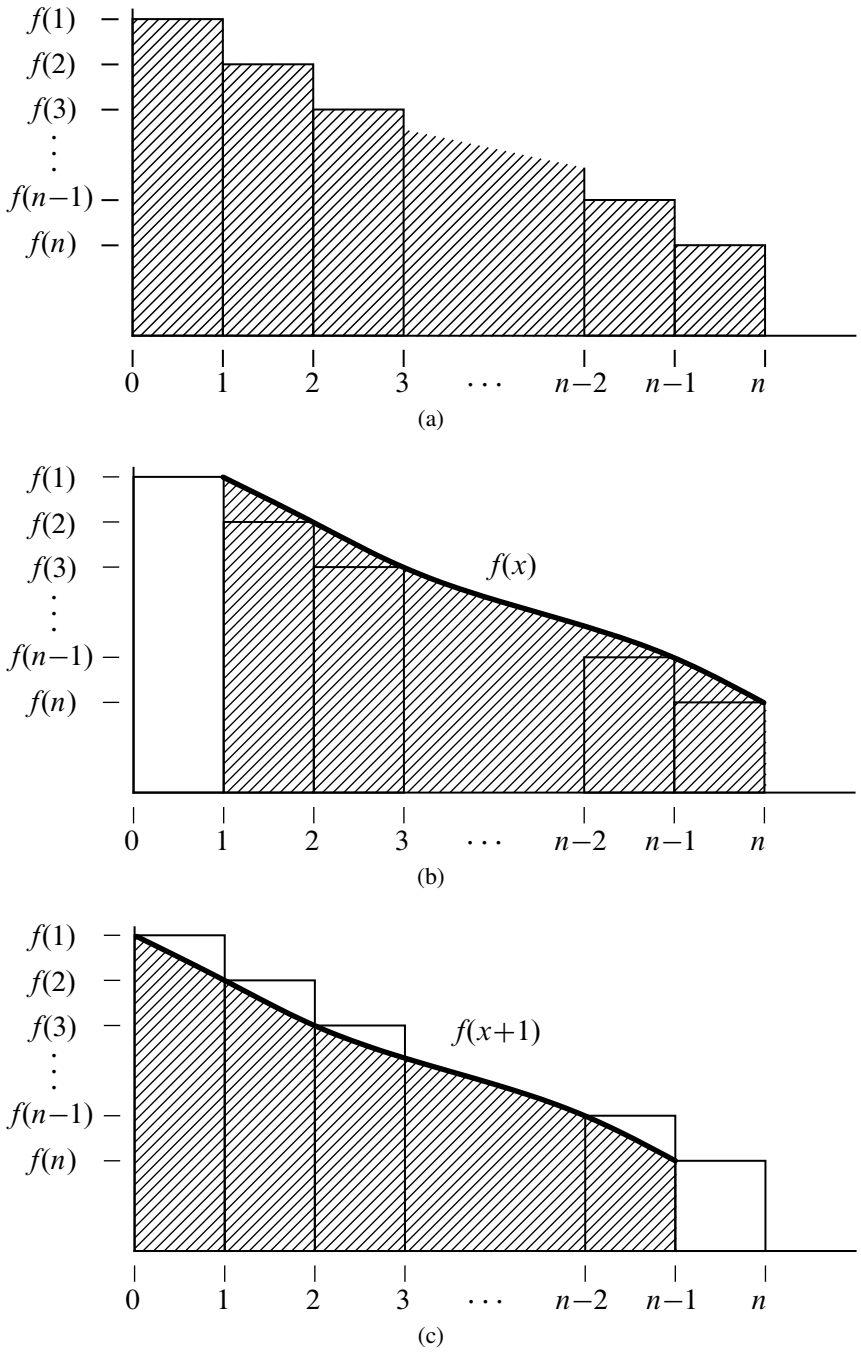


Figure 9.4 The area of the shaded region in (a) is $S = \sum_{i=1}^{n} f(i)$. The area in the shaded regions in (b) and (c) is $I = \int_1^n f(x)\,dx$.

as follows.

We begin by computing

$$\begin{aligned} I &= \int_1^n \sqrt{x}\, dx \\ &= \frac{x^{3/2}}{3/2}\Bigg|_1^n \\ &= \frac{2}{3}(n^{3/2} - 1). \end{aligned}$$

We then apply Theorem 9.3.1 to conclude that

$$\frac{2}{3}(n^{3/2} - 1) + 1 \;\le\; S \;\le\; \frac{2}{3}(n^{3/2} - 1) + \sqrt{n}$$

and thus that

$$\frac{2}{3}n^{3/2} + \frac{1}{3} \;\le\; S \;\le\; \frac{2}{3}n^{3/2} + \sqrt{n} - \frac{2}{3}.$$

In other words, the sum is very close to $\frac{2}{3}n^{3/2}$.

We'll be using Theorem 9.3.1 extensively going forward. At the end of this chapter, we will also introduce some notation that expresses phrases like "the sum is very close to" in a more precise mathematical manner. But first, we'll see how Theorem 9.3.1 can be used to resolve a classic paradox in structural engineering.

9.4 Hanging Out Over the Edge

Suppose that you have n identical blocks[8] and that you stack them one on top of the next on a table as shown in Figure 9.5. Is there some value of n for which it is possible to arrange the stack so that one of the blocks hangs out completely over the edge of the table without having the stack fall over? (You are not allowed to use glue or otherwise hold the stack in position.)

Most people's first response to this question—sometimes also their second and third responses—is "No. No block will ever get completely past the edge of the table." But in fact, if n is large enough, you can get the top block to stick out as far as you want: one block-length, two block-lengths, any number of block-lengths!

[8] We will assume that the blocks are rectangular, uniformly weighted and of length 1.

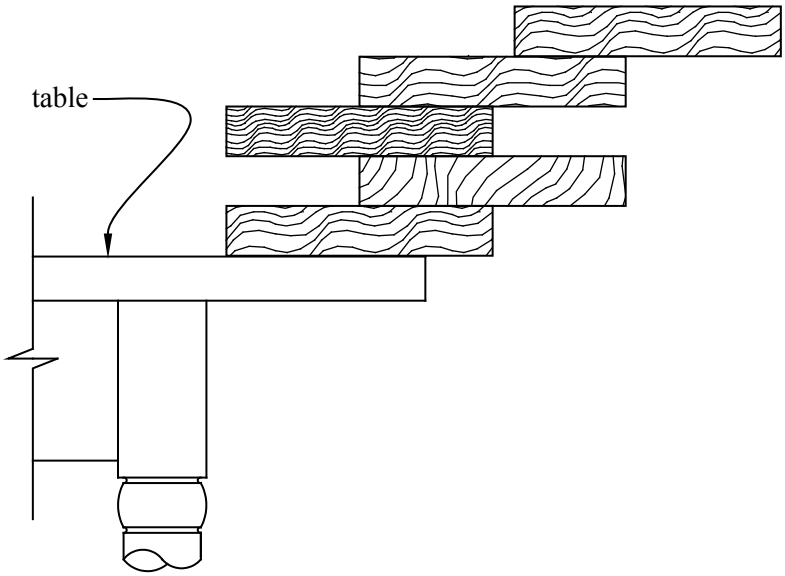


Figure 9.5 A stack of 5 identical blocks on a table. The top block is hanging out over the edge of the table, but if you try stacking the blocks this way, the stack will fall over.

