Lecture 16: Foldr and Recurrence Proofs

10:00 AM, Oct 9, 2019

Contents

Objectives

By the end of this class, you will:

• Be able to use fold to write recursive procedures
• Be able to use let expressions
• Be able to write a proof that a conjecture you’ve made about a recurrence, using plug-n-chug, is actually correct.
• Have a solid introduction to Big-O

1 Foldr

There are a few things to keep in mind about foldr expressions that will hopefully make them easier to use and understand:

• The syntax for a fold-expression is

(fold proc base alod)

where proc is a two-argument procedure, base is the base value, and alod is the list that will be recurred through.

• One way to think about the base value is it being the initial value of an accumulator. As you go through the list recursively (from right to left in foldr and left to right in foldl), you keep doing something to the accumulator based on the current element in the list. For example, say we wanted to find the product of all the elements in the following list: (list 2 3 4 5). In this case, our proc would be multiplication, and we would initially set our accumulator to 1 (base value). This way, when 5, the last element of the list, is multiplied by the base value, it would return 5. Now our accumulator is 5. The next step is applying our procedure, *, to the second right-most element, 4, and the accumulator. This outputs 20, our new accumulator. This process continues, we keep building our accumulator, until the procedure has been applied to the first element of the list and whatever the accumulator is at that point. In this case, that would be the multiplication of 2 and (foldr * 1 (list 3 4 5)), or 60, which would output 120.

• This may seem counter-intuitive because the list goes from left to right (the leftmost element is the first element, the rightmost element is the last element). Let’s take a look at what
foldr is really doing in the example above: (foldr * 1 (list 2 3 4 5)) is really (* 2 (* 3 (* 4 (* 5 1)))). Rules of evaluation say to evaluate each of the arguments first, then apply the procedure to those arguments. In this case, 2 would be evaluated first, but to evaluate the second argument, all of (* 3 (* 4 (* 5 1))) needs to be evaluated. To do this, (* 4 (* 5 1)) needs to be evaluated. Finally, both arguments in (* 5 1) are evaluated, and the multiplication procedure is applied to them. Now, the (* 4 (* 5 1)) computation can happen, because we have evaluated (* 5 1). Once this is evaluated, (* 3 (* 4 (* 5 1))) can be evaluated because we have evaluated (* 4 (* 5 1)). This process continues until we get all the way to

2 Analysis: last steps

We’ve talked about analyzing procedures using several steps,

- Describe the operation-counting function
- Derive from the program a recurrence relation on this operation counting function
- Use plug-n-chug to guess a closed-form expression for this function
- Give a proof that the guess (or 'conjecture') you made is correct
- Identify the op-counting function as lying in a particular big-O class of functions.

We’ve done all those things except the ‘give a proof’ part, and that’s today’s task.

When we looked at something like the contains17? procedure, we found that its op-counting function (which I’m going to call $H$) satisfied the recurrence

$$
H(0) = A \\
H(n) \leq B + H(n - 1)
$$

for $n > 0$ (2)

where $A$ and $B$ were nonnegative constants. From plug-n-chug, we conjectured that in general,

$$
H(n) \leq Bn + A
$$

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

We now want to prove that.

This requires two setup ingredients.

2.1 Well-ordering principle

Here’s a statement:

“A nonempty set $S$ of nonnegative integers has a smallest element.”

By a “smallest element”, I mean a number $u \in S$ with the property that for every $n \in S$, we have $u \leq n$.

Looking at this statement, you probably believe it’s true. You say to yourself, “When $S = \{2, 3, 9\}$, the number 2 is the smallest element. When $S$ is all the odd integers, the number 1 is the smallest element. Seems good to me!”
But it’s a good idea, when you encounter a claim like this, to test all the conditions. Is the statement still true if you remove the word nonnegative? What if you remove nonempty? or integers? It turns out that every one of those things is necessary. For instance, the integers themselves are a nonempty set of integers (the word ‘nonnegative’ has been dropped), but they have no least element.

If we drop ‘nonempty’, then the empty set is a set of nonnegative integers, but has no least element.

If we drop ‘nonnegative’, then the set of all integers has no least element.

If we drop ‘integers’, then the set of all real numbers $x$ with $1 < x < 2$ is a nonempty set of nonnegative numbers, but it has no least element.

