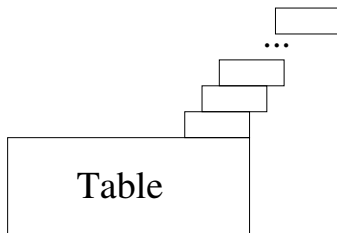


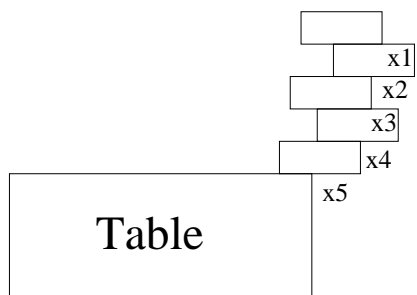
## Sums, Approximations, and Asymptotics II

### 1 Block Stacking

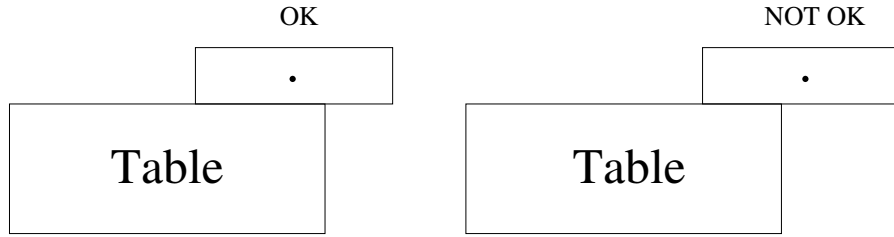
Today we will start by using the methods we covered last time to solve a physics problem. The problem is to see if we can make a stack of  $n$  blocks that won't fall over and for which some block hangs out completely off the table.



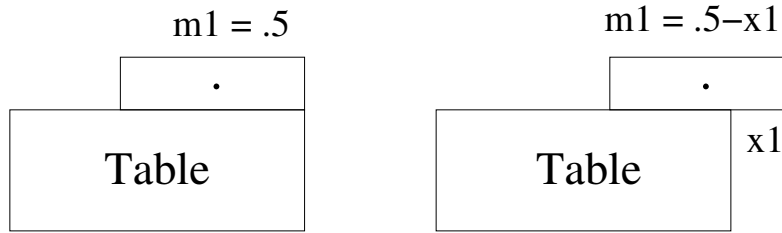
Is there some  $n$  for which the  $n$ th block hangs out completely off the table? It turns out that the answer is yes! In fact, surprisingly, you can get a block to hang out as far as you like! However, to do that, you might need to go arbitrarily high. In order to see why this is the case, we're going to have to do some analysis, which is of course why we're looking at this problem in the first place.



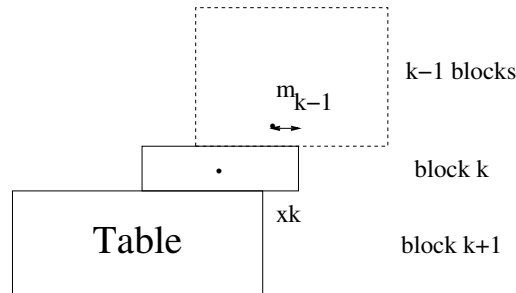
We're going to assume each block has length 1. For  $1 \leq i \leq n$ , we define  $x_i$  to be the amount that block  $i$  extends beyond block  $i + 1$ . Note that the  $(n + 1)$ st block is the table. Generally  $x_i$  is positive, but it could be negative, as with  $x_3$  above. In order to avoid blocks falling over, we need the following stability constraint. A stack of blocks is *stable* iff for  $1 \leq i \leq n$ , the center of mass of the top  $i$  blocks lies above the  $(i + 1)$ st block, i.e., it cannot exceed the left or right end. We're not going to prove that a stack won't fall over iff the stack is stable, but rather, we'll take it as an axiom of physics.



Let's express the stability constraint mathematically. Define  $m_k$  to be the amount by which the center of mass of the top  $k$  blocks is to the left of the right end of block  $k + 1$ .



What condition do we need on  $m_1$  to be stable? Well we need  $0 \leq m_1 \leq 1$ . Indeed, if  $m_1 < 0$ , then the top block will fall off the right hand side. Also, if  $m_1 > 1$ , the top block will fall off the left hand side. Note that the value of  $m_1$  which gives the maximum extension is  $m_1 = 0$ . In this case  $x_1 = 1/2$ , and this is as large as  $x_1$  can be.