9.4.1 Stability

A stack of blocks is said to be *stable* if it will not fall over of its own accord. For example, the stack illustrated in Figure 9.5 is not stable because the top block is sure to fall over. This is because the center or mass of the top block is hanging out over air.

In general, a stack of n blocks will be stable if and only if the center of mass of the top i blocks sits over the $(i+1)$st block for $i = 1, 2, \ldots, n-1$, and over the table for $i = n$.

We define the *overhang* of a stable stack to be the distance between the edge of the table and the rightmost end of the rightmost block in the stack. Our goal is thus to maximize the overhang of a stable stack.

For example, the maximum possible overhang for a single block is $1/2$. That is because the center of mass of a single block is in the middle of the block (which is distance $1/2$ from the right edge of the block). If we were to place the block so that its right edge is more than $1/2$ from the edge of the table, the center of mass would be over air and the block would tip over. But we can place the block so the center of mass is at the edge of the table, thereby achieving overhang $1/2$. This position is illustrated in Figure 9.6.

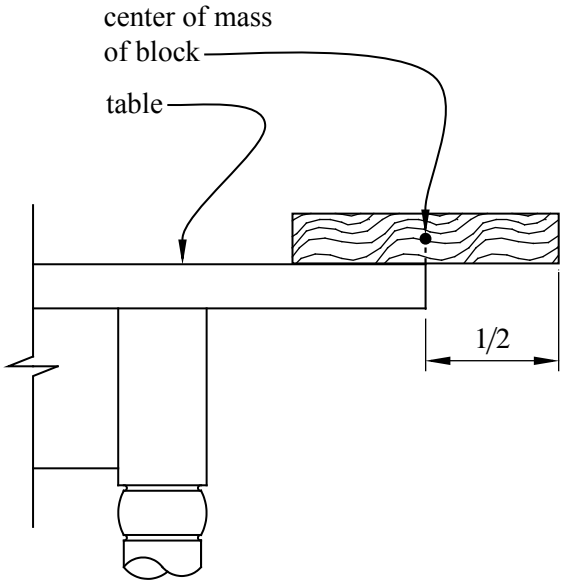


Figure 9.6 One block can overhang half a block length.

In general, the overhang of a stack of blocks is maximized by sliding the entire stack rightward until its center of mass is at the edge of the table. The overhang will then be equal to the distance between the center of mass of the stack and the rightmost edge of the rightmost block. We call this distance the *spread* of the stack. Note that the spread does not depend on the location of the stack on the table—it is purely a property of the blocks in the stack. Of course, as we just observed, the maximum possible overhang is equal to the maximum possible spread. This relationship is illustrated in Figure 9.7.

9.4.2 A Recursive Solution

Our goal is to find a formula for the maximum possible spread S_n that is achievable with a stable stack of n blocks.

We already know that $S_1 = 1/2$ since the right edge of a single block with length 1 is always distance $1/2$ from its center of mass. Let's see if we can use a recursive approach to determine S_n for all n. This means that we need to find a formula for S_n in terms of S_i where $i < n$.

Suppose we have a stable stack $\mathcal{S}$ of n blocks with maximum possible spread S_n. There are two cases to consider depending on where the rightmost block is in the stack.

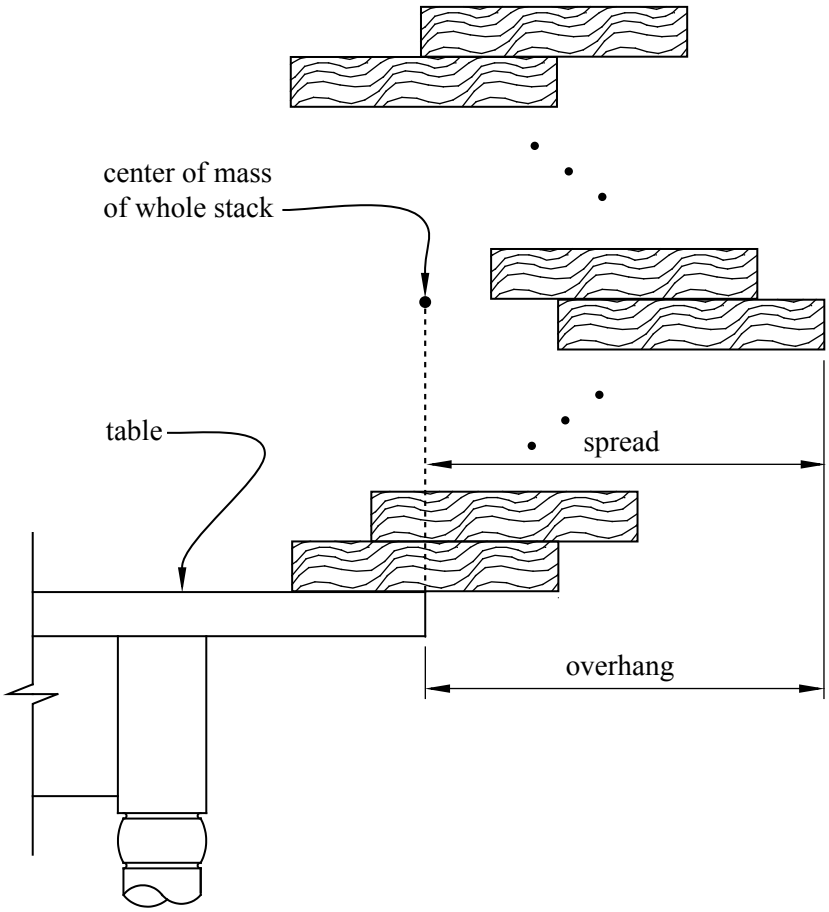


Figure 9.7 The overhang is maximized by maximizing the spread and then placing the stack so that the center of mass is at the edge of the table.

Case 1: *The rightmost block in $\mathcal{S}$ is the bottom block.* Since the center of mass of the top $n-1$ blocks must be over the bottom block for stability, the spread is maximized by having the center of mass of the top $n-1$ blocks be directly over the *left* edge of the bottom block. In this case the center of mass of $\mathcal{S}$ is[9]

$$\frac{(n-1)\cdot 1 + (1)\cdot \frac{1}{2}}{n} = 1 - \frac{1}{2n}$$

to the left of the right edge of the bottom block and so the spread for $\mathcal{S}$ is

$$1 - \frac{1}{2n}. \tag{9.17}$$

For example, see Figure 9.8.

In fact, the scenario just described is easily achieved by arranging the blocks as shown in Figure 9.9, in which case we have the spread given by Equation 9.17. For example, the spread is $3/4$ for 2 blocks, $5/6$ for 3 blocks, $7/8$ for 4 blocks, etc.

Can we do any better? The best spread in Case 1 is always less than 1, which means that we cannot get a block fully out over the edge of the table in this scenario. Maybe our intuition was right that we can't do better. Before we jump to any false conclusions, however, let's see what happens in the other case.