On the other hand, if we keep all the conditions in the statement, it appears to be solidly true, and this has motivated mathematicians to adopt it as one of the axioms of arithmetic. It’s called the Well-ordering principle or Well-ordering axiom.

A side note: If you’ve learned about induction in the past, you may have encountered the axiom of mathematical induction. From that axiom, you can prove the well-ordering principle; from the well-ordering principle, you can prove the axiom of mathematical induction. Some mathematicians choose to include one of these among their axioms; some choose the other. The number systems you get in both cases are identical. And if you really like proofs by induction, you can use them rather than the proofs I’ll describe using well-ordering. But they’ll be subject to the same degree of scrutiny as our well-ordering proofs, so you’d better get them right.

We, in C17, will adopt the well-ordering principle as an axiom, and use it in our proofs of plug-n-chug conjectures.

### 2.2 Proofs by contradiction

A standard way to prove things is to use a technique called “contradiction”: you assume something is true, reason from that to find other things that must be true, until you eventually arrive at something you know to be false. When that happens, you conclude that the thing you assumed must have been wrong.

This makes a lot of sense when you think of it not in terms of mathematics, but in terms of ordinary experience. For instance, at the end of an Agatha Christie mystery novel, you find the detective saying things like “If Joey really was on that 5:32 commuter train that was very full, then someone from Tillybridge would have seen him, and we’ve interviewed hundred of people, and not one of them saw him on that train. So he must not have been on it.”

In this case, the “assumed thing” (which we call the contradiction hypotheses) is “Joey was on the train.”

The reasoning led us to say that if that was true, someone would have seen him. But we also know that no one did see him.

That’s a contradiction, and we conclude that our original assumption — that Joey was on the train — must have been false.

Our proofs by contradiction will have this form, and they’ll all follow exactly one pattern:

- We’ll describe a set $S$. 
We’ll assume that \( S \) is nonempty.
Well show that this leads to something false.
We’ll conclude that the set \( S \) must be empty after all.

### 2.3 An example

Let’s continue with the contains17? example above. We know that

\[
\begin{align*}
H(0) &= A \\
H(n) &\leq B + H(n - 1) \quad \text{for } n > 0
\end{align*}
\]

and we have a conjecture, namely that for every \( n \in \mathbb{N} \),

\[
H(n) \leq Bn + A.
\]

Now we’re going to make a claim and prove it:

**Theorem 1.** If \( H : \mathbb{N} \rightarrow \mathbb{N} \) satisfies the recurrence

\[
\begin{align*}
H(0) &= A \\
H(n) &\leq B + H(n - 1) \quad \text{for } n > 0
\end{align*}
\]

then for every \( n \in \mathbb{N} \),

\[
H(n) \leq Bn + A. \tag{*}
\]

**Proof.** Let \( S \) be the set of natural numbers \( n \) for which Equation \((*)\) is false. I want to show that \( S \) is empty, and I’ll be done.

Suppose not. Then \( S \), being a nonempty set of natural numbers, has a least element \( k \), based on the Well-Ordering Principle.

From the recurrence relation, \( H(0) = A \). Plugging in 0 into the closed-form conjecture, \( H(0) \leq B \cdot 0 + A \). Because \( H(0) = A \leq B \cdot 0 + A \), this shows that the closed-form conjecture satisfies the recurrence relation, which we know to be true, for \( n = 0 \), so Equation\((*)\) is true for \( n = 0 \), so \( k > 0 \).

Because \( k \) is the smallest element of \( S \), Equation \((*)\) must be true for \( n = k - 1 \).

Hence, plugging in \( k - 1 \) into Equation \((*)\),

\[
H(k - 1) \leq B(k - 1) + A.
\]

Now from the recurrence relation, since \( k > 0 \), we can plug in \( k \) for \( n \) in Equation \((7)\)

\[
H(k) \leq B + H(k - 1),
\]

We can plug in the equation for \( H(k - 1) \) into this equation and perform some algebra

\[
H(k) \leq B + H(k - 1) \leq B + (B(k - 1) + A) = Bk + A.
\]

But this shows that Equation \((*)\) is actually true for \( k \), which contradicts the fact that \( k \) is an element of \( S \). hence our assumption — that the set \( S \) was nonempty — must be false, and we are done. \( \diamond \)
Aside from the specifics of the algebra, all analysis proofs will look like this, indeed, will contain many of the same words in the same order.