Now consider  $k > 1$ . Let's compute the center of mass of the top  $k$  blocks with respect to the right end of the  $k$ th block. To do this, we'll use another fact from physics that we'll treat as an axiom.

**Fact 1.** *If two objects have mass  $M_1$  and  $M_2$  and centers of mass  $C_1$  and  $C_2$ , the center of mass of the combined object is at  $\frac{C_1 M_1 + C_2 M_2}{M_1 + M_2}$ .*

So let's treat the first object as the top  $k - 1$  blocks and the second object as the  $k$ th block, and apply this fact.

**Corollary 1.** *The center of mass of the top  $k$  blocks is*

$$\frac{(k - 1)m_{k-1} + (1)\frac{1}{2}}{(k - 1) + (1)} = \frac{k - 1}{k}m_{k-1} + \frac{1}{2k}$$

*to the left of the right edge of the  $k$ th block.*

To compute  $m_k$ , we need to compute the distance from the right edge of the  $(k+1)$ st block to the center of mass of the top  $k$  blocks. By definition of  $x_k$ , it follows that

$$m_k = \frac{k-1}{k}m_{k-1} + \frac{1}{2k} - x_k.$$

Rewriting,

$$x_k = \frac{1}{2k} + \frac{k-1}{k}m_{k-1} - m_k.$$

Now we define the *total extension for block  $k$* , denoted  $E_k$ , to be the amount that block  $k$  hangs out. Then  $E_k$  is just  $x_k + x_{k+1} + \cdots + x_n = \sum_{i=k}^n x_i$ . Given  $n$  blocks, the *total extension*, which is the quantity we wish to maximize, is

$$\max_{k=1}^n E_k,$$

which is the farthest position to the right that any of the  $n$  blocks is from the table.

We first maximize  $E_k$ .

$$\begin{aligned} E_k = x_k + x_{k+1} + \cdots + x_n &= \left( \frac{1}{2k} + \frac{k-1}{k}m_{k-1} - m_k \right) + \cdots + \left( \frac{1}{2n} + \frac{n-1}{n}m_{n-1} - m_n \right) \\ &= \frac{1}{2k} + \frac{1}{2k+2} + \cdots + \frac{1}{2n} + \frac{k-1}{k}m_{k-1} \\ &\quad - \frac{1}{k+1}m_k - \frac{1}{k+2}m_{k+1} - \cdots - \frac{1}{n}m_{n-1} - m_n. \end{aligned}$$

Since  $0 \leq m_i \leq 1$  for all  $i$ ,  $E_k$  has maximum value

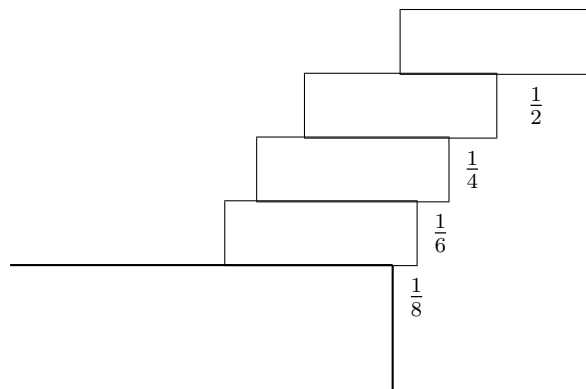
$$E_k = \frac{1}{2k} + \frac{1}{2k+2} + \cdots + \frac{1}{2n} + \frac{k-1}{k}.$$

We claim the total extension is maximized for  $k=1$  or  $k=2$ , which corresponds to either the first or second block being the farthest out from the table. Indeed,

$$E_k - E_{k+1} = \frac{1}{2k} + \frac{k-1}{k} - \frac{k}{k+1} = \frac{1}{2k} + 1 - \frac{1}{k} - 1 + \frac{1}{k+1} = \frac{1}{k+1} - \frac{1}{2k}.$$

Thus,  $E_k - E_{k+1}$  is positive iff  $k+1 < 2k$ , or if  $k > 1$ . Thus, the maximum total extension is realized either for  $k=1$  or  $k=2$ . Note that  $E_1 - E_2 = 0$ , so in fact, there are two optimal solutions, one for which the rightmost block is the first block, and one for which it is the second.