Case 2: *The rightmost block in $\mathcal{S}$ is among the top $n-1$ blocks.* In this case, the spread is maximized by placing the top $n-1$ blocks so that their center of mass is directly over the *right* end of the bottom block. This means that the center of mass for $\mathcal{S}$ is at location

$$\frac{(n-1)\cdot C + 1\cdot\left(C - \frac{1}{2}\right)}{n} = C - \frac{1}{2n}$$

where C is the location of the center of mass of the top $n-1$ blocks. In other words, the center of mass of $\mathcal{S}$ is $1/2n$ to the left of the center of mass of the top $n-1$ blocks. (The difference is due to the effect of the bottom block, whose center of mass is $1/2$ unit to the left of C.) This means that the spread of $\mathcal{S}$ is $1/2n$ greater than the spread of the top $n-1$ blocks (because we are in the case where the rightmost block is among the top $n-1$ blocks.)

Since the rightmost block is among the top $n-1$ blocks, the spread for $\mathcal{S}$ is maximized by maximizing the spread for the top $n-1$ blocks. Hence the maximum spread for $\mathcal{S}$ in this case is

$$S_{n-1} + \frac{1}{2n} \tag{9.18}$$

[9]The center of mass of a stack of blocks is the average of the centers of mass of the individual blocks.

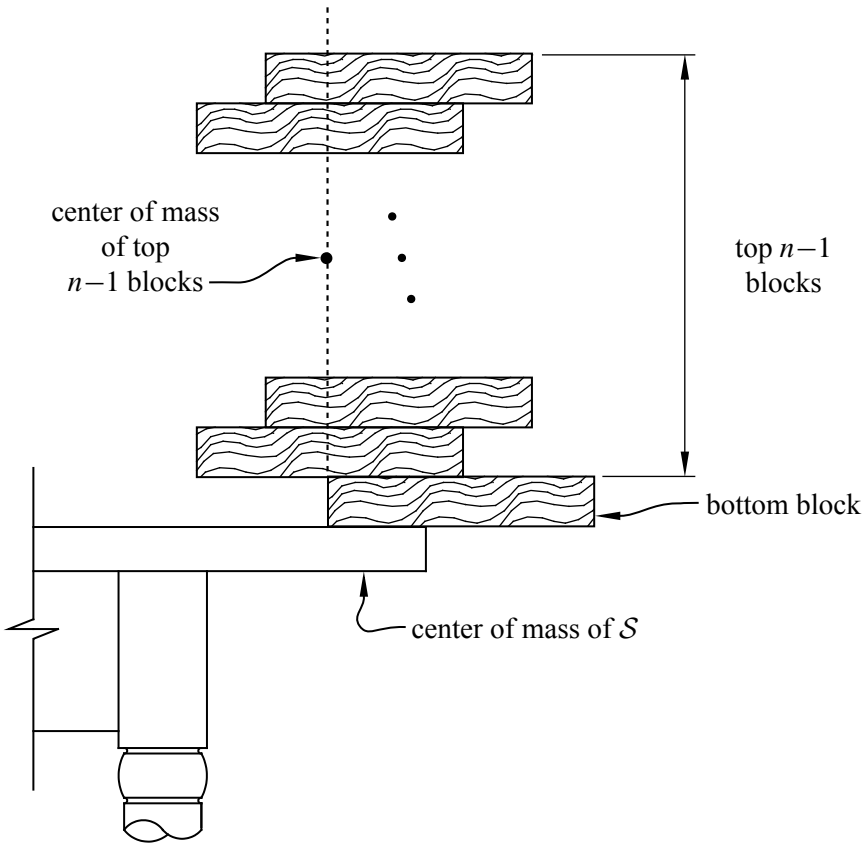


Figure 9.8 The scenario where the bottom block is the rightmost block. In this case, the spread is maximized by having the center of mass of the top $n-1$ blocks be directly over the left edge of the bottom block.

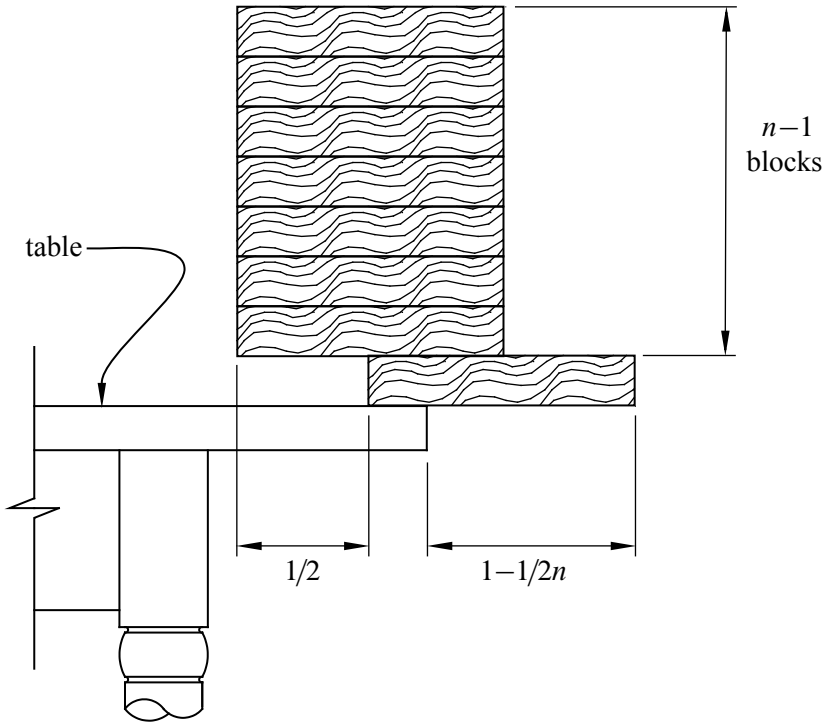


Figure 9.9 A method for achieving spread (and hence overhang) $1 - 1/2n$ with n blocks, where the bottom block is the rightmost block.

where S_{n-1} is the maximum possible spread for $n-1$ blocks (using any strategy).

We are now almost done. There are only two cases to consider when designing a stack with maximum spread and we have analyzed both of them. This means that we can combine Equation 9.17 from Case 1 with Equation 9.18 from Case 2 to conclude that

$$S_n = \max\left\{1 - \frac{1}{2n},\ S_{n-1} + \frac{1}{2n}\right\} \tag{9.19}$$

for any $n > 1$.

Uh-oh. This looks complicated. Maybe we are not almost done after all!

Equation 9.19 is an example of a *recurrence*. We will describe numerous techniques for solving recurrences in Chapter 10, but, fortunately, Equation 9.19 is simple enough that we can solve it without waiting for all the hardware in Chapter 10.

One of the first things to do when you have a recurrence is to get a feel for it by computing the first few terms. This often gives clues about a way to solve the recurrence, as it will in this case.