We can also do a two-column proof of this same claim. The nice thing about this is that almost all the steps remain the same from proof to proof. Here goes:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Let ( S = { n \in \mathbb{N} \mid (*) \text{ is false for } n } )</td>
<td>Definition of ( S )</td>
</tr>
<tr>
<td>2. Suppose that ( S ) is nonempty</td>
<td>Contradiction hypothesis</td>
</tr>
<tr>
<td>3. Let ( k ) be the least element of ( S )</td>
<td>Well-ordering</td>
</tr>
<tr>
<td>4. ((*)), for ( n = 0 ), says that ( H(0) \leq B \cdot 0 + A = A )</td>
<td>Restatement of ((*)).</td>
</tr>
<tr>
<td>5. ((*)), for ( n = 0 ), is true</td>
<td>Line 1 of recurrence</td>
</tr>
<tr>
<td>6. ( k \neq 0 )</td>
<td>If it were, then ((*)) would not be true for ( n = 0 ); S5.</td>
</tr>
<tr>
<td>7. ( k &gt; 0 )</td>
<td>( k ) is a natural number, and ( k \neq 0 ) by S6.</td>
</tr>
<tr>
<td>8. ((*)) is true for ( n = k - 1 )</td>
<td>S1, S3. ( k ) is the smallest element of ( S ), so ( k - 1 ) is not in ( S ), so ((*)) is true for ( k - 1 ).</td>
</tr>
<tr>
<td>9. ( H(k - 1) \leq B \cdot (k - 1) + A )</td>
<td>Restatement of S8 with ( k - 1 ) plugged in</td>
</tr>
<tr>
<td>10. ( H(k) \leq B + H(k - 1) )</td>
<td>Because ( k &gt; 0 ) (see S7) and line 2 of recurrence.</td>
</tr>
<tr>
<td>11. ( H(k) \leq B + [B \cdot (k - 1) + A] )</td>
<td>S10, S9.</td>
</tr>
<tr>
<td>12. ( H(k) \leq B + Bk - B + A )</td>
<td>S11, algebra.</td>
</tr>
<tr>
<td>13. ( H(k) \leq Bk + A )</td>
<td>S12, algebra.</td>
</tr>
<tr>
<td>14. ( k \notin S )</td>
<td>S13, definition of ( S ) in S1.</td>
</tr>
<tr>
<td>15. Contradiction</td>
<td>S14, S3.</td>
</tr>
<tr>
<td>16. S2 is false, so ((*)) holds for all natural numbers</td>
<td>S2, S15.</td>
</tr>
</tbody>
</table>

The great news about this is that only a few things have to change when we want to do a different proof. They’re highlighted in red here, with items that might change highlighted in green:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Let ( S = { n \in \mathbb{N} \mid (*) \text{ is false for } n } )</td>
<td>Definition of ( S )</td>
</tr>
<tr>
<td>2. Suppose that ( S ) is nonempty</td>
<td>Contradiction hypothesis</td>
</tr>
<tr>
<td>3. Let ( k ) be the least element of ( S )</td>
<td>Well-ordering</td>
</tr>
<tr>
<td>4. ((*)), for ( n = 0 ), says that ( H(0) \leq B \cdot 0 + A = A )</td>
<td>Restatement of ((*)).</td>
</tr>
<tr>
<td>5. ((*)), for ( n = 0 ), is true</td>
<td>Line 1 of recurrence</td>
</tr>
<tr>
<td>6. ( k \neq 0 )</td>
<td>If it were, then ((*)) would not be true for ( n = 0 ); S5.</td>
</tr>
<tr>
<td>7. ( k &gt; 0 )</td>
<td>( k ) is a natural number, and ( k \neq 0 ) by S6.</td>
</tr>
<tr>
<td>8. ((*)) is true for ( n = k - 1 )</td>
<td>S1, S3. ( k ) is the smallest element of ( S ), so ( k - 1 ) is not in ( S ), so ((*)) is true for ( k - 1 ).</td>
</tr>
<tr>
<td>9. ( H(k - 1) \leq B \cdot (k - 1) + A )</td>
<td>Restatement of S8 with ( k - 1 ) plugged in.</td>
</tr>
</tbody>
</table>
We’ll see how those marked lines change in a particularly trivial example, where the recurrence is easy to solve by eye.