It follows that the maximum total extension is the maximum value of  $E_1$ , which is  $\frac{1}{2} \sum_{k=1}^n \frac{1}{k}$ . For example, an optimal stacking for 5 blocks is the following.



It remains to evaluate the sum  $\frac{1}{2} \sum_{k=1}^n \frac{1}{k}$ .

## 1.1 Harmonic Numbers

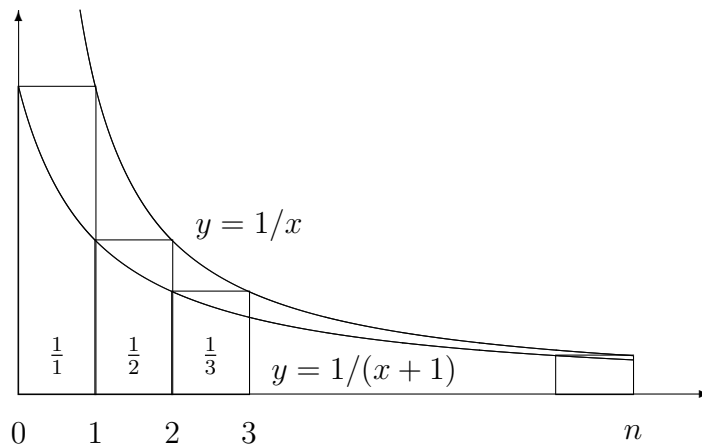
Sums like this come up all the time in computer science. In particular,

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

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 greatest overhang achievable with  $n = 1, 2, 3$  and  $4$  blocks by computing harmonic numbers:

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

We'll have to study harmonic sums more closely to determine what can be done with large numbers of blocks. Let's use integration to bound the  $n$ -th harmonic number. A picture is extremely helpful for getting the right functions and limits of integration.



We can not integrate the function  $1/x$  starting from zero. So, for the upper bound on  $H_n$  we'll take the first term explicitly ( $1/1$ ) and then upper bound the remaining terms normally. This gives:

$$\int_0^n \frac{1}{x+1} dx \leq H_n \leq 1 + \int_1^n \frac{1}{x} dx$$

$$\ln(x+1) \Big|_0^n \leq H_n \leq 1 + \left( \ln x \Big|_1^n \right)$$

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

These are good bounds; the difference between the upper and lower values is never more than 1.

Suppose we had a *million* blocks. Then the overhang we could achieve— assuming no breeze or deformation of the blocks— would be  $\frac{1}{2}H_{1000000}$ . According to our bounds, this is:

$$\text{at least } \frac{1}{2} \ln(1000001) = 6.907\dots$$

$$\text{at most } \frac{1}{2}(1 + \ln(1000000)) = 7.407\dots$$

So the top block would extend about 7 lengths past the end of the table! In fact, since the lower bound or  $\frac{1}{2} \ln(n+1)$  grows arbitrarily large, there is *no limit* on how far the stack can overhang!

Mathematicians have worked out some extremely precise approximations for the  $n$ -th harmonic number. For example:

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

where  $\gamma = 0.577215664\dots$  is **Euler's constant** and  $\epsilon(n)$  is between 0 and 1. Interestingly, no one knows whether Euler's constant is rational or irrational.

For example, suppose we wanted to know how many blocks are needed to get the top block out a distance of 100 from the table. Solving,  $H_n \geq 200$  implies  $\ln n + 1 \geq 200$ , which implies that  $n \geq e^{199}$ . This number happens to be more than the number of particles in the universe.

## 2 Products

We've now looked at many techniques for coping with sums, but no methods for dealing with products. Fortunately, we don't need to develop an entirely new set of tools. Instead, we can first convert any product into a sum by taking a logarithm:

$$\ln \left( \prod f(n) \right) = \sum \ln f(n)$$

Then we can apply our summing tools and exponentiate at the end to undo the logarithm.

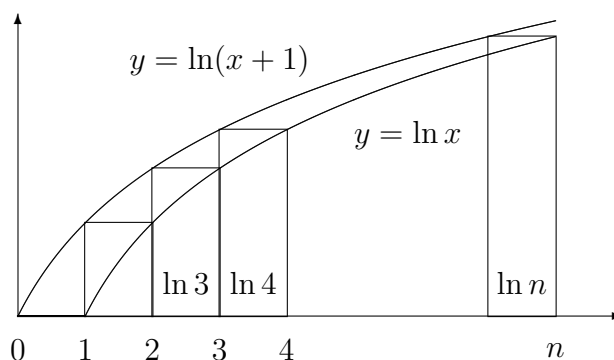
Let's apply this strategy to a product that you'll encounter almost daily hereafter:

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

First, we take a logarithm:

$$\ln n! = \ln 1 + \ln 2 + \ln 3 + \dots + \ln n$$

This sum is rather nasty, but we can still get bounds by integrating. First, we work out appropriate functions and limits of integration with a picture.