We already know that $S_1 = 1/2$. What about S_2? From Equation 9.19, we find that

$$\begin{aligned} S_2 &= \max\left\{1 - \frac{1}{4},\ \frac{1}{2} + \frac{1}{4}\right\} \\ &= 3/4. \end{aligned}$$

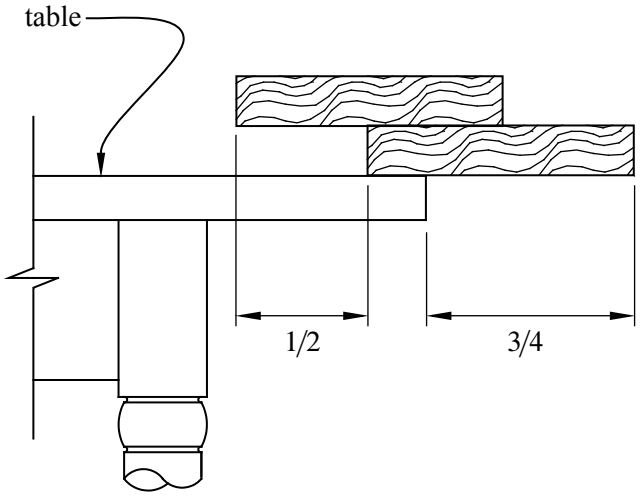
Both cases give the same spread, albeit by different approaches. For example, see Figure 9.10.

That was easy enough. What about S_3?

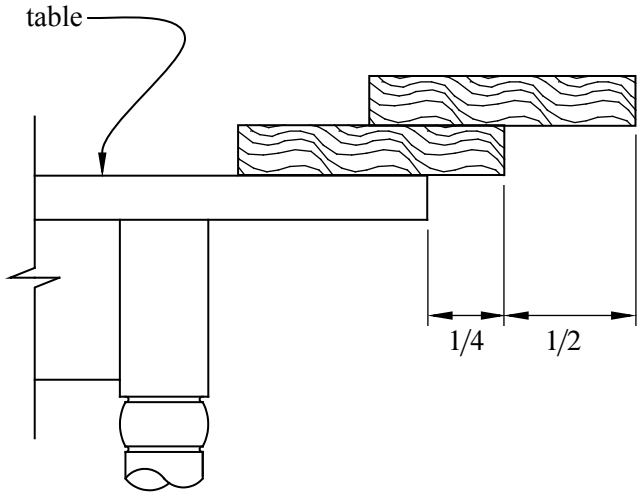
$$\begin{aligned} S_3 &= \max\left\{1 - \frac{1}{6},\ \frac{3}{4} + \frac{1}{6}\right\} \\ &= \max\left\{\frac{5}{6},\ \frac{11}{12}\right\} \\ &= \frac{11}{12}. \end{aligned}$$

As we can see, the method provided by Case 2 is the best. Let's check $n = 4$.

$$\begin{aligned} S_4 &= \max\left\{1 - \frac{1}{8},\ \frac{11}{12} + \frac{1}{8}\right\} \\ &= \frac{25}{24}. \end{aligned} \tag{9.20}$$



(a)



(b)

Figure 9.10 Two ways to achieve spread (and hence overhang) $3/4$ with $n = 2$ blocks. The first way (a) is from Case 1 and the second (b) is from Case 2.

Wow! This is a breakthrough—for two reasons. First, Equation 9.20 tells us that by using only 4 blocks, we can make a stack so that one of the blocks is hanging out completely over the edge of the table. The two ways to do this are shown in Figure 9.11.

The second reason that Equation 9.20 is important is that we now know that $S_4 > 1$, which means that we no longer have to worry about Case 1 for $n > 4$ since Case 1 never achieves spread greater than 1. Moreover, even for $n \leq 4$, we have now seen that the spread achieved by Case 1 never exceeds the spread achieved by Case 2, and they can be equal only for $n = 1$ and $n = 2$. This means that

$$S_n = S_{n-1} + \frac{1}{2n} \tag{9.21}$$

for all $n > 1$ since we have shown that the best spread can always be achieved using Case 2.

The recurrence in Equation 9.21 is much easier to solve than the one we started with in Equation 9.19. We can solve it by expanding the equation as follows:

$$\begin{aligned} S_n &= S_{n-1} + \frac{1}{2n} \\ &= S_{n-2} + \frac{1}{2(n-1)} + \frac{1}{2n} \\ &= S_{n-3} + \frac{1}{2(n-2)} + \frac{1}{2(n-1)} + \frac{1}{2n} \end{aligned}$$

and so on. This suggests that

$$S_n = \sum_{i=1}^{n} \frac{1}{2i}, \tag{9.22}$$

which is, indeed, the case.

Equation 9.22 can be verified by induction. The base case when $n = 1$ is true since we know that $S_1 = 1/2$. The inductive step follows from Equation 9.21.

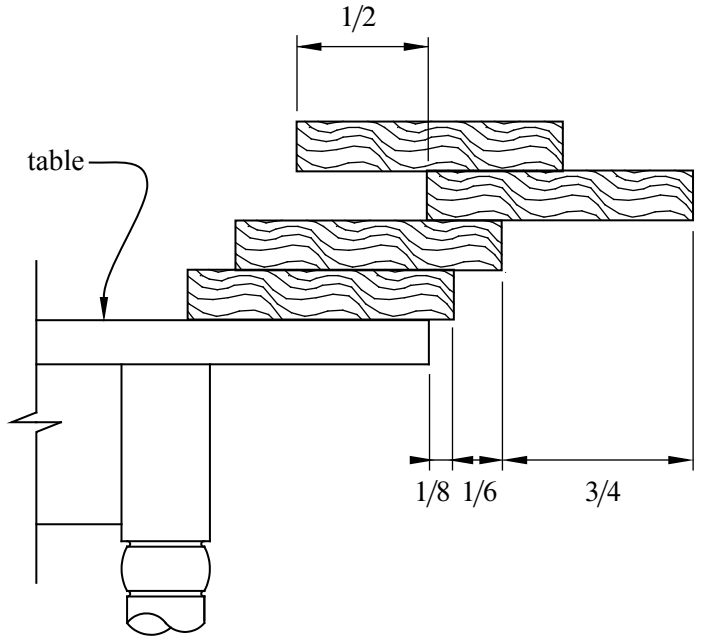
So we now know the maximum possible spread and hence the maximum possible overhang for any stable stack of books. Are we done? Not quite. Although we know that $S_4 > 1$, we still don't know how big the sum $\sum_{i=1}^{n} \frac{1}{2i}$ can get.

It turns out that S_n is very close to a famous sum known as the nth Harmonic number H_n.

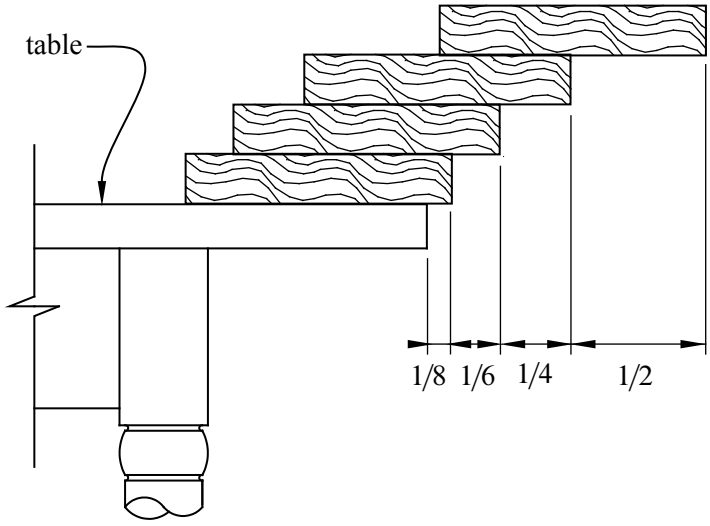
9.4.3 Harmonic Numbers

Definition 9.4.1. The *nth Harmonic number* is

$$H_n ::= \sum_{i=1}^{n} \frac{1}{i}.$$



(a)



(b)

Figure 9.11 The two ways to achieve spread (and overhang) $25/24$. The method in (a) uses Case 1 for the top 2 blocks and Case 2 for the others. The method in (b) uses Case 2 for every block that is added to the stack.