In this example, $H$ satisfies the recurrence

\[
\begin{align*}
H(0) &= A \\
H(n) &\leq H(n-1) \\
\end{align*}
\]  

for $n > 0$ \hspace{1cm} (8) \hspace{1cm} (9)

and plug-n-chug leads us to the conjecture that for all $n \in \mathbb{N}$

\[
H(n) \leq A \tag{*}
\]

Let’s prove that conjecture is correct. The two-column proof starts out with much of the same elements from the previous proof, with only lines in red that must be changed and lines in green that may be changed.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Let $S = { n \in \mathbb{N} \mid (*) \text{ is false for } n }$</td>
<td>Definition of $S$</td>
</tr>
<tr>
<td>2 Suppose that $S$ is nonempty</td>
<td>Contradiction hypothesis</td>
</tr>
<tr>
<td>3 Let $k$ be the least element of $S$</td>
<td>Well-ordering</td>
</tr>
<tr>
<td>4 $\ast$, for $n = 0$, says that $H(0) \leq \ldots$</td>
<td>Restatement of $\ast$.</td>
</tr>
<tr>
<td>5 $\ast$, for $n = 0$, is true</td>
<td>Line 1 of recurrence</td>
</tr>
<tr>
<td>6 $k \neq 0$</td>
<td>If it were, then $\ast$ would not be true for $n = 0$; S5.</td>
</tr>
<tr>
<td>7 $k &gt; 0$</td>
<td>$k$ is a natural number, and $k \neq 0$ by S6.</td>
</tr>
<tr>
<td>8 $\ast$ is true for $n = k - 1$</td>
<td>S1, S3. $k$ is the smallest element of $S$, so $k - 1$ is not in $S$, so $\ast$ is true for $k - 1$.</td>
</tr>
<tr>
<td>9 $H(k - 1) \leq \ldots$</td>
<td>Restatement of S8 with $k - 1$ plugged in.</td>
</tr>
<tr>
<td>10 $H(k) \leq \ldots$</td>
<td>Because $k &gt; 0$ (see S7) and line 2 of recurrence.</td>
</tr>
<tr>
<td>11 $H(k) \leq \ldots$</td>
<td>S10, S9.</td>
</tr>
<tr>
<td>12 $H(k) \leq \ldots$</td>
<td>S11, algebra.</td>
</tr>
<tr>
<td>13 $H(k) \leq \ldots$</td>
<td>S12, algebra.</td>
</tr>
<tr>
<td>14 $k \notin S$</td>
<td>S13, definition of $S$ in S1.</td>
</tr>
<tr>
<td>15 Contradiction</td>
<td>S14, S3.</td>
</tr>
<tr>
<td>16 S2 is false, so $\ast$ holds for all natural numbers</td>
<td>S2, S15.</td>
</tr>
</tbody>
</table>
Changing the lines in red and green gives:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  Let $S = { n \in \mathbb{N} \mid (\ast) \text{ is false for } n }$</td>
<td>Definition of $S$</td>
</tr>
<tr>
<td>2  Suppose that $S$ is nonempty</td>
<td>Contradiction hypothesis</td>
</tr>
<tr>
<td>3  Let $k$ be the least element of $S$</td>
<td>Well-ordering</td>
</tr>
<tr>
<td>4  $(\ast)$, for $n = 0$, says that $H(0) \leq A$</td>
<td>Restatement of $(\ast)$</td>
</tr>
<tr>
<td>5  $(\ast)$, for $n = 0$, is true</td>
<td>Line 1 of recurrence</td>
</tr>
<tr>
<td>6  $k \neq 0$</td>
<td>If it were, then $(\ast)$ would not be true for $n = 0$; S5.</td>
</tr>
<tr>
<td>7  $k &gt; 0$</td>
<td>$k$ is a natural number, and $k \neq 0$ by S6.</td>
</tr>
<tr>
<td>8  $(\ast)$ is true for $n = k - 1$</td>
<td>S1, S3. $k$ is the smallest element of $S$; so $k - 1$ is not in $S$, so $(\ast)$ is true for $k - 1$.</td>
</tr>
<tr>
<td>9  $H(k - 1) \leq A$</td>
<td>Restatement of S8 with $k - 1$ plugged in.</td>
</tr>
<tr>
<td>10 $H(k) \leq H(k - 1)$</td>
<td>Because $k &gt; 0$ (see S7) and line 2 of recurrence.</td>
</tr>
<tr>
<td>11 $H(k) \leq A$</td>
<td>S10, S9.</td>
</tr>
<tr>
<td>14 $k \notin S$</td>
<td>S13, definition of $S$ in S1.</td>
</tr>
<tr>
<td>15 Contradiction</td>
<td>S14, S3.</td>
</tr>
<tr>
<td>16 S2 is false, so $(\ast)$ holds for all natural numbers</td>
<td>S2, S15.</td>
</tr>
</tbody>
</table>