Now we integrate to get bounds on  $\ln n!$ .

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

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

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

$$\frac{n^n}{e^n} \leq n! \leq \frac{(n+1)^{(n+1)}}{e^n}$$

This gives some indication how big  $n!$  is: about  $(n/e)^n$ . This estimate is often good enough. However, as with the harmonic numbers, more precise bounds are known.

**Fact 2.**

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

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

$$\begin{aligned} 100! &\geq \left(\frac{100}{e}\right)^{100} \sqrt{200\pi} \underbrace{e^{1/1201}}_{=1.000832\dots} \\ 100! &\leq \left(\frac{100}{e}\right)^{100} \sqrt{200\pi} \underbrace{e^{1/1200}}_{=1.000833\dots} \end{aligned}$$

The upper bound is less than 7 hundred-thousandths of 1% greater than the lower bound! Taken together, these bounds imply ***Stirling's Formula***:

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

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; the expression on the right is your true 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  $\pi$ ?

### 3 Asymptotic Notation

Approximation is a powerful tool. It lets you sweep aside irrelevant detail without losing track of the big picture.

Approximations are particularly useful in the analysis of computer systems and algorithms. For example, suppose you wanted to know how long multiplying two  $n \times n$  matrices

takes. 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.

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 proportional to  $n^3$ . This answer is certainly less precise. However, high-precision answers are rarely needed in practice. And this approximate answer is independent of tiny implementation and hardware details; it remains valid even after you upgrade your computer.

Computer scientists make heavy use of a specialized ***asymptotic notation*** to describe the growth of functions approximately. The notation involves six weird little symbols. We've already encountered one:

$$f(n) \sim g(n) \quad \text{means} \quad \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 1$$

Less formally,  $f \sim g$  means that the functions  $f$  and  $g$  grow at essentially the same rate. We've already derived two important examples:

$$H_n \sim \ln n \qquad n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

Here are the other five symbols in the asymptotic notation system:

$$\begin{array}{ccccc} O & \Omega & \Theta & o & \omega \\ \text{oh} & \text{omega} & \text{theta} & \text{little-oh} & \text{little-omega} \end{array}$$

### 3.1 Big-Oh notation

We'll start with the most important one,  $O$ . Here's the definition: given functions  $f, g : \mathbb{R} \rightarrow \mathbb{R}$ , we say that

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

if there exist constants  $x_0$  and  $c > 0$  such that  $|f(x)| \leq c \cdot g(x)$  for all  $x \geq x_0$ . 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:

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

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 such inconvenient details under the rug.

Let's work through a sequence of examples carefully to better understand the 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  $x \geq x_0$ . Let  $c = 10$  and  $x_0 = 20$  and note that:

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

□

**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  $x \geq x_0$ . Let  $c = 1$  and  $x_0 = 1$  and note that

$$|x| \leq 1 \cdot x^2 \quad \text{for all } x \geq 1$$

□

What about the reverse? 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:

$$|x^2| \leq c \cdot x \quad \text{for all } x \geq x_0$$

Dividing both sides of the inequality by  $x$  gives:

$$x \leq c \quad \text{for all } x \geq x_0$$

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

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 are critical, and one should be very careful in this case.

**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:

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

Dividing both sides by  $2^x$  gives:

$$2^x \leq c \quad \text{for all } x \geq x_0$$

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

Here's another pitfall: is  $10 = O(1)$ ? This means that for all  $x$ , if  $f(x) = 10$  and  $g(x) = 1$ , then  $f(x) = O(g(x))$ . The answer is yes. For the proof, just set the constant  $c = 10$  in the big-Oh. What about  $100 = O(1)$ ? This is also true, for the same reason.

### 3.2 Big-Omega notation

Now suppose you wanted to say statements of the form “the running time of the algorithm is at least  $O(n^2)$ ”. Can we say  $f(n) \geq O(g(n))$ ? No! This statement is meaningless. To do this, we use a different symbol called “big-Omega”.