So Equation 9.22 means that

$$S_n = \frac{H_n}{2}. \tag{9.23}$$

The first few Harmonic numbers are easy to compute. For example,

$$H_4 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{25}{12}.$$

There is good news and bad news about Harmonic numbers. The bad news is that there is no closed-form expression known for the Harmonic numbers. The good news is that we can use Theorem 9.3.1 to get close upper and lower bounds on H_n. In particular, since

$$\int_1^n \frac{1}{x}\,dx = \ln(x)\Big|_1^n = \ln(n),$$

Theorem 9.3.1 means that

$$\ln(n) + \frac{1}{n} \leq H_n \leq \ln(n) + 1. \tag{9.24}$$

In other words, the nth Harmonic number is very close to $\ln(n)$.

Because the Harmonic numbers frequently arise in practice, mathematicians have worked hard to get even better approximations for them. In fact, it is now known that

$$H_n = \ln(n) + \gamma + \frac{1}{2n} + \frac{1}{12n^2} + \frac{\epsilon(n)}{120n^4} \tag{9.25}$$

Here γ is a value $0.577215664\ldots$ called *Euler's constant*, and $\epsilon(n)$ is between 0 and 1 for all n. We will not prove this formula.

We are now finally done with our analysis of the block stacking problem. Plugging the value of H_n into Equation 9.23, we find that the maximum overhang for n blocks is very close to $\frac{1}{2}\ln(n)$. Since $\ln(n)$ grows to infinity as n increases, this means that if we are given enough blocks (in theory anyway), we can get a block to hang out arbitrarily far over the edge of the table. Of course, the number of blocks we need will grow as an exponential function of the overhang, so it will probably take you a long time to achieve an overhang of 2 or 3, never mind an overhang of 100.

9.4.4 Asymptotic Equality

For cases like Equation 9.25 where we understand the growth of a function like H_n up to some (unimportant) error terms, we use a special notation, $\sim$, to denote the leading term of the function. For example, we say that $H_n \sim \ln(n)$ to indicate that the leading term of H_n is $\ln(n)$. More precisely:

Definition 9.4.2. For functions $f, g : \mathbb{R} \to \mathbb{R}$, we say f is *asymptotically equal* to g, in symbols,

$$f(x) \sim g(x)$$

iff

$$\lim_{x\to\infty} f(x)/g(x) = 1.$$

Although it is tempting to write $H_n \sim \ln(n) + \gamma$ to indicate the two leading terms, this is not really right. According to Definition 9.4.2, $H_n \sim \ln(n) + c$ where c is *any constant*. The correct way to indicate that γ is the second-largest term is $H_n - \ln(n) \sim \gamma$.

The reason that the $\sim$ notation is useful is that often we do not care about lower order terms. For example, if $n = 100$, then we can compute $H(n)$ to great precision using only the two leading terms:

$$|H_n - \ln(n) - \gamma| \le \left|\frac{1}{200} - \frac{1}{120000} + \frac{1}{120 \cdot 100^4}\right| < \frac{1}{200}.$$

We will spend a lot more time talking about asymptotic notation at the end of the chapter. But for now, let's get back to sums.

9.5 Double Trouble

Sometimes we have to evaluate sums of sums, otherwise known as *double summations*. This sounds hairy, and sometimes it is. But usually, it is straightforward—you just evaluate the inner sum, replace it with a closed form, and then evaluate the

outer sum (which no longer has a summation inside it). For example,[10]

$$\begin{aligned}
\sum_{n=0}^{\infty}\left(y^n\sum_{i=0}^{n}x^i\right) &= \sum_{n=0}^{\infty}\left(y^n\frac{1-x^{n+1}}{1-x}\right) && \text{Equation 9.3}\\
&= \left(\frac{1}{1-x}\right)\sum_{n=0}^{\infty}y^n - \left(\frac{1}{1-x}\right)\sum_{n=0}^{\infty}y^n x^{n+1}\\
&= \frac{1}{(1-x)(1-y)} - \left(\frac{x}{1-x}\right)\sum_{n=0}^{\infty}(xy)^n && \text{Theorem 9.1.1}\\
&= \frac{1}{(1-x)(1-y)} - \frac{x}{(1-x)(1-xy)} && \text{Theorem 9.1.1}\\
&= \frac{(1-xy)-x(1-y)}{(1-x)(1-y)(1-xy)}\\
&= \frac{1-x}{(1-x)(1-y)(1-xy)}\\
&= \frac{1}{(1-y)(1-xy)}.
\end{aligned}$$

When there's no obvious closed form for the inner sum, a special trick that is often useful is to try *exchanging the order of summation.* For example, suppose we want to compute the sum of the first n Harmonic numbers

$$\sum_{k=1}^{n}H_k = \sum_{k=1}^{n}\sum_{j=1}^{k}\frac{1}{j} \tag{9.26}$$

For intuition about this sum, we can apply Theorem 9.3.1 to Equation 9.24 to conclude that the sum is close to

$$\int_1^n \ln(x)\,dx = x\ln(x) - x\,\Big|_1^n = n\ln(n) - n + 1.$$

Now let's look for an exact answer. If we think about the pairs (k, j) over which

[10] Ok, so maybe this one is a little hairy, but it is also fairly straightforward. Wait till you see the next one!

we are summing, they form a triangle:

		j						
		1	2	3	4	5	...	n
k	1	1						
	2	1	1/2					
	3	1	1/2	1/3				
	4	1	1/2	1/3	1/4			
		...						
	n	1	1/2		...			$1/n$

The summation in Equation 9.26 is summing each row and then adding the row sums. Instead, we can sum the columns and then add the column sums. Inspecting the table we see that this double sum can be written as

$$\begin{aligned}
\sum_{k=1}^{n} H_k &= \sum_{k=1}^{n}\sum_{j=1}^{k}\frac{1}{j} \\
&= \sum_{j=1}^{n}\sum_{k=j}^{n}\frac{1}{j} \\
&= \sum_{j=1}^{n}\frac{1}{j}\sum_{k=j}^{n}1 \\
&= \sum_{j=1}^{n}\frac{1}{j}(n-j+1) \\
&= \sum_{j=1}^{n}\frac{n+1}{j} - \sum_{j=1}^{n}\frac{j}{j} \\
&= (n+1)\sum_{j=1}^{n}\frac{1}{j} - \sum_{j=1}^{n}1 \\
&= (n+1)H_n - n. \qquad (9.27)
\end{aligned}$$

9.6 Products

We've covered several techniques for finding closed forms for sums but no methods for dealing with products. Fortunately, we do not need to develop an entirely new set of tools when we encounter a product such as

$$n! ::= \prod_{i=1}^{n} i. \tag{9.28}$$

That's because we can convert any product into a sum by taking a logarithm. For example, if

$$P = \prod_{i=1}^{n} f(i),$$

then

$$\ln(P) = \sum_{i=1}^{n} \ln(f(i)).$$

We can then apply our summing tools to find a closed form (or approximate closed form) for $\ln(P)$ and then exponentiate at the end to undo the logarithm.