And now for a more interesting recurrence. In this example, $H$ satisfies the recurrence

\[
H(0) = A \tag{10}
\]
\[
H(n) \leq Pn + H(n - 1) \tag{11}
\]

and plug-n-chug leads us to the conjecture that for all $n \in \mathbb{N}$

\[
H(n) \leq \frac{Pn^2}{2} + \frac{Pn}{2} + A \tag{\ast}
\]

Let’s prove that conjecture is correct. The two-column proof starts out with much of the same elements from the previous proof, with only lines in red that must be changed and lines in green that may be changed
Let $S = \{ n \in \mathbb{N} \mid (*) \text{ is false for } n \}$ be the set of natural numbers for which (*) is false. Suppose that $S$ is nonempty. Let $k$ be the least element of $S$. (*) for $n = 0$, says that $H(0) \leq P \cdot 0 + P \cdot 0 + A = A$. (*) for $n = 0$, is true. If it were, then (*) would not be true for $n = 0$; $S5$. $k > 0$ is a natural number, and $k \neq 0$ by $S6$. (*) is true for $n = k - 1$ is the smallest element of $S$, so $k - 1$ is not in $S$, so (*) is true for $k - 1$. 

$H(k - 1) \leq P(k - 1)^2 + P(k - 1) + A$ is the smallest element of $S$, so $k - 1$ is not in $S$, so (*) is true for $k - 1$. Restatement of S8 with $k - 1$ plugged in. 

$H(k) \leq \frac{P(k-1)^2}{2} + \frac{P(k-1)}{2} + A$ is a natural number, and $k \neq 0$ by $S6$. $H(k) \leqPk + H(k - 1)$ is the smallest element of $S$, so $k - 1$ is not in $S$, so (*) is true for $k - 1$. Restatement of S8 with $k - 1$ plugged in. 

$H(k) \leq \frac{2P}{k} + \frac{P^2 - 2P + P}{2} + \frac{P - P}{2} + A$ is a natural number, and $k \neq 0$ by $S6$. $H(k) \leq \frac{2P + Pk^2 - 2P + P + Pk - P}{2} + A$ is the smallest element of $S$, so $k - 1$ is not in $S$, so (*) is true for $k - 1$. Restatement of S8 with $k - 1$ plugged in. 

$H(k) \leq \frac{2P + Pk^2 - 2P + P + P - P}{2} + A$ is a natural number, and $k \neq 0$ by $S6$. $H(k) \leq \frac{P + Pk}{2} + A$ is the smallest element of $S$, so $k - 1$ is not in $S$, so (*) is true for $k - 1$. Restatement of S8 with $k - 1$ plugged in. 

$H(k) \leq \frac{Pk^2 + Pk}{2} + A$ is a natural number, and $k \neq 0$ by $S6$. $k \notin S$ is the smallest element of $S$, so $k - 1$ is not in $S$, so (*) is true for $k - 1$. Restatement of S8 with $k - 1$ plugged in. 

$H(k) \leq \frac{Pk^2 + P}{2} + A$ is the smallest element of $S$, so $k - 1$ is not in $S$, so (*) is true for $k - 1$. Restatement of S8 with $k - 1$ plugged in. 