**Definition 1.** We say  $g(x) = \Omega(f(x))$  if  $\exists x_0, c > 0$  such that for all  $x \geq x_0$ , we have  $g(x) \geq c|f(x)|$ .

Big Omega is the opposite of big-Oh. More precisely,

**Theorem 6.**  $f(x) = O(g(x))$  if and only if  $g(x) = \Omega(f(x))$ .

*Proof.* We have  $f(x) = O(g(x))$  iff  $\exists x_0, c > 0$  such that  $\forall x \geq x_0, |f(x)| \leq cg(x)$ . This holds iff  $\exists x_0, c' > 0$  such that  $\forall x \geq x_0, g(x) \geq c'|f(x)|$ , where  $c' = \frac{1}{c}$ . Finally, this holds iff  $g(x) = \Omega(f(x))$ .  $\square$

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

### 3.3 Theta notation

Sometimes, we want to specify a running time precisely up to constant factors. For example, the running time might be at least quadratic and at most quadratic. For this, we use big-Theta notation.

**Definition 2.**  $f(x) = \Theta(g(x))$  iff  $f(x) = O(g(x))$  and  $f(x) = \Omega(g(x))$ . That is to say, iff there exist  $x_0, c_0, c_1 > 0$  such that  $\forall x \geq x_0$ ,

$$c_0|g(x_0)| \leq |f(x)| \leq c_1g(x_0).$$

For example,  $10x^3 - 20x^2 + 1 = \Theta(x^3)$ . Also,  $\frac{x}{\log x} \neq \Theta(x)$ . So keep in mind that if your running time  $T(n) = \Theta(n^2)$ , then it grows quadratically in  $n$ , that is, it is both upper and lower bounded by a quadratic function.

### 3.4 All the others...

Besides  $O, \Omega$ , and  $\Theta$ , there are two other Greek symbols we use:  $o$  and  $\omega$ . Intuitively,  $O$  corresponds to  $\leq$ ,  $\Omega$  to  $\geq$ , and  $\Theta$  to  $=$ . Here,  $o$  intuitively corresponds to  $<$ , and  $\omega$  to  $>$ .

**Definition 3.**  $f(x) = o(g(x))$  if  $\forall c > 0, \exists x_0$  such that  $\forall x \geq x_0, |f(x)| \leq cg(x)$ .

The difference here with big-Oh is that here  $g$  grows strictly faster than  $f$  since we quantify over all constants  $c > 0$ , whereas for big-Oh, we just require this to hold for some constant  $c$ , and so in big-Oh  $g$  only grows at least as fast as  $f$ .

For example,  $\frac{x}{\log x} = o(x)$ . Indeed, for all  $c > 0$ , there is an  $x_0$  such that for all  $x \geq x_0$ ,  $\frac{x}{\log x} \leq cx$ . Define  $x_0 = 2^{1/c}$ . Then  $\frac{x}{\log x} \leq \frac{x}{\log 2^{1/c}} = cx$ .

On the other hand,  $\frac{x}{100} \neq o(x)$ . We prove this by contradiction. If  $\frac{x}{100} = o(x)$ , then  $\forall c \exists x_0$  such that  $\forall x \geq x_0$ ,

$$\frac{x}{100} \leq cx.$$

But if  $c = \frac{1}{200}$ , this means  $\forall x \geq x_0$ ,  $\frac{x}{100} < \frac{x}{200}$ , which is impossible.

Finally, we define  $\omega$  as follows.

**Definition 4.**  $g(x) = \omega(f(x))$  if and only if  $f(x) = o(g(x))$ .

The  $\omega$  symbol is not used as commonly as the other 4 symbols.

*For more information about asymptotic notation see the recitation notes, and the following "Asymptotic Cheat Sheet".*

## The Asymptotic Cheat Sheet

Asymptotic notation consists of six funny symbols used to describe the relative growth rates of functions. These six symbols are defined in the table below.