For example, let's see how this works for the factorial function $n!$ We start by taking the logarithm:

$$\begin{aligned}
\ln(n!) &= \ln(1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n) \\
&= \ln(1) + \ln(2) + \ln(3) + \cdots + \ln(n-1) + \ln(n) \\
&= \sum_{i=1}^{n} \ln(i).
\end{aligned}$$

Unfortunately, no closed form for this sum is known. However, we can apply Theorem 9.3.1 to find good closed-form bounds on the sum. To do this, we first compute

$$\begin{aligned}
\int_1^n \ln(x)\, dx &= x \ln(x) - x \Big|_1^n \\
&= n \ln(n) - n + 1.
\end{aligned}$$

Plugging into Theorem 9.3.1, this means that

$$n \ln(n) - n + 1 \;\le\; \sum_{i=1}^{n} \ln(i) \;\le\; n \ln(n) - n + 1 + \ln(n).$$

Exponentiating then gives

$$\frac{n^n}{e^{n-1}} \;\le\; n! \;\le\; \frac{n^{n+1}}{e^{n-1}}. \tag{9.29}$$

This means that $n!$ is within a factor of n of n^n/e^{n-1}.

9.6.1 Stirling's Formula

$n!$ is probably the most commonly used product in discrete mathematics, and so mathematicians have put in the effort to find much better closed-form bounds on its value. The most useful bounds are given in Theorem 9.6.1.

Theorem 9.6.1 (Stirling's Formula)**.** *For all $n \ge 1$,*

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

where

$$\frac{1}{12n+1} \le \epsilon(n) \le \frac{1}{12n}.$$

Theorem 9.6.1 can be proved by induction on n, but the details are a bit painful (even for us) and so we will not go through them here.

There are several important things to notice about Stirling's Formula. First, $\epsilon(n)$ is always positive. This means that

$$n! > \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \tag{9.30}$$

for all $n \in \mathbb{N}^+$.

Second, $\epsilon(n)$ tends to zero as n gets large. This means that[11]

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n, \tag{9.31}$$

which is rather surprising. After all, who would expect both π and e to show up in a closed-form expression that is asymptotically equal to $n!$?

Third, $\epsilon(n)$ is small even for small values of n. This means that Stirling's Formula provides good approximations for $n!$ for most all values of n. For example, if we use

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

[11] The $\sim$ notation was defined in Section 9.4.4.

Approximation	$n \geq 1$	$n \geq 10$	$n \geq 100$	$n \geq 1000$
$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$	$< 10\%$	$< 1\%$	$< 0.1\%$	$< 0.01\%$
$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{1/12n}$	$< 1\%$	$< 0.01\%$	$< 0.0001\%$	$< 0.000001\%$

Table 9.1 Error bounds on common approximations for $n!$ from Theorem 9.6.1. For example, if $n \geq 100$, then $\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ approximates $n!$ to within 0.1%.

as the approximation for $n!$, as many people do, we are guaranteed to be within a factor of

$$e^{\epsilon(n)} \leq e^{\frac{1}{12n}}$$

of the correct value. For $n \geq 10$, this means we will be within 1% of the correct value. For $n \geq 100$, the error will be less than 0.1%.

If we need an even closer approximation for $n!$, then we could use either

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

or

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

depending on whether we want an upper bound or a lower bound, respectively. By Theorem 9.6.1, we know that both bounds will be within a factor of

$$e^{\frac{1}{12n}-\frac{1}{12n+1}} = e^{\frac{1}{144n^2+12n}}$$

of the correct value. For $n \geq 10$, this means that either bound will be within 0.01% of the correct value. For $n \geq 100$, the error will be less than 0.0001%.

For quick future reference, these facts are summarized in Corollary 9.6.2 and Table 9.1.

Corollary 9.6.2. *For $n \geq 1$,*

$$n! < 1.09\sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

For $n \geq 10$,

$$n! < 1.009\sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

For $n \geq 100$,

$$n! < 1.0009\sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

9.7 Asymptotic Notation

Asymptotic notation is a shorthand used to give a quick measure of the behavior of a function $f(n)$ as n grows large. For example, the asymptotic notation $\sim$ of Definition 9.4.2 is a binary relation indicating that two functions grow at the *same* rate. There is also a binary relation indicating that one function grows at a significantly *slower* rate than another.

9.7.1 Little Oh

Definition 9.7.1. For functions $f, g : \mathbb{R} \to \mathbb{R}$, with g nonnegative, we say f is *asymptotically smaller* than g, in symbols,

$$f(x) = o(g(x)),$$

iff

$$\lim_{x\to\infty} f(x)/g(x) = 0.$$

For example, $1000x^{1.9} = o(x^2)$, because $1000x^{1.9}/x^2 = 1000/x^{0.1}$ and since $x^{0.1}$ goes to infinity with x and 1000 is constant, we have $\lim_{x\to\infty} 1000x^{1.9}/x^2 = 0$. This argument generalizes directly to yield

Lemma 9.7.2. *$x^a = o(x^b)$ for all nonnegative constants $a < b$.*

Using the familiar fact that $\log x < x$ for all $x > 1$, we can prove

Lemma 9.7.3. *$\log x = o(x^\epsilon)$ for all $\epsilon > 0$.*

Proof. Choose $\epsilon > \delta > 0$ and let $x = z^\delta$ in the inequality $\log x < x$. This implies

$$\log z < z^\delta/\delta = o(z^\epsilon) \qquad \text{by Lemma 9.7.2.} \tag{9.32}$$

■

Corollary 9.7.4. *$x^b = o(a^x)$ for any $a, b \in \mathbb{R}$ with $a > 1$.*

Lemma 9.7.3 and Corollary 9.7.4 can also be proved using l'Hôpital's Rule or the McLaurin Series for $\log x$ and e^x. Proofs can be found in most calculus texts.

9.7.2 Big Oh

Big Oh is the most frequently used asymptotic notation. It is used to give an upper bound on the growth of a function, such as the running time of an algorithm.

Definition 9.7.5. Given nonnegative functions $f, g : \mathbb{R} \to \mathbb{R}$, we say that

$$f = O(g)$$

iff

$$\limsup_{x\to\infty} f(x)/g(x) < \infty.$$

This definition[12] makes it clear that

Lemma 9.7.6. *If $f = o(g)$ or $f \sim g$, then $f = O(g)$.*

Proof. $\lim f/g = 0$ or $\lim f/g = 1$ implies $\lim f/g < \infty$. ■

It is easy to see that the converse of Lemma 9.7.6 is not true. For example, $2x = O(x)$, but $2x \not\sim x$ and $2x \neq o(x)$.

The usual formulation of Big Oh spells out the definition of lim sup without mentioning it. Namely, here is an equivalent definition:

Definition 9.7.7. Given functions $f, g : \mathbb{R} \to \mathbb{R}$, we say that

$$f = O(g)$$

iff there exists a constant $c \geq 0$ and an x_0 such that for all $x \geq x_0$, $|f(x)| \leq cg(x)$.

This definition is rather complicated, but the idea is simple: $f(x) = O(g(x))$ means $f(x)$ is less than or equal to $g(x)$, except that we're willing to ignore a constant factor, namely, c, and to allow exceptions for small x, namely, $x < x_0$.