Contradiction $S2$ is false, so (*) holds for all natural numbers. 

Changing the lines in red and green gives:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S = { n \in \mathbb{N} \mid (*) \text{ is false for } n }$</td>
<td>Definition of $S$</td>
</tr>
<tr>
<td>Suppose that $S$ is nonempty</td>
<td>Contradiction hypothesis</td>
</tr>
<tr>
<td>Let $k$ be the least element of $S$</td>
<td>Well-ordering</td>
</tr>
<tr>
<td>(*) for $n = 0$, says that $H(0) \leq P \cdot 0 + P \cdot 0 + A = A$</td>
<td>Restatement of (*).</td>
</tr>
<tr>
<td>(*) for $n = 0$, is true</td>
<td>Line 1 of recurrence</td>
</tr>
<tr>
<td>$k &gt; 0$ is a natural number, and $k \neq 0$ by $S6$.</td>
<td>If it were, then (*) would not be true for $n = 0$; $S5$.</td>
</tr>
<tr>
<td>(*) is true for $n = k - 1$</td>
<td>$S1$, $S3$. $k$ is the smallest element of $S$, so $k - 1$ is not in $S$, so (*) is true for $k - 1$.</td>
</tr>
<tr>
<td>$H(k - 1) \leq P(k - 1)^2 + P(k - 1) + A$</td>
<td>Restatement of S8 with $k - 1$ plugged in.</td>
</tr>
<tr>
<td>$H(k) \leq \frac{P(k-1)^2}{2} + \frac{P(k-1)}{2} + A$</td>
<td>Because $k &gt; 0$ (see S7) and line 2 of recurrence.</td>
</tr>
<tr>
<td>$H(k) \leq \frac{2P}{k} + \frac{P^2 - 2P + P}{2} + \frac{P - P}{2} + A$</td>
<td>$S10$, $S9$.</td>
</tr>
<tr>
<td>$H(k) \leq \frac{2P + Pk^2 - 2P + P + Pk - P}{2} + A$</td>
<td>$S11$, algebra.</td>
</tr>
<tr>
<td>$H(k) \leq \frac{P + Pk}{2} + A$</td>
<td>$S12$, algebra.</td>
</tr>
<tr>
<td>$H(k) \leq \frac{Pk^2 + Pk}{2} + A$</td>
<td>$S12A$, algebra.</td>
</tr>
<tr>
<td>$k \notin S$</td>
<td>$S13$, definition of $S$ in $S1$.</td>
</tr>
<tr>
<td>Contradiction</td>
<td>$S14$, $S3$.</td>
</tr>
<tr>
<td>$S2$ is false, so (*) holds for all natural numbers</td>
<td>$S2$, $S15$.</td>
</tr>
</tbody>
</table>
3 Comparing functions, and big-O

We’re going to hold off on proving the conjecture is correct, and instead talk about the last step. To do so, I have to say what a big-O class is.

To do so, I have to go back to the things you learned on the Homeworks: the notion of “f is eventually less than g” and “f is less than g up to constants”. Notice that both of these are statements about whole functions, which is an extension of the kinds of things you used to do in algebra class, where you mostly talked about numbers, and eventually proved things like \( x < 2y \), i.e., that two numbers had a particular relationship. We’re talking about functions having a particular relationship to each other.

In everything that follows, I’m going to be talking about functions from \( \mathbb{N} \) to \( \mathbb{N} \), because all we care about in this class are operation-counting functions for procedures. The size of the data consumed by the procedure (typically a list) will be a natural number; the number of operations will also be a natural number. So everything, from here on in this discussion of analysis, is an \( \mathbb{N} \rightarrow \mathbb{N} \) function.

We’re going to combine these two ideas into one, and say that “f is eventually less than g, up to constants”, if there are numbers \( M, c > 0 \) with the property that for every integer \( n > M \), we have

\[
 f(n) \leq c \cdot g(n).
\]

(Again: this definition applies only to \( \mathbb{N} \rightarrow \mathbb{N} \) functions!)

Thinking of f and g as op-counting functions for a couple of procedures that produce the same result (e.g., two different ways of implementing “set-intersection”), this dominance notion says that f’s procedure may not be as good as g’s for small problems \( (n \leq M) \), but that once the problems are large enough \( (n > M) \), f performs at least as well as g, at least if we’re willing to ignore a constant multiple.