|                 |                                   |   |
|-----------------|-----------------------------------|---|
| $f = \Theta(g)$ | $f$ grows at the same rate as $g$ | There exists an $n_0$ and constants $c_1, c_2 > 0$ such that for all $n > n_0$ , $c_1g(n) \leq  f(n)  \leq c_2g(n)$ . |
| $f = O(g)$      | $f$ grows no faster than $g$      | There exists an $n_0$ and a constant $c > 0$ such that for all $n > n_0$ , $ f(n)  \leq cg(n)$ .                      |
| $f = \Omega(g)$ | $f$ grows at least as fast as $g$ | There exists an $n_0$ and a constant $c > 0$ such that for all $n > n_0$ , $cg(n) \leq  f(n) $ .                      |
| $f = o(g)$      | $f$ grows slower than $g$         | For all $c > 0$ , there exists an $n_0$ such that for all $n > n_0$ , $ f(n)  \leq cg(n)$ .                           |
| $f = \omega(g)$ | $f$ grows faster than $g$         | For all $c > 0$ , there exists an $n_0$ such that for all $n > n_0$ , $cg(n) \leq  f(n) $ .                           |
| $f \sim g$      | $f/g$ approaches 1                | $\lim_{n \rightarrow \infty} f(n)/g(n) = 1$   |

The  $\sim$  and  $\Theta$  notations are confusingly similar; qualitatively, functions related by  $\sim$  must be even more nearly alike than functions related by  $\Theta$ . The  $\omega$  notation makes the table nice and symmetric, but is almost never used in practice. Some asymptotic relationships between functions imply other relationships. Some examples are listed below.

$$\begin{array}{ll} f = O(g) \text{ and } f = \Omega(g) & \Leftrightarrow f = \Theta(g) & f = o(g) & \Rightarrow f = O(g) \\ f = O(g) & \Leftrightarrow g = \Omega(f) & f = \omega(g) & \Rightarrow f = \Omega(g) \\ f = o(g) & \Leftrightarrow g = \omega(f) & f \sim g & \Rightarrow f = \Theta(g) \end{array}$$

## Limits

The definitions of the various asymptotic notations are closely related to the definition of a limit. As a result,  $\lim_{n \rightarrow \infty} f(n)/g(n)$  reveals a lot about the asymptotic relationship between  $f$  and  $g$ , provided the limit exists. The table below translates facts about the limit of  $f/g$  into facts about the asymptotic relationship between  $f$  and  $g$ .

$$\begin{array}{ll} \lim_{n \rightarrow \infty} f(n)/g(n) \neq 0, \infty & \Rightarrow f = \Theta(g) & \lim_{n \rightarrow \infty} f(n)/g(n) = 1 & \Rightarrow f \sim g \\ \lim_{n \rightarrow \infty} f(n)/g(n) \neq \infty & \Rightarrow f = O(g) & \lim_{n \rightarrow \infty} f(n)/g(n) = 0 & \Rightarrow f = o(g) \\ \lim_{n \rightarrow \infty} f(n)/g(n) \neq 0 & \Rightarrow f = \Omega(g) & \lim_{n \rightarrow \infty} f(n)/g(n) = \infty & \Rightarrow f = \omega(g) \end{array}$$

Therefore, skill with limits can be helpful in working out asymptotic relationships. In particular, recall L'Hospital's Rule:

$$\text{If } \lim_{n \rightarrow \infty} f(n) = \infty \text{ and } \lim_{n \rightarrow \infty} g(n) = \infty, \text{ then } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)}.$$

Every computer scientist knows two rules of thumb about asymptotics: logarithms grow more slowly than polynomials and polynomials grow more slowly than exponentials. We'll prove these facts using limits.

**Theorem.** For all  $\alpha, k > 0$ :

$$(\ln n)^k = o(n^\alpha) \tag{1}$$

$$n^k = o((1 + \alpha)^n) \tag{2}$$

*Proof.*

$$\lim_{n \rightarrow \infty} \frac{(\ln n)^k}{n^\alpha} = \left( \lim_{n \rightarrow \infty} \frac{\ln n}{n^{\alpha/k}} \right)^k \stackrel{*}{=} \left( \lim_{n \rightarrow \infty} \frac{1/n}{(\alpha/k)n^{\alpha/k-1}} \right)^k = \left( \lim_{n \rightarrow \infty} \frac{1}{(\alpha/k)n^{\alpha/k}} \right)^k = 0$$

$$\lim_{n \rightarrow \infty} \frac{n^k}{(1 + \alpha)^n} = \left( \lim_{n \rightarrow \infty} \frac{n}{(1 + \alpha)^{n/k}} \right)^k \stackrel{*}{=} \left( \lim_{n \rightarrow \infty} \frac{1}{(n/k) \cdot (1 + \alpha)^{n/k-1}} \right)^k = 0$$

The starred equalities follow from L'Hospital's Rule. □