We observe,

Lemma 9.7.8. *If $f = o(g)$, then it is* not *true that $g = O(f)$.*

[12] We can't simply use the limit as $x \to \infty$ in the definition of $O()$, because if $f(x)/g(x)$ oscillates between, say, 3 and 5 as x grows, then $f = O(g)$ because $f \leq 5g$, but $\lim_{x\to\infty} f(x)/g(x)$ does not exist. So instead of limit, we use the technical notion of lim sup. In this oscillating case, $\limsup_{x\to\infty} f(x)/g(x) = 5$.

The precise definition of lim sup is

$$\limsup_{x\to\infty} h(x) ::= \lim_{x\to\infty} \text{lub}_{y\geq x} h(y),$$

where "lub" abbreviates "least upper bound."

Proof.

$$\lim_{x\to\infty} \frac{g(x)}{f(x)} = \frac{1}{\lim_{x\to\infty} f(x)/g(x)} = \frac{1}{0} = \infty,$$

so $g \neq O(f)$.

■

Proposition 9.7.9. $100x^2 = O(x^2)$.

Proof. Choose $c = 100$ and $x_0 = 1$. Then the proposition holds, since for all $x \geq 1$, $\left|100x^2\right| \leq 100x^2$. ■

Proposition 9.7.10. $x^2 + 100x + 10 = O(x^2)$.

Proof. $(x^2+100x+10)/x^2 = 1+100/x+10/x^2$ and so its limit as x approaches infinity is $1+0+0 = 1$. So in fact, $x^2+100x+10 \sim x^2$, and therefore $x^2+100x+10 = O(x^2)$. Indeed, it's conversely true that $x^2 = O(x^2 + 100x + 10)$. ■

Proposition 9.7.10 generalizes to an arbitrary polynomial:

Proposition 9.7.11. $a_k x^k + a_{k-1}x^{k-1} + \cdots + a_1 x + a_0 = O(x^k)$.

We'll omit the routine proof.

Big Oh notation is especially useful when describing the running time of an algorithm. For example, the usual algorithm for multiplying $n \times n$ matrices uses a number of operations proportional to n^3 in the worst case. This fact can be expressed concisely by saying that the running time is $O(n^3)$. So this asymptotic notation allows the speed of the algorithm to be discussed without reference to constant factors or lower-order terms that might be machine specific. It turns out that there is another, ingenious matrix multiplication procedure that uses $O(n^{2.55})$ operations. This procedure will therefore be much more efficient on large enough matrices. Unfortunately, the $O(n^{2.55})$-operation multiplication procedure is almost never used in practice because it happens to be less efficient than the usual $O(n^3)$ procedure on matrices of practical size.[13]

9.7.3 Omega

Suppose you want to make a statement of the form "the running time of the algorithm is a least…". Can you say it is "at least $O(n^2)$"? No! This statement is meaningless since big-oh can only be used for *upper* bounds. For lower bounds, we use a different symbol, called "big-Omega."

[13] It is even conceivable that there is an $O(n^2)$ matrix multiplication procedure, but none is known.

Definition 9.7.12. Given functions $f, g : \mathbb{R} \to \mathbb{R}$, we say that

$$f = \Omega(g)$$

iff there exists a constant $c > 0$ and an x_0 such that for all $x \geq x_0$, we have $f(x) \geq c|g(x)|$.

In other words, $f(x) = \Omega(g(x))$ means that $f(x)$ is greater than or equal to $g(x)$, except that we are willing to ignore a constant factor and to allow exceptions for small x.

If all this sounds a lot like big-Oh, only in reverse, that's because big-Omega is the opposite of big-Oh. More precisely,

Theorem 9.7.13. *$f(x) = O(g(x))$ if and only if $g(x) = \Omega(f(x))$.*

Proof.

$$\begin{aligned}
&f(x) = O(g(x)) \\
&\quad \text{iff} \quad \exists c > 0, x_0.\ \forall x \geq x_0.\ |f(x)| \leq c g(x) && \text{(Definition 9.7.7)} \\
&\quad \text{iff} \quad \exists c > 0, x_0.\ \forall x \geq x_0.\ g(x) \geq \frac{1}{c}|f(x)| \\
&\quad \text{iff} \quad \exists c' > 0, x_0.\ \forall x \geq x_0.\ g(x) \geq c'|f(x)| && \text{(set } c' = 1/c\text{)} \\
&\quad \text{iff} \quad g(x) = \Omega(f(x)) && \text{(Definition 9.7.12)} \quad \blacksquare
\end{aligned}$$

For example, $x^2 = \Omega(x)$, $2^x = \Omega(x^2)$, and $x/100 = \Omega(100x + \sqrt{x})$.

So if the running time of your algorithm on inputs of size n is $T(n)$, and you want to say it is at least quadratic, say

$$T(n) = \Omega(n^2).$$

Little Omega

There is also a symbol called little-omega, analogous to little-oh, to denote that one function grows strictly faster than another function.

Definition 9.7.14. For functions $f, g : \mathbb{R} \to \mathbb{R}$ with f nonnegative, we say that

$$f(x) = \omega(g(x))$$

iff

$$\lim_{x \to \infty} \frac{g(x)}{f(x)} = 0.$$

In other words,

$$f(x) = \omega(g(x))$$

iff

$$g(x) = o(f(x)).$$

For example, $x^{1.5} = \omega(x)$ and $\sqrt{x} = \omega(\ln^2(x))$.

The little-omega symbol is not as widely used as the other asymptotic symbols we have been discussing.

9.7.4 Theta

Sometimes we want to specify that a running time $T(n)$ is precisely quadratic up to constant factors (both upper bound *and* lower bound). We could do this by saying that $T(n) = O(n^2)$ and $T(n) = \Omega(n^2)$, but rather than say both, mathematicians have devised yet another symbol, Θ, to do the job.

Definition 9.7.15.

$$f = \Theta(g) \quad \text{iff} \quad f = O(g) \text{ and } g = O(f).$$

The statement $f = \Theta(g)$ can be paraphrased intuitively as "f and g are equal to within a constant factor." Indeed, by Theorem 9.7.13, we know that

$$f = \Theta(g) \quad \text{iff} \quad f = O(g) \text{ and } f = \Omega(g).$$

The Theta notation allows us to highlight growth rates and allow suppression of distracting factors and low-order terms. For example, if the running time of an algorithm is

$$T(n) = 10n^3 - 20n^2 + 1,$$

then we can more simply write

$$T(n) = \Theta(n^3).$$

In this case, we would say that T *is of order* n^3 or that $T(n)$ *grows cubically*, which is probably what we really want to know. Another such example is

$$\pi^2 3^{x-7} + \frac{(2.7x^{113} + x^9 - 86)^4}{\sqrt{x}} - 1.08^{3x} = \Theta(3^x).$$

Just knowing that the running time of an algorithm is $\Theta(n^3)$, for example, is useful, because if n doubles we can predict that the running time will *by and large*[14] increase by a factor of at most 8 for large n. In this way, Theta notation preserves information about the scalability of an algorithm or system. Scalability is, of course, a big issue in the design of algorithms and systems.