If we write (for the next few minutes), \( f << g \) to mean “f is eventually less than g, up to constants”, then there are some basic facts that are pretty useful.

- For any function f, we have \( f << f \).
- If \( f << g \) and \( g << h \), then \( f << h \).
- If \( f(n) = 0 \) for every \( n \), then \( f << g \) for every function g.

The last of these is easy to prove: you pick \( M = 0 \) and \( c = 1 \).

That’s a general form for these proofs: to show that there are numbers \( M \) and \( c \) with the desired property, you say what they are, and then prove that those numbers have the desired property.

I’ve made some claims here, but haven’t proved them, because those proofs are easier once you’ve seen a couple of very concrete examples.

Let’s do a first example. I claim that the function \( f(n) = 3n \) is eventually less than the function \( g(n) = n \), up to constants.

To prove this, I have to show that they satisfy the definition, so I have to exhibit numbers \( M \) and \( c \) with the required property.

To find those numbers, I do a little background work: I draw the graphs (use Desmos!) and see that g looks as if it’s mostly less than f. But if I multiply g by some constant, I can make
it just as big as \( f \) . . . and a good constant for doing that is 3. Once I do so, I also have to pick a value \( M \) whose meaning is “to the right of this value, I can see that my multiple of \( g \) is at least as big as \( f \).” Well, since \( f = 3g \), just about any value of \( M \) will work, and I’m going to choose one, namely \( M = 1 \), to make things concrete.

OK. Now I’m ready to go.

I claim that \( f \) is eventually less than \( g \) up to constants, with \( M = 1 \) and \( c = 3 \) as proof of that. I can restate this claim:

Claim: Picking \( M = 1 \) and \( c = 3 \), for \( n > M \), \( f(n) \leq cg(n) \).

I can restate that again, simplifying a little bit:

Claim: for \( n > 1 \), \( f(n) \leq 3 \cdot g(n) \).

Is this last claim true? Well, for any \( n > 1 \) (indeed, for any \( n \) at all), we have

\[
f(n) = 3n
\]

by the definition of \( f \). And we have

\[
3 \cdot g(n) = 3 \cdot n
\]

by the definition of \( g \). So

\[
f(n) = 3n \leq 3 \cdot n = 3 \cdot g(n),
\]

and we’ve proved our claim.

Let’s do that again for a slightly more interesting case, where “eventually” will come into play.

Let’s look at \( f(n) = 3n + 5 \) and \( g(n) = n \). Once again, graphing shows that not only is \( f \) always bigger than \( g \), by no matter how much you multiply \( g \) by, \( f(0) \) will always be bigger than \( c \cdot g(0) \). So we’ll definitely need to consider the “eventually” part of things. If, as above, we multiply \( g \) by 3, we’ll be comparing \( 3n + 5 \) to \( 3n \), and the \( 3n \) will always be less. But who says we have to use 3? Let’s pick \( c = 4 \), because graphing shows that when we do that, \( 3n + 5 \) is eventually less than \( 4n \).

Let’s continue with our preparation work by asking “How big does \( n \) have to be for \( 3n + 5 \) to be less than or equal to \( 4n \)?” Well, if we solve

\[
3n + 5 \leq 4n \\
5 \leq 4n - 3n \\
5 \leq n
\]

it appears that \( n > 4 \) is sufficient. So we’ll pick \( M = 4 \) and \( c = 4 \). (In today’s slides, I picked \( M = 5 \); both work fine!)

**Question:** Why can you pick \( M = 4 \) when \( n \) must be greater than 5?

**Answer:** Recall that \( n > M \), and since we are working with natural numbers, if \( n > 4 \) then \( n \geq 5 \).

So my claim is this:

Claim: for \( M = 4, c = 4 \), we have that for \( n > M \), \( f(n) \leq c \cdot g(n) \).
We restate this by plugging in the values of $M$ and $c$ to get

Claim: $n > 4$, $f(n) \leq 4 \cdot g(n)$.

Is this true? Well, suppose $n > 4$. Adding $3n$ to both sides, we get $3n + n > 3n + 4$, i.e. $4n > 3n + 4$. Now that means that $4n \geq 3n + 5$, which is the same as $3n + 5 \leq 4n$, which is the same as $f(n) \leq 4g(n)$, and we’re done.

This idea — one function eventually being no more than another, up to constants, is so special that it gets its own name and notation.

We define the set $O(g)$ to be all functions $h$ with the property that $h << g$, i.e., all functions that are eventually less than $g$.

So we’ve shown, above, that

\[
(n \mapsto 3n) \in O(n \mapsto n) \\
(n \mapsto 3n + 5) \in O(n \mapsto n)
\]

$\text{f(n) = 6n + 10 g(n) = 2n}$ Pick values for $M$ and $c$ that show that $f(n)$ is eventually less than $g$ up to constants. Let $c$ be the smallest constant that shows this.

### 3.1 A brief discussion on notation

You may have heard or seen the expression $O(n)$ before. However, this notation is misleading, as $n$ is not a function. In this class, we write $O(n \mapsto n)$ to show that we are talking about a set of functions.

I’m going to write down two functions, and I want you to think about whether they are the same function or different. You might want to review the definition of “sameness” for functions before committing yourself to an answer.

\[
h : \mathbb{N} \rightarrow \mathbb{N} : n \mapsto n + 1 \\
k : \mathbb{N} \rightarrow \mathbb{N} : k \mapsto k + 1
\]

So ... are $h$ and $k$ the same function?

They are, because they have the same domain, and the same codomain, and for every element $x$ in the domain, the value $h(x)$ is the same as the value $k(x)$.

That means that the sets $O(n \mapsto n)$ and $O(k \mapsto k)$ denote exactly the same thing. By long tradition, computer scientists use the letter $n$, but the letter $k$ (or any other) is equally good.

### 3.2 A practical consequence

We already showed (except for a missing proof of the conjecture) that the operation-counting function, $L$, for the length procedure satisfied a particular recurrence, and that in fact there are nonnegative numbers $A$ and $B$ with $L(n) \leq Bn + A$ for all $n \in \mathbb{N}$.

Using a proof like the one above (picking $c = B + A$, for instance, and $M = 1$), we can show that $L$ is eventually less than (up to constants) the function $n \mapsto n$, hence that $L \in O(n \mapsto n)$. Informally, we say that “length is a $O(n \mapsto n)$ procedure.” It’s important to remember that what this really means is that the op-counting function for length is in the class $O(n \mapsto n)$.

Let’s dignify that example with a theorem that we can apply to lots of other examples as well.
Theorem 2. For any $A, B \geq 0$, the function $f(n) = An + B$ is in $O(k \mapsto k)$ (or, equivalently, $O(n \mapsto n)$).

To see the proof for this theorem, check out the slides.

We went over one other theorem:

Theorem 3. For any $A, B, C \geq 0$, if $f$ satisfies a recurrence of the form:

\[
\begin{align*}
f(0) &= C \\
f(n) &\leq An + B + f(n-1)
\end{align*}
\]

for $n > 0$, then $f \in O(n \mapsto n^2)$.

4 Summary

Ideas

- Proof by contradiction and the well-ordering principle are powerful tools in proving a conjecture.
- Procedures are data.
- “Mapping” a procedure over a list of data, or filtering data, is a powerful way to do many of the things we’ve done by writing recursive procedures up until now.
- `let` expressions can be used to create temporary bindings, and can really improve runtime in certain cases.
- Modified the rules for applying a user-defined proc to its args to accommodate lambda.

Skills

- You now know how to write the “type” for a procedure that’s used as the input or output of another procedure: the type is simply something containing an arrow and put in parens, like `(int -> bool)` or `(str * str -> str)`.
- You know how to map procedures over data, and how to filter data.
- You know how to write an expression whose value is a user-defined procedure that is not in the top-level environment (i.e. a `lambda`).
- Understanding new rules of evaluation that include lambda.
- Understanding situations where `let` expressions can be used.
- You know how to prove a conjecture using proof by contradiction and the well-ordering principle.

Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS 17 document by filling out the anonymous feedback form: [http://cs.brown.edu/courses/csci0170/feedback](http://cs.brown.edu/courses/csci0170/feedback).