[14]Since $\Theta(n^3)$ only implies that the running time, $T(n)$, is between cn^3 and dn^3 for constants $0 < c < d$, the time $T(2n)$ could regularly exceed $T(n)$ by a factor as large as $8d/c$. The factor is sure to be close to 8 for all large n only if $T(n) \sim n^3$.

9.7.5 Pitfalls with Asymptotic Notation

There is a long list of ways to make mistakes with asymptotic notation. This section presents some of the ways that Big Oh notation can lead to ruin and despair. With minimal effort, you can cause just as much chaos with the other symbols.

The Exponential Fiasco

Sometimes relationships involving Big Oh are not so obvious. For example, one might guess that $4^x = O(2^x)$ since 4 is only a constant factor larger than 2. This reasoning is incorrect, however; 4^x actually grows as the square of 2^x.

Constant Confusion

Every constant is $O(1)$. For example, $17 = O(1)$. This is true because if we let $f(x) = 17$ and $g(x) = 1$, then there exists a $c > 0$ and an x_0 such that $|f(x)| \leq cg(x)$. In particular, we could choose $c = 17$ and $x_0 = 1$, since $|17| \leq 17 \cdot 1$ for all $x \geq 1$. We can construct a false theorem that exploits this fact.

False Theorem 9.7.16.

$$\sum_{i=1}^{n} i = O(n)$$

Bogus proof. Define $f(n) = \sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + n$. Since we have shown that every constant i is $O(1)$, $f(n) = O(1) + O(1) + \cdots + O(1) = O(n)$. ■

Of course in reality $\sum_{i=1}^{n} i = n(n+1)/2 \neq O(n)$.

The error stems from confusion over what is meant in the statement $i = O(1)$. For any *constant* $i \in \mathbb{N}$ it is true that $i = O(1)$. More precisely, if f is any constant function, then $f = O(1)$. But in this False Theorem, i is not constant—it ranges over a set of values 0, 1,..., n that depends on n.

And anyway, we should not be adding $O(1)$'s as though they were numbers. We never even defined what $O(g)$ means by itself; it should only be used in the context "$f = O(g)$" to describe a relation between functions f and g.

Lower Bound Blunder

Sometimes people incorrectly use Big Oh in the context of a lower bound. For example, they might say, "The running time, $T(n)$, is at least $O(n^2)$," when they probably mean[15] "$T(n) = \Omega(n^2)$."

[15] This can also be correctly expressed as $n^2 = O(T(n))$, but such notation is rare.

Equality Blunder

The notation $f = O(g)$ is too firmly entrenched to avoid, but the use of "=" is really regrettable. For example, if $f = O(g)$, it seems quite reasonable to write $O(g) = f$. But doing so might tempt us to the following blunder: because $2n = O(n)$, we can say $O(n) = 2n$. But $n = O(n)$, so we conclude that $n = O(n) = 2n$, and therefore $n = 2n$. To avoid such nonsense, we will never write "$O(f) = g$."

Similarly, you will often see statements like

$$H_n = \ln(n) + \gamma + O\left(\frac{1}{n}\right)$$

or

$$n! = (1 + o(1))\sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

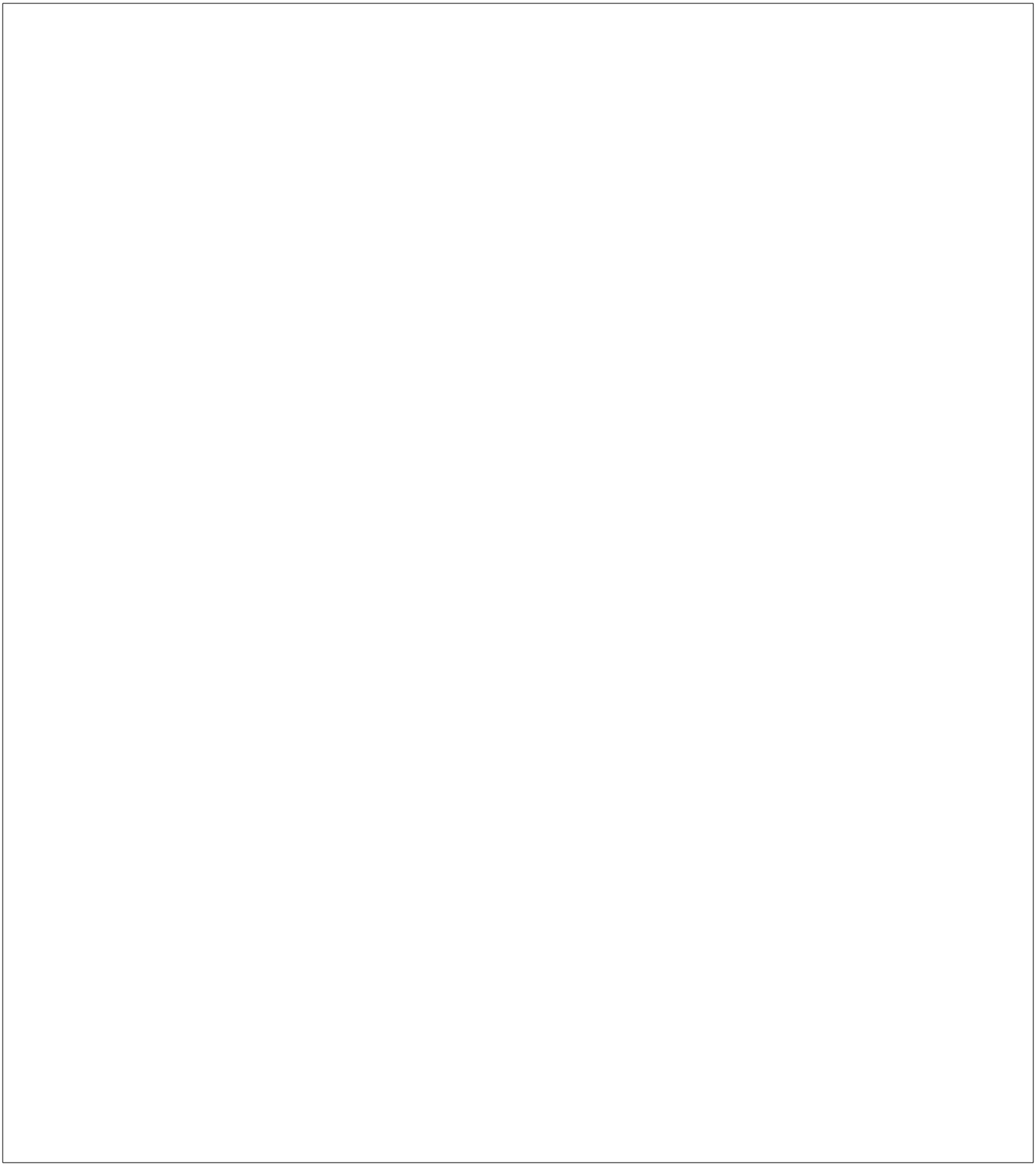
In such cases, the true meaning is

$$H_n = \ln(n) + \gamma + f(n)$$

for some $f(n)$ where $f(n) = O(1/n)$, and

$$n! = (1 + g(n))\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

where $g(n) = o(1)$. These transgressions are OK as long as you (and you reader) know what you mean.



6.042J / 18.062J Mathematics for Computer Science
Fall 2010